
World Wide Web, 4: 49–77 (2001)

A Conceptual Model and Rule-Based Query Language for HTML

MENGCHI LIU mengchi@scs.carleton.ca
School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, ON, Canada K1S 5B6

TOK WANG LING lingtw@comp.nus.edu.sg
School of Computing, National University of Singapore, Lower Kent Ridge Road, Singapore 119260

Abstract

Most documents available over the Web conform to the HTML specification. Such documents are hierarchically structured in nature. The existing data models for the Web either fail to capture the hierarchically structure within the documents or can only provide a very low level representation of such hierarchical structure. How to represent and query HTML documents at a higher level are important issues. In this paper, we first propose a novel conceptual model for HTML. This conceptual model has only a few simple constructs but is able to represent the complex hierarchical structure within HTML documents at a level that is close to human conceptualization/visualization of the documents. We also describe how to convert HTML documents based on this conceptual model. Using the conceptual model and conversion method, one can capture the essence (i.e., semistructure) of HTML documents in a natural and simple way. Based on this conceptual model, we then present a rule-based language to query HTML documents over the Internet. This language provides a simple but very powerful way to query both intra-document structures and inter-document structures and allows the query results to be restructured. Being rule-based, it naturally supports negation and recursion and therefore is more expressive than SQL-based languages. A logical semantics is also provided.

Keywords: data model for HTML, conceptual modeling, HTML query language, rule-based query language, fixpoint logical semantics

1. Introduction

Most documents available over the Web conform to the HTML specification. They are intended to be human readable through a browser and thus are constructed following some common conventions and often exhibit some hierarchical structure. How to represent such documents at the conceptual level is an important issue.

In the past several years, a number of data models have been developed in the database community in order to retrieve data from the Web, such as UnQL [6], OEM [27], Strudel [9], etc. For a survey, see [12]. These proposals mainly use relational, graph-based or tree-based data models to represent the Web data. They focus on inter-document structures, with little attention to intra-document structures and thus fail to capture the hierarchically structure within HTML documents. For example, none of the existing data models can represent Michael Ley's DBLP bibliography HTML document at `http://www.`

informatik.uni-trier.de/~ley/db shown in Figure 1 in a natural and simple way.

Because of the difficulties with HTML, XML [5] is fast emerging as the dominant standard for data representation and exchange over the Web. It provides natural support for describing the hierarchical structure in Web documents and allows specific markup to be created for specific data.

The World Wide Web Consortium (W3C) has recently recommended the Document Object Model (DOM) as an application programming interface for HTML and XML documents [30]. DOM defines the logical structure of documents and the way a document is accessed and manipulated. It represents documents as a hierarchy of various node objects. With DOM, programmers can access, change, delete, add or build HTML or XML documents, and navigate their structure. Nevertheless, DOM is a very low level data model for HTML/XML documents as it is intended for programmers to write programs to access and manipulate HTML and XML documents, rather than for the user to find the information within the documents.

W3C has also recommended XPath [7] as a language for operating on the abstract, logical structure of an XML document, rather than its surface syntax. It models an XML document as a tree of various nodes at a level higher than DOM, such as element nodes, attribute nodes, text nodes, etc. However, how to describe the abstract, logical structure of an HTML document still remains an open problem.

Another important issue is how to query large HTML documents and perform inference, not only based on their inter-document structures but also on their intra-document structures to obtain useful information.

In the past few years, a number of query languages and systems have been developed in the database community to retrieve data from the Web, such as W3QS [8], WebSQL [23], WebLog [16], UnQL [6], Lorel [2], WebQOL [3], Strudel [9] and Florid [14]. For surveys, see [1,11,24]. Because they are based on the data models that are limited as mentioned above, they fail to support queries over intra-document structures. For example, CIA World Factbook Web site at http://www.odci.gov/cia/publications/factbook/ contains abundant detailed information about each country (and region) in the world in an HTML document, such as its location, geographic coordinates, area, land boundaries, coastline, population, age structure, birth rate, GDP, budget, etc. But there is no simple way to query such information, let alone to infer any useful information.

The purpose of this paper is twofold. We first propose a novel conceptual model for HTML, called HTML-CM, which stands for HTML conceptual model. This conceptual model has only a few simple constructs but is able to represent the complex hierarchical structure within HTML documents at a level that is close to human conceptualization/visualization of the documents. Based on this conceptual model, we then present a rule-based language to query HTML documents over the Internet, called HTML-QL, which stands for HTML query language. This language provides a simple but very powerful way to query both intra-document structures and inter-document structures and allows the query results to be restructured. Being rule-based, it naturally supports negation and recursion and therefore is more expressive than SQL-based languages. A logical semantics for HTML-QL is also provided.

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 2

The rest of the paper is organized as follows. Section 2 proposes the conceptual model HTML-CM and shows how to automatically convert HTML documents into HTML-CM. Section 3 presents the rule-based query language HTML-QL, including its syntax, query examples, and logical semantics. Section 4 briefly describes the implementation of our Web search and inference system that supports HTML-CM and HTML-QL. Section 5 summarizes and points out further research issues. This paper is a major revision of the two conference papers [19,21].

2. HTML-CM: A conceptual model for HTML

In this section, we first introduce HTML-CM, our conceptual model for HTML. Then we discuss how to convert HTML documents into HTML-CM. We assume the existence of two kinds of symbols: a set $\mathcal{U}$ of URLs, and a set $\mathcal{C}$ of constants. Note that $\mathcal{U}$ is a subset of $\mathcal{C}$.

2.1. Conceptual model

Each Web document in HTML is considered structured in HTML-CM. They have a title and a body. Our purpose is to extract conceptual structures within HTML documents. To this end, we ignore the features that are used to enhance the visual aspects of HTML documents, such as fonts, colors, style, etc.

Definition 1. The notion of *objects* is defined recursively as follows:

1. A constant $c \in \mathcal{C}$ is a *lexical* object.
2. Let o be an object and $u \in \mathcal{U}$. Then $o\langle u \rangle$ is a *linking* object, and o is called the *label* and u is called the *anchor* of the linking object.
3. Let a, o be objects. Then $a \Rightarrow o$ is an *attributed* object, and a is called the *attribute* and o is called the *value* of the attributed object.
4. If $o_1, \ldots, o_n$ are objects with $n > 1$, then $\{o_1, \ldots, o_n\}$ is a *list* object.

In an object-relational database, we can have homogeneous tuples and sets. Tuple elements can be accessed using attribute names while set elements are directly accessed. In an HTML document, we may have attributed objects together with other objects.

Thus, it is sometimes impossible to distinguish tuples from sets. In addition, HTML documents conceptually supports lists instead of sets as duplication is allowed and the order of elements exists. Thus, we use list objects for tuples, sets and lists in HTML-CM. The attribute names can be used to access components of list objects that matches them.

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 3

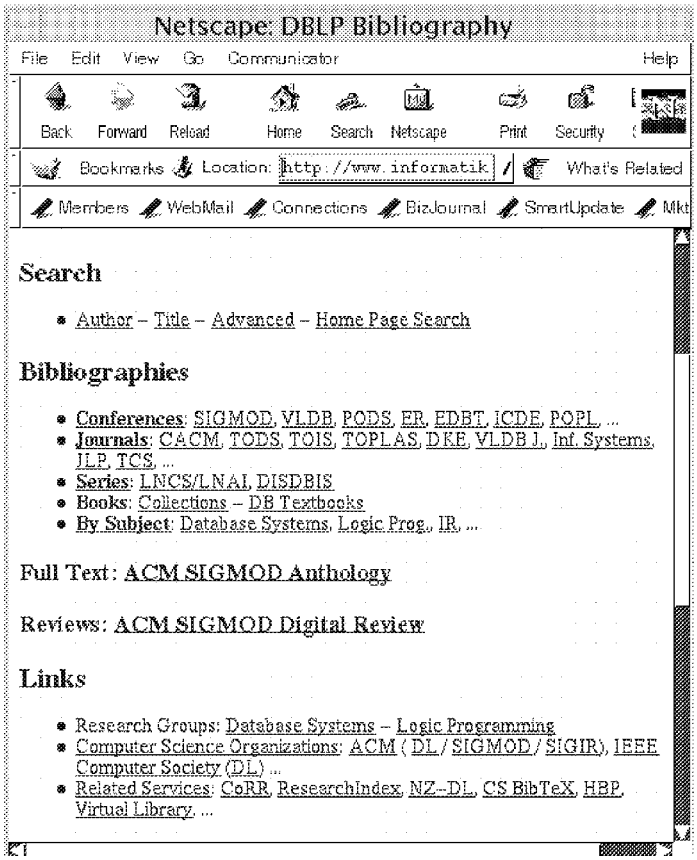

Figure 1. DBLP bibliography.

Example 1. The following are examples of objects:

Lexical objects:	*Computer Science, Database Systems*
Linking objects:	*Faculty*⟨*faculty.html*⟩, *Jim Carter*⟨ */faculty/carter/* ⟩
Attributed objects:	*Title* ⇒ *CS Department*, *Location* ⇒ *Europe*
List object:	{*Title* ⇒ *CS Dept*, *Research*⟨*research.html* ⟩}

A Web document available over the Internet has an associated URL that is used to access the Web document.

Definition 2. Let u be a URL and t an object. Then $u : t$ is a Web object.

The Web documents have many kinds, such as HTML, postscript, pdf, audio, video, etc. In this paper, we mainly focus on HTML documents. We model an HTML document as a list object.

Example 2. The following is an example of Web object that represents an HTML document at a given URL:

csdept.html : {
  Title ⇒*CSDept,*
  People⟨*people.html*⟩⇒ {
    Faculty⟨*faculty.html*⟩,
    Staff⟨*staff.html*⟩,
    Students⟨*students.html* ⟩},
  Programs ⇒ {
    Ph.D Program⟨*phd.html*⟩,
    M.Sc Program⟨*msc.html*⟩,
    B.Sc Program⟨*bsc.html*⟩},
  Research⟨*research.html*⟩
}

Note that a linking object is different from a Web object even though it is somewhat similar to a Web object. A Web object corresponds to a Web document while a linking object is simply part of a Web object that represents an HTML document.

http://www.informatik.uni-trier.de/~ley/db : {
  Title ⇒*DBLP Bibliography*,
  Search ⇒ {*Author*⟨a_1⟩, *Title*⟨a_2⟩, *Advanced*⟨a_3⟩, *Home Page Search*⟨a_4⟩},
  Bibliographies ⇒ {
    Conferences⟨b_1⟩ ⇒{*SIGMOD*⟨b_{11}⟩, *VLDB*⟨b_{12}⟩, *PODS*⟨b_{13}⟩, ... },
    Journals⟨b_2⟩ ⇒{*CACM*⟨b_{21}⟩, *TODS*⟨b_{22}⟩, *TOIS*⟨b_{23}⟩, ... },
    Series⟨b_3⟩ ⇒{*LNCS/LNAI*⟨b_{41}⟩, *DISDBIS*⟨b_{42}⟩},
    Books ⇒{*Collections*⟨b_{51}⟩, *DB Textbook*⟨b_{52}⟩},
    By Subjects⟨b_4⟩ ⇒{*Database Systems*⟨b_{61}⟩, *Logic Prog*⟨b_{62}⟩, *IR*⟨b_{63}⟩}},
  Full Text ⇒*ACM SIGMOD Anthology*⟨c_1⟩,
  Reviews ⇒*ACM SIGMOD Digital Review*⟨c_2⟩,
  Links ⇒{
    Research Groups ⇒{*Database Systems*⟨*d1*⟩, *Logic Programming*⟨*d2*⟩},
    Computer Science Organization⟨e_1⟩ ⇒{
      ACM⟨e_{11}⟩ (*DL*⟨e_{12} ⟩, *SIGMOD*⟨e_{13} ⟩, *SIGIR* ⟨e_{14}⟩),
      IEEE Computer Society⟨e_{15}⟩(*DL*⟨e_{16}⟩)}
    Related Services ⟨f_1⟩ ⇒{
      CoRR⟨f_{11}⟩, *Research Index*⟨f_{12}⟩, *NZ-DL*⟨f_{13}⟩,
      CS BibTex⟨f_{14}⟩, *HBP*⟨f_{15}⟩, *Virtual Library* ⟨f_{16}⟩}}
  ...
}

Figure 2. DBLP Web object with simplified URLs.

Example 3. Consider part of the latest DBLP bibliography server of Michael Ley at `http://www.informatik.uni-trier.de/~ley/db` shown in Figure 1. It can be represented as a Web object in HTML-CM as shown in Figure 2 with simplified URLs such as a_1, b_1, etc. to fit in the paper.

As the above examples show, HTML-CM provides an intuitive representation of the complex hierarchical structure within HTML documents at a high level that is close to human conceptualization/visualization of the documents. None of the other data models can do so in such a simple way.

2.2. *Converting HTML documents*

HTML documents are intended to be human readable though a browser. They are constructed following some common conventions and often exhibit some hierarchical structure. By examining a large number of HTML documents, we have discovered a set of general rules that can be used to convert HTML documents into Web objects in HTML-CM.

In this subsection, we present the conversion rules top-down and recursively using the operator C and give some examples.

As HTML documents are not parsed, they may have syntactic errors. To make our presentation simple, we assume that HTML documents are syntactically correct, while the implementation can be fault-tolerant like most Web browsers. We focus on HTML 4.01 [28], the latest version of HTML.

Our conversion method is mainly based on the contents shown with a Web browser and therefore we ignore the features that are used to enhance the visual aspects of the Web documents, such as fonts, colors, style, etc. We assume that there is an HTML interpreter that can be used to parse the HTML files, remove optional tags and irrelevant components, and automatically identify the corresponding components for our definition below in HTML documents based on various tags, such as <p>, <h*n*>,
, etc.

An HTML document starts with <html> tag and ends with </html> tag. It normally consists of two parts: *head* and *body*. The head mainly specifies the *title* of the document while the body specifies the *contents* of the HTML document. HTML documents can be classified into two kinds based on the kind of the contents: *Frame-based documents* whose body starts with <frameset> and ends with </frameset>, and *regular documents* whose body starts with <body> and ends with </body>.

Frame-based documents. The purpose of frame-based documents is to allow the visitor to see more than one page at a time, without completely cluttering up their screen. Frame-based documents have a number of frames in their body and each frame contains its own HTML document. We convert frame-based document as follows.

Rule 1. Let D_{F} be a frame-based document with frames or nested framesets $F_1, \ldots, F_n$ in its body:

$D_{\mathrm{F}} =$ <html><head><title> T </title></head>
<frameset ...> $F_1, \ldots, F_n$ </frameset> </html>

and $\uplus$ be list concatenation operator. Then D_{F} is converted into a list of two attributed objects using the operator C as follows:

$$C(D_{\mathrm{F}}) = \{\mathit{Title} \Rightarrow T, \mathit{Frameset} \Rightarrow C(F_1) \uplus \cdots \uplus C(F_n)\}.$$

Rule 2. Let $F =$ <frame name = "N" src = "U"> be a frame, where N is the name and U is the URL of the frame. Then $C(F) = \{N\langle U\rangle\}$.

Rule 3. Let $Fs =$ <frameset ...>$F_1, \ldots, F_n$</frameset> be a frame set, where each F_i for $1 \leqslant i \leqslant n$ is a frame or a nested frameset. Then $C(Fs) = C(F_1) \uplus \cdots \uplus C(F_n)$.

In other words, we convert a possibly nested frameset into a list object and each frame in a frameset into a linking object in the list object.

Regular documents. Regular documents are the main ones that provide various information over the Internet. We convert them into list objects as follows.

Rule 4. Let D_{R} be a regular document with sections $S_1, \ldots, S_n$ in its body:

$D_{\mathrm{R}} =$<html><head><title> T </title></head>
<body> $S_1 \ldots S_n$ </body> </html>

Then $C(D_{\mathrm{R}}) = \{\mathit{Title} \Rightarrow T, C(S_1), \ldots, C(S_n)\}$.

Sections. In HTML documents, there can be two kinds of sections: sections with a heading and sections without a heading. If a section has a heading, then we can convert it into an attributed objects with the heading as its attribute and the rest as its value. If it has no heading, we just convert its contents.

Rule 5. Let $S =$ <hn> H </hn> T be a section with a heading H and contents T and $S' = T'$ a section without a heading. Then $C(S) = C(H) \Rightarrow C(T)$, $C(S') = C(T')$.

Example 4. Consider the following HTML document:

```
<html>
<head><title>Computer Science Department</title></head>
<body>
<h2>History</h2>
The department was founded in 1970.
<h2>Programs</h2>
The department offers B.Sc. M.Sc. and Ph.D degrees
<h2>Facilities</h2>
The department has up to date equipment and software
</body>
</html>
```

This HTML document has 3 sections with headings. Using rules 4 and 5, we can obtain the following list object:

{*Title* ⇒*Computer Science Department,*
History ⇒*The department was founded in 1970*,
Programs ⇒*The department offers B.Sc., M.Sc. and Ph.D. degrees*,
Facilities ⇒*The department has up to date equipment and software*}

Within a section, there may be a sequence of paragraphs, lists, tables, etc. In what follows, we discuss how to convert them using the operator C.

Paragraphs. A paragraph in a section immediately follows the tag <p>. Some paragraphs may have an emphasized beginning that is usually bold, italic, etc. or is followed by a colon ':'. We convert them into attributed objects as follows.

Rule 6. Let $P = <t>B</t> : R$ or $P = <t>B : </t>R$ be a paragraph with an emphasized beginning and $P' = R$ a paragraph without an emphasized beginning, where t is either b, i, em, or strong. Then $C(P) = C(B) \Rightarrow C(R)$, and $C(P') = C(R)$. If R has logical parts $R_1, \ldots, R_n$ with $n \geqslant 1$, then

$$\begin{aligned} C(R) &= \{C(R_1), \ldots, C(R_n)\} && \text{if } n > 1, \\ C(R) &= C(R_1) && \text{if } n = 1. \end{aligned}$$

Each logical part R_i for $1 \leqslant i \leqslant n$ is converted as a paragraph recursively.

Example 5. The following is a section of CIA World Factbook page about Canada at http://www.odci.gov/cia/publications/factbook/. It consists of a sequence of paragraphs.

```
<p>
<b>Location:</b> Northern North America
<p>
<b>Geographic coordinates:</b> 60 00 N, 95 00 W
<p>
<b>Map references:</b> North America
<p>
<b>Area:</b>
<br><i>total:</i> 9,976,140 sq km
<br><i>land:</i> 9,220,970 sq km
<p>
<b>Land use:</b>
<br><i>arable land:</i> 5%
<br><i>permanent crops:</i> 0%
<br><i>permanent pastures:</i> 3%
<br><i>forests and woodland:</i> 54%
<br><i>other:</i> 38% (1993 est.)
```

Using rule 6, we obtain the following list object:

{*Location* ⇒*Northern North America,*
Geographic coordinates ⇒*60 00 N, 95 00 W*
Map references ⇒*North America,*
Area ⇒ {
total ⇒*9,976,140 sq km,*
land ⇒*9,220,970 sq km*}
Land use ⇒ {
arable land ⇒*5%,*
permanent crops ⇒*0%,*
permanent pastures ⇒*3%,*
forests and woodland ⇒*54%,*
other ⇒*38% (1993 est.)*}}

Multimedia features and hypertexts. HTML documents can contain multimedia features such as images, applets, video clips, sound clips, etc. In HTML-CM, we convert such features into attributed objects using keywords *image, applet, video, sound*, etc. as their attributes.

In HTML, image files are referenced using image links. At the conceptual level, we only care about the information related to the images in the HTML document.

Rule 7. Let $I =$ `<img src = "`U`" alt="`T`">` be an image link, where U is a URL and T is a string. Then $C(I) = \mathit{Image} \Rightarrow T\langle U\rangle$. When the `alt` field is missing, we treat it as `alt=""`.

The cases for other multimedia features are handled in the similar way.

In HTML, hypertext links are used to link to other documents. We convert such links into linking objects as follows.

Rule 8. Let $H =$ `<a href = "U">` T `</a>` be a hypertext link, where U is a URL and T is a string. Then $C(H) = C(T)\langle U\rangle$.

Rule 9. Let S be a character string. Then $C(S) = S$.

Example 6. Consider the following paragraph without emphasized heading:

```
<a href="dick.html">Dick</a> likes
<a href="jane.html">Jane</a>
```

It has three logical units and thus can be converted into the following object:

{*Dick*⟨*dick.html*⟩, *likes, Jane*⟨*jane.html*⟩}

Lists. HTML supports three kinds of lists: ordered lists, unordered lists and definition lists. One can also nest one kind of list inside another.

Rule 10. Let $L =$ <t>
$L_1 \ldots$ L_n</t> be an ordered or unordered list, where t is either `ol` or `ul`. Then

$$C(L) = \{C(L_1), \ldots, C(L_n)\} \quad \text{when } n > 1,$$
$$C(L) = C(L_1) \quad \text{when } n = 1.$$

Rule 11. Let $L_D =$ <dl><dt>N_1<dd>$L_1 \ldots$ <dt>N_m<dd>L_m</dl> be a definition list. Then

$$C(L_D) = \{C(N_1) \Rightarrow C(L_1), \ldots, C(N_m) \Rightarrow C(L_m)\} \quad \text{when } n > 1,$$
$$C(L_D) = C(N_1) \Rightarrow C(L_1) \quad \text{when } n = 1.$$

Rule 12. Let $I = T$<t>
$T_1 \ldots$ T_m</t> be a nested ordered or unordered list item where t is either `ol` or `ul`. Then

$$C(I) = C(T) \Rightarrow \{C(T_1), \ldots, C(T_m)\}.$$

Example 7. Consider the following HTML document containing nested lists:

```
<html>
<head><title>CS Department Research</title></head>
<body>
<h2>Research Areas</a></h2>
<ol>
<li>Artificial Intelligence
<ul>
<li>Cognitive Science <li>Linguistics <li>Reasoning
</ul>
<li>Database Systems
<ul>
<li>Query Processing  <li>Data Models <li>Active DB
</ul></ol>
<h2>Research Groups and Labs</a></h2>
<ol>
<li>Programming Languages
<li>Intelligent Systems
<li>Natural Language Lab
</ol>
</body>
</html>
```

Using rules 4, 5, 10, and 12, we obtain the following list object:

{*Title* ⇒*CS Department Research,*
Research Areas ⇒{
Artificial Intelligence ⇒ {
Cognitive Science,
Linguistics,
Reasoning},
Database Systems ⇒ {
Query Processing,
Data Models,
Active DB}},
Research Groups and Labs ⇒{
Programming Languages,
Intelligent Systems,
Natural Language Lab}}}

Example 8. Consider the following HTML document containing a definition list:

```
<html>
<head><title>Graduate Studies in CS</title></head>
<body>
<h2></h2>
<dl>
<dt>General Information
<dd>The department was established in 1970
<dt>Programs of Study
<dd>It offers M.Sc, and Ph.D in Computer Science
<dt>Financial Support
<dd>A variety of scholarships are available
<dt>Facilities
<dd>The research labs have all kinds of state-of-the-art
    equipment
</dl>
</body>
</html>
```

Using rules 4 and 11, we obtain the following list object:

{*Title* ⇒*Graduate Studies in CS,*
General Information ⇒*The department was established in 1970*,
Programs of Study ⇒*It offers M.Sc. and Ph.D. in Computer Science*,
Financial Support ⇒*A variety of scholarships are available*,
Facilities ⇒*The research labs have all kinds of state-of-the-art equipment*}}

Tables. Tables in HTML are used to arrange data into rows and columns of cells. Tables may have caption and column/row headings. Tables can be used in two different ways: to present a list with a better layout or to present real tabular information. We convert tables as follows.

Rule 13. Let $T =$ <table><caption>H</caption>TC</table> be a table with a caption and $T' =$ <table>TC</table> be a table without a caption, where H is the caption and TC is the table contents. Then

$$C(T) = C(H) \Rightarrow C(TC),$$
$$C(T') = C(TC).$$

Let $R_1, \ldots, R_n$ be rows other than the row for column headings in the table contents TC. Then $C(TC) = \{C(R_1), \ldots, C(R_n)\}$. For each row R_i with $1 \leqslant i \leqslant n$:

1. If the table has column headings $H_1, \ldots, H_n$ and each row R_i has a row heading H: $R_i =$ <tr><th>H<td>$C_1 \ldots$ <td>C_n, then

   $$C(R_i) = C(H) \Rightarrow \{C(H_1) \Rightarrow C(C_1), \ldots, C(H_n) \Rightarrow C(C_n)\}.$$

2. If the table has column headings $H_1, \ldots, H_n$, but each row R_i has no row heading: $R_i =$ <tr><td>$C_1 \ldots$ <td>C_n, then

   $$C(R_i) = \{C(H_1) \Rightarrow C(C_1), \ldots, C(H_n) \Rightarrow C(C_n)\}.$$

3. If the table does not have column headings but each row R_i has a row heading: $R_i =$ <tr><th>H<td>$C_1 \ldots$ <td>C_n, then

   $$C(R) = C(H) \Rightarrow \{C(C_1), \ldots, C(C_n)\}.$$

4. If the table has neither column nor row headings, then for each row $R_i =$ <td>$C_1 \ldots$ <td>C_n,

   $$C(R) = \{C(C_1), \ldots, C(C_n)\}.$$

Example 9. Consider the following table in an HTML document:

```
<table>
<caption align = top>Bear Sightings</caption>
<tr>
<td><br><th>Babies<th>Adults<th>Total
<tr><th>Northampton<td>2<td>4<td>6
<tr><th>Becket<td>5<td>22<td>27
<tr><th>Worthington<td>7<td>5<td>12
</table>
```

Using rule 13 case 1, we can obtain the following attributed object:

Bear Sightings $\Rightarrow$ {
Northampton $\Rightarrow$ {*Babies* $\Rightarrow$ *2, Adults* $\Rightarrow$ *4, Total* $\Rightarrow$ *6* },
Becket $\Rightarrow$ {*Babies* $\Rightarrow$ *5, Adults* $\Rightarrow$ *22,Total* $\Rightarrow$ *27*},
Worthington $\Rightarrow$ {*Babies* $\Rightarrow$ *7, Adults* $\Rightarrow$ *5, Total* $\Rightarrow$ *12*}}

In HTML documents, tables without column headings and row headings are often used to arrange items for visual effects.

Example 10. Consider the following portion of an HTML document:

```
<h2 align="center"> Faculty Profiles</h2>
<table border="0" cellpadding=3 cellspacing=3 align=center>
<tr>
<td colspan=50% align=left>
<a href="/faculty/bunt/">Rick Bunt</A>
<td colspan=50% align=left>
<a href="/faculty/carter/">Jim Carter</A>
<tr>
<td colspan=50% align=left>
<a href="/faculty/cheston/">Grant Cheston</A>
<td colspan=50% align=left>
<a href="/faculty/cooke/">John Cooke</A>
...
</table>
```

Using rule 13 case 4 and rule 8, we can obtain the following attributed object:

Faculty Profiles $\Rightarrow$ {
Rick Bunt⟨*/faculty/bunt/*⟩,
Jim Carter⟨*/faculty/carter/*⟩,
Grant Cheston⟨*/faculty/cheston/*⟩,
John Cooke⟨*/faculty/cooke/*⟩,
...
}

Forms. HTML forms enable visitors to communicate with the Web server. There are two basic parts of a form: the structure that consists of text fields, text areas, buttons, and menus that the visitor sees and fills out on a page, and the processing part that process the information the visitor fills out using CGI script typically written in Perl or some other programming language.

Conceptually, we are interested in the structure part; that is, what kind of information is presented and can be communicated with the Web server. Each element on a form has a name and a value/type associated with it. The name identifies the data that is being sent, the value is the data that is built-in the HTML document, the type specifies the kind of value that comes from the visitor.

HTML supports two ways to send information to the Web server: *GET* and *POST*. The GET method appends the name–value pairs to the end of the URL and is for small amount of data. The POST method sends a data file with the name–value pairs to the server's standard input without size limit.

Using either GET or POST is mainly a physical level concern and does not have much conceptual value.

A form normally has two special buttons: *submit* and *reset*. The *submit* button is used to send the information and the *reset* button is used to reset the form.

Rule 14. Let F be a form as follows where $E_1, \ldots, E_n$ are the elements in the form.

$F =$ <form ...>
$E_1, \ldots, E_n$
<input type = "submit" ...>
<input type = "reset" ...>
</form>

Then $C(F) = \mathit{Form} \Rightarrow \{C(E_1), \ldots, C(E_n)\}$.

Note that we don't keep information about the *submit* and *reset* buttons in the result as we make *Form* a reserved attribute in HTML-CM to indicate that its value are used for communication with the Web server and those two buttons are implied.

We now discuss how to convert elements on forms. First, a text field is a one line area that allow the user to input text.

Rule 15. Let $T_F = L$:<input type="T" name="N" value="V"> be a text field, where L is the label visible to the user, T is the type for the input text, N is the name for the text field that is not visible to the user, and V is the default value. Then $C(T_F) = \mathit{TextField} \Rightarrow \{\mathit{Label} \Rightarrow L, \mathit{Name} \Rightarrow N, \mathit{Type} \Rightarrow T, \mathit{Value} \Rightarrow V\}$.

Text areas in HTML are text fields that can span several lines. They are converted in a similar way.

Radio buttons on forms are used to let the user make only one choice from a set of alternatives.

Rule 16. Let R be a group of radio buttons with the same name of the following form, where L is the label for the group, N is the name, V_i is the value, and L_i is the label visible to the user.

$R = L$:<input type = "radio" name = "N_1" value = "V_1">L_1
...
<input type = "radio" name = "N_m" value = "V_m">L_m

Then

$$C(R) = \mathit{Radio\ Buttons} \Rightarrow \{\mathit{Label} \Rightarrow L, \mathit{Name} \Rightarrow N, \mathit{Options} \Rightarrow \{\{\mathit{Label} \Rightarrow L_1, \mathit{Value} \Rightarrow V_1\}, \ldots \{\mathit{Label} \Rightarrow L_n, \mathit{Value} \Rightarrow V_n\}\}.$$

While radio buttons can accept only one answer per set, check boxes allow the visitor to check as many check boxes as they like. It is converted in a similar way.

Menus allow visitors of HTML document to enter information easily. It serves the same purpose as radio buttons or check boxes but takes up less space.

Rule 17. Let M be a menu of the following form, where L is the label for the menu visible to the user, N is the name for the menu, V_i is the value, and L_i is the label visible to the user.

$M = L$ <select name = "N" ...>
<option value = "V_1">L_1
...
<option value = "V_n">L_n
</select>

Then

$$C(M) = Menu \Rightarrow \{Label \Rightarrow L, Name \Rightarrow N \; Options \Rightarrow \{\{Label \Rightarrow L_1, Value \Rightarrow V_1\}, \ldots \{Label \Rightarrow L_n, Value \Rightarrow V_n\}\}.$$

Example 11. Consider the following portion of an HTML document:

```
<form method=post action="http://site.com/cgi-bin/get_menu">
  <br>
<p>Username:<input type="Text" name="Name"
  value="Enter a name here">
<p><strong>Age Category</string><br>
<select name = "Category">
<option value = "teenager">13-19
<option value = "adult">20-60
<option value = "senior">over 60
</select>
<p>Sex:
<input type = "radio" name = "Gender" value = "female">Female
<input type = "radio" name = "Gender" value = "male">Male <br>
<input type = "submit" value = "Send info">
<input type = "reset" value = "Start over">
</form>
```

It is converted into the following object in HTML-CM:

Form ⇒{
  Text Field ⇒{
    Label ⇒*Username*,
    Name ⇒*Name*,
    Type ⇒*Text*,
    Value ⇒*Enter a name here*},
  Menu ⇒{
    Label ⇒*Age Category*,
    Name ⇒*Category*,
    Options ⇒{

$\{Label \Rightarrow 13\text{–}19, \ Value \Rightarrow teenager\},$
$\{Label \Rightarrow 20\text{–}60, \ Value \Rightarrow adult\},$
$\{Label \Rightarrow over\ 60, Value \Rightarrow senior\}\}$
$Radio\ Button \Rightarrow \{$
$Label \Rightarrow Sex,$
$Name \Rightarrow Gender,$
$Options \Rightarrow \{$
$\{Label \Rightarrow Female, Value \Rightarrow female\},$
$\{Label \Rightarrow Male, \ Value \Rightarrow male\}\}\}$

3. HTML-QL: A rule-based query language for HTML

In this section, we present a rule-based language called HTML-QL to query Web objects, based on the conceptual model HTML-CM discussed above. First, we define the syntax; then we give some examples; finally, we define the semantics. To make our presentation simpler, we use sets instead of lists in HTML-QL, but the results can be extended to lists too.

3.1. Syntax of HTML-QL

Besides the set $\mathcal{U}$ of URLs and the set $\mathcal{C}$ of constants, we also assume the existence of a set $\mathcal{V}$ of variables started with '$' followed by a string and '$' itself is an anonymous variable.

Definition 3. The *terms* are defined recursively as follows:

1. A constant is a *lexical* term.
2. If X is a constant or a variable, then $X\langle\rangle$ is a *label* term.
3. If Y is a URL or a variable, then $\langle Y\rangle$ is a *anchor* term.
4. If X is a constant or a variable, and Y is a URL or a variable, then $X\langle Y\rangle$ is a *linking* term, and X is called the *label* and Y is called the *anchor* of the linking term.
5. If X and Y are terms, then $X \Rightarrow Y$ is an *attributed* term.
6. If $X_1, \ldots, X_n$ are distinct terms with $(n \geqslant 0)$, then $\{X_1, \ldots, X_n\}$ is a *set* term.
7. A variable is either an *lexical* term, a *linking* term, a *label*, an *anchor*, an *attributed* term, or a *set* term depending on the context.

Example 12. The following are several examples of terms:

Lexical terms:	*CS Dept, John Smith,* $Name
Label terms:	*Faculty*$\langle\rangle$*, Journals*$\langle\rangle$*,* $Label$\langle\rangle$
Anchor terms:	$\langle$*fac.html*$\rangle$*,* $\langle$*journals.html*$\rangle$*,* $\langle$$URL$$\rangle$
Linking terms:	*Faculty*$\langle$*fac.html*$\rangle$*, Faculty*$\langle$$U$$\rangle$*,* $Label$\langle$$URL$$\rangle$
Attributed terms:	*Title* $\Rightarrow$*CS Dept*, *Program* $\Rightarrow$ $\{$$D$$\}$,
	$A $\Rightarrow$$V, $A $\Rightarrow$$L\langle$$U$$\rangle$, $A\langle$$U$$\rangle$$\Rightarrow$$V
Set terms:	$\{$$X$$\}$, $\{$$X$*, John*$\}$, $\{$*Author*$\langle$$U$$\rangle\}$

A term is *ground* if it has no variables. Note that ground terms are simply objects.

Definition 4. The *expressions* are defined as follows:

1. Let U be a URL or a variable, and T a term. Then $U : T$ is a *positive expression*.
2. If P is a positive expression, then $\neg P$ is a *negative expression*.
3. Arithmetic, string and set operation expressions are defined using terms in the usual way.

For readers familiar with deductive databases or logic programming, The URL u in positive expression $u : T$ and negative expression $\neg u : T$ functions as a predicate and T is $u : T$ as a term. As we can use a variable in the place of u, HTML-QL is in fact a higher-order language.

Example 13. The following are several examples of expressions where u stands for some constant URL:

Positive expressions:	$u : \$X,\ u : \{\$X\},\ \ u : \{Answer \Rightarrow \$X\},\ \$U : \$V$
Negative expressions:	$\neg\, u : \$X,\ \neg\, u : \{Faculty \Rightarrow \{John\}\}, \neg\, u : \{Faculty\langle\rangle\}$
Arithmetic expressions:	$\$A = \$B * 2,\ \ \$Age = 2001 - \$Birthyear$
String expressions:	$\$FName = John + \$LName,\ John \in \$FName$
Set expressions:	$John \in \$Faculty,\ \$S = \$S1 \cup \$S2$

An expression is *ground* if it contains no variables. A ground positive expression is a Web object.

Definition 5. A *rule* has the form $A : -L_1, \ldots, L_n$, where A is a positive expression $u : T$, each L_i is a positive expression, a negative expression, or an arithmetic, string or set operation expression defined using terms. A rule is *safe* if all variables in the head are covered or limited as defined in [4,17,18,29].

For a negative expression with a set term in the body of a rule, we can move the negation sign into the set for convenience. For example, we can use $u : \{\neg Faculty \Rightarrow \{John\}\}$ to stand for $\neg\, u : \{Faculty \Rightarrow \{John\}\}$. We can also combine positive and negative expressions with the same URL for convenience. For example, we can use the expression $u : \{Faculty \Rightarrow \{John\}, \neg Faculty \Rightarrow \{Mary\}\}$ to stand for $u : \{Faculty \Rightarrow \{John\}$, $u : \{\neg\, Faculty \Rightarrow \{Mary\}\}$ in the body of a rule.

Note that the anonymous variable \$ may appear several times in a rule and their different occurrences in general stand for different variables. Thus, it cannot appear in the head of a safe rule.

In deductive database languages, a query is normally defined as a rule with empty head. If the query contains no variables, the query result is either true or false. If the query contains variables, the query result is a set of bindings that make each ground query true. However, for the Web queries, we want not only the set of bindings that make the query true but also proper restructuring of the query results. The head of the rule can be used for

this purpose. Also, complex queries over the Web objects may need more than one rule to express. Thus, we introduce our notion of query as follows.

Definition 6. A *query* is a set of safe rules whose heads have the same URL.

In order to make queries easier to express, we introduce the following shorthands for rules, terms and expressions appearing in rules:

1. $X.$ stands for $X \Rightarrow \$$
2. $X_1.X_2 \ldots X_n$ stands for $X_1 \Rightarrow \{X_2 \Rightarrow \ldots \{X_n\} \ldots\}$
3. $A :- \ldots *^n X \ldots$ stands for the following $n + 1$ rules:
   - $A :- \ldots X \ldots$
   - $A :- \ldots \$.X \ldots$
   - $\vdots$
   - $A :- \ldots \underbrace{\$ \ldots \$}_{n}.X \ldots$

   If there are several such dot notations in a rule, then it stands for their various combinations as outlined above.
4. $A :- \ldots * X \ldots$ stands for $A :- \ldots *^c X \ldots$ for some fixed number c.
5. $A :- \ldots \langle X \rangle : Y$ stands for $A :- \ldots \langle X \rangle, X : Y$
6. $A :- \ldots \langle X \rangle.Y$ stands for $A :- \ldots \langle X \rangle, X : \{Y\}$

In other words, $*^n$ stands for 0–n anonymous variables in the path.

3.2. *Query examples*

The following queries are based on the DBLP Web object shown in Figure 2. To make them simple, we use u_l to stand for the URL `http://www.informatik.uni-trier.de/~ley/db` and u_o for the URL of the query result.

(Q_1) Copy the contents of the document at the given URL u_l into a local file given by the URL u_o:

$$u_o : \$X \ :- \ u_l : \$X$$

Note that no matter what document pointed by u_l is, such as HTML, XML, postscript, image, executable, etc., it is copied to the destination u_o without any processing. If we know that the document is in HTML and we want to convert it into HTML-CM, then we can use the following query instead:

$$u_o : \{\$X\} \ :- \ u_l : \{\$X\}$$

It says that every element denoted by $\$X$ in the set is also an element in the result set. The notion $\{\$X\}$ in the body of the rule means that $\$X$ is an element in the corresponding set whereas the notion $\{\$X\}$ in the head of the rule is used to group the result into a set. It corresponds to a partial set term in Relationlog [18].

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 18

(Q_2) List the objects under the attribute *Search*:

$$u_o : \{Answer \Rightarrow \$X\} \ :- \ u_l : \{Search \Rightarrow \$X\}$$

The result to this query based on the Web object in Figure 2 is as follows:

$$\{Answer \Rightarrow \{Author\langle a_1\rangle,\ Title\langle a_2\rangle,\ Advanced\langle a_3\rangle,\ Home\ Page\ Search\langle a_4\rangle\}\}$$

(Q_3) List the anchors (URLs) under the attribute *Search*:

$$u_o : \{Answer \Rightarrow \{\$X\}\} \ :- \ u_l : \{Search \Rightarrow \{\langle \$X \rangle\}\}$$

The result is $\{Answer \Rightarrow \{a_1, a_2, a_3, a_4\}\}$

(Q_4) List the labels under the attribute *Search*:

$$u_o : \{Answer \Rightarrow \{\$X\}\} \ :- \ u_l : \{Search \Rightarrow \{\$X\langle\rangle\}\}$$

The result is $\{Answer \Rightarrow \{Author,\ Title,\ Advanced,\ Home\ Page\ Search\}\}$. This query can also be represented equivalently using the dot notation in HTML-QL as follows:

$$u_o : \{Answer \Rightarrow \{\$X\}\} \ :- \ u_l : \{Search.\$X\langle\rangle\}$$

(Q_5) List all the attributes at the first and second levels:

$$u_o : \{Answer \Rightarrow \{\$X\}\} \ :- \ u_l : \{\$X \Rightarrow \$Y\}$$
$$u_o : \{Answer \Rightarrow \{\$Y\}\} \ :- \ u_l : \{\$X \Rightarrow \$Y \Rightarrow \$Z\}$$

The result is $\{Answer \Rightarrow \{Title,\ Search,\ Bibliographies,\ \ldots\}\}$. This query can also be represented equivalently using the dot notation as follows:

$$u_o : \{Answer \Rightarrow \{\$X\}\} \ :- \ u_l : \{\$X.\}$$
$$u_o : \{Answer \Rightarrow \{\$Y\}\} \ :- \ u_l : \{\$X.\$Y.\}$$

(Q_6) Obtain the URL of TODS:

$$u_o : \{Answer \Rightarrow \$X\} \ :- \ u_l : \{*TODS\langle \$X \rangle\}$$

The result is $\{Answer \Rightarrow b_{22}\}$

(Q_7) Obtain all the URLs in the page.

$$u_o : \{Answer \Rightarrow \{\$U\}\} \ :- \ u_l : \{*\langle \$U \rangle\}$$

(Q_8) Obtain all the URLs together with their labels.

$$u_o : \{Answer \Rightarrow \{\$L\langle \$U \rangle\}\} \ :- \ u_l : \{*\$L\langle \$U \rangle\}$$

(Q_9) Get all the URLs reachable from the page.

$$u_r : \{\$X\} \ :- \ u_l : \{*\langle \$X \rangle\}$$
$$u_r : \{\$X\} \ :- \ u_r : \{\$Y\},\ \$Y : \{*\langle \$X \rangle\}$$

Note that this query involves multiple Web objects and the rules are recursive since the result Web object defined is used in the body of the second rule.

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 19

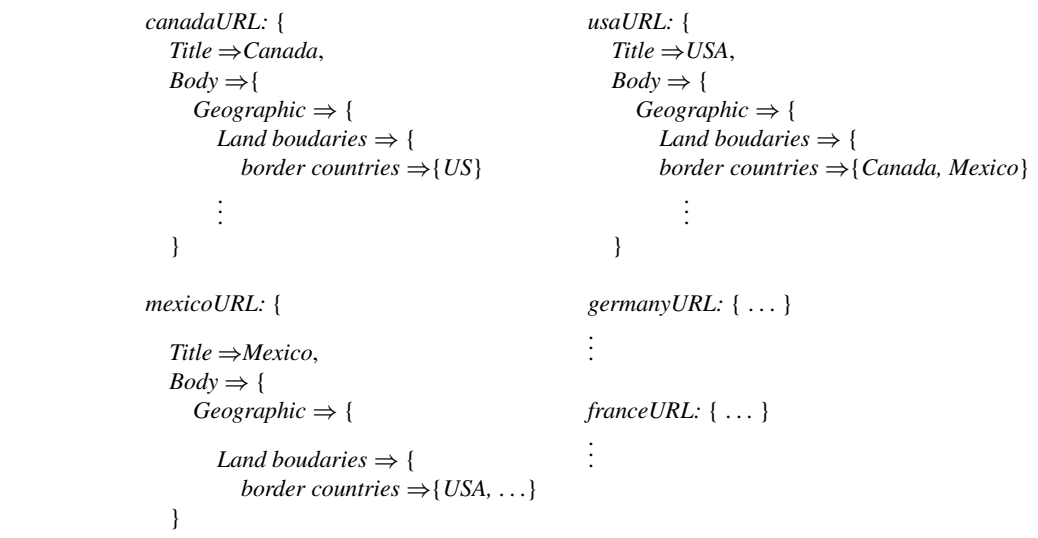

Figure 3. CIA world factbook.

(Q_{10}) Get all the URLs together with their labels reachable from the page.

$$u_r : \{\$L\langle\$U\rangle\} \;:\!-\; u_l : \{*\$L\langle\$U\rangle\}$$
$$u_r : \{\$L\langle\$U\rangle\} \;:\!-\; u_r : \{*\$L'\langle\$U'\rangle\},\ \$U' : \{*\$L\langle\$U\rangle\}$$

In order to demonstrate the expressive power of HTML-QL, let us consider the CIA world factbook 2000 at `http://www.odci.gov/cia/publications/factbook`. This Web server contains detailed information about each country (and region) in the world in HTML format, such as its location, geographic coordinates, area, land boundaries, costline, population, age structure, birth rate, GDP, budget, etc. We can view the Web server as a set of simplified Web objects as shown in Figure 3 and therefore we can query them and infer useful information.

(Q_{11}) Find countries that border both Germany and France.

$$u_o : \{Answer \Rightarrow \{\$N\}\} \;:\!-\; \$U : \{Title \Rightarrow \$N,\ *border\ country \Rightarrow \{Germany,\ France\}\}$$

The result is $\{Answer \Rightarrow \{Belgium,\ Luxemburg,\ Switzerland\}\}$

(Q_{12}) Find countries that border Germany but not France.

$$u_o : \{Answer \Rightarrow \{\$N\}\} \;:\!-\; \$U : \{Title \Rightarrow \$N,\ *border\ country \Rightarrow \{Germany\},\ \neg\, *border\ country \Rightarrow \{France\}\}$$

The result is $\{Answer \Rightarrow \{Austria,\ Czech\ Republic,\ Denmark,\ Frence,\ Netherlands,\ Poland\}\}$. Note that this query involves negation.

(Q_{13}) Find pairs of countries/regions that border the same countries.

$$u_o : \{Answer\Rightarrow \{\{Region1\Rightarrow \$N1,\ Region2\Rightarrow \$N2\}\}\} :-$$
$$\$U1 : \{Title \Rightarrow \$N1,\ *border\ countries \Rightarrow \$Cs\},$$
$$\$U2 : \{Title \Rightarrow \$N2,\ *border\ countries \Rightarrow \$Cs\},$$
$$N1 < N2$$

The result to this query is as follows:

$$\{Answer \Rightarrow \{\{Region1 \Rightarrow Bhutan,\ Region2 \Rightarrow Nepal\},$$
$$\{Region1 \Rightarrow Gibraltar,\ Region2 \Rightarrow Portugal\},$$
$$\{Region1 \Rightarrow Holy\ See\ (Vatican\ City),\ Region2 \Rightarrow San\ Marino\},$$
$$\{Region1 \Rightarrow Hong\ Kong,\ Region2 \Rightarrow Macau\},$$
$$\{Region1 \Rightarrow United\ Arab\ Emirates,\ Region2 \Rightarrow Yemen\}\}$$

Note that in this query, the construct $\{Region1\Rightarrow \$N1,\ Region2\Rightarrow \$N2\}$ in the hread is used to form a pair of countries (or regions) while the outside pair of brackets is used to group the result tuples into the set.

(Q_{14}) Find all the countries that can be reached from Canada by land transportation means.

$$u_r : \{Answer\Rightarrow \{\$C\}\} :- \$U : \{Title \Rightarrow Canada,\ *border\ countries \Rightarrow \{\$C\}\}$$
$$u_r : \{Answer\Rightarrow \{\$C\}\} :-\ u_r : \{Answer \Rightarrow \{\$X\}\},$$
$$\$U : \{Title \Rightarrow \$X,\ *border\ countries \Rightarrow \{\$C\}\}$$

Note that this is another recursive query.

3.3. Semantics of HTML-QL

In this subsection, we define the Herbrand-like logical semantics for HTML-QL. We first define terminology that is needed later in this subsection.

Definition 7. The *Herbrand universe* U_{H} of HTML-QL is the set of all ground terms that can be formed.

In other words, U_{H} is the domain of all possible objects.

Definition 8. The *Herbrand base* B_{H} of HTML-QL is the set of all ground positive expressions that can be formed using terms in U_{H}.

That is, B_{H} is the set of all possible Web objects that can be formed.

Definition 9. A *Web database* WDB is a subset of B_{H}.

In other words, a Web database is a set of Web objects. For example, the CIA world factbook shown in Figure 3 is a Web database.

Definition 10. A *ground substitution* θ is a mapping from the set of Web variables $\mathcal{V} - \{\$\}$ to U_{H}.

The use of anonymous variables and dot notations in the body of rules allows us to express queries easier as demonstrated in the examples above. However, when dealing with the semantics, a ground substitution cannot map the anonymous variable to several different objects. To simplify our presentation, we assume that each occurrence of the anonymous variable is replaced by a non-anonymous variable that never occur in the query rules so we do not have to map anonymous variable \$ to any object. We also assume that the rules here do not contain shorthands.

Given a Web database, the positive expressions in the body of a query rule is actually a query that is used to match part of one or more Web objects. Thus, we introduce the following notions.

Definition 11. An object o is *part-of* an object o', denoted by $o \preceq o'$, if and only if one of the following hold:

(1) both are constants and $o = o'$;
(2) both are linking objects such that one of the following holds:
   - $o = l\langle\rangle$ and $o' = l'\langle u\rangle$ such that $l \preceq l'$;
   - $o = l\langle u\rangle$ and $o' = l'\langle u\rangle$ such that $l \preceq l'$;
   - $o = \langle u\rangle$ and $o' = l'\langle u\rangle$;
(3) both are attributed objects: $o = a \Rightarrow v$ and $o' = a' \Rightarrow v'$ such that $a \preceq a'$ and $v \preceq v'$;
(4) both are set objects such that for each $o_i \in o - o'$, there exists $o_i' \in o' - o$ such that $o_i \preceq o_i'$.

The part-of relationship between objects o and o' captures the fact that o is part of o'.

Example 14. The following are several examples:

$$\begin{aligned}
&\mathit{Faculty} \preceq \mathit{Faculty}\\
&\mathit{Faculty} \preceq \mathit{Faculty}\langle \mathit{fac.html}\rangle\\
&\langle \mathit{fac.html}\rangle \preceq \mathit{Faculty}\langle \mathit{fac.html}\rangle\\
&\mathit{Programs} \Rightarrow \{\mathit{M.Sc\ Program}\} \preceq \mathit{Programs} \Rightarrow \{\mathit{Ph.D\ Program,\ M.Sc\ Program}\}\\
&\{\mathit{Title} \Rightarrow \mathit{CS\ Dept}\} \preceq \{\mathit{Title} \Rightarrow \mathit{CS\ Dept,\ Faculty}\langle \mathit{fac.html}\rangle\}
\end{aligned}$$

We extend the part-of relationship to Web objects and Web databases as follows.

Definition 12. Let $W = u : t$, $W' = u' : t'$ be two Web objects. Then W is *part-of* W', denoted by $W \preceq W'$, if and only if $u = u'$ and $t \preceq t'$.

Definition 13. Let *WDB* and *WDB*$'$ be two Web databases. Then *WDB* is *part-of* *WDB*$'$, denoted by $\mathit{WDB} \preceq \mathit{WDB}'$, if and only if for each $W \in \mathit{WDB} - \mathit{WDB}'$, there exists $W' \in \mathit{WDB}' - \mathit{WDB}$ such that $W \preceq W'$.

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 22

Definition 14. Let *WDB* be a Web database. The notion of satisfaction (denoted by $\models$) and its negation (denoted by $\not\models$) based on *WDB* are defined as follows.

1. For a ground positive expression $u : t$, $WDB \models u : t$ if and only if there exists $u : t' \in WDB$ such that $t \preceq t'$.
2. For a ground negative expression $\neg u : t$, $WDB \models \neg u : t$ if and only if $WDB \not\models u : t$.
3. For each ground arithmetic, string, or set operation expression ψ, $WDB \models \psi$ if and only if ψ is true in the usual sense.
4. For a rule r of the form $A :- L_1, \ldots, L_n$, $WDB \models r$ if and only if for every ground substitution θ, $WDB \models \theta L_1, \ldots, WDB \models \theta L_n$ implies $WDB \models \theta A$.

Example 15. Let *WDB* denote the Web database containing the DBLP Web object in Example 3. Then we have

$$
\begin{aligned}
&WDB \models u_l : \{Search \Rightarrow \{Author\langle\rangle\}\}\\
&WDB \models u_l : \{Bibliographies \Rightarrow \{Journals\langle\rangle\}\}\\
&WDB \models u_l : \{Bibliographies \Rightarrow \{Journals\langle\rangle \Rightarrow \{TODS\langle\rangle\}\}\}\\
&WDB \models \neg u_l : \{Search \Rightarrow \{SIGMOD\langle\rangle\}\}\\
&WDB \models \neg u_l : \{Bibliographies \Rightarrow \{Journals \Rightarrow \{Author\langle\rangle\}\}\}\\
&WDB \models 6 = 3 * 2\\
&WDB \models John\ Smith = John + Smith\\
&WDB \models John \in \{John, Mary, Tony\}
\end{aligned}
$$

Definition 15. Let Q be a query. A *model* M of Q is a Web database that satisfies Q. A model M of Q is *minimal* if and only if for each model N of Q, $M \preceq N$.

As in deductive databases, we are interested in a minimal model of the query that can be computed bottom-up. We now continue to define the bottom-up semantics.

Definition 16. Let *WDB* be a Web database and Q a set of rules. The *immediate logical consequence* operator T_Q over *WDB* is defined as follows:

$$
\begin{aligned}
T_Q(WDB) = \{\theta A \mid\ & A :- L_1, \ldots, L_n \in Q \text{ and there exists a ground}\\
& \text{substitution } \theta \text{ such that } WDB \models \theta L_1, \ldots, WDB \models \theta L_n\}
\end{aligned}
$$

Example 16. Consider query Q_4 in the last subsection and the Web database *WDB* the same as in Example 15, we have

$$
\begin{aligned}
T_{Q_4}(WDB) = \{&u_o : \{Answer \Rightarrow \{Author\}\},\\
&u_o : \{Answer \Rightarrow \{Title\}\},\\
&u_o : \{Answer \Rightarrow \{Advanced\}\},\\
&u_o : \{Answer \Rightarrow \{HomePageSearch\}\}\}
\end{aligned}
$$

Note that the operator T_Q does not perform grouping. Therefore, we introduce the following notions.

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 23

Definition 17. Two objects o and o' are *compatible* if and only if one of the following holds:

1. both are constants and are equal;
2. $o = a \Rightarrow v$ and $o' = a \Rightarrow v'$ such that v and v' are compatible;
3. both are set objects.

A set of objects are *compatible* if and only each pair in it is compatible.

Example 17. The following pairs are compatible:

Author and *Author*
{*Author*} and {*Title*}
Answer $\Rightarrow$ {*Author*} and *Answer* $\Rightarrow$ {*Title*}

Definition 18. Two Web objects $u : t$ and $u' : t'$ are *compatible* if and only if $u = u'$ and t and t' are compatible. A set of Web objects are *compatible* if and only if each pair of them is compatible.

Example 18. The following set of Web objects are compatible.

u_o : {*Answer* $\Rightarrow$ {*Author*}},
u_o : {*Answer* $\Rightarrow$ {*Title*}},
u_o : {*Answer* $\Rightarrow$ {*Advanced*}},
u_o : {*Answer* $\Rightarrow$ {*Home Page Search*}}

Definition 19. Let S be a set of (Web) objects and S' a compatible subset of S. Then S' is a *maximal* compatible set in S if there does not exist a (Web) object $o \in S - S'$ that is compatible with each object in S'.

Definition 20. Let S be a set of objects. The *grouping* operator G is defined recursively on S as follows:

1. If S is a singleton set $S = \{o\}$, then $G(S) = o$.
2. If S is a compatible set of attributed objects $S = \{a \Rightarrow v_1, \ldots, a \Rightarrow v_n\}$, then $G(S) = a \Rightarrow G(\{v_1, \ldots, v_n)$.
3. S is a set of set objects, then

$$G(S) = \bigcup \big\{ G\big(S'\big) \mid S' = \{o \mid o \in s, \text{ and } s \in S\} \text{ is a maximal compatible set of objects} \big\}.$$

It is extended to a set of Web objects as follows:

1. If S is a compatible set of Web objects of the form $u : o_1, \ldots, u : o_n$, then $G(S) = u : G(\{o_1, \ldots, o_n\})$.

2. If S is divided into maximal compatible subsets $S_1, \ldots, S_n$ such that $S = S_1 \cup \cdots \cup S_n$, then $G(S) = G(S_1) \cup \cdots \cup G(S_n)$.

Definition 21. The *powers* of the operation T_Q over the Web database *WDB* are defined as follows:

$$\begin{aligned} T_Q \uparrow 0(WDB) &= WDB \\ T_Q \uparrow n(WDB) &= T_Q(G(T_Q \uparrow n-1(WDB))) \cup T_Q \uparrow n-1(WDB) \\ T_Q \uparrow \omega(WDB) &= \bigcup_{n=0}^{\infty} T_Q \uparrow n(WDB) \end{aligned}$$

Example 19. Continuing with the Example 16, we have

$$\begin{aligned} G(T_{Q_4} \uparrow \omega(WDB)) &= G(\{ \\ &\qquad u_o : \{Answer \Rightarrow \{Author\}\}, \\ &\qquad u_o : \{Answer \Rightarrow \{Title\}\}, \\ &\qquad u_o : \{Answer \Rightarrow \{Advanced\}\}, \\ &\qquad u_o : \{Answer \Rightarrow \{Home\ Page\ Search\}\}\}) \\ &= u_o : G(\{ \\ &\qquad \{Answer \Rightarrow \{Author\}\}, \\ &\qquad \{Answer \Rightarrow \{Title\}\}, \\ &\qquad \{Answer \Rightarrow \{Advanced\}\}, \\ &\qquad \{Answer \Rightarrow \{Home\ Page\ Search\}\}\}) \\ &= u_o : \{Answer \Rightarrow G(\{Author\}, \{Title\}, \{Advanced\}, \{Home\ Page\ Search\})\} \\ &= u_o : \{Answer \Rightarrow \{Author, Title, Advanced, Home\ Page\ Search\}\} \end{aligned}$$

Theorem 1. Let *WDB* be a Web database and Q a query. Then $G(T_Q \uparrow \omega(WDB))$ is a minimal model of Q.

Proof: It is essentially the same as the proof in [18]. □

Definition 22. Let *WDB* be a Web database and Q a query. Then the *semantics* of Q under *WDB* is given by $G(T_Q \uparrow \omega(WDB))$.

4. Implementation

Based on the conceptual model HTML-CM and the rule-based query language HTML-QL presented in the previous sections, we have developed a prototype system for Web search and inference. The system will soon be available from the Web site `http://www.scs.carleton.ca/~mliu/HTML-QL`. In this section, we briefly describe the system.

The high level architecture of the system is shown in Figure 4.

The system is organized into four layers. The first layer is the entire World-Wide Web on the Internet. Because HTML documents are not only generated automatically by specific application programs such as FrontPage and CoffeeCup, but also written by human

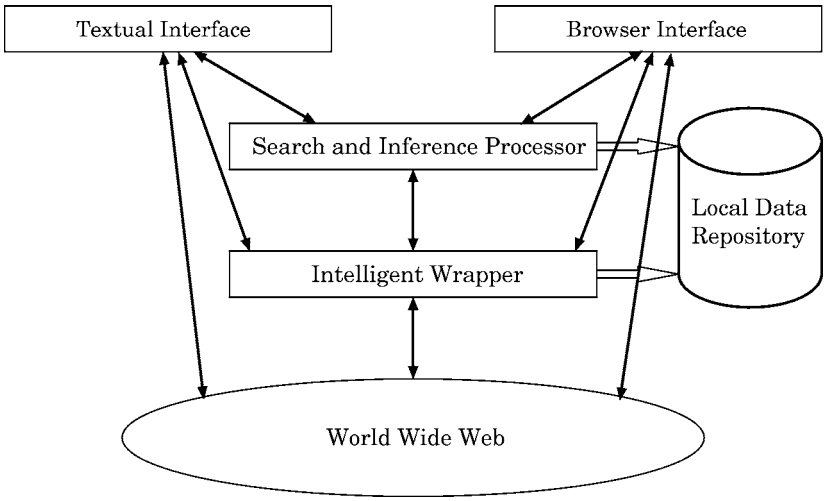

Figure 4. High level system architecture.

manually, it is quite common that some documents have syntactic errors. Like commercial Web browsers such as Internet Explorer and Netscape, our system can tolerate syntactic errors.

The second layer is the Intelligent wrapper that implements HTML-CM. It accesses the World-Wide Web through the Internet, fetches HTML documents and converts them into Web objects in HTML-CM. In order to speed-up query processing, the Intelligent Wrapper also caches Web objects in the local data repository and builds proper indexes on these Web objects. It allows the user to adjust Web objects by adding, adjusting or deleting attributes or objects and it can also learn the pattern from the user's interaction and then process similar HTML documents accordingly. For these reasons, it is called the intelligent wrapper.

The third layer of the system is the Search and Inference Processor that implements HTML-QL. It is mainly in charge of query processing. It obtains queries to be processed from the user interface layer and checks whether or not Web objects involved are in the local data repository. If Web objects involved are not in the local data repository, then it invokes the Intelligent Wrapper to fetch them. Then it uses semi-naive bottom-up fix-point computation to generate the result and sends the query result to the user interface layer for proper display.

The last layer is the User Interface. Two kinds of interfaces are supported: textual interface and browser interface. They provide different kinds of environment for the user to issue commands and express queries, perform syntactical analysis, and pass valid commands in internal format to the Search and Inference Processor. They also display query results generated by the Search and Inference Processor to the user in different format, the Web objects generated by the Intelligent Wrapper, and original Web document like lynx and Netscape, respectively.

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 26

5. Conclusion

The main contributions of the paper are the following. First, we have proposed a conceptual model HTML-CM for HTML documents. Unlike other data models that mainly focus on inter-document structures and hyperlink navigation, our conceptual model focuses on the complex hierarchical structure within the HTML documents (intra-document structure). It only has a few simple but powerful high level constructs that can be used to best describe the contents in HTML documents. The HTML documents represented in this conceptual model are close to human conceptualization/visualization of the documents. We have also presented a set of generic rules to automatically convert HTML documents into this conceptual model. Discovering rules to structure the HTML documents has already been addressed in the literature such as [10,13,15,25,26]. However, the rules presented in this paper are more general and systematic. During the conversion, we ignore many features that are used to enhance the visual aspects of the Web documents. In other words, we lose some information during conversion. Thus, we cannot convert Web documents in our conceptual model back into HTML format. However, we can capture the essence of HTML documents in a natural and intuitive way.

Second, we have presented HTML-QL, a rule-based language for querying HTML documents over the Internet based on HTML-CM. Unlike other Web query languages, HTML-QL provides a simple but very powerful way to query both the intra-document structures and inter-document structures and allows the query results to be restructured. Furthermore, HTML-QL has a firm logical foundation, which is lacking in most other Web query languages. The prototype system that supports HTML-CM and HTML-QL have been implemented and will soon be available from the Web site `http://www.scs.carleton.ca/~mliu/HTML-QL`. The query language and system can be easily extended to handle XML documents.

We note that although the approach works well on many HTML documents, it is still the case that HTML pages can be found where it is difficult to extract useful information. We would like to extend the functionality of HTML-QL by incorporating other useful features to make it a really useful tool for Web search and inference and investigate the computability and complexity issues of HTML-QL queries. Using HTML-QL, we would also like to develop data extraction and data integration tools based on the method proposed in [20,22]. Our objective is to build an intelligent Web search engine on top of the search and inference system.

Acknowledgements

The research was partially supported by research and equipment grants from the Natural Sciences and Engineering Research Council of Canada (NSERC). The authors are also grateful to Yibin Su for implementing the system.

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 27

References

[1] S. Abiteboul, "Querying semistructured data," in *Proc. of the Internat. Conf. on Data Base Theory*, Lecture Notes in Computer Science 1186, Springer: New York, 1997, pp. 1–18.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener, "The Lorel query language for semistructured data," *Internat. J. Digital Libraries* 1(1), 1997, 68–88.

[3] G. Arocena and A. Mendelzon, "WebOQL: Restructuring documents, databases and Webs," in *Proc. of the Internat. Conf. on Data Engineering*, IEEE Computer Soc., 1998, pp. 24–33.

[4] C. Beeri, S. Naqvi, O. Shmueli, and S. Tsur, "Set construction in a logic database language," *J. Logic Programming* 10(3,4), 1991, 181–232.

[5] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, Extensible markup language (XML) 1.0, *W3C Recommendation*; see `http://www.w3c.org/TR/1999/REC-xml-19980210`, February 1998.

[6] P. Buneman, S. Davidson, G. Hilebrand, and D. Suciu, "A query language and optimization techniques for unstructured data," in *Proc. of the ACM SIGMOD Internat. Conf. on Management of Data*, 1996, pp. 505–516.

[7] J. Clark and S. DeRose, XML path language (XPath) version 1.0, *W3C Recommendation*; see `http://www.w3c.org/TR/1999/REC-xpath-19991116`, November 1999.

[8] O. Shmueli and D. Konopnicki, "W3QS: A query system for the World-Wide Web," in *Proc. of the Internat. Conf. on Very Large Data Bases*, Zurich, Switzerland, Morgan Kaufmann, 1995, pp. 54–65.

[9] M. Fernandez, D. Florescu, A. Levy, and D. Suciu, "A query language for a Web-site management system," *SIGMOD Record*, 1997, 4–11.

[10] M. Fernandez, D. Florescu, A. Levy, and D. Suciu, "Reasoning about Web-site structure," in *Proc. of AAAI'98 Workshop on AI and Information Integration*, 1998.

[11] D. Florescu, A. Levy, and A. Mendelzon, "Database techniques for the World-Wide Web: A survey," *SIGMOD Record* 26(3), 1997.

[12] D. Florescu, A. Levy, and A. Mendelzon, "Database techniques for the World-Wide Web: A survey," *SIGMOD Record* 27(3), 1998, 59–74.

[13] J. Hammer, H. Garcia-Molina, J. Cho, A. Crespo, and R. Aranha, "Extracting semistructured information from the Web," in *Proc. of the Workshop on Management of Semistructured Data*, 1997.

[14] R. Himmeroder, G. Lausen, B. Ludascher, and C. Schlepphorst, "On a declarative semantics for Web queries," in *Proc. of the Internat. Conf. on Deductive and Object-Oriented Databases*, Switzerland, 1997, Lecture Notes in Computer Science, Springer: New York, pp. 386–398.

[15] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, P. J. Modi, I. Muslea, A. G. Philpot, and S. Tejada, "Modeling Web sources for information integration," in *Proc. of the 15th National Conf. on AI*, 1998.

[16] L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian, "A declarative language for querying and restructuring the Web," in *Proc. of the 6th Internat. Workshop on Research Issues in Data Engineering*, 1996.

[17] M. Liu, "ROL: A deductive object base language," *Information Systems* 21(5), 1996, 431–457.

[18] M. Liu, "Relationlog: A typed extension to datalog with sets and tuples," *J. Logic Programming* 36(3), 1998, 271–299.

[19] M. Liu and T. W. Ling, "A conceptual model for the Web," in *Proc. of the Internat. Conf. on Conceptual Modeling (ER 2000)*, Salt Lake City, 9–12 October 2000, Lecture Notes in Computer Science, Springer: New York, pp. 225–238.

[20] M. Liu and T. W. Ling, "A data model for semistructured data with partial and inconsistent information," in *Proc. of the Internat. Conf. on Advances in Database Technology (EDBT 2000)*, Konstanz, Germany, 27–31 March 2000, Lecture Notes in Computer Science 1777, Springer: New York, pp. 317–331.

[21] M. Liu and T. W. Ling, "A rule-based query language for the Web," in *Proc. of the 7th Internat. Conf. on Database Systems for Advanced Applications (DASFAA 2001)*, Hong Kong, China, 18–20 April 2001, IEEE Computer Soc. Press: Silver Spring, MD, pp. 6–13.

[22] M. Liu, T. W. Ling, and T. Guan, "Integration of semistructured data with partial and inconsistent information," in *Proc. of the Internat. Database Engineering and Application Symposium (IDEAS '99)*, Montreal, Canada, 2–4 August 1999, IEEE Computer Soc. Press: Silver Spring, MD, pp. 44–52.

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 28

[23] A. Mendelzon, G. Mihaila, and T. Milo, "Querying the World Wide Web," in *Proc. of the 1st Internat. Conf. on Parellel and Distributed Information System*, 1996, pp. 80–91.
[24] A. O. Mendelzon and T. Milo, "Formal models of Web queries," in *Proc. of the ACM Symposium on Principles of Database Systems*, 1997.
[25] I. Muslea, S. Minton, and C. A. Knoblock, "Hierarchical wrapper induction for semistructured information sources," *J. Autonom. Agents Multi-Agent Systems* 4(1/2), 2001, 93–114.
[26] J. Myllymaki, "Effective Web data extraction with standard XML technologies," in *Proc. of the 10th Internat. World Wide Web Conf.*, Hong Kong, China, 2001, ACM: New York, pp. 689–696.
[27] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, "Object exchange across heterogeneous information," in *Proc. of the Internat. Conf. on Data Engineering*, IEEE Computer Soc. Press: Silver Spring, MD, 1995, pp. 251–260.
[28] D. Raggett, A. L. Hors, and I. Jacobs, "HTML 4.01 specification," W3C Recommendation; see `http://www.w3c.org/TR/html401`, December 1999.
[29] J. D. Ullman, *Principles of Database and Knowledge-Base Systems*, Vol. 1, Computer Soc. Press: Silver Spring, MD, 1988.
[30] L. Wood, A. L. Hors et al., "Document Object Model (DOM) Level 2 Specification," W3C Recommendation; see `http://www.w3c.org/TR/2000/CR-DOM-Level-2-20000307`, March 2000.

VTEX(CG) PIPS No:381690 artty:ra (Kluwer BO v.2001/08/30)
WJ381690.tex; 18/09/2001; 10:46; p. 29