# On Berners-Lee computer, hints on navigability semantics and lack of referential integrity*

Joaquim Gabarró
Departament de LSI
Univ. Politècnica de Catalunya
Jordi Girona 1-3, E-08034
Barcelona, Spain
gabarro@lsi.upc.es

Isabel Vallejo
Departament de LSI
Univ. Politècnica de Catalunya
Jordi Girona 1-3, E-08034
Barcelona, Spain

Fatos Xhafa
Departament de LSI
Univ. Politècnica de Catalunya
Jordi Girona 1-3, E-08034
Barcelona, Spain
fatos@lsi.upc.es

## ABSTRACT

In this paper we deal with semantics issues related to the stability, navigability and extensibility of web applications. To this end, we define a *web application* as a deterministic labelled transition system in which states are the `html` pages and transitions are `urls`. This definition allows us, on the one hand, to characterize the temporal evolution of a web application and, on the other, to classify web applications into several types according to how the information is spread over the web application. This last classification captures interesting semantic properties related to the navigability and web application extensibility. We use partial orders to define and characterize web applications extensibility as web program refinements in a way that preserve navigability and *improve* the information reached through browsing.

We apply these ideas to construct a simple web application, namely, a small virtual museum based on approximations of three original paintings. Despite of being very simple, this example evidences the need for the semantic properties while designing the web application in order to assure navigability and obtaining better information while browsing. Based on the extensibility characterization, we were able to extend the virtual museum with different paintings approximations while preserving navigability properties as well as browsing of paintings' approximations of increased quality resolution.

Finally, based on our model for web applications we formally define the web program refinements through partial orders.

## Keywords

Web applications, semantic web, web program refinements, partial orders.

*Work partially supported by Spanish CICYT under grant TIC2002-04498-C05-03 (Tracer), IST program of the EU under contract IST-2001-33116 (Flags) and DELLIS.

## 1. WORLD WIDE WEB AND BERNERS-LEE COMPUTER

The World Wide Web (WWW) has turned out to be an important medium for the distribution and use of information by individuals, organizations, and communities over the world. T. Berners-Lee et al. [1] define the WWW as a "boundless information world" accessible via URLs, HTTP and HTML. WWW provides a reach source of information but can be extremely difficult to navigate, especially due to its continuous growth and the dynamic structure of the information and documents maintained by complex web applications.

Navigability is an important aspect of the web applications. Navigation is usually done selectively by following links having certain "properties". In a sense, the navigation requires by our brains to use associative and selection mechanisms to follow links that lead to relevant information from those that may eventually lead to irrelevant information. Since in practice, finding information or documents of interest is reduced to/depends on browsing the web by following links, the browsing shows its limitations and often produce the syndrome of "lost-in-hyperspace". These limitations has much to do with the lack of referential integrity, as introduced by L. Cardelli et al. [2], who proposed the *Berners-Lee computer* as a new model for Web computing. One of the main characteristics of this new kind of computer is the *lack of sequential integrity* meaning that an URL can be seen as a kind of network pointer, but it does not always point to the same entity, and occasionally it does not point at all. Other aspects closely related to this are the *stability* and the *extensibility* properties of web applications. Certainly, a lot of web applications suffer continuous changes and extensions causing lack of stability, lack of referential integrity, not preserving thus the navigability properties.

In this paper we deal with semantics issues related to the stability, navigability and extensibility of web applications. Our motivation is to formally define a model for web applications that would allow us to relate navigability and extensibility of web applications, for instance, how can we make extensions of web applications in a way that it preserves navigability properties and permits reaching "better" information while browsing. To this end, we define a *web application* as a deterministic labelled transition system in which states are the `html` programs and transitions are `urls`. On

the one hand, this definition allows us to characterize the temporal evolution of a web application in terms of its semantic properties regarding the navigability. On the other hand, using this model we can classify web applications into several types according to how the information is spread over the web application. This last classification captures interesting semantic properties related to the navigability and web application extensibility. We use partial orders to define and characterize web applications extensibility as web program refinements in a way that preserve navigability and *improve* the information reached through browsing.

We apply these ideas to design a simple web application, namely, a small virtual museum based on three original paintings. Since paintings' approximations can be downloded/displayed faster, we consider for each painting a chain of nine approximations in increasing order of resolution. Then, we use these three chains to obtain extensions of our small museum. Despite of being very simple, this example evidences the need for the semantic properties while designing the web application in order to assure navigability and obtaining images of increased quality while browsing. Indeed, different approaches could be followed in order to design the virtual museum each of them showing different navigability properties. Based on the extensibility characterization through partial orders on paintings, we were able to extend the virtual museum with different paintings approximations while preserving navigability properties as well as browsing of increased quality images. Interestingly, this way of designing web applications can be found also in real web applications, as we show by some examples from WWW.

The small virtual museum example gives us hints on web applications extensions. Indeed, the chain of paintings' approximations can be seen as obtained through a refinement process. Thus, our initial small museum was further refined by adding paintings' approximations of increased quality. Motivated by this and using our model for web applications we formally define the web program refinements through partial orders.

The paper is organized as follows. We give in Sect. 2 the definition of our model for web applications as a deterministic labelled transition system and provide characterization of the *semantics* of web applications as well as a classifications of web applications into different types. We exemplify these definitions and the results on different types of web applications through a small virtual museum in Sect. 3. We discuss in Sect. 4 issues related to the lack of referential integrality and introduce the web application refinements. We conclude in Sect. 5 with some remarks and further work.

## 2. WEB APPLICATIONS AND SEMANTICS

In this section we give the basic definitions and notations on our model of web applications. We will consider the temporal evolution of a web application. To do this, first of all we define a web at some given time.

*Definition 1.* (**Transition System of a Web Application**) A web application is a labelled transition system $\texttt{WEB} = (\texttt{url}_0, \texttt{HTML}, \texttt{URL})$ where nodes are `html` pages and labels are `url` links such that:

− There is a finite set of static `html` pages

$$\texttt{HTML} = \{\texttt{html}_0, \texttt{html}_1 \dots\}.$$

The starting page is $\texttt{html}_0$, this is usually a webmap (the `index.html`).
− There is a finite set of `url` links through which we can access and navigate across the application. The link $\texttt{url}_0$ give us access to the application.

$$\texttt{URL} = \{\texttt{url}_0\} \cup \{\texttt{url} \mid \exists \texttt{html} \in \texttt{HTML} \text{ having } \texttt{url} \text{ as hyperlink.}\}$$

We will use the following notation. We note $\texttt{*url}$ the site pointed by `url`, therefore $\texttt{*url}_0 = \texttt{html}_0$. When a link `url` connects the pages $\texttt{html}_i$ and $\texttt{html}_j$, we write

$$(\ \texttt{html}_i,\ \texttt{url},\ \texttt{html}_j\ ) \quad \text{or} \quad \texttt{html}_i \xrightarrow{\texttt{url}} \texttt{html}_j.$$

Observe that since an `url` is a physical address (and not just a label), the transition system is deterministic [4]:

$$\texttt{html} \xrightarrow{\texttt{url}} \texttt{html}_i \text{ and } \texttt{html} \xrightarrow{\texttt{url}} \texttt{html}_j \implies \texttt{html}_i = \texttt{html}_j$$

and `url` points to a unique site:

$$\texttt{html}_i \xrightarrow{\texttt{url}} \texttt{html}_j \text{ and } \texttt{html}_k \xrightarrow{\texttt{url}} \texttt{html}_l \implies \texttt{html}_j = \texttt{html}_l.$$

In our model, several programs can use the same `url`, transitions as $\texttt{html}_i \xrightarrow{\texttt{url}} \texttt{html}$ and $\texttt{html}_j \xrightarrow{\texttt{url}} \texttt{html}$ can be legal. Many applications have direct links to $\texttt{html}_0$ modelled as

$$\forall\, \texttt{html} \in \texttt{HTML} \ :\ \texttt{html} \xrightarrow{\texttt{url}_0} \texttt{html}_0.$$

We can easily model some well known browser capacities. Thus, if it is possible to reload an `html` page we introduce a special link $\texttt{html\_rld}$ such that $(\texttt{html},\ \texttt{html\_rld},\ \texttt{html})$. If, given a transition $(\texttt{html}_i,\ \texttt{url},\ \texttt{html}_j)$, it is possible to go back, we define $\texttt{url}^{-1}$ such that $(\texttt{html}_j,\ \texttt{url}^{-1},\ \texttt{html}_i)$. Further, we identify navigation with the trace of the links. Thus, $\texttt{nvg} = \texttt{url}_1\ \texttt{url}_2 \cdots \texttt{url}_n$ is defined as

$$\texttt{html}_0 \xrightarrow{\texttt{nvg}} \texttt{html}_n =_{def} \exists\, \texttt{html}_1, \dots, \texttt{html}_n \ :\ \texttt{html}_0 \xrightarrow{\texttt{url}_1} \xrightarrow{\texttt{url}_1} \texttt{html}_1 \xrightarrow{\texttt{url}_2} \texttt{html}_2 \cdots \texttt{html}_{n-1} \xrightarrow{\texttt{url}_n} \texttt{html}_n.$$

As a `WEB` transition system is deterministic we write $\texttt{html}_0 \cdot \texttt{nvg} = \texttt{html}_n$ and the set of possible navigations is

$$\texttt{NVG}(\texttt{WEB}) = \{\texttt{nvg} \mid \exists\, \texttt{html} \in \texttt{HTML} \ :\ \texttt{html}_0 \xrightarrow{\texttt{nvg}} \texttt{html}\}.$$

Using the above notations, we can easily define the web pages accessible through a path of length $k$:

$$\texttt{HTML}^k_{\texttt{WEB}} = \{\texttt{html} \in \texttt{HTML} \mid \exists\ \texttt{nvg} \in \texttt{NVG} \ :\ \texttt{html}_0 \xrightarrow{\texttt{nvg}} \texttt{html} \text{ and } \mid \texttt{nvg} \mid = k\}.$$

In general, navigation is an intentional activity, users navigate in order to obtain or increase information about some topic. A `WEB` application, then, should display information in an ordered and easy to find way. We assume that the informational contents of web pages can be modelled as elements of a partial order $(\texttt{INF}, \bot, \sqcup, \sqcap, \sqsubseteq)$ abstracting the

informational content of the topic (see e.g. [3] for partial orders). Let the elements of $\texttt{INF}$ be $\bot, I_0, I_1, \ldots$, then:

$-$ $\bot$ is the minimal bit of information related to the application, in the sense that, it is very general and imprecise.
$-$ $I_i \sqsubset I_j$ means that $I_j$ has more (or better/improved) information than $I_i$.
$-$ $I_i \sqcup I_j$ means that a *good joint organization* of information coming form $I_i$ and $I_j$. For instance, repetitions should be avoided, a *bad explanation* in $I_i$ should be deleted if we dispose of a *better explanation* in $I_j$ and so on.
$-$ $I_j \sqcap I_j$ isolates the common information in $I_i$ and $I_j$.

*Definition 2.* (**Semantic web application**) Let $\texttt{WEB}$ be a web application and assume that the information is displayed as elements of a partial order $(\texttt{INF}, \bot, \sqsubseteq)$. Therefore there is a map giving the informational content of every $\texttt{html}$ page and link $\texttt{url}$:

$$inf : \texttt{HTMLS} \cup \texttt{URLS} \ \rightarrow \ \texttt{INF}.$$

A semantic web is a tuple $\mathcal{WEB} = (\texttt{WEB}, \texttt{INF}, inf)$ such that:

$-$ The information associated to an $\texttt{url}$ is $inf(\texttt{url})$ and the information associated to a web page $\texttt{html}$ is $inf(\texttt{html})$.
$-$ The $\texttt{url}_0$ giving access to the application give us the minimal information $inf(\texttt{url}_0) = \bot$.
$-$ Every $\texttt{url}$ contains some "minimal information", it is a kind of thumbnail signaling the information appearing in $\texttt{*url}$, $inf(\texttt{url}) \sqsubset inf(\texttt{*url})$.
$-$ The information content associated to a web application is defined as:

$$inf(\mathcal{WEB}) = \bigsqcup_{\texttt{nvg} \in \texttt{NVG}} inf(\texttt{html}_0 \cdot \texttt{nvg}).$$

When needed we write extensively as:

$$\mathcal{WEB} = (\texttt{WEB}, \texttt{INF}, inf) =$$
$$(\ (\texttt{url}_0, \texttt{HTML}, \texttt{URL}),\ (\texttt{INF}, \bot, \sqsubseteq),\ inf : \texttt{HTMLS} \cup \texttt{URLS} \ \rightarrow \ \texttt{INF}).$$

Given a web page $\texttt{html}$ having an information content $\texttt{I} \in \texttt{INF}$ (that means $inf(\texttt{html}) = \texttt{I}$) and $k$ hyperlinks $\texttt{url}_1$,...,$\texttt{url}_k$ we represent this web as a record with field selectors as

$$\texttt{html} = [inf = \texttt{I}, link_1 = \texttt{url}_1, \cdots, link_k = \texttt{url}_k]$$

such that $\texttt{html} \cdot inf = \texttt{I}$, $\texttt{html} \cdot link_1 = \texttt{url}_1$ and, so on, $\texttt{html} \cdot link_k = \texttt{url}_k$. When there is no ambiguity we will shortly denote $\texttt{html} = [\texttt{I}, \texttt{url}_1, \cdots, \texttt{url}_k]$.

We proceed now with the classification of web applications into different types according to how the information is spread over the web application and following we provide their main properties.

*Definition 3.* (**Web application types**) Taking into account how the information is spread along the web application, we have several classes of web applications:
$-$ A $\mathcal{WEB}$ is *no loss*, when for every web page $\texttt{html} = [\texttt{I}, \texttt{url}_1, \ldots, \texttt{url}_k]$ of $\texttt{HTML}$ it holds

$$I \ \sqsubseteq \quad inf(\texttt{*url}_1)) \sqcup \cdots \sqcup inf(\texttt{*url}_k) \quad .$$

$-$ We say that $\mathcal{WEB}$ has *perfect recall*, when for every web page $\texttt{html} = [\texttt{I}, \texttt{url}_1, \cdots, \texttt{url}_k]$ of $\texttt{HTML}$ it holds

$$\forall\, i \ : \ 1 \leq i \leq k \ : \ I \sqsubseteq inf(\texttt{*url}_i).$$

$-$ A $\mathcal{WEB}$ has *perfect split of information*, when every web page $\texttt{html} = [\texttt{I}, \texttt{url}_1, \ldots, \texttt{url}_i, \ldots, \texttt{url}_j, \ldots, \texttt{url}_k]$ verifies

$$\forall\, i, j \ : \ i \neq j \ : \ inf(\texttt{*url}_i) \sqcap inf(\texttt{*url}_j) = \bot.$$

$-$ A $\mathcal{WEB}$ application is both *perfect split* and *no loss* when it verifies at the same time the *perfect split* and the *no loss* conditions.

The intuition behind these types of web applications is the following. In a *no loss* $\mathcal{WEB}$ the information is spread along the links but it is not lost; in a *perfect recall* $\mathcal{WEB}$ when choosing a link we keep all the preceding information and, in a *perfect split* $\mathcal{WEB}$ different links from a page do not share any common information. As for the last type, it seems that in real applications one should consider *no loss and perfect split* $\mathcal{WEB}$s.

We give in the following three lemmas the basic properties of a *no loss* $\mathcal{WEB}$ and *perfect recall* $\mathcal{WEB}$ (their proofs are straightforward and we omit them).

*Lemma 1.* In a *no loss* $\mathcal{WEB}$, it holds:

$$\bot = inf(\texttt{HTML}^0_{\texttt{WEB}}) \sqsubseteq inf(\texttt{HTML}^1_{\texttt{WEB}}) \sqsubseteq inf(\texttt{HTML}^2_{\texttt{WEB}}) \sqsubseteq \cdots \sqsubseteq inf(\mathcal{WEB}).$$

*Lemma 2.* In a *perfect recall* $\mathcal{WEB}$, it holds:

$-$ A navigation

$$\texttt{html}_0 \xrightarrow{\texttt{url}_1} \texttt{html}_1 \xrightarrow{\texttt{url}_2} \texttt{html}_2 \cdots \texttt{html}_{n-1} \xrightarrow{\texttt{url}_n} \texttt{html}_n$$

give us the ascending chain

$inf(\texttt{html}_0) \sqsubseteq inf(\texttt{html}_1) \sqsubseteq inf(\texttt{html}_2) \sqsubseteq \cdots \sqsubseteq inf(\texttt{html}_{n-1}) \sqsubseteq inf(\texttt{html}_n)$.
$-$ If a page $\texttt{html}$ maintains a link to $\texttt{html}_0$ then any path going from $\texttt{html}_0$ to $\texttt{html}$

$$\texttt{html}_0 \xrightarrow{\texttt{url}_2} \texttt{html}_1 \cdots \texttt{html}_{n-1} \xrightarrow{\texttt{url}_n} \texttt{html}$$

provides no information because

$$\bot = inf(\texttt{html}_0) = inf(\texttt{html}_1) = \cdots = inf(\texttt{html}_{n-1}) = inf(\texttt{html}).$$

$-$ If there is a cycle through any web page, for instance $\texttt{html}_1$ like

$$\texttt{html}_1 \xrightarrow{\texttt{url}_2} \texttt{html}_2 \cdots \texttt{html}_{n-1} \xrightarrow{\texttt{url}_n} \texttt{html}_1$$

the pages $\texttt{html}_2, \ldots, \texttt{html}_{n-1}$ are redundant because

$$inf(\texttt{html}_1) = inf(\texttt{html}_2) = \cdots = inf(\texttt{html}_{n-1}) = inf(\texttt{html}_1).$$

From this lemma we have that in a perfect recall web, cycles creates redundant information, therefore should be avoided.

*Lemma 3.* For any perfect recall $\mathcal{WEB}$ there is a $\mathcal{TRIM\_WEB\_DAG}$ with no cycles and no redundant information (all ascending chains are strict).

# 3. DESIGN OF A SMALL VIRTUAL MUSEUM

We apply the concepts and ideas of previous section to design a simple web application, namely, a small virtual museum. Our starting point is the selection of three original paintings: *Autumn* of the Albanian painter *Vangjush Mio*, *On the Riviera* of the painter *John Lavery* from Northern Ireland and *Tandem* of the Spanish painter *Ramon Casas*. Since paintings' approximations can be downloded/displayed faster, we constructed for each painting a chain of nine `jpg` approximations[1] of increased quality[2] as partial orders: $\text{Mio}_1$.jpg $\sqsubset$ $\text{Mio}_2$.jpg $\sqsubset \cdots \sqsubset$ $\text{Mio}_9$.jpg for the painting *Autumn*; $\text{Lavery}_1$.jpg $\sqsubset$ $\text{Lavery}_2$.jpg $\sqsubset \cdots \sqsubset$ $\text{Lavery}_9$.jpg for the painting *On the Riviera* and $\text{Casas}_1$.jpg $\sqsubset$ $\text{Casas}_2$.jpg $\sqsubset \cdots \sqsubset$ $\text{Casas}_9$.jpg for the painting *Tandem*.

Having three chains of paintings' approximations, we can consider sets of different quantity and quality information items to be included (displayed) in our web application. So, first of all we have to decide how these items can be *grouped* and *related* in the web application. We consider that the museum can display at most three paintings at a given moment, therefore the possible information items to be displayed (a snapshot of the web page) can be:

− Just one painting having some approximation degree, for instance $\text{Casas}_3$.jpg
− Two paintings of some (same/different) approximation degree, e.g. [$\text{Lavery}_2$.jpg, $\text{Mio}_7$.jpg].
− Three paintings, say, [$\text{Casas}_5$.jpg, $\text{Lavery}_2$.jpg, $\text{Mio}_8$.jpg].

Therefore the possible items of displayable information are:

$$\begin{aligned}&\texttt{PAINTINGS} = \{\text{Casas}_i\text{.jpg} \mid 1 \le i \le 9\} \cup \\ &\cup\{\text{Lavery}_i\text{.jpg} \mid 1 \le i \le 9\} \cup \{\text{Mio}_i\text{.jpg} \mid 1 \le i \le 9\} \\ &\cup\{\ [\text{Casas}_i\text{.jpg}, \text{Lavery}_j\text{.jpg}]\ \mid\ 1 \le i, j \le 9\} \\ &\cup\{\ [\text{Casas}_i\text{.jpg}, \text{Mio}_j\text{.jpg}]\ \mid\ 1 \le i, j \le 9\} \cup \cdots \\ &\cup\{\ [\text{Casas}_i\text{.jpg}, \text{Lavery}_j\text{.jpg}, \text{Mio}_j\text{.jpg}]\ \mid\ 1 \le i, j, k \le 9\}.\end{aligned}$$

The number of items in `PAINTINGS` is quite large. In order to guide the web design, we must decide how to relate the different items of information, concretely, we organize `PAINTINGS` as a partial order. Clearly, given $p, p' \in$ `PAINTINGS` we have to fix when $p \sqsubset p'$. There are several possibilities to fix it but since when we navigate through the web application we should get better (or at least more precise) information we consider only "perfect recall" partial order.

## 3.1 Partial orders on `PAINTINGS`

[1]The approximations can be seen as refinements of images of increased resolution.
[2]See www.lsi.upc.es/∼gabarro/SmallMuseum/

As we consider a perfect recall approach, we cannot loose any part of the information but moreover we have to improve it somehow. Thus, given $p, p' \in$ `PAINTINGS` we can define the desired partial order $p \sqsubset p'$ in two different ways:

(a) display better images: $p \sqsubset_{\texttt{B}} p' =_{def}$. The quality of information given in $p'$ is **better** than in $p$.
(b) display better images or add to the display new paintings. $p \sqsubset_{\texttt{BorA}} p' =_{def}$. The quality of information given in $p'$ is **better** than in $p$ **OR** if information quality is kept unchanged, new paintings should be **added** to $p'$.

*Display better images.* The current display is updated improving the quality of some displayed image and we note the partial order $\texttt{PAINTINGS}_{\texttt{B}} = (\texttt{PAINTINGS}, \sqsubset_{\texttt{B}})$. We have the following cases.
− *One painting.* The painting can be improved, no other information is added, for instance $\text{Casas}_i$.jpg $\sqsubset_{\texttt{B}}$ $\text{Casas}_{i'}$.jpg, iff $i < i'$.
− *Two paintings.* At least one painting is improved. For example

$$[\text{Casas}_i\text{.jpg}, \text{Lavery}_j\text{.jpg}] \sqsubset_{\texttt{B}} [\text{Casas}_{i'}\text{.jpg}, \text{Lavery}_{j'}\text{.jpg}]$$

iff $i \le i'$, $j \le j'$ and at least one inequality is strict.
− *Three paintings* The information about the three paintings is maintained and at least one is improved
$[\text{Casas}_i\text{.jpg}, \text{Lavery}_j\text{.jpg}, \text{Mio}_k\text{.jpg}]$
$\sqsubset_{\texttt{B}}$ $[\text{Casas}_{i'}\text{.jpg}, \text{Lavery}_{j'}\text{.jpg}, \text{Mio}_{k'}\text{.jpg}]$ iff $i \le i'$, $j \le j'$, $k \le k'$ and at least one inequality is strict.

*Display better images or add to the display new paintings.* We can improve information in two different ways, either by increasing the quality of a given image or by adding new paintings not displayed before. We denote the partial order $\texttt{PAINTINGS}_{\texttt{BorA}} = (\texttt{PAINTINGS}, \sqsubset_{\texttt{BorA}})$. As before we have several cases.

− *One painting.* The quality is improved, e.g. $\text{Casas}_i$.jpg $\sqsubset_{\texttt{BorA}}$ $\text{Casas}_{i'}$.jpg iff $i < i'$.
− *One versus two paintings.* The quality of the display can remain unchanged but new information about an other painting need to be added, for instance $\text{Casas}_i$.jpg $\sqsubset_{\texttt{BorA}}$ $[\text{Casas}_{i'}\text{.jpg}, \text{Lavery}_j\text{.jpg}]$ iff $i \le i'$.
− *Two paintings.* At least one painting is improved, for instance $[\text{Casas}_i\text{.jpg}, \text{Lavery}_j\text{.jpg}] \sqsubset_{\texttt{BorA}} [\text{Casas}_{i'}\text{.jpg}, \text{Lavery}_{j'}\text{.jpg}]$ iff $i \le i'$, $j \le j'$ and at least one inequality is strict.
− *Two versus three paintings.* The information about preceding paintings is not worse and new information is added, for instance
$[\text{Casas}_i\text{.jpg}, \text{Lavery}_j\text{.jpg}]$
$\sqsubset_{\texttt{BorA}}$ $[\text{Casas}_{i'}\text{.jpg}, \text{Lavery}_{j'}\text{.jpg}, \text{Mio}_k\text{.jpg}]$ iff $i \le i'$, $j \le j'$.
− *Tree paintings.* The information about the tree paintings is maintained and at least one is improved, for instance
$[\text{Casas}_i\text{.jpg}, \text{Lavery}_j\text{.jpg}, \text{Mio}_k\text{.jpg}] \sqsubset_{\texttt{BorA}}$
$[\text{Casas}_{i'}\text{.jpg}, \text{Lavery}_{j'}\text{.jpg}, \text{Mio}_{k'}\text{.jpg}]$ iff $i \le i'$, $j \le j'$, $k \le k'$ and at least one inequality is strict. Fig. 1 shows this partial order for the case of 2 approximations.

## 3.2 The museum design

To get small examples, for each painting we take just two approximations. We consider the first one as a thumbnail and the second one is a the final image:

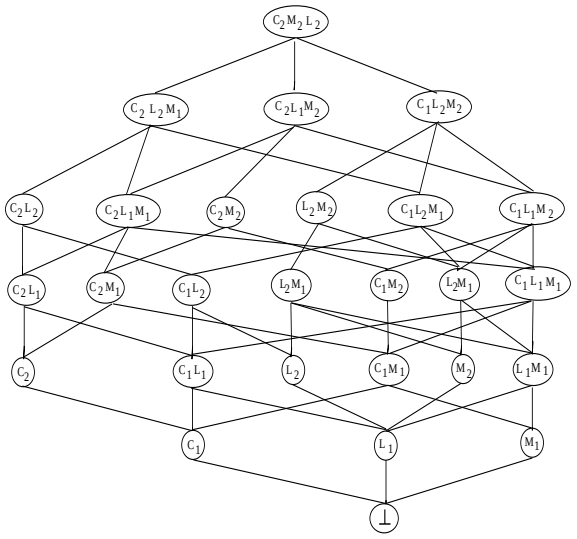

**Figure 1: Hasse diagram of $\texttt{PAINTINGS}_{\texttt{BorA}} = (\texttt{PAINTINGS}, \sqsubset_{\texttt{BorA}})$ in the case of display better images or add to the display new paintings. Two approximations for each painting are considered.**

$\text{Casas}_6\text{.jpg} \sqsubset \text{Casas}_9\text{.jpg}$ $\text{Lavery}_7\text{.jpg} \sqsubset \text{Lavery}_9\text{.jpg}$
$\text{Mio}_8\text{.jpg} \sqsubset \text{Mio}_9\text{.jpg}$

In order to get easier notations we rename the figures as

$$\text{Casas}_6\text{.jpg} = \texttt{C}_1,\ \text{Casas}_9\text{.jpg} = \texttt{C}_2,\ \text{Lavery}_7\text{.jpg} = \texttt{L}_1$$
$$\text{Lavery}_9\text{.jpg} = \texttt{L}_2,\ \text{Mio}_8\text{.jpg} = \texttt{M}_1,\ \text{Mio}_9\text{.jpg} = \texttt{M}_2$$

With the new notation we have $\texttt{C}_1 \sqsubset \texttt{C}_2$, $\texttt{L}_1 \sqsubset \texttt{L}_2$ and $\texttt{M}_1 \sqsubset \texttt{M}_2$. In the following, we index the links and web pages with the information displayed, for instance

$\texttt{html}_{111} = [\,[\texttt{C}_1, \texttt{L}_1, \texttt{M}_1],\ \texttt{url}_{211},\ \ldots\,]$ and $\texttt{*url}_{111} = \texttt{html}_{111}$.

In order to build some small applications (see e.g. Fig. 2) we have to determine the information content of web pages and links:
– The information content of a web page is the displayed image, for instance $inf(\texttt{html}_{111}) = [\texttt{C}_1, \texttt{L}_1, \texttt{M}_1]$.
– The information content of a link will be an image of the `html` page. We adopt the usual design rule: to get better information about one item, click on that item. For instance, in $\texttt{html}_{111}$ the link $\texttt{url}_{211}$ points to $\texttt{html}_{211}$ having a better Casas painting image, therefore we take $inf(\texttt{url}_{211}) = \texttt{C}_1$. This is easily implemented as

```
<A href="html211.html">
    <img width=244 src=Casas6.jpg border=1>
</A>
```

In this way, by just clicking into the image $\texttt{C}_1$ we get a new page containing a better display.

We will take $\texttt{PAINTINGS}_{\texttt{BorA}}$ as a partial order to compare the displayed information. Recall that $p \sqsubset_{\texttt{BorA}} p'$ means that information in $p'$ has better quality than information in $p$ or some new information has been added in $p'$.

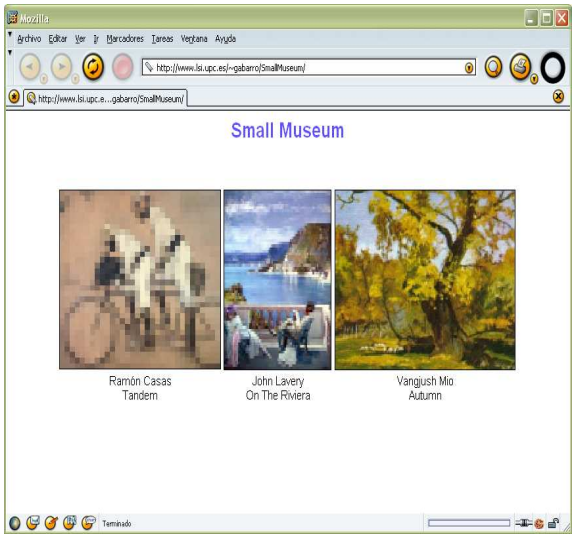


**Figure 2: Small Museum web application.**

To build the applications we develop two different approaches. First we consider perfect recall designs. In this case, there is an initial thumbnail for each painting, some image can be improved, but information about other images is maintained. Second, we consider a no loss design. Again, initially there is a thumbnail image for each painting. When one of them is clicked a better image is given just for that painting.

*Perfect recall design.* First we consider an implementation $\texttt{WEB}_{\texttt{Free}}$ giving us a maximum of freedom in navigation. The initial link is $\texttt{url}_{111}$ and the web pages are

$\texttt{html}_{111} = [\,[\texttt{C}_1, \texttt{L}_1, \texttt{M}_1],\ \texttt{url}_{211},\ \texttt{url}_{121},\ \texttt{url}_{112}]$
$\texttt{html}_{211} = [\,[\texttt{C}_2, \texttt{L}_1, \texttt{M}_1],\ \texttt{url}_{221},\ \texttt{url}_{212}]$
$\texttt{html}_{121} = [\,[\texttt{C}_1, \texttt{L}_2, \texttt{M}_1],\ \texttt{url}_{221},\ \texttt{url}_{122}], \cdots$
$\texttt{html}_{221} = [\,[\texttt{C}_2, \texttt{L}_2, \texttt{M}_1],\ \texttt{url}_{222}]$
$\texttt{html}_{212} = [\,[\texttt{C}_2, \texttt{L}_1, \texttt{M}_2],\ \texttt{url}_{222}],\ \texttt{html}_{122} = \cdots$
$\texttt{html}_{222} = [\,[\texttt{C}_2, \texttt{L}_2, \texttt{M}_2]\,]$.

The web pages can be seen as the vertices of a cube and one navigation step corresponds to moving along one edge in this cube. The semantic web is

$$\mathcal{WEB}_{\texttt{Free}} = (\texttt{WEB}_{\texttt{Free}}, \texttt{PAINTINGS}_{\texttt{BorA}}, inf).$$

This application has perfect recall because, for instance,

$[\texttt{C}_1, \texttt{L}_1, \texttt{M}_1] = inf(\texttt{html}_{111}) \ \sqsubset_{\texttt{BorA}}\ inf(\texttt{*url}_{211}) = [\texttt{C}_2, \texttt{L}_1, \texttt{M}_1]$.

Note that the application displays quite a small content, 3 items, but has a big number of pages $2^3 = 8$. Clearly, freedom is expensive in design. To decrease the number of pages, let us consider a guided tour application $\texttt{WEB}_{\texttt{Guided}}$ where updates follows a fixed order:

$\texttt{html}_{111} = [\,[\texttt{C}_1, \texttt{L}_1, \texttt{M}_1]$
$\texttt{url}_{211}\,],\ \texttt{html}_{211} = [\,[\texttt{C}_2, \texttt{L}_1, \texttt{M}_1]\ \texttt{url}_{221}\,]$
$\texttt{html}_{221} = [\,[\texttt{C}_2, \texttt{L}_2, \texttt{M}_1]$
$\texttt{url}_{222}],\ \texttt{html}_{222} = [\,[\texttt{C}_2, \texttt{L}_2, \texttt{M}_2]\,]$.

As before, $\mathcal{WEB}_{\texttt{Guided}} = (\texttt{WEB}_{\texttt{Guided}}, \texttt{PAINTINGS}_{\texttt{BorA}}, inf)$ has perfect recall. Now the size of the application is smaller, just 4 pages.

*No loss design.* This is a usual web design, first there are thumbnails, clicking on an image we get a better display:

$\texttt{html}_{111} = [\,[\mathrm{C}_1, \mathrm{L}_1, \mathrm{M}_1],\ \texttt{url}_{2**},\ \texttt{url}_{*2*},\ \texttt{url}_{**2}]$
$\texttt{html}_{2**} = [\,\mathrm{C}_2\,],\ \texttt{html}_{*2*} = [\,\mathrm{L}_2\,],\ \texttt{html}_{**2} = [\,\mathrm{M}_2\,].$

The $\mathcal{WEB}_{\texttt{NoLoss}} = (\texttt{WEB}_{\texttt{NoLoss}}, \texttt{PAINTINGS}_{\texttt{BorA}}, inf)$ is no loss because

$[\mathrm{C}_1, \mathrm{L}_1, \mathrm{M}_1] = inf(\texttt{html}_{111}) \sqsubset_{\texttt{BorA}} (\, inf(\texttt{*url}_{2**}) \sqcup inf(\texttt{*url}_{*2*}) \sqcup inf(\texttt{*url}_{**2})\,) = \mathrm{C}_2 \sqcup \mathrm{L}_2 \sqcup \mathrm{M}_2$
$= [\mathrm{C}_2, \mathrm{L}_2, \mathrm{M}_2].$

Note that all the three applications have the same information content

$inf(\mathcal{WEB}_{\texttt{Free}}) = inf(\mathcal{WEB}_{\texttt{Guided}}) = inf(\mathcal{WEB}_{\texttt{NoLoss}})$
$= [\mathrm{C}_2, \mathrm{L}_2, \mathrm{M}_2],$

however, in the first two cases this information is directly accessible by just navigating through a path while in the last case it is spread along the whole web.

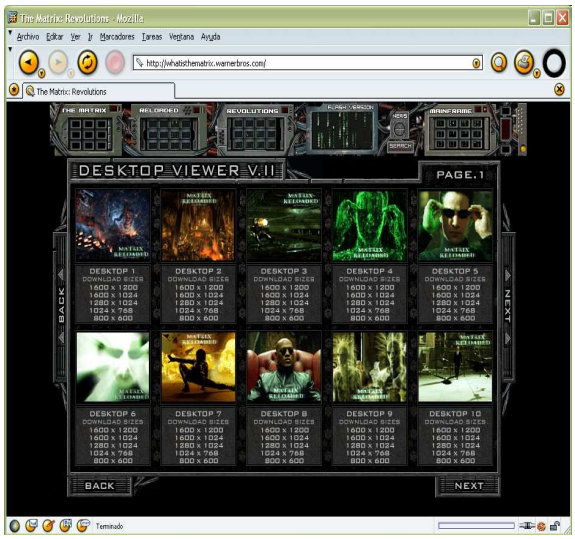


**Figure 3: Desktop viewer from "Reloaded" Warner-Bros**

## 3.3 Real examples of "no loss design" from WWW

Many web applications providing wallpapers maintain a structure quite close to the $\mathcal{WEB}_{\texttt{NoLoss}}$ application given in the preceding example. Consider for instance the page[3] (see Fig. 3) "The Matrix: Revolutions". On this page there is a set of thumbnail images and below every image there is a set of links corresp. to different desktop sizes. The structure of $\mathcal{WEB}_{\texttt{Matrix}}$ has to offer wallpapers based on images and for each image, wallpapers of different sizes are offered:

[3]http://whatisthematrix.warnerbros.com/

$\texttt{IMAGES} = \{\texttt{NEO}_{\perp}, \texttt{NEO}_{1600\times1200}, \ldots, \texttt{NEO}_{900\times600}, \texttt{MORFEO}_{\perp}, \texttt{MORFEO}_{1600\times1200}, \ldots\}$

the web contains pages as:

$\texttt{html}_{\texttt{Matrix}} = [\,[\texttt{NEO}_{\perp}, \texttt{MORFEO}_{\perp}, \ldots],$
$\texttt{url}_{(\texttt{NEO},1600\times1200)}, \ldots\,]$
$\texttt{html}_{(\texttt{NEO},1600\times1200)} = [\,\texttt{NEO}_{1600\times1200}\,].$

The KDE page `http://www.kde-look.org/` has a similar structure.

# 4. REFERENTIAL INTEGRITY AND WEB APPLICATION REFINEMENTS

One of the main characteristics of the *Berners-Lee computer* as a new model for Web computing is the *lack of sequential integrity* meaning that an URL can be seen as a kind of network pointer, but it does not always point to the same entity, and occasionally it does not point at all. However, a close look to the WWW shows that this point of view can be nuanced at least in two aspects:

– There are some `url` addresses that tend to be stable and always points to something, e.g. `http://www.microsoft.com/`, `http://www.sun.com/`. Not only addresses belonging to big industries remain stable but also basic addresses belonging to local institutions are also quite stables, for instance, Government of Catalonia `http://www.gencat.net/`. We guess that this process will continue and in the future more addresses will become stable.
– For stable addresses it is true that they do not point always to the same entity. But, from a theoretical point of view, if in a $\mathcal{WEB}$ application page `html` is updated into $\texttt{html}'$ designers say that the new $\mathcal{WEB}'$ is better than the old one, they could write $\mathcal{WEB} \sqsubset \mathcal{WEB}'$ where $\sqsubset$ means "better".

On the other hand, the web applications evolve in time, not only as regards their addresses but also concerning their information contents. Again, we can observe that for many web applications, for instance those of governmental institutions, big parts of them tend to remain almost stables. In this respect, we observe that:

– Many times changes are mainly "cosmetic", that is, the same information is displayed in a better way. In these cases the "semantic content" remains unchanged. For instance, the initial page of many institutions is a webmap fashioned in different ways.
– It might be argued that the names of hyperlinks change when the page is updated. In fact, it is not always the case since browsers display the `url` when the link is clicked.
– Most of the times changes to information content support the idea of "give more detailed information". For instance one year a Computer Science Department give just a list of courses, the next year, each course has a link giving more details.

## 4.1 Web refinements as extensions

These observations hint us to define a special class of web refinements. Let us consider what does it mean that an application $\mathcal{WEB}$ is updated yielding to another $\mathcal{WEB}'$. We

assume that in one update step a web page

$$\texttt{html} = [\texttt{I}, \texttt{url}_1, \cdots, \texttt{url}_k]$$

can be improved in two complementary ways:

– The displayed information $I$ can be improved.
– New links can be added together with the corresponding new pages. In order to simplify definitions, when a new `url` is added we assume that `*url` has no links. This corresponds to the quite common design rule: add information in small steps. Note that in another update the page `*url` can be updated by adding links.

*Definition 4.* (**Page update partial order**) Given a web page $\texttt{html} = [\texttt{I}_1, \texttt{url}_1, \cdots, \texttt{url}_k]$ belonging to a $\mathcal{WEB}$, an information $I_2$ such that $I_1 \sqsubset I_2$, and new links $\texttt{url}_{k+1}, \ldots, \texttt{url}_{k+l}$ such that

$$inf(\texttt{url}_{k+1}) \sqsubset inf(\texttt{*url}_{k+1}), \ldots, inf(\texttt{url}_{k+l}) \sqsubset inf(\texttt{*url}_{k+l})$$

and such that pages $\texttt{*url}_{k+1}, \cdots, \texttt{*url}_{k+l}$ have no outgoing links, the `html` page can be updated as:

$$\texttt{html}[update : \texttt{I}_2,\ add : \texttt{url}_{k+1}, \cdots,\ add : \texttt{url}_{k+l}]$$
$$= [\texttt{I}_2, \texttt{url}_1, \cdots, \texttt{url}_k, \texttt{url}_{k+1}, \cdots, \texttt{url}_{k+l}] = \texttt{html}'.$$

We denote $\mathcal{WEB}' = \mathcal{WEB}[\ \texttt{html} \leftarrow \texttt{html}'\ ]$ the application where `html` has been replaced by $\texttt{html}'$. Page update induces a partial order $\mathcal{WEB} \sqsubseteq \mathcal{WEB}'$.

*Lemma 4.* Updates on different pages `html` and $\texttt{html}'$ belonging to the same $\mathcal{WEB}$ commute

$$\mathcal{WEB}[\ \texttt{html} \leftarrow \texttt{html}_1\ ][\ \texttt{html}' \leftarrow \texttt{html}_2\ ] =$$
$$\mathcal{WEB}[\ \texttt{html}' \leftarrow \texttt{html}_2\ ][\ \texttt{html} \leftarrow \texttt{html}_1\ ].$$

Updates can also be composed sequentially and we denote

$$\mathcal{WEB}[\ \texttt{html} \leftarrow \texttt{html}'\ ][\ \texttt{html}' \leftarrow \texttt{html}''\ ] =$$
$$\mathcal{WEB}[\ \texttt{html} \leftarrow \texttt{html}'\ ;\ \texttt{html}' \leftarrow \texttt{html}''\ ].$$

*Lemma 5.* Given a web extension $\mathcal{WEB}' = \mathcal{WEB}[\ \texttt{html} \leftarrow \texttt{html}'\ ]$, let $inf'$ and $inf$ be the information functions corresponding to $\mathcal{WEB}'$ and $\mathcal{WEB}$ then

– navigability is extended, $\texttt{NVG(WEB)} \subseteq \texttt{NVG(WEB}')$,
– information is locally improved, for any $\texttt{nvg} \in \texttt{NVG(WEB)}$ it holds

$$inf(\texttt{*url}_0 \cdot \texttt{nvg}) \sqsubseteq inf'(\texttt{*url}'_0 \cdot \texttt{nvg}),$$

– and, global information is also improved, $inf(\mathcal{WEB}) \sqsubseteq inf'(\mathcal{WEB}')$.

*Lemma 6.* Consider two different web dags (no cycles) $\mathcal{WEB\_DAG}$ and $\mathcal{WEB\_DAG}'$ such that $\mathcal{WEB\_DAG}$ is "an older version" of $\mathcal{WEB\_DAG}'$, that means

– both run into a common url space address, we model this as $\texttt{URL} \subseteq \texttt{URL}'$,
– and both give information about the same topic, therefore the partial order $(\texttt{INF}, \bot, \sqsubseteq)$ is the same in both cases.

Therefore:

$$\mathcal{WEB\_DAG} = (\ (\texttt{url}_0, \texttt{HTML}, \texttt{URL}),\ (\texttt{INF}, \bot, \sqsubseteq),\ inf\ )$$
$$\mathcal{WEB\_DAG}' = (\ (\texttt{url}_0, \texttt{HTML}', \texttt{URL}'),\ (\texttt{INF}, \bot, \sqsubseteq),\ inf'\ ).$$

Moreover, assuming that:

– navigability has been preserved, modelled as $\texttt{NVG(WEB\_DAG)} \subseteq \texttt{NVG(WEB\_DAG}')$,
– and information has been locally improved, modelled as $inf(\texttt{*url}_0 \cdot \texttt{nvg}) \sqsubseteq inf'(\texttt{*url}'_0 \cdot \texttt{nvg})$ for any $\texttt{nvg} \in \texttt{NVG(WEB\_DAG)}$

then $\mathcal{WEB\_DAG}$ can be updated into $\mathcal{WEB\_DAG}'$ through a sequence of updates and therefore $\mathcal{WEB\_DAG} \sqsubseteq \mathcal{WEB\_DAG}'$.

Proof. Update from the end of the free. □

*Lemma 7.* Given a web extension $\mathcal{WEB}_2 = \mathcal{WEB}_1[\ \texttt{html} \leftarrow \texttt{html}'\ ]$ such that

$$\texttt{html}' = \texttt{html}[update : \texttt{I}_2,\ add : \texttt{url}_{k+1}, \cdots,\ add : \texttt{url}_{k+l}]$$

it holds

– If $\mathcal{WEB}_1$ has perfect recall and the updated page verifies $I_2 \sqsubseteq inf(\texttt{*url}_i)$ for $1 \leq i \leq k+l$ then $\mathcal{WEB}_2$ has perfect recall.
– If $\mathcal{WEB}_1$ is no loss and $I_2 \sqsubset (\ inf(\texttt{*url}_l) \sqcup \cdots \sqcup inf(\texttt{*url}_{k+l})\ )$ then $\mathcal{WEB}_2$ is no loss.

Note that an "easy" way to maintain perfect recall is by taking updates that maintain current information given in the page and just creates new links with more information

$$\texttt{html}' = \texttt{html}[add : \texttt{url}_{k+1}, \cdots,\ add : \texttt{url}_{k+l}].$$

In this case it is enough to assure $inf(\texttt{html}) \sqsubseteq inf(\texttt{*url}_i)$ for $k+1 \leq i \leq k+l$. No loss information is even easier to maintain, for instance it suffices to assure

$$I_2 \sqsubset (\ inf(\texttt{*url}_{k+1}) \sqcup \cdots \sqcup inf(\texttt{*url}_{k+l})\ ).$$

As we can see in the following lemma we can add links to the home page in a no loss web application.

*Lemma 8.* In a no loss $\mathcal{WEB}$ we can update an $\texttt{html} = [\texttt{I}_1, \texttt{url}_1, \cdots, \texttt{url}_k]$ with no link to the home page $\texttt{html}_0$ by adding such a link, $\texttt{html}' = \texttt{html}[add : \texttt{url}_0]$ therefore $\mathcal{WEB}' \sqsubseteq \mathcal{WEB}[\ \texttt{html} \leftarrow \texttt{html}'\ ] = \mathcal{WEB}'$ and $\mathcal{WEB}'$ is no loss.

Proof. As $inf(\texttt{url}_0) = \bot$ and $\mathcal{WEB}$ is no loss it holds that

$$inf(\texttt{html}') = inf(\texttt{html}) \sqsubseteq (\ inf(\texttt{*url}_1) \sqcup \cdots \sqcup inf(\texttt{*url}_k)\ )$$
$$= (\ inf(\texttt{*url}_1) \sqcup \cdots \sqcup inf(\texttt{*url}_k) \sqcup \bot\ )$$
$$= (\ inf(\texttt{*url}_1) \sqcup \cdots \sqcup inf(\texttt{*url}_k) \sqcup inf(\texttt{*url}_0)\ ).$$

□

## 4.2 Small museums extensions

We have considered in Sect. 3 three small museums webs $\mathcal{WEB}_{\texttt{NoLoss}}, \mathcal{WEB}_{\texttt{Guided}}, \mathcal{WEB}_{\texttt{Free}}$ for which we can prove the following extensions

$$\mathcal{WEB}_{\texttt{NoLoss}} \sqsubset \mathcal{WEB}_{\texttt{Guided}} \sqsubset \mathcal{WEB}_{\texttt{Free}}.$$

Of course, we could add more resolution levels, and for instance we can build a version of $\mathcal{WEB}_{\texttt{Free}}$ with the 9 resolution levels for each painter. In general for any $i, j, k < 9$ web pages will have the form

$\texttt{html}_{ijk} = [\ [\text{Casas}_i, \text{Lavery}_j, \text{Mio}_k],\ \texttt{url}_{i+1,j,k},\ \texttt{url}_{i,j+1,k},\ \texttt{url}_{i,j,k+1}]$.

Let us call this new version $\mathcal{WEB}_{\texttt{Free(9)}}$; the number of pages of this application would be $9^3 = 729$. Clearly, this number of pages is too large. This means that we can design web applications where information arrives in a small increments and such that the resulting web applications have very high connectivity degree.

# 5. CONCLUSIONS AND FURTHER WORK

We have defined a formal model for web applications as a deterministic labelled transition system that allows to study and characterize the semantics of basic properties of web applications such as temporal evolution, navigability and extensibility. We show the usefulness of the model by exemplifying with a simple web application, namely, a small virtual museum evidencing the need for the semantic properties while designing the web application in order to assure navigability and obtaining better information while browsing. Using partial orders we are able to formally define the web program refinements that gives us useful criteria for the extensibility of web applications.

We believe that our model is also useful to formalize other issues related to web applications such as *navigation complexity*. In this context it would be interesting to formalize the algorithmic behavior of browsing of web applications and find algorithms for an *intelligent browsing*.

# 6. REFERENCES


[1] Berners-Lee, T; Cailliau, R; Luotonen, A; Frystyk, H; Secret, A: The World-Wide Web. CACM, Vol 37, No. 8, 1994.

[2] Cardelli, L.; Davis, R.: Service Combinators for Web Computing. Proc. of the First Usenix Conference on Domain Specific Languages, Santa Barbara, 1997.

[3] Davey, B.A.; Priestley, H.A: *Introduction to lattices and order*. Second edition, Cambridge University Press, 2002.

[4] Eilenberg, S: *Automata languages and machines*. Vol A. Academic Press, 1974.