Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 235 (2009) 3–18
www.elsevier.com/locate/entcs

A Visual Technique for Web Pages Comparison

María Alpuente Daniel Romero[1,2]

Technical University of Valencia
Camino de Vera s/n, Apdo 22012,
46071 Valencia, Spain.

Abstract

Despite the exponential WWW growth and the success of the Semantic Web, there is limited support today to handle the information found on the Web. In this scenario, techniques and tools that support effective information retrieval are becoming increasingly important. In this work, we present a technique for recognizing and comparing the visual structural information of Web pages, The technique is based on a classification of the set of html–tags which is guided by the visual effect of each tag in the whole structure of the page. This allows us to translate the web page to a normalized form where groups of html tags are mapped into a common canonical one. A metric to compute the distance between two different pages is also introduced. Then, by means of a compression process we are also able to reduce the complexity of recognizing similar structures as well as the processing time when comparing the differences between two Web pages. Finally, we briefly describe a prototype implementation of our tool along with several examples that demonstrate the feasibility of our approach.

Keywords: Web page comparison, visual structure, Web page compression.

1 Introduction

HTML is designed to visualize structure and information contents in an understandable way to humans. The main problem with the use of HTML, is its mixtures of semantic content, page structure and layout [9].

The WWW is today a huge repository of information. Typical scenarios in which a suitable notion of similarity of Web pages is desirable include: search engines, testing tools, documents wrappers, detection of duplicated Web pages, and Web data mining.

Given a query for retrieving a piece of information from the web, the search for this information typically involves three aspects: textual information within the

[1] This work has been partially supported by the EU (FEDER) and the Spanish MEC TIN2007-68093-C02-02 project, UPV PAID-06-07 project, and Generalitat Valenciana GV06/285. Daniel Romero is also supported by FPI–MEC grant BES–2008–004860.

[2] Email: {alpuente,dromero}@dsic.upv.es

doi:10.1016/j.entcs.2009.03.002

Web page, page structure layout, and the patterns of the query. However, an extra factor that is hardly considered by current tools is whether two different pieces of code can express the same visual sensation.

A quantification of this visual similarity could lead to the querying of a documental database using a documental pattern as the query. The answer to this query can be either the most (visually) similar document of the database or the most similar document from a pre-defined set. This last type of answer can be used as a classification method [5]. Moreover, the importance of visual features of Web pages is increasing from the viewpoint of search engines optimization.

When we look at a Web page, we are not aware of the underlying HTML code, but are only able to distinguish the visual structure given by the groupings, columns, rows and data. This suggests us the idea to define as "visual structure of a Web page" the apparent structure of a Web page that is perceived by a human independently of the source code that produces it.

In this work, we develop a technique for Web pages comparison that considers its visual structure. First, we present a translation for HTML code that highlights the visible structure of a Web page given by some of HTML tags. Then, we formalize two compression transformations for a Web page. The *horizontal compression* packs together those subterms which represent repetitive structures. The *vertical compression* shrinks those chains of tags that does not influence visually the perceived result. After applying the Web page compression, we obtain an *irreducible term* that represents the "essence" of the Web page concerning its visual aspect. Finally, since Web pages are provided with a tree–like structure, we define a quantitative measure of similarity of two pages based on "edit distance" between two trees.

Related work. Although there have been other recent efforts to define new techniques for comparing Web pages [3,8,11,13,17], only few works have addressed the recognition of their visual structural information. Given a query for retrieving a piece of information from the web, the search for this information typically involves three aspects: textual information in the Web page, page structure layout, and the patterns of the query. A study on how the three factors affect to the Web page similarity is presented in [13]. With respect to the page structure, the authors define classes for grouping the tags, then the comparison is based on counting how many elements of each class are in a given page. However, the visual aspects of each element are not considered in the work. In [8], a Web page is transformed into a sequence of tags called "HTML–string". Then a distance measure between two strings determines the similarity degree of two pages. The comparison is defined tag–to–tag without considering a classification of tags according to their visual affect.

In [11] a methodology for visual comparison is proposed. The method segments a page image into several regions based on image processing, and the result is presented as a graph. The similarity is then calculated by a graph matching algorithm. Unlike us, the structure based on the tags is not considered. Finally, a method for

recognizing structures is presented in [3]. The method is based on automatically constructing a suitable wrapper, including the detection of repeated sequences of tags, and detecting repeated patterns of strings.

The work more closely related to ours is [17], which analyzes the structure of pages based on detecting visual similarities of the tags inside of a page. The main difference with our own work lies in how the analysis of the tags is performed. Our method is top–down thus favoring the inspection of outer structure, whereas in [17] the analysis of the tags is performed bottom–up, detecting frequent patterns of visual similarity.

Plan of paper. The paper is organized as follows. Section 2 recalls some standard notions, and introduces Web page descriptions. In Section 3, we present a transformation of HTML code that allows us to get a clear visual structure of a Web page. Section 4 formalizes a compression technique for Web pages. This technique packs together those subterms which represent repetitive structures and shrinks those chains of tags that does not influence visually. In Section 5, we define a measure of similarity between two Web pages based on the *tree edit distance* algorithm [7,18]. Section 6 describes the main features of our prototypical implementation. Section 7 concludes and discusses future work.

2 Preliminaries

In this section, we briefly recall the essential notions and terminology used in this paper. We call a finite set of symbols *alphabet*. Σ denotes a set of *function symbols* (also called *operators*), or *signature*. We consider varyadic signatures as in [4] (i.e., signatures in which symbols do not have a fixed arity).

Terms are viewed as labelled trees in the usual way. $\tau(\Sigma)$ denote the *term algebra* built on Σ. Syntactic equality between objects is represented by $\equiv$. We also need $max(x, y)$ and $\lceil n \rceil$, $max(x, y) =$ if $(x \geq y)$ x, else y; and $\lceil n \rceil$ is the least integer number greater than or equal to n.

2.1 Web page description

In this work, a *Web page* is either an XML [14] or an XHTML [15] document, which we assume to be well-formed, since there are plenty of programs and online services which can be used to validate XHTML/XML syntax and perform links checking (e.g. [10,16]). 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 $\tau(\mathcal{T}ext \cup \mathcal{T}ag)$.

Note that XML/XHTML tag attributes can be considered as common tagged elements, and hence translated in the same way.

(a)

(b)

```
<html><body>
<table border="0">
  <tr><td>
    <font size="5.5"><b>
      My favorite writers
    </b></font>
  </td></tr>
  <tr><td>
    <table border="0">
      <tr><td>
        <img ../> José Hernandez <br />
        <img ../> Miguel de Cervantes <br />
        <img ../> Jorge Luis Borges <br />
        <img ../> Paulo Coelho <br />
      </td></tr>
    </table>
  </td>
 </tr>
</table>
powered by the ELP Group
</body></html>
```

(c)

```
<html>
<body>
<h2>My favorite books</h2>
<ul>
  <li>El Martin Fierro</li>
  <li>Don Quijote de la Mancha </li>
  <li>Ficciones</li>
  <li>El Alquimista</li>
</ul>
</body>
</html>
```

(d)

Fig. 1. Example of similar Web pages

3 Visual structure of Web pages

When we look at a Web page, we are not aware of the underline HTML code, but are only able to distinguish its visual structure. However, two different pieces of HTML code can express the same visual sensation. Let us illustrate this by means of a rather intuitive example.

Example 3.1 Consider the Web pages in Figure 1(a) and Figure 1(b). At a glance, these two pages have similar appearance, however the corresponding HTML code that defines their structure is very different. Figure 1(a) is defined using tables (tag `<table>`), whereas Figure 1(b) uses a list (tag `<ul>`), see Figure 1(c) and Figure 1(d) respectively.

Example 3.1 illustrates that, in order to provide for appropriate comparison among pages, the visual effect of each of the HTML tags should be considered. In the following we present an abstraction of Web pages, which translates each Web page to a canonical representative according to its visual structure.

3.1 Translation

The basic idea for the translation is to infer the visual structure of the page from the HTML tags in it. In Table 1 we present a classification of the HTML tags with respect to the visual effect that they produce [3]. The elements *grp*, *col*, and *row* describe the essential properties of the pictorical structure of a Web page: *grp* corresponds to an element grouping other elements, *col* is an element that divides into columns, and *row* is an element that splits into rows. Finally, the element *text* represents the concrete data contained in the Web page. Note that, in Table 1, each text or tag that does not influence visually the aspect of the page is defined as *text*. The list of tags is not exhaustive, but only taken as "proof of concept" that could be eventually enlarged with new ones.

Visual (class) tag	**HTML tags**
grp	`table,ul,html,body,tbody,div,p`
row	`tr,li,h1,h2,hr`
col	`td`
text	otherwise

Table 1
Classification of HTML tags

Let $\Sigma_{\mathbb{V}} = \{grp, col, row, text\}$ be a signature of abstract (visual) HTML tags, where each of *grp*, *col*, *row*, and *text* can be seen as the "abstraction" of a numbers of different concrete HTML tags according to the classification given in the Table 1. It is straightforward to translate the Web pages into ordinary terms of the term algebra $\tau(\Sigma_{\mathbb{V}})$. This is done by means of the function *trn*.

Definition 3.2 (translate) Let $f(t_1, \ldots, t_n) \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ be a Web page. The translation *trn* function is defined as:

$$trn :: \tau(\mathcal{T}ext \cup \mathcal{T}ag) \to \tau(\Sigma_{\mathbb{V}})$$

$$trn(f(t_1, \ldots, t_n)) = \begin{cases} \alpha(f) & n = 0 \\ \alpha(f)(trn(t_1), \ldots, trn(t_n)) & otherwise \end{cases}$$

where $\alpha :: (\mathcal{T}ext \cup \mathcal{T}ag) \to \Sigma_{\mathbb{V}}$ replace a concrete HTML tag by the corresponding visual (class) tag, according to the classification given in Table 1.

Let us illustrate Definition 3.2 by means of an example.

Example 3.3 Let $page \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ be the term describing the HTML code of Figure 1(d). The translated *page* is shown in Figure 2.

[3] We use a number of most common HTML tags that typically found in Web pages which allow to define its structure.

$$
\begin{aligned}
trn(page) = grp(grp(&row(text),\\
&grp(row(text),\\
&\quad row(text),\\
&\quad row(text),\\
&\quad row(text))\\
&)\\
)&
\end{aligned}
$$

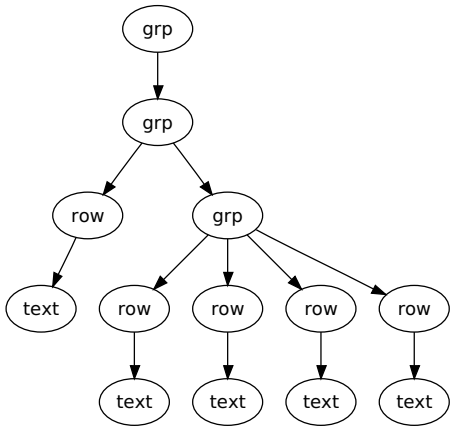

Fig. 2. Example of translation of the Web page of Figure 1(d)

4 Web page compression

The translation given in Section 3.1 allows us identify different HTML tags thus obtaining a clear (and simple to handle) visual structure of the Web page. Moreover, translation brings to light both the repetitive structures occurring into the Web pages and those chains of tags that do not influence visually the visual aspect of the page. In our approach, comparison of Web pages does not depend on the concrete number of child elements of a given classes, e.g. rows, provided some of them exist in the document. In the case when the considered document contains a single column, we simplify the HTML code by "using up" one of the tags.

In this section, we present two compression functions for the term that represents a Web page which dramatically reduce the size of each singular page. Horizontal compression (hrz) packs together those subterms which represent repetitive structures. Vertical compression (vrt) shrinks those chains of tags that does not influence visually the result. First of all, let us introduce the notion of *marked term*, which will be used in the following sections.

4.1 Marked term

In order to avoid losing information after compressing a Web page, we need to record the number of nodes before applying the compression. In the following, we consider terms of the marked term algebra $\tau([\mathbb{N}]\Sigma_{\mathbb{V}})$, where "$[\mathbb{N}]$" represents the number of times that the term t is duplicated "in the marked term $[\mathbb{N}]t$". For instance, consider the term of Figure 3(a). The corresponding marked term is shown in Figure 3(b). The subterm "$[2]row([1]text)$" represents that the term $row([1]text)$ appears twice. Note that, the horizontal compression is not commutative, thus (in Figure 3(a)) the first subterm $row(text)$ cannot be packed together with the last two. When no confusion can arise, we just write $[1]grp([2]row([1]text)) = grp([2]row(text))$.

In the following, we say that two marked terms are equal if the corresponding

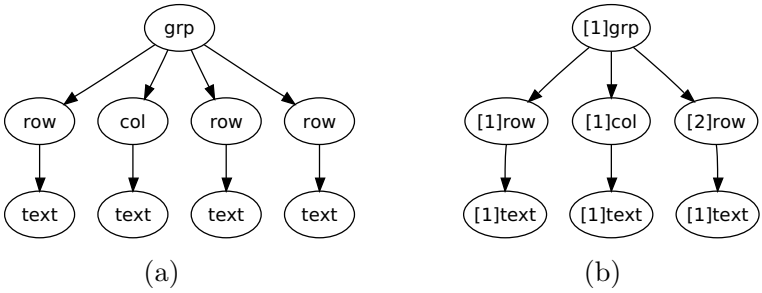

Fig. 3. Example of marked algebra

(unmarked) terms are equal. In symbols

$$[r_1]f \equiv_{\Sigma_{\mathbb{V}}} [r_2]g \quad \text{iff} \quad f \equiv g$$

The equality between two marked trees is define as follows. Let $t = f(t_1, \ldots, t_n)$, $s = g(v_1, \ldots, v_n) \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be two marked trees. Then

$$t \equiv_{\Sigma_{\mathbb{V}}} s \quad \text{iff} \quad f \equiv_{\Sigma_{\mathbb{V}}} g \ \text{ and } \ t_i \equiv_{\Sigma_{\mathbb{V}}} v_i,\ 1 \leq i \leq n$$

The equivalence between function symbols is given by the fact that the tags are abstracted by the same visual tag or not (see Section 3.1). Given function symbols f and g, $f \equiv_{\Sigma_{\mathbb{V}}} g$ iff $\alpha(f) \equiv \alpha(g)$.

4.2 Horizontal compression

In order to achieve a uniform style for Web documents, it is common in practice to repeat some pieces of code. Another common use of repetitive structures appears when we represent lists of elements (for instance, "a books list"). In these circumstances, repetitive structures may typically speed down the analysis of the comparison of Web pages for visual similarity. In the following, we present a compression function for terms that allows us to overcome both drawbacks. Let us give an example.

Example 4.1 Given the term of Figure 4(a), the associated marked term is shown in Figure 4(b).

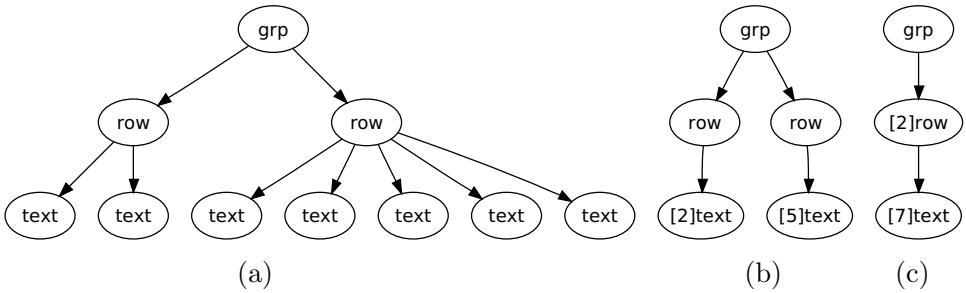

Fig. 4. Naïve term compression

However, a further compression would be possible which is shown in Figure 4(c). In order to pack together some subterms without losing meaningful information about the structure of the page, we could distinguish several alternatives.

(i) Sum number of subterms: $grp([2]row([7]text))$ or $grp([2]row([2]text, [5]text))$.

(ii) Sum only the number of leafs: $grp(row([7]text))$.

(iii) Sum the number of roots and approximate number of the leafs: $grp([2]row([4]text))$.

Let us analyze each of the above options considering the number of nodes that are in the original term (see Figure 4(a)): 1 *grp* node, 2 *row* nodes, and 7 *text* nodes.

(i) This naïve solution does not preserve the original number of nodes. In the example, we would get: 1 *grp* node, 2 *row* nodes, and 14 *text* nodes (7 for each *row*) (see Figure 4(c)).

(ii) In this case, the compression does not respect the visual structure of the page, that originally consisted of two rows.

(iii) This is more conservative compression that keeps both, the overall structure of the page as well as the approximated number of nodes. In the example, we get: 1 *grp* node, 2 *row* nodes, and 8 *text* nodes (4 for each *row*). This can be seen as the more accurate "approximation" of the original number of nodes.

Definition 4.2 (join for terms) Let $t = [r_1]f(t_1, \ldots, t_n)$, $s = [r_2]f(v_1, \ldots, v_n) \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be two terms such that $t \equiv_{\Sigma_{\mathbb{V}}} s$. Then, the *join* between t and s is defined as follows:

$$join :: \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \times \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \to \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$$
$$join(t, s) = \widehat{join}(t, s, 1, 1, 1)$$

where the auxiliary function $\widehat{join}$ is given by

$$\widehat{join} :: \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \times \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$$

$$\widehat{join}(t, s, k_1, k_2, p) = \begin{cases} [m]f & n = 0 \quad (1) \\ [m]f(\widehat{join}(t_1, v_1, r_1, r_2, m), \ldots, & n > 0 \quad (2) \\ \qquad \widehat{join}(t_n, v_n, r_1, r_2, m)\ ) & \end{cases}$$

$$\text{where } m = \lceil (r_1 * k_1 + r_2 * k_2)/p \rceil$$

Note that, in favour of the largest numbers of repetitions, we use the integer division with round up $\lceil \_ \rceil$. Roughly speaking, Definition 4.2 computes the mark, i.e., the number of repetitions of the currently processed node by using both the new and old values of its father. Let us illustrate this by means of an example.

Example 4.3 Consider the terms of Figure 5. Then, the *join* between the terms in Figure 5(a) and Figure 5(b), produces the term of Figure 5(c).

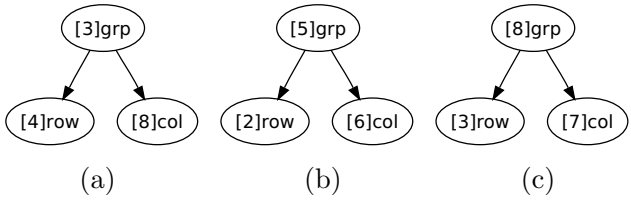

Fig. 5. Join of terms

Definition 4.2 can be generalized to a number $n \geq 2$ of terms in the obvious way. In symbols

$$join(f_1, \ldots, f_n) = join(\ldots(join(f_1, f_2), \ldots), f_n)$$

Now, we are ready to define the horizontal compression function.

Definition 4.4 (horizontal compression) Let $t = f(t_1, \ldots, t_n) \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be a term (Web page). The horizontal compression function hrz on marked terms is defined as:

$$hrz :: \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \to \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$$

$$hrz(t) = \begin{cases} t & n = 0 \quad (1) \\ hrz(f(t_1, \ldots t_{i-1}, s, t_{j+1}, \ldots t_n)) & (\ (1 \leq i \leq j \leq n)\ and \\ \quad \text{where } s = join(t_i, \ldots, t_j) & \quad (t_i \equiv_{\Sigma_{\mathbb{V}}} t_{i+1} \ldots t_{j-1} \equiv_{\Sigma_{\mathbb{V}}} t_j)\ ) \quad (2) \\ f(hrz(t_1), \ldots, hrz(t_n)) & \text{otherwise} \quad (3) \end{cases}$$

In the above definition, the second recursive equation ensures that the general structure of the input term is preserved. That is, the consecutive terms that are equal module $\equiv_{\Sigma_{\mathbb{V}}}$ are packed together. Roughly speaking, Definition 4.4 states that all the arguments that are equal w.r.t. $\equiv_{\Sigma_{\mathbb{V}}}$ and occur at the same level i are packed together. Then, compression recursively proceeds to level $(i + 1)$.

4.3 Vertical compression

XML/XHTML is a markup language for documents containing semi–structured information. In XHTML, all elements must be properly nested within each other, like this

```
<b><i>This text is bold and italic</i></b>
```

Considering the translation given in Section 3.1, this structured information favours the formation of chains of tags that does not influence in the overall structure of the page. In the following, we describe how to shrink the chains of tags while preserving the overall structure.

Let us begin by characterizing the condition we need for the vertical compression. We let $root(t)$ denote the funcion symbol occurring at the top position of t.

Definition 4.5 (safe vertical compression) Let $t = [r]f([r_1]t_1, \ldots, [r_n]t_n) \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be a term and let $\texttt{grp}, \texttt{text} \in \Sigma_{\mathbb{V}}$. Then t obeys the condition of *safe vertical compression* iff the following requirements are fulfilled:

$$
\begin{array}{ll}
r = 1 & (1) \\
n = 1 & (2) \\
\neg(\ f \equiv \texttt{grp} \ \wedge \ root(t_1) \not\equiv \texttt{grp}\ ) & (3)\ \text{(preserve the structure of the page)} \\
root(t_1) \not\equiv \texttt{text} & (4)\ \text{(preserve the information in the page)}
\end{array}
$$

In Definition 4.5, the first condition ensures that no repetitions are disregarded. The second condition ensures that t consists of a chain of tags (only having one child). The third condition states that grouping (`grp`) has a particular higher status than other elements, and thus, should not be compressed. The last condition allows us to keep information of the term. Also note that the repetitions $[r_1] \ldots [r_n]$ of subterms of t are not considered.

Definition 4.6 (shrinking) Let $t = [r]f([m]g(t1, \ldots, t_n)) \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ a term that obeys the *safe vertical condition*. Then, the *shrinking* of t is defined as:

$$
\begin{array}{l}
shr :: \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \to \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \\
shr([r]f([m]g(t1, \ldots, t_n))) = \left\{ \begin{array}{lll}
[r]f(t1, \ldots, t_n) & m = 1 \ \wedge \ g \not\equiv \texttt{grp} & (1) \\
[m]g(t1, \ldots, t_n) & \text{otherwise} & (2)
\end{array} \right.
\end{array}
$$

The idea behind Definition 4.6 is trying to preserve the outer structure of the term whenever possible. Note that, as in the previous definition, grouping (`grp`) has a particular higher status than other elements.

Now, we are ready to formalize the vertical compression transformation.

Definition 4.7 (vertical compression) Let $t = [r]f(t_1, \ldots, t_n) \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be a Web page. Then vertical compression vrt is defined as:

$$
\begin{array}{l}
vrt :: \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \to \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \\
vrt(t) = \left\{ \begin{array}{lll}
t & n = 0 & (1) \\
vrt(shr(t)) & t \text{ obeys the } \textit{safe vertical condition} & (2) \\
[r]f(vrt(t_1), \ldots, vrt(t_n)) & \text{otherwise} & (3)
\end{array} \right.
\end{array}
$$

Roughly speaking, Definition 4.7 moves the inner more influential nodes as higher as possible in the tree. Let us illustrate this definition by means of an example.

Example 4.8 Let $t \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be the term corresponding to Figure 6(a). The vertical compression of t is shown in Figure 6(b).

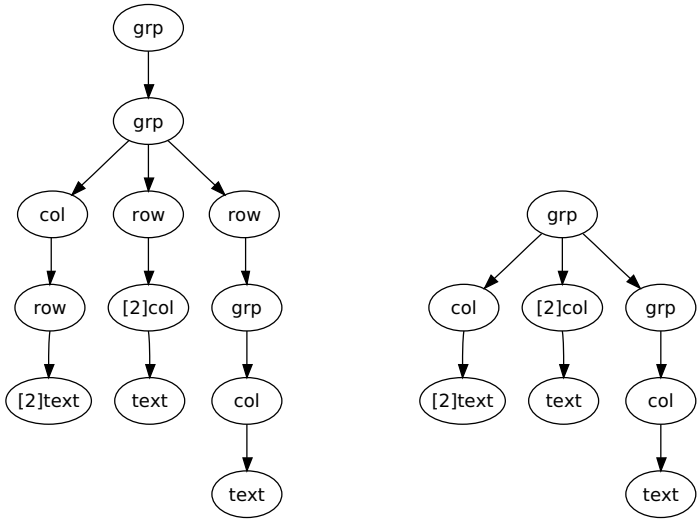

Fig. 6. Vertical compression

4.4 Shrinking and Join

The operators given in Section 4.2 and Section 4.3 allow us to compute the horizontal and vertical compression of a term. This is done by shrinking chains and joining subterms. In order to formalize the overall compression of a term, we define the following operator.

Definition 4.9 (compress) Let $f \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be a Web page. Let $hrz, vrt :: \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \to \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be the two compression functions (horizontal and vertical, respectively) given in Definitions 4.4 and 4.7. Then, the *compress* operator is defined as:

$$compress :: \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \to \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$$
$$compress(f) = hrz(vrt(f))$$

Roughly speaking, we first remove the tags that belong to a chain of tags that doesn't influence the aspect of the resulting page and then join the subterms. Since both the vertical and horizontal transformations are confluent and terminating by repeatedly applying this operation we obtain an irreducible term after an arbitrary number of steps. Given a term f, we say that the resulting term f_{zip} is an *irreducible term* for f.

Example 4.10 Consider again the term t of Example 4.8. Then, we get the *irreducible term* in Figure 7.

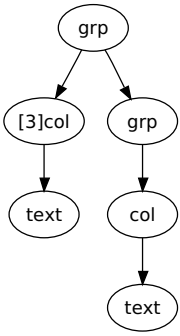

Fig. 7. Irreducible term

The compression technique given in this section generates a representative term for the visual content of the original page as mentioned above. Starting from this term, in the following section we formulate a method to compare Web pages based on their visual structure.

5 Comparison based on visual structure

In our context, the problem of comparing Web pages essentially boils down to comparing trees. In the literature, the most widely used approach for comparing trees consists in computing the "edit distance" between the two trees, i.e., the minimum cost sequence of edit operations (node insertion, node deletion and label change) that transforms one tree into another (see [7,18]). See [1] for a deep discussion on the *tree edit distance problem*.

5.1 Tree edit distance

The *tree edit distance problem* assumes that there exists a *cost function* defined on each *edit operation*. In the following, we follow the approach of [1] and define a *metric cost function* on pairs of nodes.

First to all, let us define the *edit operations*. Let λ denote a fresh constant symbol that represents the *empty* marked term, i.e., the term $[0]t$ for any t. Let $nd_1, nd_2 \in [\mathbb{N}]\Sigma_{\mathbb{V}}$ be two marked trees. Then, each edit operation is represented as:

$$(nd_1 \to nd_2) \in ([\mathbb{N}]\Sigma_{\mathbb{V}} \times [\mathbb{N}]\Sigma_{\mathbb{V}})\backslash(\lambda, \lambda)$$

We say that $(nd_1 \to nd_2)$ is a *relabeling* if $nd_1 \not\equiv \lambda$ and $nd_2 \not\equiv \lambda$, a *deletion* if $nd_2 \equiv \lambda$, and an *insertion* if $nd_1 \equiv \lambda$.

Definition 5.1 (metric cost function) Let $nd_1 = [r_1]f,\ nd_2 = [r_2]g \in [\mathbb{N}]\Sigma_{\mathbb{V}}$ be

two marked terms. Then, a *metric cost function* on an edit operation is defined as:

$$\gamma :: ([\mathbb{N}]\Sigma_{\mathbb{V}} \times [\mathbb{N}]\Sigma_{\mathbb{V}})\backslash(\lambda,\lambda) \to \mathbb{R}$$

$$\gamma(nd_1 \to nd_2) = \begin{cases} 0 & nd_1 \equiv_{\Sigma_{\mathbb{V}}} nd_2 \\ r_2 & nd_1 \equiv_{\Sigma_{\mathbb{V}}} \lambda \quad \text{(insertion)} \\ r_1 & nd_2 \equiv_{\Sigma_{\mathbb{V}}} \lambda \quad \text{(deletion)} \\ max(r_1, r_2) & otherwise \quad \text{(relabeling)} \end{cases}$$

Note that the *metric cost function* assigns identical costs along with insertion, deletion as well as relabeling transformations. Roughly speaking, Definition 5.1 states that the cost of an *edit operation* between two nodes is given by the largest number of repetitions of the nodes. Also note that, γ is a distance metric [4].

The cost of a sequence $S = s_1, \ldots, s_n$ of *edit operations* is given by $\gamma(S) = \sum_{i=1}^{n} \gamma(s_i)$. The *edit distance*, $\delta(t_1, t_2)$, between two trees t_1 and t_2 is formally defined as:

$$\delta(t_1, t_2) = min\{\gamma(S) \mid S \text{ is a sequence of operations transforming } t_1 \text{ into } t_2 \}$$

Since γ is a distance metric, δ is a distance metric too.

5.2 Comparison of Web pages

The *edit distance* δ allows us compute the distance between two trees (Web pages) t_1 and t_2 depending on the *edit operations* that needed for transforming t_1 into t_2. To measure the similarity between two Web pages, we endow the concept of *edit distance* with the number of nodes of the pages. More formally, the Web pages comparison is defined as follows.

Definition 5.2 (Web pages comparison) Let $t, s \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be two Web pages. Let t_{zip} and s_{zip} be two irreducible visual representatives of t and s respectively. The comparison between t and s is formulated as:

$$cmp :: \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \times \tau([\mathbb{N}]\Sigma_{\mathbb{V}}) \to [0..1]$$
$$cmp(t, s) = 1 - \frac{\delta(t_{zip}, s_{zip})}{|t_{zip}| + |s_{zip}|}$$

Definition 5.2 assigns a quantitative measure, a real number between 0 and 1, that expresses the similarity of two pages. Note that, a central question in the tree edit distance algorithm (and therefore in our comparison) is how to choose the cost values of single operations. In this work, we choose the natural (and intuitive) measure that assigns identical cost to insertion and deletion as well as relabeling

[4] A distance metric on a set X is a function $d :: X \times X \to \mathbb{R}$ that satisfies the following conditions: $\forall\ x, y, z \in X.\ d(x, x) = 0$, $d(x, y) \geq 0$, $d(x, y) = d(y, x)$, and $d(x, z) \leq d(x, y) + d(x, z)$.

(see Definition 5.1). A particular study about cost models on edit operations is discussed in [12].

Example 5.3 Consider again the two Web pages of Example 3.1. Let $t, s \in \tau([\mathbb{N}]\Sigma_{\mathbb{V}})$ be the two marked terms corresponding to two different Web pages. The irreducible visual representatives t_{zip} and s_{zip} are shown in Figures 8(a) and 8(b), respectively.

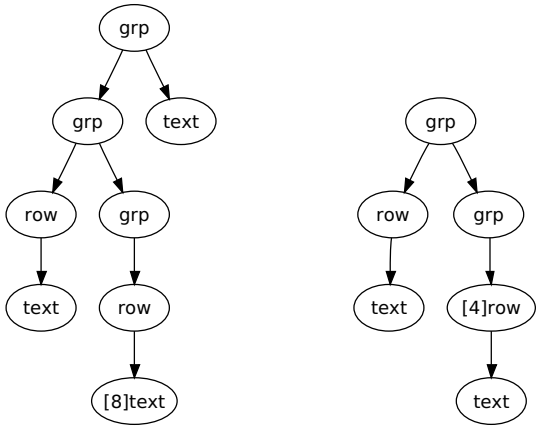

Fig. 8. Visual representatives of two different pages

Consider also the *cost function* given in Definition 5.1. Then

$$|t_{zip}| = 15 \text{ and } |s_{zip}| = 12$$
$$\delta(t_{zip}, s_{zip}) = 2$$
$$cmp(t, s) = 0.92 \sim$$

We say that the similarity between t and s is 92%.

6 Implementation

The comparison technique presented here has been implemented in Maude [2], with an interface developed in Java. The experimental system is publicly available at `http://www.dsic.upv.es/~dromero/cmp.html`. The Maude programming language, which implements the semantic as well as logical framework of rewriting logic, provides a formal analysis infrastructure (such as state-space breadth-first search) with competitive performance (see [6]).

The main features of the our tool are:

- The implementation consists of approximately 560 lines of source code written in Maude. It includes the modules for trees and lists processing extracted from [2].

- The online parser for semistructured expressions (i.e. XML/XHTML documents) is also written in Java. Also a Java class provides a single access point that hides the technical detail to every possible user.
- The tool includes a *config file* to configure the initial settings for some parameters, e.g., location of the Web pages and output folder.

Preliminary experiments with our tool demonstrate that the system works very satisfactorily on several experiments, including all the examples in this paper. We are currently coupling the system with a Web interface, with the aim of making our tool available to every Internet user.

7 Conclusion

HTML was designed to visualize structure and information in an understandable way to humans. The main problem is its mixtures of semantic content, page structure and layout. Web pages comparison is currently an open problem, whose importance extends from search engines to Web data mining.

In this paper, we presented a top–down technique for comparison of Web pages. The key idea behind our method is that two different pieces of code can express the same visual sensation. First, a transformation of Web pages, which translates each Web page according to its visual structure, is defined. By means of two compression functions we obtain a visual representative for the original Web page. Then, we define a measure of similarity between two Web pages based on the *tree edit distance* algorithm. We have developed a prototype implementation that demonstrates the feasibility of our approach. As future work, we plant to extend our analysis by also considering stylesheets.

References

[1] P. Bille. A survey on tree edit distance and related problems. *Theor. Comput. Sci.*, 337(1-3):217–239, 2005.

[2] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer-Verlag, 2007.

[3] W. W. Cohen. Recognizing structure in Web pages using similarity queries. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence*, pages 59–66, Menlo Park, CA, USA, 1999.

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

[5] V. Eglin and S. Bres. Document page similarity based on layout visual saliency: Application to query by example and document classification. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 03)*, page 1208. IEEE Computer Society, 2003.

[6] A. Farzan, F. Chen, J. Meseguer, and G. Rosu. Formal analysis of Java programs in JavaFAN. In *CAV*, pages 501–505, 2004.

[7] P. N. Klein. Computing the Edit–Distance between Unrooted Ordered Trees. In *ESA '98: Proceedings of the 6th Annual European Symposium on Algorithms*, pages 91–102, London, UK, 1998. Springer-Verlag.

[8] G. A. Di Lucca, M. Di Penta, and A. R. Fasolino. An Approach to Identify Duplicated Web Pages. In *COMPSAC '02: Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*, pages 481–486, Washington DC, USA, 2002. IEEE Computer Society.

[9] D. Siegel. The Web Is Ruined and I Ruined It. *World Wide Web Journal*, 2(4):13–21, 1997.

[10] AI Internet Solutions. CSE HTML validator, 2008. Available at http://www.htmlvalidator.com/.

[11] Y. Takama and N. Mitsuhashi. Visual Similarity Comparison for Web Page Retrieval. In *WI '05: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 301–304, Washington, DC, USA, 2005. IEEE Computer Society.

[12] J. Tekl, R. Chbeir, and K. Yetongnon. Semantic and Structure Based XML Similarity: An integrated Approach. In *13th International Conference on Management of Data (COMAD), New Delhi, India*, 2006.

[13] A. Tombros and Z. Ali. Factors Affecting Web Page Similarity. In *Advances in Information Retrieval, 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005*, volume 3408 of *LNCS*, pages 487–501, 2005.

[14] World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.0, second edition, 1999. Available at: http://www.w3.org.

[15] World Wide Web Consortium (W3C). Extensible HyperText Markup Language (XHTML), 2000. Available at: http://www.w3.org.

[16] World Wide Web Consortium (W3C). Markup Validation Service, 2005. Available at: http://validator.w3.org/.

[17] Y. Yang and H. Zhang. HTML Page Analysis Based on Visual Cues. In *ICDAR '01: Proceedings of the Sixth International Conference on Document Analysis and Recognition*, page 859, Washington, DC, USA, 2001. IEEE Computer Society.

[18] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.