GVERDI-R A Tool for Repairing Faulty Web Sites

María Alpuente, Demis Ballis
Moreno Falaschi, and Daniel Romero

DSIC, Technical University of Valencia, Spain
Dip. Matematica e Informatica, Udine, Italy
Dip. di Scienze Matematiche e Informatiche, Siena, Italy

PROLE 2006 (Sitges, Spain)

Contents

1. Introduction
2. What GVERDI-R is and how it works
3. The Web Repairing Methodology
4. Strategies
5. Prototype
6. Conclusions and Future work
Purpose

- The increasing complexity of Web sites has made their design and construction a challenging problem.
- The development and maintenance of Web sites are difficult tasks.
- To maintain the consistency of ever-larger and complex Web sites, Web administrators need effective mechanisms to assist them in fixing every possible inconsistency.
- Systematic, formal approaches can bring many benefits to Web site construction, giving support to automated Web site verification and repair.
What is GVERDI-R?

- In previous works, we developed GVERDI-R (Graphical Verification and Rewriting for Debuging Internet sites), which is based on a rewriting-like approach to Web site specification and verification.

- It does the following:
  - Allows us to specify the integrity conditions for the Web sites.
  - Diagnoses errors by computing the requirements not fulfilled by a given Web site.
  - Finds incorrect/forbidden patterns and missing/incomplete Web pages.

- In this work, we present a methodology that allows us to obtain a correct and complete Web site by applying appropriate repair actions.
What we need to verify

- **A Web site**
  - Since Web pages are provided with a tree-like structure, they can be straightforwardly translated into ordinary terms of a given term algebra $\tau(\text{Text} \cup \text{Tag})$.
  - A *Web site* is a finite collection of ground terms $\{p_1 \ldots p_n\}$.
  - A web page is represented as a tree-like structure.

- **A specification**
  - A Web specification is a triple $(I_N, I_M, R)$,
  - $I_N$ is a finite set of correctness rules.
  - $I_M$ is a finite set of completeness rules.
  - $R$ contains the definition of some auxiliary functions (arithmetics, string processing).
  - Each rule is represented as a Web page template.
What we need to verify

- **A Web site**
  - Since Web pages are provided with a tree-like structure, they can be straightforwardly translated into ordinary terms of a given term algebra $\tau(\text{Text} \cup \text{Tag})$.
  - A *Web site* is a finite collection of ground terms $\{p_1 \ldots p_n\}$.
  - A web page is represented as a tree-like structure.

- **A specification**
  - A Web specification is a triple $(I_N, I_M, R)$,
  - $I_N$ is a finite set of correctness rules.
  - $I_M$ is a finite set of completeness rules.
  - $R$ contains the definition of some auxiliary functions (arithmetics, string processing).
  - Each rule is represented as a Web page template.
What we need to verify

- **A Web site**
  - Since Web pages are provided with a tree-like structure, they can be straightforwardly translated into ordinary terms of a given term algebra $\tau(\text{Text} \cup \text{Tag})$.
  - A *Web site* is a finite collection of ground terms $\{p_1 \ldots p_n\}$.
  - A web page is represented as a tree-like structure.

- **A specification**
  - A Web specification is a triple $(I_N, I_M, R)$,
  - $I_N$ is a finite set of correctness rules.
  - $I_M$ is a finite set of completeness rules.
  - $R$ contains the definition of some auxiliary functions (arithmetics, string processing).
  - Each rule is represented as a Web page template.
Example of a Web page - XML

```xml
<blog>
  <entry>
    <subject>culture</subject>
    <date>02-12-06</date>
    <content>blablabla1</content>
    <comments>1</comments>
  </entry>
  <entry>
    <subject>history</subject>
    <date>01-08-06</date>
    <content>blablabla2</content>
    <comments>0</comments>
  </entry>
</blog>
```
Example of a Web page - Algebra

\[
\text{blog(entry(subject(culture), date(02-12-06),} \\
\text{content(blablabla1), comments(1)),} \\
\text{entry(subject(history), date(01-08-06),} \\
\text{content(blablabla2), comments(0)) )}
\]
Example of a Web site

\[ p1 \] blog(entry(subject(culture), date(02-12-06),
content(blablabla1), comments(1)),
entry(subject(history), date(01-08-06),
content(blablabla2), comments(0)))

\[ p2 \] entry(subject(culture), date(02-12-06),
comments(commentary(date(02-15-06),
author(blink(DanielRomero)),
text(blabla))))

\[ p3 \] entry(subject(history), date(01-08-06), comments())

\[ p4 \] members(member(name(DanielRomero)),
member(name(DemisBallis)) )
Example of a specification

r1) Correctness rule:
blink(X) -> error
   /* It simply states that blinking text
   is forbidden in the whole Web Site. */
blog(X) -> error | X in[:TextTag:]*sex[:TextTag:]*
   /* The word “sex” is not allowed
   in comments. */

r2) Completeness rule:
entry(commentary(author(X)))->#members(member(name(X)))<E>
   /* if somebody enters a comment, then
   s/he must be a member. */
The verification process

Simulation and Partial rewriting

Each specification rule which is represented as a Web page template and is used as a pattern to search inside a Web page. To do this, we use simulation and partial rewriting.
The Web Repairing Methodology

Repairing a Faulty Web site

Given a faulty Web site $W$ and the sets of detected errors, $E_N$ and $E_M$, our goal is to modify the Web site by adding, changing, or removing information in order to produce a Web site that is correct and complete w.r.t. its Web specification.
Kinds of errors

Correcteness errors
- Erroneous information embedded in a Web page.

Completeness errors
- Missing Web pages.
- Universal errors.
- Existential errors.
The primitive repair actions

- **change**\((p, w, t)\) replaces the subterm \(p|_w\) in \(p\) with the term \(t\) and returns the modified Web page.

- **insert**\((p, w, t)\) modifies the term \(p\) by adding the term \(t\) to \(p|_w\) and returns the modified Web page.

- **add**\((p, W)\) adds the Web page \(p\) to the Web site \(W\) and returns the Web page \(p\).

- **delete**\((p, t)\) deletes all the occurrences of the term \(t\) in the Web page \(p\) and returns the modified Web page.
Fixing errors

The procedure is as follows: whenever an error is found, a possible repair action is selected and is executed in order to remove the erroneous information, provided that it does not introduce any new bugs.

- **First step**: Fixing correctness errors
  - “Correctness through Change” strategy
  - “Correctness through Deletion” strategy

- **Second step**: Fixing completeness errors
  - “Completeness through Insertion” strategy
  - “Completeness through Deletion” strategy
Fixing errors

The procedure is as follows: whenever an error is found, a possible repair action is selected and is executed in order to remove the erroneous information, provided that it does not introduce any new bugs.

- **First step**: Fixing correctness errors
  - “Correctness through Change” strategy
  - “Correctness through Deletion” strategy

- **Second step**: Fixing completeness errors
  - “Completeness through Insertion” strategy
  - “Completeness through Deletion” strategy
Fixing errors

The procedure is as follows: whenever an error is found, a possible repair action is selected and is executed in order to remove the erroneous information, provided that it does not introduce any new bugs.

- **First step**: Fixing correctness errors
  - “Correctness through Change” strategy
  - “Correctness through Deletion” strategy

- **Second step**: Fixing completeness errors
  - “Completeness through Insertion” strategy
  - “Completeness through Deletion” strategy
Strategies

Fixing correctness errors

- “Correctness through change” strategy
The “Correctness through Change" strategy

Given a correctness error \( e = (p, w, \varnothing ) \in E_N \), the subterm \( p|_w \) of the Web page \( p \) is replaced with a new term \( t \) that is introduced by the user.

- The new term must not introduce any new error
  - Local correctness property
  - Global correctness property
The “Correctness through Change” strategy

Given a correctness error $e = (p, w, l\sigma) \in E_N$, the subterm $p|w$ of the Web page $p$ is replaced with a new term $t$ that is introduced by the user.

- The new term must not introduce any new error
  - Local correctness property
  - Global correctness property
The “Correctness through Change” strategy

Local correctness property

- \( e = (p, w, l, \sigma) \in E_N \) and the associated repair action \( \text{change}(p, w, t) \).

Conditional rules

\[ CS_e \equiv \{ \neg c \mid \exists (l \rightarrow r \mid c) \in I_N, \text{a position } w', \text{a substitution } \sigma \text{ s.t. } l\sigma \sqsubseteq p|_{w.w'} \} \]

Unconditional rules

\[ \forall l \rightarrow r \in I_N, w \in O_{Tag}(t), \text{substitution } \sigma, l\sigma \not\sqsubseteq t|_{w} \]
The “Correctness through Change” strategy

Global correctness property

- Whenever incorrect data in \( t \) is fixed, \( t \) must also be considered within the context that surrounds it in \( p \).

- The idea behind the Global correctness property is to raise the Local correctness property to the Web page level where the error occurs.
The “Correctness through Change” strategy

The execution of a change action that obeys the global correctness property as well as the local correctness property decreases the number of correctness errors of the original Web site.

\[ | E_N(w') | < | E_N(w) | \text{ where } w' \equiv w \setminus \{p\} \cup \{\text{change}(p, w, t)\}. \]
Strategies

The development for the other strategies is similar

- “Correctness through Change” strategy
- “Correctness through Deletion” strategy
- “Completeness through Insertion” strategy
- “Completeness through Deletion” strategy
The basic methodology presented so far has been partially implemented in the preliminary prototype GVERDI-R (Graphical Verification and Rewriting for Debuging Internet sites), which is written in Haskell (GHC v6.2.2) with the rewriting motor in Maude.

It is publicly available together with a set of examples at http://www.dsic.upv.es/users/elp/GVerdi.
GVerdi-R Screenshot

Specification

```
 blink(X) -> error
"blog(X) -> error : &match X [a-zA-Z]*&
"entry(commentary(author(X))) -> #members(member(name(X)))"
```
GVERDI-R A Tool for Repairing Faulty Web Sites

Introduction
What GVERDI-R is and how it works

The Web Repairing Methodology

Strategies

Prototype

Conclusions and Future work

GVerdi-R Screenshot

Correctness error

GVERDI-R 1.0

GVERDI-R: Verification and Rewriting for Debugging Internet sites

Input Files
Specification filename: /proleExample/specification.spec
Website directory: /proleExample

[Load/Load]

CHECK! Show Specs Show Web Clear Quit Help About

CHECK in MAUDE!

Errors Found

Errors Correctness:

Page: entryCulture.html
Position: [1,3,1,2]
Error: <blink>Daniel Romero</blink>
GVerdi-R Screenshot

Repairing
GVerdi-R A Tool for Repairing Faulty Web Sites

Introduction
What GVERDI-R is and how it works
The Web Repairing Methodology
Strategies
Prototype
Conclusions and Future Work

GVerdi-R Screenshot

Repairing

Maria Alpuente, Demis Ballis, Moreno Falaschi, and Daniel Romero
Conclusions

- This methodology allows us to predict whether a repair action may cause a new error, and assists the user in reformulating the action.
- To resolve the problem associated to the conditions of conditional rules, this methodology helps the user by suggesting ranges of correct values (using CiME).
- Our methodology is smoothly integrated on top of an existing Web Verification Framework, that offers the expressiveness and computational power of functions.