Search, Analysis, and Integration of Web Documents: A Case Study with *FLORID*

Rainer Himmeröder Paul-Th. Kandzia Bertram Ludäscher Wolfgang May Georg Lausen

Institut für Informatik, Universität Freiburg, Germany

Overview

- Introduction/Motivation
- *FLORID* Web model
- Integration: CIA WORLD FACTBOOK and WORLD ONLINE
- (Semistructured Data)
- Conclusions

Goal: A uniform framework for

- *Querying the Web:*
  - express declaratively how to query/navigate on the Web
  - extract data from Web pages for populating a database (*Web-data warehousing*)
- *Management of Semistructured Data:*
  - structure is irregular, partial, unknown, implicit in the data
  - example: HTML pages
  - querying/navigation using *general path expressions*
  - discover structure
- *Information Integration:*
  - heterogeneous sources with different structure
  - wrappers, mediators

DOOD Paradigm:

- *deduction*: for data-driven exploration of the Web and high level querying
- *object-orientation*: for flexible modeling of semistructured data (optional methods instead of NULLs)

***Web-FLORID*:** extension of *F-logic* for querying and restructuring the Web:

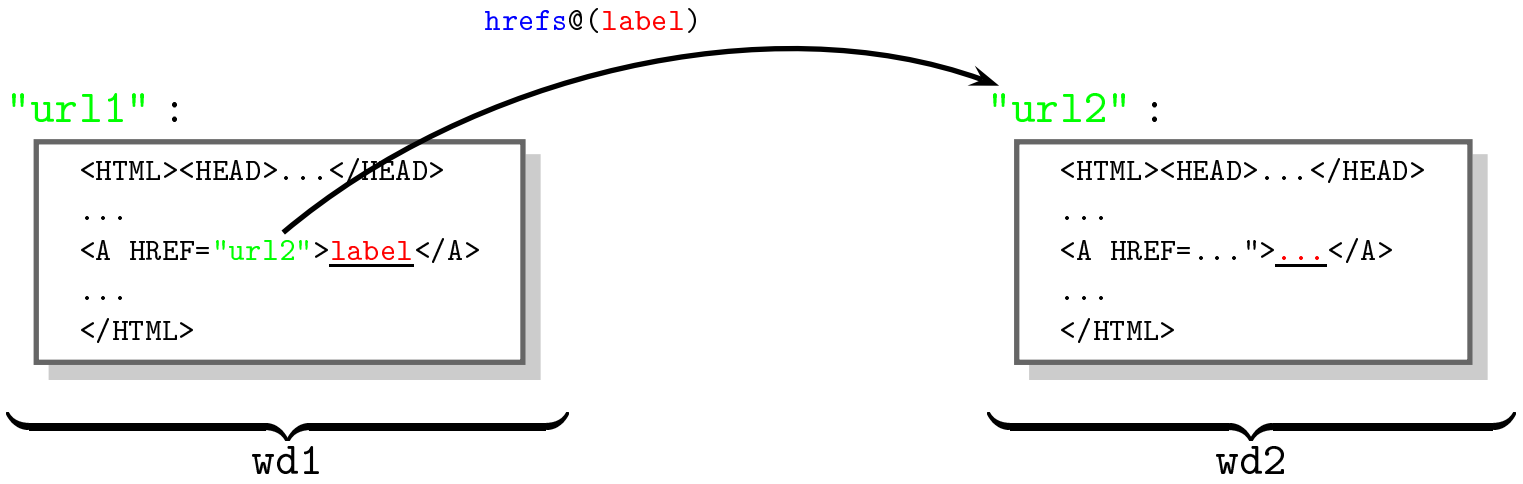
- declarative rule-based programming style: uniform language for wrappers & mediators
- meta features: schema browsing/reasoning, variables at class/method positions
- restructuring of information
- navigation by (general) path expressions
- uniform access to local db & Web data $\Rightarrow$ integration of heterogenous information

Basic Constructs:

```
Object:Class                                  % ISA-relation, "∈"
SubClass::Class                               % SUBCLASS-relation "⊆"

Class[Method@(P-types) => R-type]             % SIGNATURE: single-valued
Class[Method@(P-types) =>> R-types]           % ... and multi-valued

Object[Method@(Params) -> R]                  % DATA: single-valued
Object[Method@(Params) ->> {R1,R2}]           % ... and multi-valued

Obj.M1@(P1)[Spec1]..M2@(P2)[Spec2]            % PATH EXPRESSION
```

Object Creation via Path Expressions in the Head:

```
X.father:man  ←  X:person.
X.mother:woman  ←  X:person.
?-_:person.M:C.
M=father, C=man;
M=mother, C=woman
```



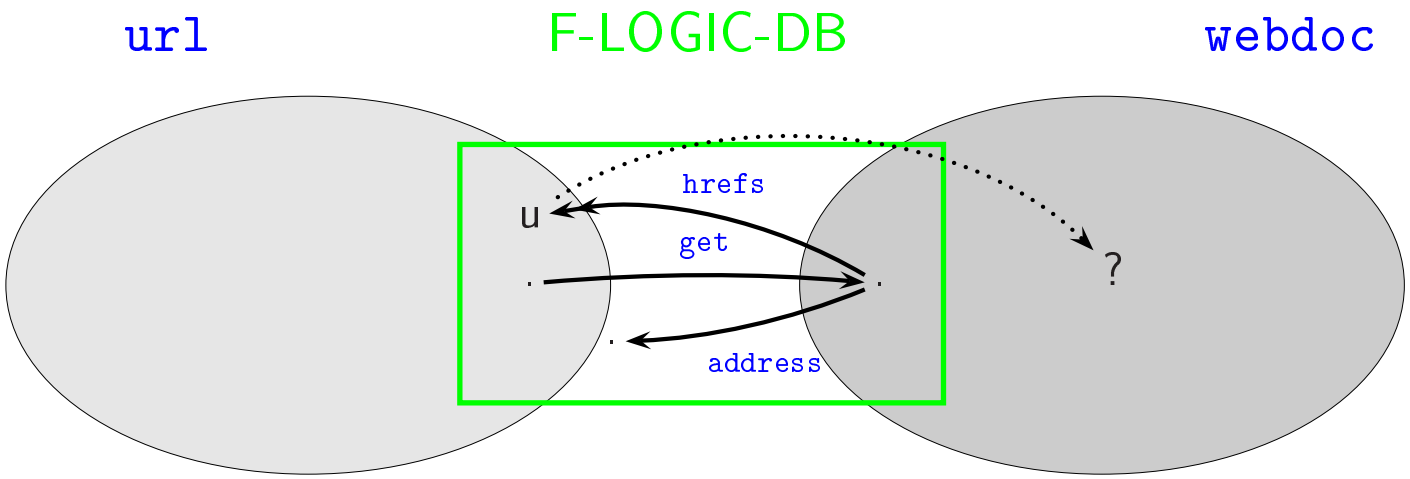
Link Structure:

Signature: `webdoc[hrefs@(string) =>> url]`

Example: `wd1:webdoc[hrefs@("label") ->> "url2"]`

Further Attributes:

```
webdoc[self => url; address => string; modif => string; ... ; error =>> string].
```




```
url::string[get => webdoc]
```

Rule-Based Exploration:

```
U.get[] ← U:url, ...                    % generate OID
⇒  U.get:webdoc                         % ... add to webdoc
⇒  U.get[address -> ...; hrefs@(...) ->> ...]   % ... fill in slots
```

```
U:explored   ← U:url.get[].
U:unexplored ← U:url, not U:explored.
```

Extension of F-logic by

- **Path Expressions** [FLU-VLDB-94]
  closure axioms $\Rightarrow$ extended Herbrand universe $\overline{U}$, Herbrand base $\overline{\mathcal{HB}}$
- **Web Interface**
  - set of *reserved names* R (`get`, `url`, `hrefs` ...)
  - *explore*: $\mathcal{URL} \rightarrow \mathfrak{P}(\overline{\mathcal{HB}(\mathcal{URL} \cup R)})$ *% maps URLs to sets of new facts*
- **Web Access Axiom**: for $H \subset \overline{\mathcal{HB}}$:

  $H \models u\texttt{:url} \wedge u\texttt{.get} \quad \Rightarrow \quad H \models new$ for all $new \in$ *explore*(u)

  "if `get` *is defined for a URL* u*, then all explored data is in H"*

$\Rightarrow$ **minimal Herbrand Web Model**

- **Integration with Bottom-up Evaluation**:

$$T_{\bar{P}}^{\mathcal{W}}(H) := H \cup T_{\bar{P}}(H) \cup \bigcup \{ \textit{explore}(u) \mid u\texttt{:url}, u\texttt{.get} \in T_{\bar{P}}(H) \}$$

$\Rightarrow$ **declarative semantics**: if *explore* $:= \emptyset$ then *Web-FLORID*$=$ *FLORID*

CIA WORLD FACTBOOK (CIA)

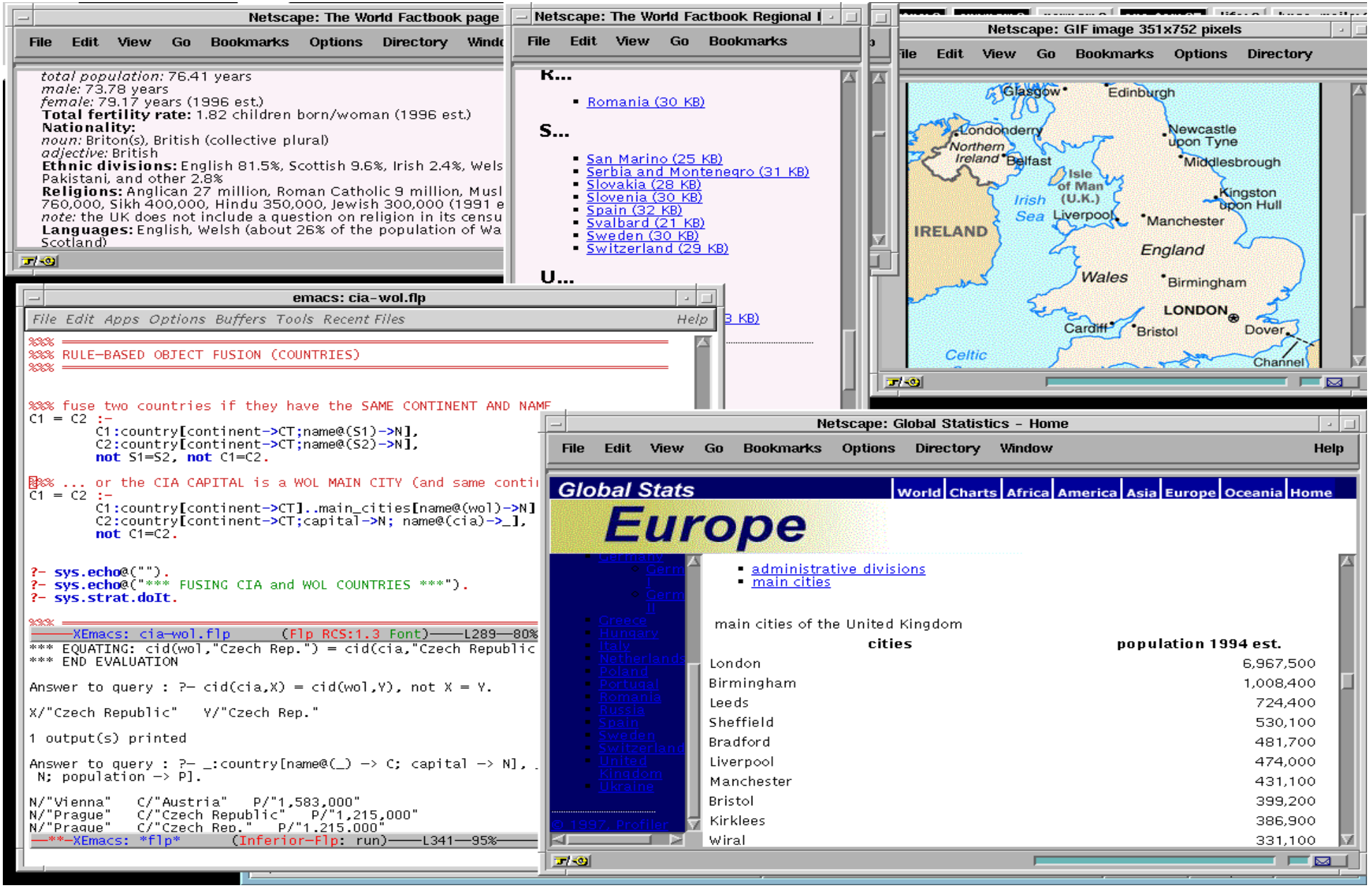
- geography, people, government, economy, ... **no cities** (apart from country capitals)
- information: link structure, **formatted text**
- very structured and regular
- complete

WORLD ONLINE (WOL)

- administrative divisions, **main cities**
- information: link structure, **tables**
- not very regular
- incomplete[1]

[1] WOL author: *"All visitors must realize that this site (i.e. collecting the data and putting it up here) is a logical development of one of my hobbies, you therefore cannot expect all data to be of academic standard. What you see is what you get, although I try to be as thorough as possible."*

EXAMPLE: INTEGRATION CIA WORLD FACTBOOK and WORLD ONLINE



```
=========================
ACCESSING RELEVANT PAGES:
=========================
C[url@(cia)->U] :- C:continent[file@(cia)->FN], strcat(cia.src,FN,U).
U:url.get :- C:continent[url@(cia)->U].

========================
EXTRACTING ‘‘RAW DATA’’:
========================
pattern(capital,"/Capital:.*\n(.*)/").
pattern(total_area,"/total area:.*\n(.*sq km)/").

C[Method -> X] :- pattern(Method, RegEx),
                  pmatch(C:country.url@(cia).get, RegEx, "$1", X).

================================
RESTRUCTURING AND DATA CLEANING:
================================
C:real_country :-  C:country[capital->CA], not substr("none", CA).

====================================
INTEGRATION OF SOURCES, OBJECT FUSION:
====================================
C1 = C2 :- C1:country[continent->CT]..main_cities[name@(wol)->N],
           C2:country[continent->CT;capital->N; name@(cia)->_], not C1=C2.
```

QUERY: *"Name the capitals (from CIA) with their population (from WOL)"*

```
?- _:country[name@(cia) -> Country; capital -> City],
   _:city[name@(wol) -> City; population -> P].

P/"1,583,000"   City/"Vienna"   Country/"Austria"
P/"1,215,000"   City/"Prague"   Country/"Czech Republic"
P/"2,152,423"   City/"Paris"   Country/"France"
P/"3,472,009"   City/"Berlin"   Country/"Germany"
P/"2,016,000"   City/"Budapest"   Country/"Hungary"
P/"3,041,101"   City/"Madrid"   Country/"Spain"
P/"711,119"   City/"Stockholm"   Country/"Sweden"
P/"134,393"   City/"Bern"   Country/"Switzerland"
P/"6,967,500"   City/"London"   Country/"United Kingdom"

9 output(s) printed
```

So far: Structure of documents known in advance $\Rightarrow$ *content-based* queries (data extraction).

However: document structure is often unknown/irregular/partial,... $\Rightarrow$ semistructured data.

Def. A **semistructured database** is a finite set of labeled edges:

$$(x, \ell, y) \in D \quad \Leftrightarrow \quad x \overset{\ell}{\text{->>}} y \quad \Leftrightarrow \quad x[\ell \text{->>} \{y\}] \ .$$

Mapping a ssdb to F-logic:

```
X:node, Y:node, L:label, X[L ->> {Y}]  ←  ssdb(X,L,Y).
```

Example: Web Skeleton Extractor

P_{ext}:

```
root[src ->> {u_1, ..., u_n}].          % define root nodes
node :: url.                            % nodes are urls
(U:node).get  ←  root[src ->> {U}].     % get root nodes
-------------------------------------------------------
Y:node, L:label, X[L ->> {Y}]  ←        % define new nodes/lables/links...
        X:node.get[hrefs@(L) ->> {Y}], φ.   % ...by following hrefs
Y.get  ←  Y:node, ψ.                    % access nodes which satisfy ψ
```

Specialization of the Skeleton Extractor for DBLP

- root[src ->> {dblp}]. dblp = "http://www.informatik.uni-trier.de/~ley/db/".
- φ = substr("trier",Y), and (*consider only* `url`*'s containing "trier"*)
- ψ = substr("/db/journals/is/",Y), (*restrict to IS journal*)

⇒ Queries with **path expressions**: `?- dblp.."Inf. Systems"..L.."Michael E. Senko".`

Def. General path expressions GPE:

- $\mathcal{L} \cup \{\text{any}\} \subseteq GPE$,
- if $M, N \in GPE$ and $n \in \mathbb{N}_0$, then the following are in GPE:
  $(M \cdot N)$, $(M|N)$, $(M)^*$, $(M)^+$, $(M)^?$, $(M)^{-1}$, $(M)^n$,
- if φ is binary relation symbol, then if$(\varphi) \in GPE$,
- if $\ell \in \mathcal{L}$ and ψ is a unary relation symbol then $\mu(\ell), \mu(\ell, \psi) \in GPE$.

⇒ specification/implementation by simple path expressions + rules

Summary

- DOOD paradigm attractive for querying and restructuring the Web
- uniform access to local db & Web data $\Rightarrow$ integration of heterogenous information
- reasoning about document structure and Web structure
- use of search engines (AltaVista)
- Implementation in  *Web-FLORID* (= Florid 2.0):
  `http://www.informatik.uni-freiburg.de/~dbis/florid/`

Outlook

- SGML parser
- Output primitives
- ...