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

Wolfgang May
Institut für Informatik
Universität Freiburg
Germany
may@informatik.uni-freiburg.de

ADC 2000, Canberra, Australia
Jan. 31th, 2000

Overview

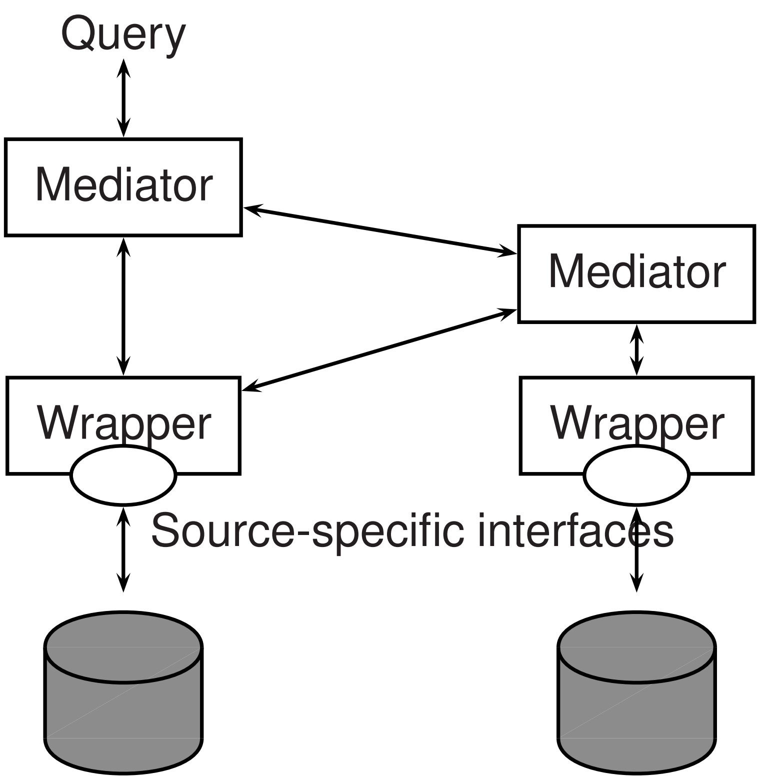
- Integrated Architecture for
  - Web querying
  - Web Data Extraction
- Unified World Model: Web structure + application level
- Unified Language for Exploration, Wrapping, Mediating, Querying
- Implementation: F-Logic/FLORID
- Examples / Case Studies:
  - DBLP Publications Web Server (single-site)
  - Geographical Information (multi-site)

Architectures

Early Web Querying Systems

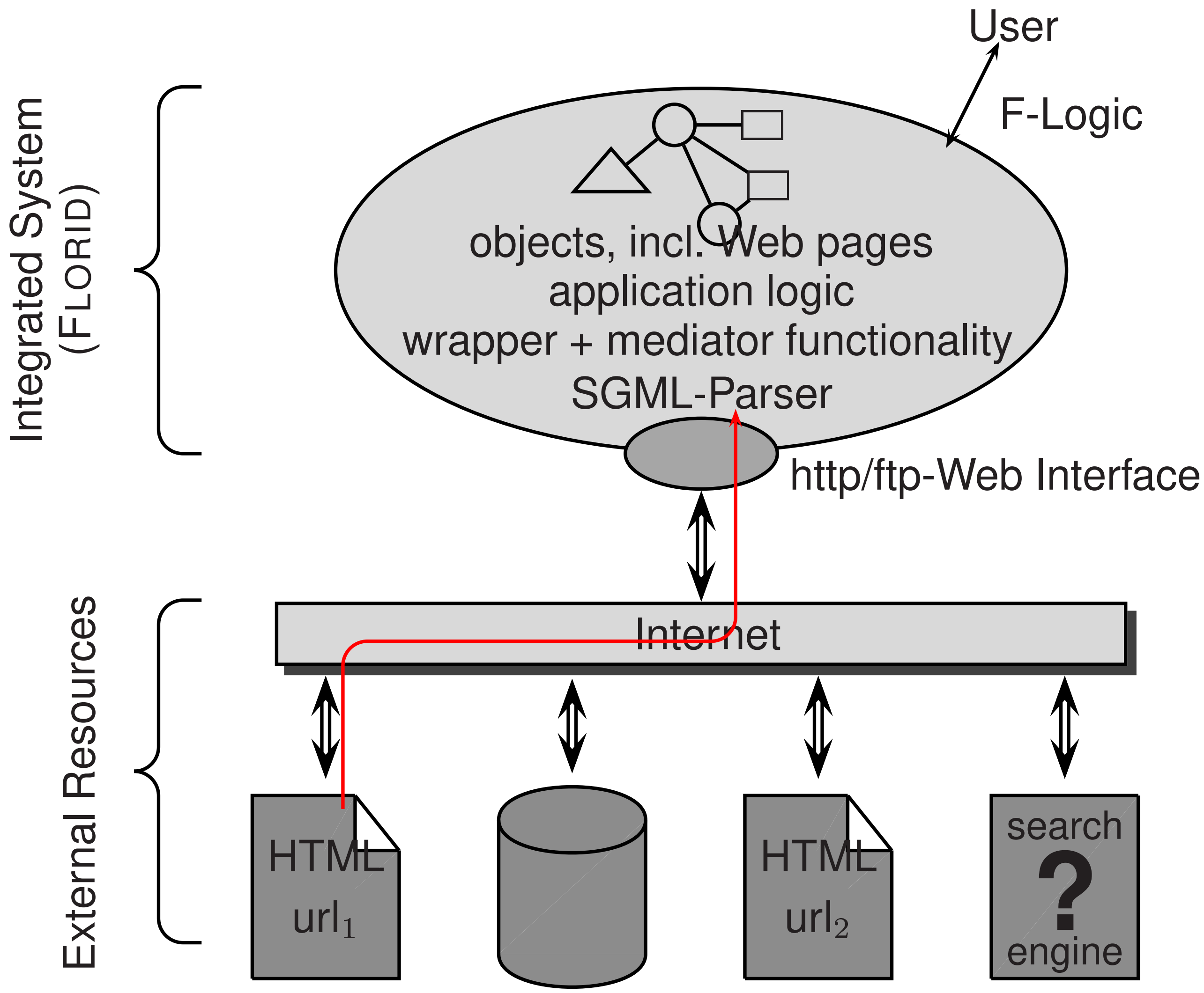
- Generic Interfaces to the Web
- Restricted internal capabilities (querying only)

Two-Level Wrapper & Mediator – Architecture



- different languages and data models for wrapping/mediating/querying
- predefined wrappers → sources known a priori

Architecture of Web Access with FLORID



- Unified, integrated framework for wrappers and mediators
- F-Logic: unified data model, wrapper, mediator, and querying language
- Data Model: Representation of the Web fragment and application-level representation.
  Structure + Contents of the Web as a unit

Web Querying and Integration functionality

- Internal World Model contains information about Web structure (*Web skeleton*):
  - autonomous exploration of the Web via generic HTTP-interface
  - Use of Web Search engines (AltaVista)
- Internal World Model contains a representation of Web contents:
  - Web Structure representation: wrapping in the internal model
  - Web Contents representation: mediation/integration based on the internal model (all data sources)
  - Querying the internal model both on structure and contents level

The Web Model

- unified object-oriented model in F-Logic
  - the Web (carrier of information)
  - the application domain (carried information)
- graph-based model
- inter-document level
  topology of the Web ((Web) skeleton):
  - nodes: Web documents,
  - (labeled) edges: hyperlinks between documents.
  - skeleton: no information apart from the link structure is available;
- intra-document level
  The *page markup* (tags):
  induces a tree structure of the page contents.
- Web skeleton and parse-trees: application-independent
- an object-oriented model of the application domain.

The Skeleton $\mathcal{SK}$: URL's and Web Documents

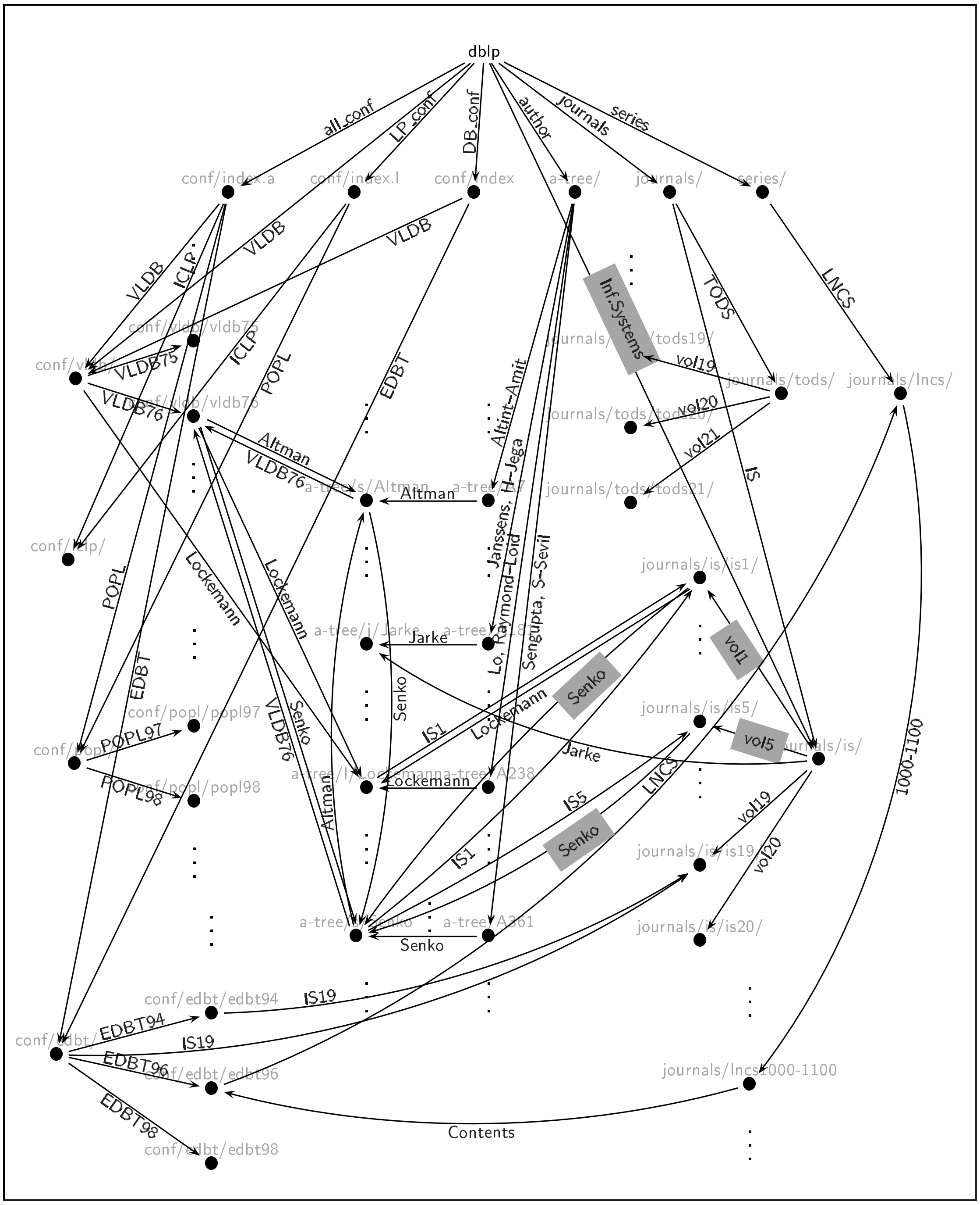
- Every resource in the Web has a unique *url*.
- document associated with a url contains *hyperlinks* to other url's

$$(x, \ell, y) \in \mathcal{SK} \Leftrightarrow \text{the Web document } x \text{ contains a hyperlink labeled with } \ell \text{ to the Web document } y.$$

(“$<\texttt{a href} = y > \ell </\texttt{a}>$”)

Example: The DBLP Server

Example: The DBLP Server

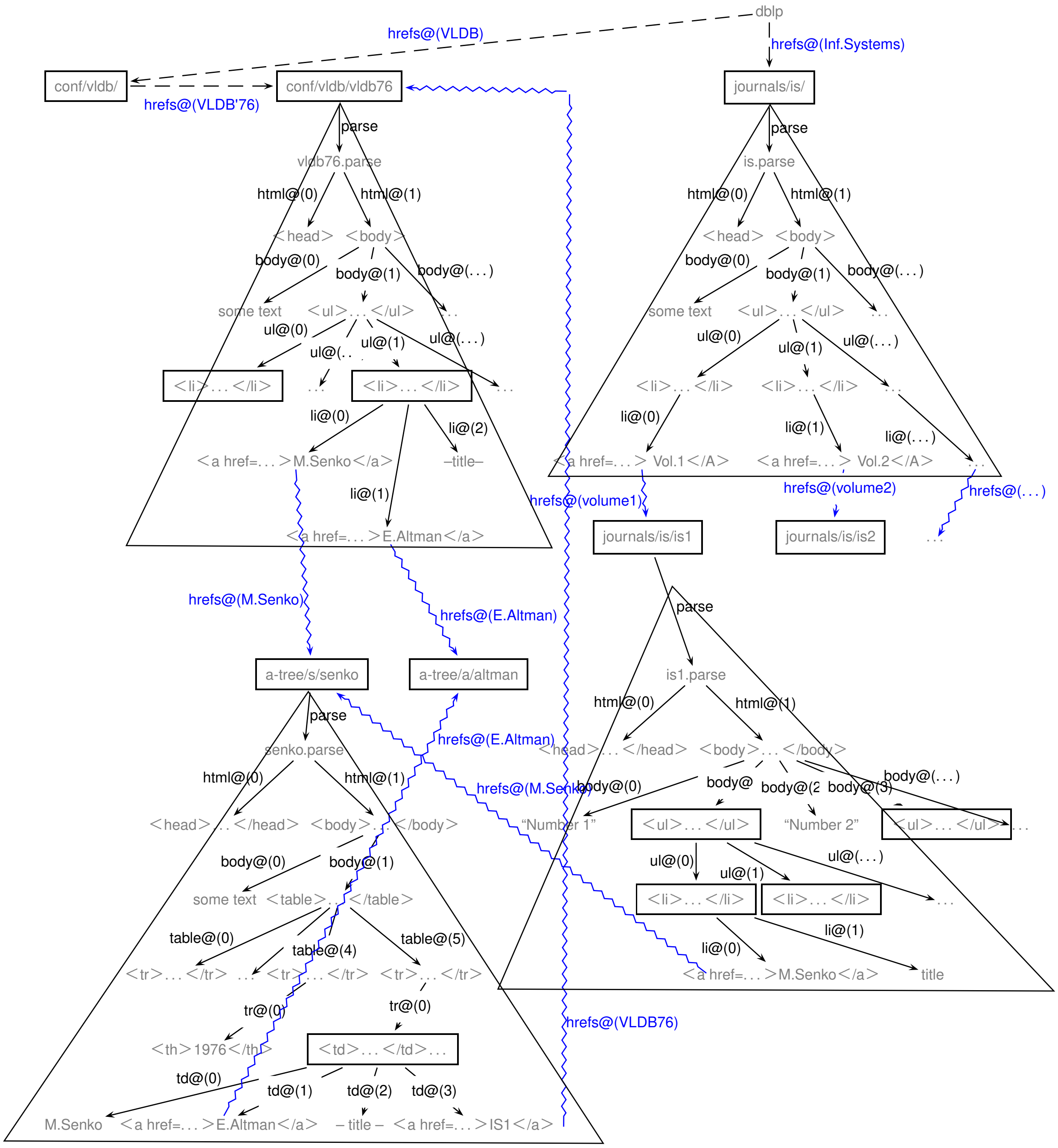


- skeleton: Web pages and hyperlinks
- corresponds to real world objects: journals, conferences, books, and authors

Extending the Web Skeleton: Parse-trees

- real-world objects are represented as individual Web pages, or by substructures.
- integration of *parse-trees*

Example: Extended Web Skeleton of DBLP



Extended Web Skeleton

- *extended Web skeleton*: unified – but still based on the Web representation, not on the application semantics.

$\Rightarrow$ base for (structural) *Web querying*

- (semantical) Web data extraction:
  many objects have already a direct counterpart in the extended Web skeleton.
  - objects have a Web representation as a Web page (referencable via url).
  - objects correspond to nodes in a parse-tree (journal volumes and papers).
    (referencable in HTML via page#anchor).
  - counterparts in several parsetrees
    $\Rightarrow$ Object fusion:
    Objects as objects in the Web representation *and* in the application model.
  - mapping between nodes/arcs of the extended Web skeleton and instances of distinguished classes/relationships of the application modeling.
- XML?
  Parse-tree $\equiv$ application-semantic model?

Requirements

- Logical issues:
- Suitable modeling (classes)
- Object creation on demand
- Object fusion
- Navigation in the model
- Powerful, flexible reasoning

- non-logical features ⇒ built-ins:
- Web Access via http-protocol,
- Parsing of HTML/SGML/XML,
- Matching with Perl Regular Expressions,

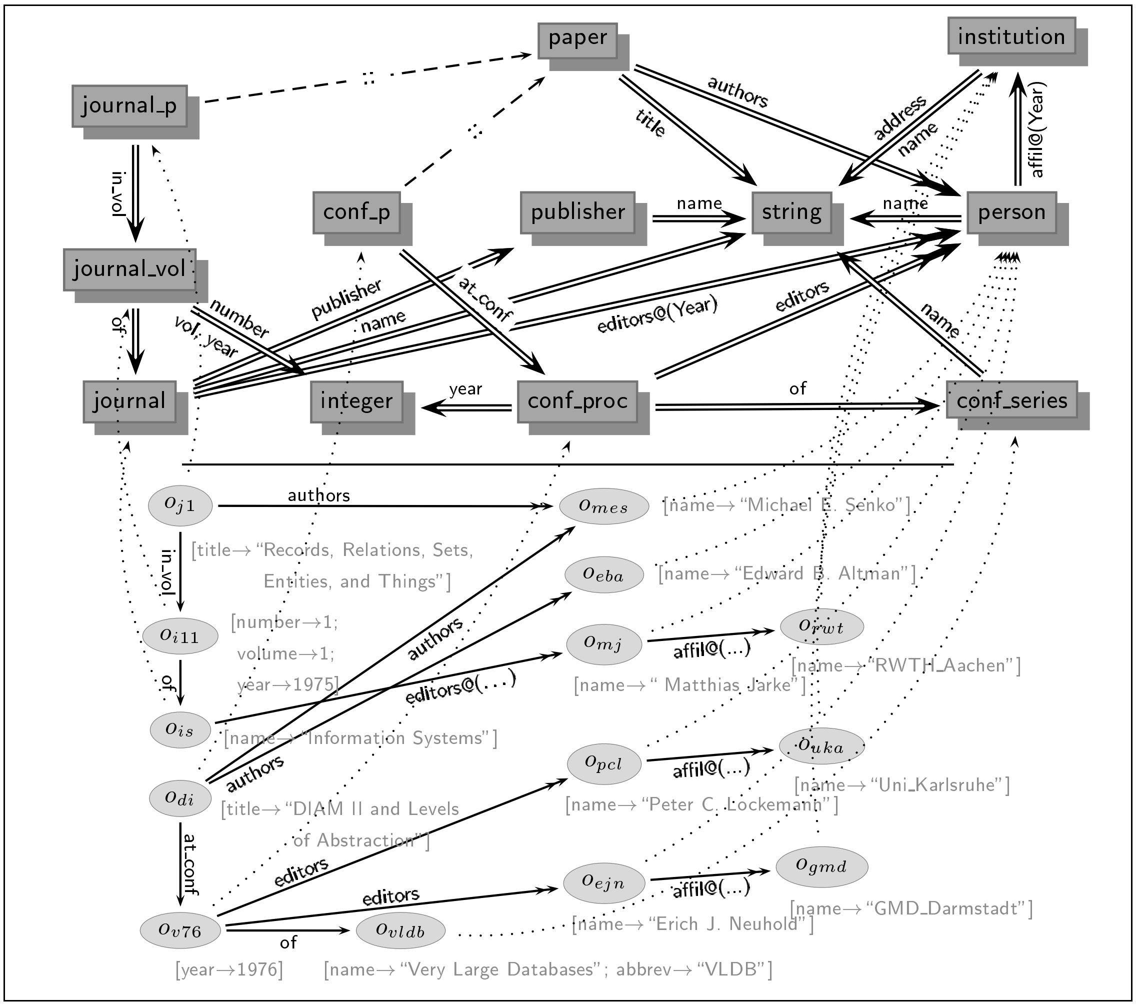
Choice: Deductive Object-Oriented Database Language

- rule-based: rapid prototyping, declarativity
- ideas from active databases (for Web Access)

Formal Framework: F-Logic

- *id-terms* are composed from object constructors and variables as usual.
- is-a atoms: $o : c$
- subclass atoms: $c :: d$
- Method applications to objects:
  $o[m \rightarrow v]$ (scalar)
  $o[m \twoheadrightarrow v]$ (multivalued)
  analogous with arguments: $o[m@(x_1, \ldots, x_n) \rightarrow v]$.
  inheritable:
  $c[m \bullet\!\!\rightarrow v]$
  $c[m \bullet\!\!\twoheadrightarrow v]$
- Signatures of methods:
  $c[m \Rightarrow v]$ (scalar)
  $c[m \Rightarrow\!\!\!\!\Rightarrow v]$ (multivalued)
- Variables allowed at all positions
- Entities can act at the same time as classes, objects and methods
- Rules over atoms: <head> :- <body>.
- Program: a set of rules

Example: F-Logic Model of DBLP



journal_p :: paper.   conf_p :: paper.

paper[title⇒string; authors⇒⇒person].

journal_p[in_vol⇒journal_volume].

conf_p[at_conf⇒conf_proc].

o_{j1} : journal_p[title→"Records, Relations, Sets, Entities, and Things"; authors↠{o_{mes}}; in_vol→o_{i11}].

o_{di} : conf_p[title→"DIAM II and Levels of Abstraction"; authors↠{o_{mes}, o_{eba}}; at_conf→o_{v76}].

o_{i11} : journal_vol[of→o_{is}; number→1; volume→1; year→1975].

o_{is} : journal[name→"Information Systems"; editors@(...)↠{o_{mj}}].

o_{v76} : conf_proc[of→o_{vldb}; year→1976; editors↠{o_{pcl}, o_{ejn}}].

o_{vldb} : conf_series[name→"Very Large Databases"].

o_{mes} : person[name→"Michael E. Senko"]. o_{mj} : person[name→"Matthias Jarke"; affil@(...)→o_{rwt}].

o_{rwt} : institution[name→"RWTH_Aachen"].   ...

Further F-Logic Features

- path expressions:
  $[\![o.m]\!]$: that o' s.t. o[m $\rightarrow$ o']
  $[\![o..m]\!]$: all o' s.t. o[m $\twoheadrightarrow$ o']

  ?- P : conf_proc.of[abbrev→"VLDB"], P[year→1976],
  P..editors[affil@(1976)→A].

- object creation by path expressions in the head:
  o.m[. . .] $\leftarrow$ . . .

- Derived equality via object fusion:
  $o_1 = o_2 \leftarrow \ldots$

- Aggregates: sum, count

- nonmonotonic inheritance, equivalent to Default Logic.

- FLORID: bottom-up inflationary semantics with user-defined stratification
  (deductive + procedural elements)

Exploration of the Web

- classes url and webdoc
  (subclasses of string).
- class url: url.get implemented as an *active* method (C++):

  u.get :- . . .

  - accesses the Web document which is accessible via u
  - assigns it to u.get (object creation)
  - becomes an instance of class webdoc
  - and several properties are automatically filled in.

u.get[hrefs@(ℓ) $\twoheadrightarrow$ u'] $\Leftrightarrow$

u.get contains "<a href $= u' >$ ℓ " .

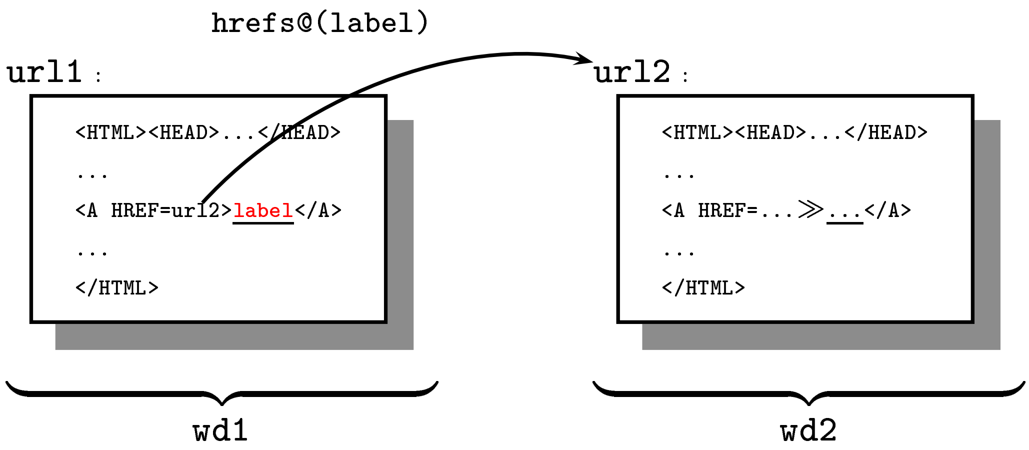
```
url::string[get⇒webdoc].

webdoc[url ⇒url; author ⇒string;
         type ⇒string; hrefs@(string) ⇒⇒url;
         modif ⇒string; error ⇒⇒string].
---------------------------------------------------
url1[get→wd1].        url2[get→wd2]
wd1:webdoc[url→url1; hrefs@("label")→→{url2; ...}]
wd2:webdoc[url→url2; type→html; ...]
```

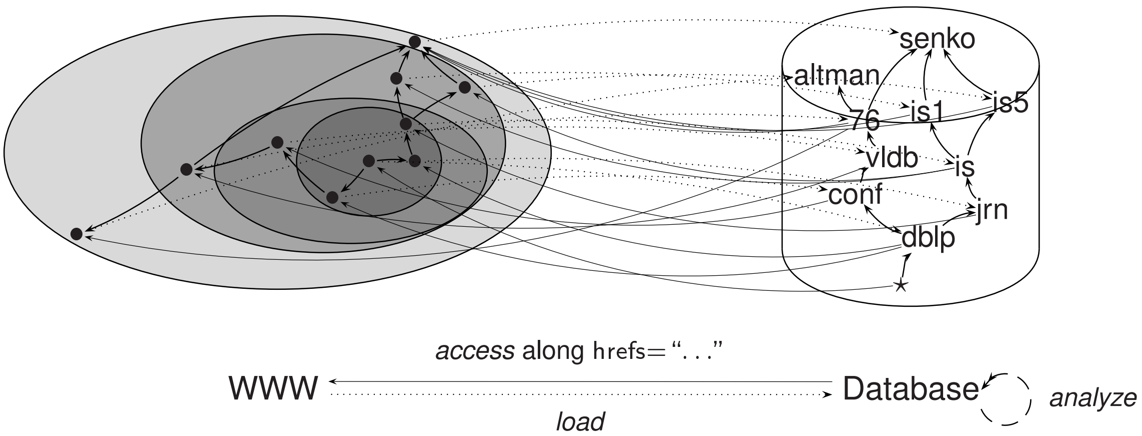
Data-Driven Web Exploration

- in course of the information extraction and restructuring process, additional pages are recognized to be relevant:

  U.get ← A:author[homepage→U].



- approach implements a hybrid concept by embedding *data-driven wrapping* into a *warehouse* approach



Parsing of Web Pages

- url.parse generates F-Logic representation of the parse-tree,
  assigns it to the object u.parse : parsetree
- *SGML-tagged* groups $<tag> \ldots </tag>$ become objects,
- classes $webdoc.<tag>$,
- navigation: $o.<tag>$@(0), $\ldots$, $o.<tag>$@(n)
  are the segments inside $o.<tag>$
- attributes: $o.<attr>$

Wrapping

- url.get, url.parse: *raw*, uninterpreted data

$\Rightarrow$ (Extended) Web Skeleton
allows for simple syntax-based and structure-based Web querying

$+$ semantic wrapping by F-Logic Rules

- Logical Markup:
  Parser-based
  DBLP-server: sufficiently well-structured HTML

- direct correspondence between HTML-nodes and objects (extended Web skeleton).

- Optical and Syntactical Markup:
  pattern matching via regular expressions

- construction of object model from scratch/identifying new objects

Internal Document Analysis

- SGML/HTML Parsing
- not all Web pages provide logical markup
- well-structured pages need further wrapping:
  - keywords,
  - commalists,
  - text search for relevant words

Pattern Matching in FLORID

Perl regular expressions by the built-in predicate

pmatch(<string>,“/<regexp>/”, [<fmt-list>], [X_1,. . . , X_n])

Generic Wrapping Tasks

Extracting contents of the pages:

- Logical Markup
  - HTML-Lists
  - HTML-Tables: Headers, Columns
- Optical Markup
  - Paragraphs
  - Boldfacing, Emphasizing
- Syntactical Markup:
  - Commalists, Semicolons, Parentheses

$\Rightarrow$ Generic Rules for these tasks

+ Rules for application-specific interpretation
+ Refined rules dealing with exceptions

- Modular Design

Finetuning: Perl-Interface

- String Manipulation

perl(<perl-routine-as-string>,
    [<variable-list$_1$>],
    [<variable-list$_2$>])

Input: <variable-list$_1$>
Output: <variable-list$_2$>

Also used for

- Output Redirection
- Prototyping of built-ins (e.g., Aggregates)

Case Study: The MONDIAL Geographic Database

- CIA World Factbook: Countries
- CIA World Factbook: Organizations
- GlobalStatistics: Cities and Administrative Divisions
- ... some additional smaller sources

⇒ Integration

- different layouts of data sources
- not “best-case”, but “real-world”

Querying Forms

- Search engines, HTML interfaces to database Web servers (e.g., Oracle)

- uses CGI scripts via <form>-tag
- query transmission:
  - GET method: customized url with parameters
    $\Rightarrow$ feasible inside the current framework
  - POST method: parameters are received by the cgi script via standard input (perl-www-library).
- Data-driven Web exploration in combination with Web search engines:
- Refining Search Results from Web search engines,
- Combining the results of different Web search engines,
- Data-Driven queries on Web Indexes:
  query Web indexes depending on previously extracted data.

Conclusion

- integrated architecture,
- unified model for Web representation and model of the application,
- integrated data model/language for wrapping, mediating, and querying.
- advantages of rule-based languages:
  - Prototyping, Refinement, Adaptability (autonomous sources)
  - Generic rules (logical variables)
- Further Work:
  - XML