An Integrated Architecture for Exploring, Wrapping, Mediating and Restructuring Information from the Web

Wolfgang May

Institut für Informatik, Universität Freiburg, Germany
E-mail: `may@informatik.uni-freiburg.de`

Abstract

The goal of information extraction from the Web is to provide an integrated view on heterogeneous information sources. A main problem with current wrapper/mediator approaches is that they rely on very different formalisms and tools for wrappers and mediators, thus leading to an "impedance mismatch" between the wrapper and mediator level. Additionally, most approaches currently are tailored to access information from a fixed set of sources.

In this paper, we discuss an architecture where Web exploration, wrapping, mediation, and querying is done in an integrated system. Such an architecture reveals significant advantages in combination with a unified framework – i.e., data model and language – in which all tasks are done. Our approach is based on a unified model of the application-level information and the relevant fragment of the Web, and on an integrated language for accessing the Web, wrapping, mediating, and querying information.

In this world model, in contrast to other approaches, the relevant part of the Web becomes a part of the internal world model of the system. This allows for a data-driven Web exploration which is independent from a given network of individual predefined wrappers and mediators. Thus, in addition to the classical wrapping and mediating functionality, a system in this architecture can be equipped with Web navigation *and* exploration *functionality.*

In an abstract sense, the system comprises a universal wrapper which can be applied to arbitrary Web data sources which become known to the system during information processing. Equipped with suitably intelligent rules, the system can potentially explore before unknown parts of the Web, thus coping with the steady growth of the Web.
The architecture is implemented in the FLORID *system [17].*

1. Introduction

The Web is now the most popular information repository and there is a strong need for integration of data from different sources. A standard approach for this is the mediator architecture [31] which comprises *wrappers* for translating data from different local languages into a common format, and *mediators* for providing an integrated view on the data. Existing approaches have a strictly layered architecture and employ different languages for wrappers and mediators. Every data source is assigned a wrapper; mediators are connected to wrappers which provide relevant information for them. There the network of sources, wrappers, and mediators has to be fixed a priori. Such systems in general rely on very different formalisms and tools for wrappers and mediators, leading to an "impedance mismatch" between the wrapper and mediator level. In general, their world model does not contain knowledge about the Web structure itself, which is required for autonomous Web exploration in search for specific information and adaptability.

In this paper, we propose an architecture where Web exploration, wrapping, mediation, and querying is done in an integrated system. In this architecture, the application-level information and the relevant fragment of the Web are represented in a unified world model. By providing an integrated rule-based language for wrappers, mediators, and for querying, an impedance mismatch due to separate languages for these tasks can be avoided. This also allows for data-driven Web exploration based on previously accessed pages.

The architecture is implemented in FLORID[17], an implementation of the deductive object-oriented database language F-Logic [19] extended with Web access functionality. F-Logic serves as data model, and as wrapper, mediator, and query language. Generic rule patterns are used for *structured document analysis* based on an SGML parser, and for processing raw data by pattern matching. Some issues of handling semistructured data and Web access have already been described in [26]. In [27] and [28] we dealt with the integrated data model and generic wrapping techniques, respectively. In the present paper, we focus on the architecture and its merits and consequences wrt. other approaches.

The paper is structured as follows: After introducing our architecture in Section 2, we present the formal framework underlying our implementation in Section 3. Section 4 describes the underlying Web model and the realization of the Web access in this framework. In Section 5, we first present the internal object-oriented data model which unifies the Web structure and the application-level model. Then, we describe how wrapping and mediating tasks are executed on this model. In Section 6, we focus on the autonomous Web exploration functionality which is supported by the integrated architecture. An overview of related work is given in Section 7. The conclusion relates our work to the standard layered architecture and gives an abstract interpretation.

2. The Architecture

Figure 1 shows our architecture of a system for processing information from the Web. In contrast to the common layered wrapper-mediator-architecture (cf. [31]), our architecture is based on a single interface to the Web as a whole instead of multiple interfaces to individual Web sources via individual, independent wrappers. The http/ftp-protocol-based Web interface provides *generic* (i.e., source-independent) operations for fetching and parsing information from the Web into the system. Via this interface, the Web can be seen as a part of the system which can be queried on-demand.

The reasoning module of the system maintains an integrated world model comprising a model of the relevant fragment of the Web as the *carrier of information* and an application-level model of the *carried information* itself. The world model is handled by a powerful, user-definable internal logic which combines wrapper, mediator, and querying functionality in a single framework (cf. Section 5):

- Exploration and Navigation: Since the internal world model contains information about the Web structure, the system can explore the Web autonomously. Relevant urls can be found and fetched into the system based on hyperlinks on already known pages, or by using Web search machines, such as AltaVista.
- Wrapping: the wrappers operate on the representation of Web pages as strings or parse-trees, generating an intertwined application-level model of the information. Here, standard tasks in wrapping, e.g., tables, lists, commalists, etc., can handled by generic patterns. These are complemented by application-specific supplements and refinements for handling exceptional cases to obtain the required flexibility.
- Integration and mediation: The model comprises the collected wrapped information from several sources transformed in a common application-level model.

Having navigation, web access, wrapping, and mediator functionality together, a cross-fertilization of these tasks is possible. Especially, this also allows for data-driven Web exploration based on previously accessed pages as shown in Section 6. This can especially be exploited in a rule-based framework as presented in this work.

In the following sections, we describe the FLORID system [12][1] which follows this architecture. Based on the object-oriented deductive language F-Logic, it demonstrates the ideas and advantages of the architecture. Although, other implementations (E.g., in contrast to our deductive system, a "functional" system providing a lisp-like manipulation and querying language using a KIF [15]-like data model) could be possible, if they provide a Web interface, an internal data model, and abstract functionality as described in Sections 4 and 5.

3. Formal Framework of the FLORID System

By combining the rich modeling capabilities (objects, methods, class hierarchy, inheritance, signatures) of the object-oriented data model with the advantages of deductive database languages, F-Logic [19] provides a suitable framework for modeling and handling Web information. At a short glance, the syntax is as follows (we omit inheritable methods and signature specifications; for the full syntax and semantics, the reader is referred to [19]):

- The language is based on variables, constants, and *object constructors* from which *id-terms* are composed as usual. Id-terms are interpreted as elements of the universe. For convention, object constructors start with lowercase letters whereas variables start with uppercase ones. Ground id-terms play the role of *logical* object identifiers (*oids*).

In the sequel, let O, C, D, Q_i, S, S_i, *Sc*, and *Mv* stand for id-terms.

- An *is-a assertion* is an expression of the form $O : C$ (object O is a member of class C), or $C :: D$ (class C is a subclass of class D).
- The following are *object atoms*:
- $O[\mathit{Sc}@(Q_1, \ldots, Q_k) \rightarrow S]$: the *single-valued* method *Sc* with arguments $Q_1, \ldots, Q_k$ to O results in S,
- $O[\mathit{Mv}@(Q_1, \ldots, Q_k) \twoheadrightarrow \{S_1, \ldots, S_n\}]$: the *multi-valued* method *Mv* with arguments $Q_1, \ldots, Q_k$ of O has the values S_i.
- An F-Logic *rule* is a logic rule $\mathsf{h} \leftarrow \mathsf{b}$ over F-Logic's atoms; an F-Logic *program* is a set of rules.

Example 1 *Figure 2 shows a fragment the F-Logic representation of the DBLP server [8]. For readability, we use mnemonic oid's of the form o_{name}.*

In addition to the basic F-Logic syntax, the FLORID system also supports *path expressions* [13] in place of id-terms for navigating in the object-oriented model. Especially, single-valued references can create *anonymous objects* when used in the *head* of rules. A rule of the form

O.M[<properties>] ← <body(O,M)>

[1]available at http://www.informatik.uni-freiburg.de/~dbis/florid/.

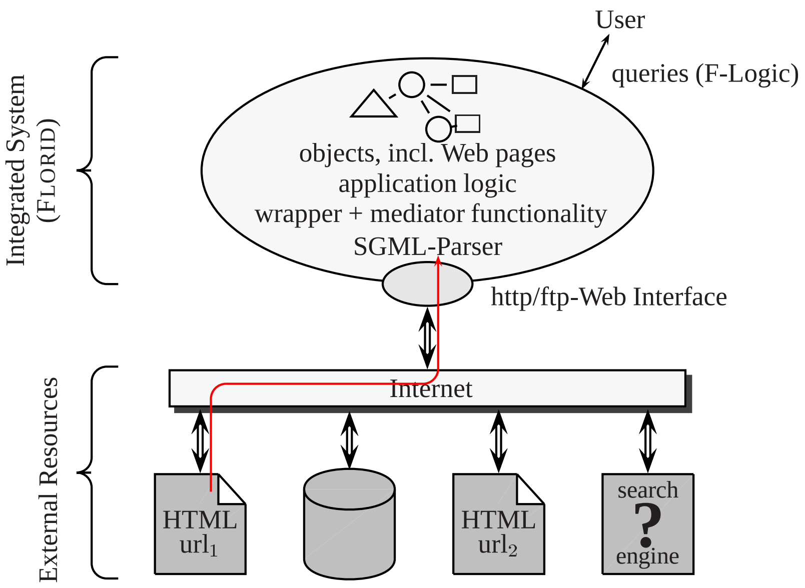

Figure 1. An Integrated Architecture for Web Access

o_{is} : journal[name→"Information Systems";
eds@(...)↠{o_{mj}}].
o_{i11} : journal_vol[of→o_{is}; year→1975;
number→1; volume→1;].
o_{vldb} : conf_series[name→"Very Large Databases"].
o_{v76} : conf_proc[of→o_{vldb}; year→1976; eds↠{o_{pcl}, o_{ejn}}].
journal_p : paper. conf_p : paper.
o_{j1} : journal_p[title→". . . "; authors↠{o_{mes}}; in_vol→o_{i11}].
o_{di} : conf_p[title→". . . "; authors↠{o_{mes}, o_{eba}};
at_conf→o_{v76}].
o_{mes} : person[name→"Michael E. Senko"].
o_{mj} : person[name→"Matthias Jarke"; affil→o_{rwt}].
o_{rwth} : institution[name→"RWTH Aachen"]. . . .

Figure 2. Excerpt of an F-Logic database

creates an object x such that o[m→x] and x[<properties>] hold whenever <body(o,m)> is satisfied. Object creation is used for constructing an application-level model.

Negation-free F-Logic programs have a standard logic programming semantics [19]. For programs with negation, FLORID allows *inflationary* and user-defined *stratified* semantics.

Since path expressions and F-Logic atoms may be arbitrarily nested, a concise and extremely flexible specification language for object properties is obtained. F-Logic supports reasoning about structure and schema by allowing variables at property and class positions. Such "meta-queries" plus the possibility to fuse objects by equating their oids are crucial and unique features of our integration approach.

In the following, F-Logic serves as a framework for building a comprehensive model of the relevant Web fragment and its application-level representation, and for extracting, integrating, and querying data in this model. The FLORID system provides an implementation of F-Logic extended with Web access functionality as described below.

4. The Web Interface

The Web interface maps the relevant Web documents into the unified world model *without interpreting them*. This mapping is based on the generic idea of modeling the Web by two classes url and webdoc (cf. [26]). Thus, Web access via the interface is performed by a single generic method which is based on Internet access using the http/ftp protocols: Every member u of class url provides a special method get which makes the associated Web document wd : webdoc a part of the world model of the system. get is implemented as an *active* method: By evaluating rules of the form

u.get:- <body> ,

the internal database is extended by a new Web document:

- the Web document which is accessible via the url u is accessed,
- it is assigned to the newly created object u.get (conceived as a large string),
- it is made an instance of class webdoc, and
- several properties are automatically filled in (cf. Figure 3):
  – Typically, the document associated with a url contains *hyperlinks* to other url's which refer to further Web documents. This is modeled by the built-in attribute

hrefs:

u.get[hrefs@(ℓ) $\twoheadrightarrow$ u'] $\Leftrightarrow$
u.get contains "<a href = u' > ℓ </a>".

– If an error occurs when a document is accessed via its url, u.get[error $\twoheadrightarrow$ <error_message>] holds for the resulting error message.

url::string[get⇒webdoc].
webdoc[url ⇒url; author ⇒string;
    type ⇒string; hrefs@(string) ⇒⇒url; ... ;
    modif ⇒string; error ⇒⇒string].

wd1:webdoc[url→"url1"; hrefs@("label")↠{"url2"}]
wd2:webdoc[url→"url2"; type→"html"; ...]

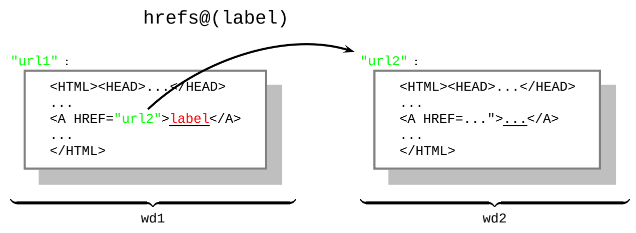

Figure 3. F-Logic Web model: signature and example data

Thus, u.get[hrefs@(ℓ) $\twoheadrightarrow$ u'] contains the link structure (*Web skeleton*) of the relevant part of the Web fragment. Via *path expressions*, the user can then navigate through the Web skeleton. By formulating appropriate rules of the form

U'.get :- U.get[hrefs@(L)↠U'], <filter(L,U,U')> ,

where <filter> specifies a *filter* on urls to be followed, the system can autonomously explore the Web. Using u.get.error, the system can also react on access errors, i.e., with accessing alternative data sources.

By the *Web interface method* u.get, *raw* (i.e., uninterpreted) data from the Web is fetched into the system. This handling reflects only the very basic properties of Web documents, without further exploiting the document type or structure.

Generic pre-processing of Web Data. The active method u.parse generates the F-Logic representation of the parse-tree of an SGML document (for this, FLORID employs the SGML-parser *nsgmls*). Although this already happens inside the system, we consider u.parse as part of the Web interface since it provides a source-independent and application-independent functionality which is tightly coupled with Web-specific notions. The parse-tree is assigned to the object u.parse : parsetree which is linked into the Web skeleton:

- the parse-tree is made an object u.parse : parsetree and is transformed into an object-oriented representation:
- every *SGML-tagged* group $<tag>$ ... $</tag>$ is made an object o;
- the class $wd.<tag>$ contains all such groups $<tag>$ ... $</tag>$ on wd, e.g., x : wd.table holds for all tables on wd.
- each tag induces a method for navigation in a parse-tree: Let x be a node of the parse-tree, then $x.<tag>$@(0), ..., $x.<tag>$@(n) address the distinguished segments inside an instance of $x.<tag>$, e.g.,

- For suitable $k = 0, 1, 2, \ldots$, wd.html@(k) addresses all distinguished segments of wd between <html> and </html>, e.g. wd.html@(0) is the head, providing wd.html@(0).head@(i) and wd.html@(1) is the body, providing wd.html@(1).body@(j).

Example 2 (Tables) *Tables are represented in HTML by the tags <table>, <tr> (table row), <td> (column elements containing data), and <th> (column elements containing header entries). Then,*

- *all tables which contain '1998' in some column of some header row are identified by*
  T : (wd.table), T.table@(R).tr@(C)[th@(0)→S],
  substr(S,"1998").
- *the third column of the 17th row of a given table tab is addressed by* tab.table@(17).tr@(3).

As a future extension to XML documents, a special generic wrapper will generate a schema of the data source based on the DTD, and an object-oriented representation of the data based on the XML file.

5. The Internal Data Model and Functionality

The whole inside of the system is uniformly designed in an object-oriented framework: the information is represented in the F-Logic data model; Web exploration, wrapping, mediation, and querying is done by F-Logic rules.

Organization of the Web Model. Via the Web interface, the relevant Web pages are accessed and mapped to an internal representation as given in the previous section. Web documents are connected by references derived from hyperlinks. In our framework, the parse-trees together with the hyperlinks form the *extended Web skeleton* [27] of the relevant fragment of the Web which is the basis for generating an application-level model.

Example 3 (DBLP Server: Extended Web Skeleton) *Figure 4 shows a fragment of the extended skeleton of the DBLP server. Dashed lines denote hyperlinks on the page level, full lines denote links in the parse-tree representation, and zigzag links denote hyperlinks emanating from parse-tree nodes. Boxed nodes correspond to real-world objects.*

Note that the extended Web skeleton can be generated automatically, solely by url.get and url.parse via rules con-

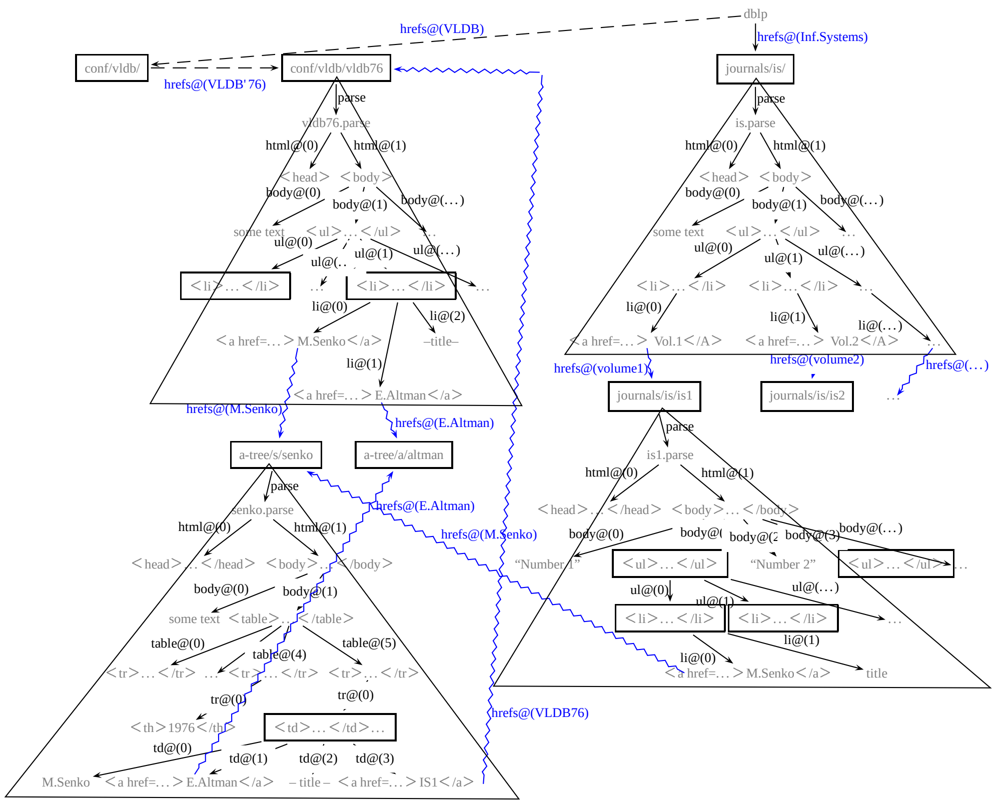

Figure 4. DBLP Database: Skeleton + Parse-trees

trolling the exploration process (cf. Section 6). With the extended Web skeleton, the information to be wrapped is contained in the model, allowing for a flexible internal wrapping process. By wrapping and mediating, from the (still uninterpreted) extended Web skeleton, an application-level model is derived and superposed on the skeleton.

Wrapper Functionality. In our approach, the construction of wrappers and mediators is based on *generic rule schemata* [28] for standard tasks which can be complemented by application-specific rules and refinements for handling exceptional cases to obtain the required flexibility. For the wrapping task, depending on the structure of Web pages, *parser*-based, *matching*-based, or *combined* wrapper rules are preferable. For logical markup such as lists or tables which is subject to a well-specified grammar, wrapping is based on u.parse, the object-oriented representation of the *parse-tree*. Such generic patterns are, e.g., defined for lists and tables. If the structure is only partially tagged (paragraphs, line breaks, fonts, commas), the wrapper is based on pattern matching via regular expressions, provided by a built-in matching predicate

pmatch(<string>,"/<regexp>/", [<fmt-list>], [$X_1, \ldots, X_n$])

which extracts substrings by Perl's regular expressions into Perl-variables and binds them to F-Logic variables. Generic matching patterns are used, e.g., for decomposing "pseudo-lists" which are structured via emphasizing or boldfacing, for commalists, and for name-value pairs. Additionally, arbitrary predicates (e.g., for string processing) can be defined via FLORID's built-in *Perl* interface.

These generic wrapper rules are *not* designed to apply for a specific Web site/page, but can be applied to every Web page which is considered to be relevant – thus, also previously unknown Web pages can be wrapped. The power and flexibility of generic rule patterns depends on the usage of variables ranging over objects (including host objects), classes, and methods which is allowed in F-Logic.

Mediator Functionality. Mediators are also specified by F-Logic rules. Here, typical tasks, e.g., fusing equivalent objects from different sources, or replacing string(name)-valued attributes by links to appropriate objects are covered by generic rules. Additionally, the semantics of the application is encoded into rules which generate an integrated object-oriented model of the application domain from the partial models derived by the wrapper rules.

Based on the integrated world model, rules can be defined which operate (simultaneously) on the Web skeleton, its hyperlinks, the pages HTML/XML sources and their parse-trees, and on the application-level model itself. A distinguishing feature of our integrated architecture – using a unified data model and manipulation language – is that these rules can interfere seamlessly, combining information from different data sources. By inter-source joins, information already extracted from one source can be applied for specific extraction of information of other sources (including exploring other sources which provide the required information).

In general, for being successful with such an architecture, the data manipulation language has to be concise and flexible; note again that also a system based on a functional language instead of F-Logic could be possible.

6. Autonomous Data-Driven Web Exploration

Due to the architecture with an integrated Web interface, the functionality is not restricted to wrapping and mediating tasks. Since the Web structure and the parse-trees are contained in the internal world model, this information can also be used by rules. To our knowledge, FLORID is the only system which is aware of the Web structure itself in its data model.

Especially, a system in this architecture can be equipped with *Web navigation* and *exploration* functionality. Since Web access and data manipulation are specified in the same framework (here, by rules containing u.get in the head), Web exploration and navigation can be done in a data-driven way. The internal database is generated by iteratively accessing Web documents, analyzing them, and then following relevant hyperlinks (Figure 5). Relevant hyperlinks can be found based on hyperlinks on already known pages, or by using Web search machines, such as AltaVista. In both cases, the set of urls to be actually accessed can be restricted using filtering rules of the form

```
U.get:- U:url, <u qualifies as a relevant url> .
```

Especially, wrapping one source can be driven by the information which is extracted from other sources. The extracted data can be reorganized in this loop or in a separate subsequent step.

For wrapping Web pages which are not known a priori (i.e., it is not possible to define a wrapper for them at program design time), the power of the generic "universal" wrapper rules is crucial.

Example 4 (Searching specific information) *Suppose we are interested in the affiliations, addresses, mail-addresses, photos etc. of all DBLP authors. These informations are available from their homepages. These are not part of the DBLP server, and do not share a common structure. Additionally, the address is often not given explicitly on the homepage, but on the homepage of a person's institution.*

The url of an author's homepage is given on the DBLP server under the "Homepage" hyperlink:

```
A[homepage→Url] ←
     A:author.url.get[hrefs@("Homepage")↠Url].
Url.get :- A:author[homepage→Url].
Url.parse :- A:author[homepage→Url].
```

Then, heuristics can be given for extracting the address from the homepage, searching an "address" keyword (possibly depending on the native language of the Web pages host server), or using the HTML <address> tag. Similar, mailto-links can be used to extract mail addresses, and <img>-tags can be used for detecting photos.

By combining rule-based Web exploration and data extraction, the architecture allows also for implementing Web search engines: In contrast to the current index-based search engines (which can be used by the programs) which allow only simple queries, such a system can extract and process semantical information and use it for a semantics-based search on the Web by following hyperlinks.

7. Related Work

Among the first generation Web query languages, some systems followed an integrated approach: WebLog [23] is a deductive language operating in an integrated graph-based framework. Although its syntax resembles F-Logic, it is not fully object-oriented; the only objects are *"rel-infons"* which simulate in some sense the nodes of the parse-tree. There is no reported implementation of WebLog. The idea of UnQL [6] can also be seen as an integrated approach, using a graph-based model for semistructured data and defining a language for navigating and querying this model.

Most recent prototypes for information extraction follow the layered approach, using separate programs for wrapping (often, an individual wrapper is designed for every source) and mediating.

In the STRUDEL [10] system, the Web is also mapped to a graph representation. Here, all relevant pages must be accessed at the same time before the querying (and reasoning) phase is started. In the TSIMMIS project [14], the graph-based *OEM* (Object Exchange Model) is used as a common data model for the extracted, application-level information. The Web structure and the page markup is not modeled. Instead, every Web source is mapped to OEM by a wrapper. Different languages are used for wrapper and mediator specification, and for querying. [16] describes a grammar-based tool for coding wrappers producing OEM output. In the ARIADNE project, e.g., [3], the Web is also not modeled explicitly. Instead, for every relevant Web page, a grammar for wrapping it directly into the target model is derived semi-automatically. Their approach does not support HTML tables or any textual formatting. As a non-graph based approach, the ARANEUS project [4] uses a hypertext-based model. Different languages are used for extracting data and defining views on it. A hierarchical graph model

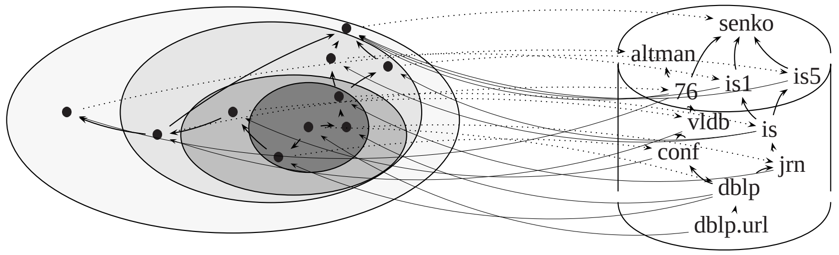

Figure 5. Exploring the Web by iteratively following hyperlinks

of the Web, similar to our approach, consisting of the sites, Web pages, and intra-page structuring, is presented in [21]; an extension of W3QS/W3QL [20] to this model is planned. Also, WebOQL [2] makes parse-trees first-class citizens of the model which can be queried directly. W3QL and WebOQL use tags in SQL-style SELECT clauses – unlike FLORID, which uses rules and path expressions for parse-tree navigation. The YATL language of the YAT system [7] is a rule-based querying and transformation language for XML/SGML wich is also based on a tree representation. Here, generic trees can be instantiated to be used in mediator programs. The Web structure is not modeled, also the relevant part of the Web cannot be extended at run-time. An early approach for using F-Logic for data integration has been reported in [24].

Jedi [18] is a tool for manually specifying Web access and wrappers for HTML pages by a framework combining grammars and rules. W4F [30] is a toolkit for interactively generating wrappers for HTML pages using HEL, a DOM-based language operating on the parse-tree of a document. Comma-lists and name-value pairs are regarded as atomic – i.e., the result is not completely mapped to the application domain, requiring a manually specified application-specific postprocessing. HEL focusses on wrapping single Web pages; Web exploration and information integration is left to the application. Another tool for interactive generation of wrappers using the DOM model is presented in *NoDoSE* [1]. [22] and [9] present methods for automatical wrapper generation.

Most of the above models focus either on the *Web representation* of information (UnQL, STRUDEL), neglecting the application semantics of the information, or on Web access and wrapping only (Jedi, W4F), or on mediation (TSIMMIS/OEM), then leaving the mapping from the Web representation to the data model to external wrappers. The above approaches do not deal with Web exploration. For a complete overview of existing systems, see [11].

Information brokering using several sources is investigated in *InfoSleuth* [5], *Information Manifold* [25], and *Observer* [29]. In all these approaches, it is assumed that site descriptions and appropriate wrappers are given for each accessible data source – i.e., they cannot be used for exploring the Web itself. Also, the Web structure is not part of their world model.

8. Conclusion

In the common layered wrapper-mediator-architecture, each wrapper is associated with a data source. Against the mediator, it provides an interface for exchanging application-level information. The Web structure (both the Web skeleton, and the parse-trees of individual Web pages) is *not* part of the world model of the systems. The network structure of the wrappers and mediators has to be designed a priori for the relevant Web fragment – which thus has to be known a-priori.

Instead, in our approach with an integrated system incorporating a generic Web interface, the Web can be seen as a part of the system, i.e., arbitrary Web pages can be decided to be accessed at runtime. Using an integrated internal world model including the Web structure, there is no gap between wrappers and mediators. Functionality for Web exploration, wrapping, and mediating functionality can be combined to operate on the Web skeleton, its hyperlinks, the page HTML/XML sources and their parsetrees, and on the application-level model itself.

Especially, in addition to the classical wrapping and mediating functionality, a system with this architecture can be equipped with *Web navigation* and *exploration* functionality. Web pages not known a priori can be detected to contain potentially relevant information. Since wrappers can be written on a generic, markup level, such pages can also be processed using suitable heuristics: The generic wrapping schemata form a universal wrapper which can be applied to arbitrary Web pages when they become known to the system during information processing. This can be interpreted as a dynamical generation and configuration of *agents* inside the main system: the wrapper functionality which applies to an information source automagically forms the (wrapper) agent for wrapping this source. Equipped with suitably intelligent internal logic, the system – with its virtual agents – can potentially explore before unknown parts of the Web, thus coping with the steady growth of the Web.

With this, the architecture allows also for implementing data-driven, hyperlink-oriented Web search engines which also can extract and process semantical information.

We presented an object-oriented deductive implementation of the architecture. Alternatively, implementations based on a functional language could be possible, or an implementation using XML as internal data format together with a powerful XML-based data manipulation language.

Acknowledgements. I want to thank G. Lausen and the former team members R. Himmeröder, P.-T. Kandzia, B. Ludäscher, M. Seilnacht, and C. Schlepphorst for the fruitful cooperation in the project.

References

[1] B. Adelberg. Nodose – a tool for semi-automatically extracting semi-structured data from text documents. In *ACM Intl. Conf. on Management of Data (SIGMOD)*, 1998.

[2] G. O. Arocena and A. O. Mendelzon. WebOQL: Restructureing Documents, Databases, and Webs. In *Intl. Conf. on Data Engineering (ICDE)*, 1998.

[3] N. Ashish and C. A. Knoblock. Wrapper generation for semi-structured internet sources. *SIGMOD Record*, 26(4):8–15, 1997.

[4] P. Atzeni, G. Mecca, and P. Merialdo. To weave the web. In *Intl. Conf. on Very Large Data Bases (VLDB), 1997*.

[5] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk:. Infosleuth: Agent-based semantic integration of information in open and dynamic environments. In *ACM Intl. Conf. on Management of Data (SIGMOD)*, 1997.

[6] P. Buneman, S. Davidson, G. Hillebrandt, and D. Suciu. A query language and optimization techniques for unstructured data. In *ACM Intl. Conf. on Management of Data (SIGMOD)*, pages 505–516, 1996.

[7] S. CLuet, C. Delobel, J. Siméon, and K. Smaga. Your Mediators need data Conversion. In *ACM Intl. Conf. on Management of Data (SIGMOD)*, 1999.

[8] DBLP computer science bibliography. `http://dblp.uni-trier.de`, 1998.

[9] D. Embley, D. Campbell, Y. Yiang, Y.-K. Ng, and R. Smith. A conceptual-modeling approach to extracting data from the web. In *Intl. Conf. on the Entity Relationship Approach (ER)*, Springer LNCS 1507, 1998.

[10] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy, and D. Suciu. Catching the boat with strudel: Experiences with a web-site management system. In *ACM Intl. Conf. on Management of Data (SIGMOD)*, 1998.

[11] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world-wide web: A survey. *SIGMOD Record*, 27(3), 1998.

[12] J. Frohn, R. Himmeröder, P.-T. Kandzia, G. Lausen, and C. Schlepphorst. FLORID: A prototype for F–Logic. *Intl. Conf. on Data Engineering (ICDE)*, 1997.

[13] J. Frohn, G. Lausen, and H. Uphoff. Access to objects by path expressions and rules. In *Intl. Conf. on Very Large Data Bases (VLDB)*, pages 273–284, 1994.

[14] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. 8(2), 1997.

[15] M. R. Genesereth. Knowledge interchange format. *Intl. Conf. on Principles of Knowledge Representation and Reasoning*, 1991. Morgan Kaufmann.

[16] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semistructured information from the web. In *Intl. Workshop on the Web and Databases (WebDB)*, pages 18–25, 1997.

[17] R. Himmeröder, P. Kandzia, B. Ludäscher, W. May, and G. Lausen. Search, analysis, and integration of web documents: A case study with FLORID. *Intl. Workshop on Deductive Databases and Logic Programming*, 1998.

[18] G. Huck, P. Fankhauser, K. Aberer, and E. J. Neuhold. Jedi: Extracting and synthesizing information from the web. In *Intl. Conf. on Cooperative Information Systems (CoopIS)*. IEEE Computer Science Press, 1998.

[19] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[20] D. Konopnicki and O. Shmueli. W3QS: A Query System for the World-Wide Web. In *Intl. Conf. on Very Large Data Bases (VLDB)*, pages 54–65, 1995.

[21] D. Konopnicki and O. Shmueli. Bringing database functionality to the www. In *Intl. Workshop on the Web and Databases (WebDB)*, pages 49–55, 1998.

[22] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. 1997.

[23] L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian. A Declarative Language for Querying and Restructuring the Web. In *Intl. Workshop on Research Issues in Data Engineering (RIDE)*, 1996.

[24] A. Lefebvre, P. Bernus, R. Topor. Querying Heterogeneous Databases: A Case Study. In *Australasian Database Conf. (ADC)*, 1993.

[25] A. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 2(5), 1995.

[26] B. Ludäscher, R. Himmeröder, G. Lausen, W. May, and C. Schlepphorst. Managing semistructured data with FLORID: A deductive object-oriented perspective. *Information Systems*, 23(8):589–612, 1998.

[27] W. May. Modeling and querying structure and contents of the web. In *Workshop on Internet Data Modeling (IDM'99), DEXA 99 Workshops*. IEEE Computer Science Press, 1999.

[28] W. May, R. Himmeröder, G. Lausen, and B. Ludäscher. A unified framework for wrapping, mediating and restructuring information from the web. In *Intl. Workshop on the World-Wide Web and Conceptual Modeling (WWWCM)*. To appear in Springer LNCS, 1999.

[29] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *Intl. Conf. on Cooperative Information Systems (CoopIS `96)*, 1996.

[30] A. Sahuguet and F. Azavant. Looking at the web through xml glasses. In *Intl. Conf. on Cooperative Information Systems (CoopIS)*. IEEE Computer Science Press, 1999.

[31] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 1992.