Sink web pages of web application

Doru Anastasiu Popescu¹, Zoltan Szabo²

¹Faculty of Mathematics and Computer Sciences
University of Pitești, ROMANIA
E-mail: dopopan@yahoo.com

²High School “Petru Maior”, Reghin, ROMANIA
E-mail: szabozoliposta@yahoo.com

Abstract
In this paper we define a new notion: sink web page for web application. We define a partial order relation between the web pages of a web application, for this. With the help of this relation, we construct a directed graph associated to a web application. In general, web applications contain many web pages. Their verification and testing is usually performed by using different verification mechanisms. Because many of the components of a web application are in the partial order relation, it follows that the time for verification and testing is considerably reduced if we verify only the sink web pages.

Keywords: Verification, Relation, Web Application, Graph, HTML, Java

Introduction
The starting point in writing this paper is represented by the fact that many web applications use their static web pages’ code, as well as the codes of other web pages. It is obvious that in testing and verifying ([6.1], [6.4] and [6.5]) such an application (which can have a large number of static web pages, of different complexity) it is useful to find a way of reducing the number of static web pages to be tested and verified. In this sense, we can consider that for two web pages p and q in which p’s code is included in q’s code, it is enough to verify only q’s code.

In section 2 of this paper, we will present a relation between the web pages of the web application, together with an example, after which we will continue with the construction of a directed graph (section 3) associated to this relation and a method (section 4) which determines the web pages needed to be tested in order to verify the correctness of the web application.

The relation defined in section 2 is different from that of [6.2]. In [6.2] relation is a relation of equivalence, and this article is a reflexive relationship, transitive but not symmetric. With this relation the number of web pages to be verified and tested shrinks greatly. For web sites tested with program developed based on these notions, number of web pages checked and tested decreased by 60 percent.

In section 5, I described a Java programme, created by me, which uses the concepts from the previous sections.

A relation among the web pages of a web application

Let P={p₁, p₂,…, pₙ} be the set of web pages in an web application and T be a set of unimportant tags. Below, we define the relation R on the set P.

Definition
We say that p R q, when p and q are from P, if:
a) all tags in p which are not in T are in q in the same order.
b) for any tag $<Tg>$ from p and q, which are not in T, if $<\text{Tg}>$ is in q, then $<\text{Tg}>$ is in p.

For this relation, we construct a directed graph like below:

- the vertices of the graph are the indices of the static web pages: $p_1, p_2, \ldots, p_n$.
- the edges of the graph correspond to pairs of pages for which the R relation applies, such that edge $(i,j)$ exists if $p_i R p_j$.

We call this graph DGR (directed graph associated to the relation R).

The DGR can have circuits of length 2, i.e. circuits of the form $(i,j,i)$. In this case, the web pages for $i$ and $j$, meaning $p_i$ and $p_j$, have the properties:

- $p_i R p_j$
- $p_j R p_i$.

Thus, $p_i$ and $p_j$ consist of the same scripts and tags.

Because it suffices to verify only one of pages $p_i$ or $p_j$, we will fuse the vertices $i$ and $j$ in DGR. The graph obtained after fusing all the pairs of nodes with the above property will be named RDGR (reduced directed graph associated to the relation R).

**Example**

Let us consider the following DGR for a web application:

![Figure 1](image1.png)

After fusing the nodes of the DGR, we obtain the RDGR:

![Figure 2](image2.png)

**Observation**

If in a web application there are no web pages involved in the relation, then the DGR and RDGR are the same as the graph formed only of $n$ isolated vertices.

**Definition**

Let a web application $P = \{p_1, p_2, \ldots, p_n\}$ be the set of web pages in an web application. We say that $p_i$ is sink web page if, exterior degree for vertex $i$, from RDGR is 0.
Algorithm for reducing the number of components that need verification

Using the notions presented in section 2, for a web application (with a connected DGR graph) containing web pages \( p_1, p_2, \ldots, p_n \), we give the following algorithm:

**Step 1**
Construct the DGR of the web application.

**Step 2**
Fuse the vertices in DGR to obtain RDGR. Each node \( i, 1 \leq i \leq m \), of this new directed graph will be associated with page \( p_{x[i]} \).

**Step 3**
Calculate the exterior degree of the \( i \)th vertex, in variable \( d[i] \), \( i=1,2,\ldots,n \).

**Step 4**
Find vertices with exterior degree 0 (sink web pages) and store them in array \( t \).

**Step 5**
Test and verify the page \( x[t[i]] \), for \( i=1,2,\ldots,k \).

The correctness of the algorithm

The algorithm determines the static web pages associated to nodes with exterior degree 0. These are used for testing and verifying (sink web pages). Consider that the result of testing and verifying is positive, meaning that the pages corresponding to the vertices with exterior degree equal to 0 are correctly built. Find a page \( p_i \) which corresponds to a vertex \( i \), not isolated, with exterior degree different from 0. From RDGR, it follows that there exists a path from \( i \) to \( j \), meaning that there exists the string:

\[
p_i=q_1, q_2, \ldots, q_h, q_h=p_j, \text{ with } q_1 \text{ R } q_2 \text{ R } q_3, \ldots, q_h \text{ R } q_h.
\]

From the way we defined the R relation, we obtain \( p_i \text{ R } p_j \) and since \( p_j \) is correct, it follows that \( p_i \) is also correct.

**Observations**

1. If the DGR of a web application is not a connected graph, we apply this method for each connected component.

2. Verifying the correctness of the sink web pages can be done with different validations (for example: [6.8], [6.9], [6.10]).

Implementation

Using the results from previous sections, I have realized a Java programme which determines the sink pages of a web application existing in a given folder.

The programme does the following:
- Creates a text file containing the names of the files with the extension .htm or .html, using a depth first search through the folders. These files will be coded using the numbers 1, 2, ..., \( n \), where \( n \) represents the number of these pages.
- Constructs the DGR graph, using a method which verifies the relationship between two pages (the execution time is \( O(n^2 \cdot m) \), where \( m \) is the maximum number of characters of a HTML file).
- Constructs the RDGR graph, from DGR graf.
- Determines the sink web pages and the web pages that these ones solve.

The program created the TEST.TXT file with the following information:

1) Number of files from web application
2) Number of .html and .htm files from web application
3) Web pages for verification (sink pages)
4) Web pages solve for any sink pages.

The following table presents some results:

<table>
<thead>
<tr>
<th>Website</th>
<th>Number of .html and .htm files from web application (N)</th>
<th>Number of sink web pages (K)</th>
<th>Percent sink web pages in web application (P)</th>
<th>Number of web pages solve with sink web pages (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>60</td>
<td>16</td>
<td>26.26</td>
<td>44</td>
</tr>
<tr>
<td>S2</td>
<td>22</td>
<td>20</td>
<td>90.90</td>
<td>2</td>
</tr>
<tr>
<td>S3</td>
<td>20</td>
<td>15</td>
<td>75.00</td>
<td>5</td>
</tr>
<tr>
<td>S4</td>
<td>87</td>
<td>4</td>
<td>4.59</td>
<td>83</td>
</tr>
<tr>
<td>S5</td>
<td>93</td>
<td>7</td>
<td>7.53</td>
<td>84</td>
</tr>
<tr>
<td>SumN=282</td>
<td>SumK=62</td>
<td></td>
<td>Average=40.86</td>
<td>Average=43.6</td>
</tr>
</tbody>
</table>

**Conclusions and future work**

In general, web applications contain many static web pages. Their verification and testing is usually performed by using different verification mechanisms. Because many of the components of a web application are in the R relation, it follows that the time for verification and testing is considerably reduced if we verify only the web pages corresponding to the vertices with exterior degree equal to 0. In the future, I intend to realize a more complex application which uses the results from this article, with algorithms that have an execution time as small as possible.

**References**

**Books:**

**Conference Proceedings:**

**Conference Proceedings:**

**Conference Proceedings:**
Conference Proceedings:

Journal Articles:

Journal Articles:

Internet Sources:
Alpine HTML Doctor: http://www.alpineinternet.com/

Internet Sources:
Validome HTML/XHTML/… http://www.validome.org/

Internet Sources:
W3C Markup Validation Service: http://validator.w3.org