



Datalog-related Aspects in Lixto Visual Developer

Robert Baumgartner, 17th of March 2010

Lixto empowers better decisions by searching
& aggregating information in real-time and
delivering end-to-end connectivity solutions

- **Lixto Overview and Architecture**
- **Short Live Demo Visual Developer**
- **Visual Wrapper Generation**
- **Elog Data Extraction Language**



Company Overview and Core Competence



GUTBROD

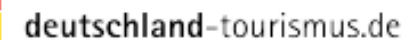
VOSS



hama



FUJITSU



Headquarters in Vienna,
Austria

Providing end-to-end solutions
in

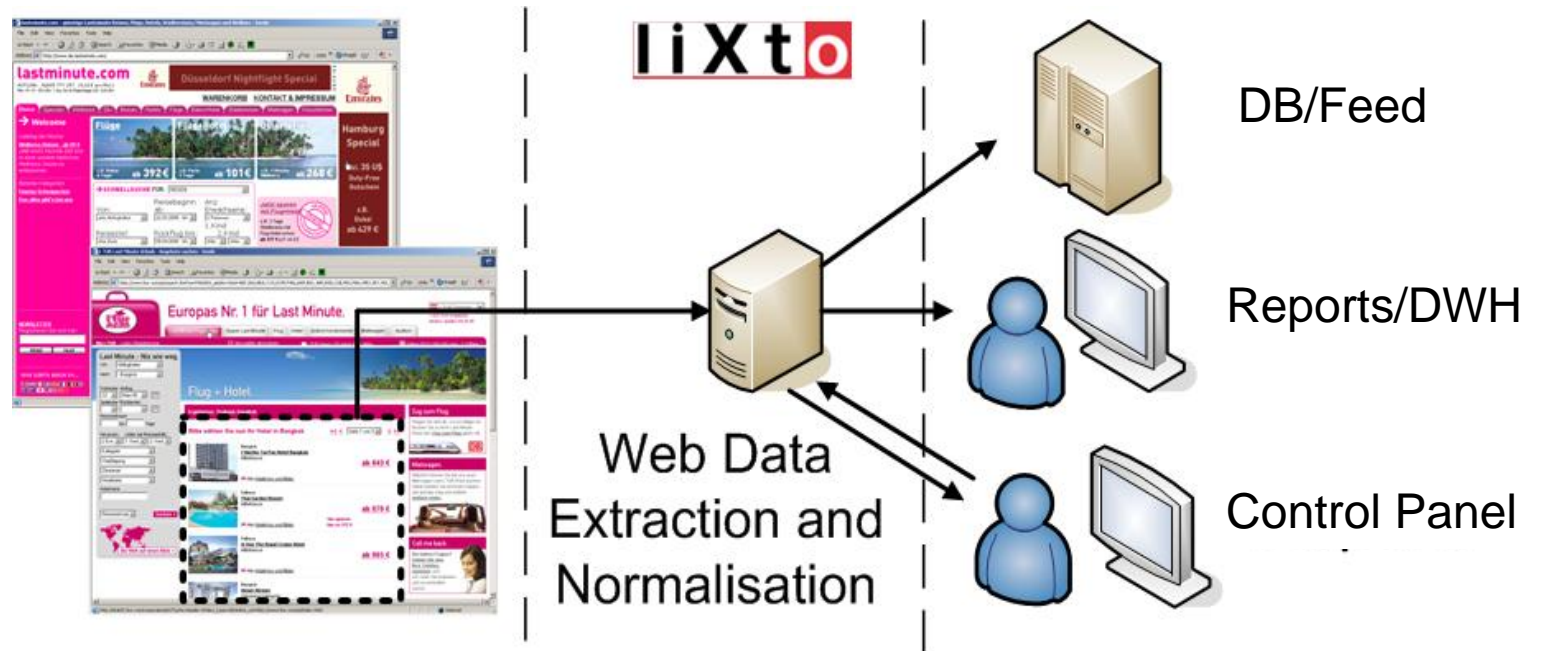
Online Market Intelligence
Web Process Integration
Meta Search

and

tools for Web Data Extraction



Lixto Solution Architecture: Bridge the Gap



Web Data Harvesting

Integration and Matching
Execution Plan

Analysis and Reporting
Consumption as Service

Visual Wrapper
Generation

Configuration
Introspection



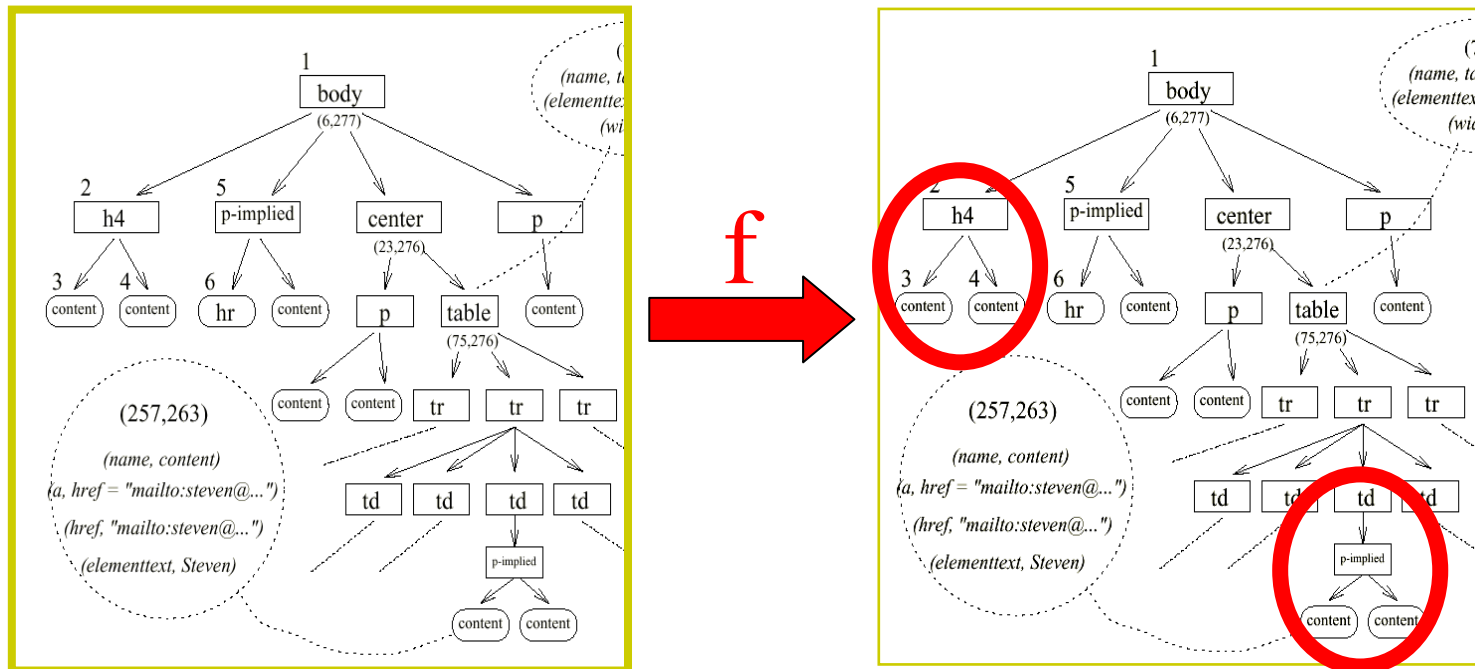
Desiderata of a Web Wrapping Language

- (i) has a solid and well understood **theoretical foundation**,
- (ii) provides a good trade-of between **complexity** and the number of practical wrappers that can be **expressed**,
- (iii) is **easy to use** as a wrapper programming language, and
- (iv) is **suitable for being incorporated into visual tools**, since ideally all constructs of a wrapping language can be realized through corresponding visual primitives.



Extraction from Tree Structure

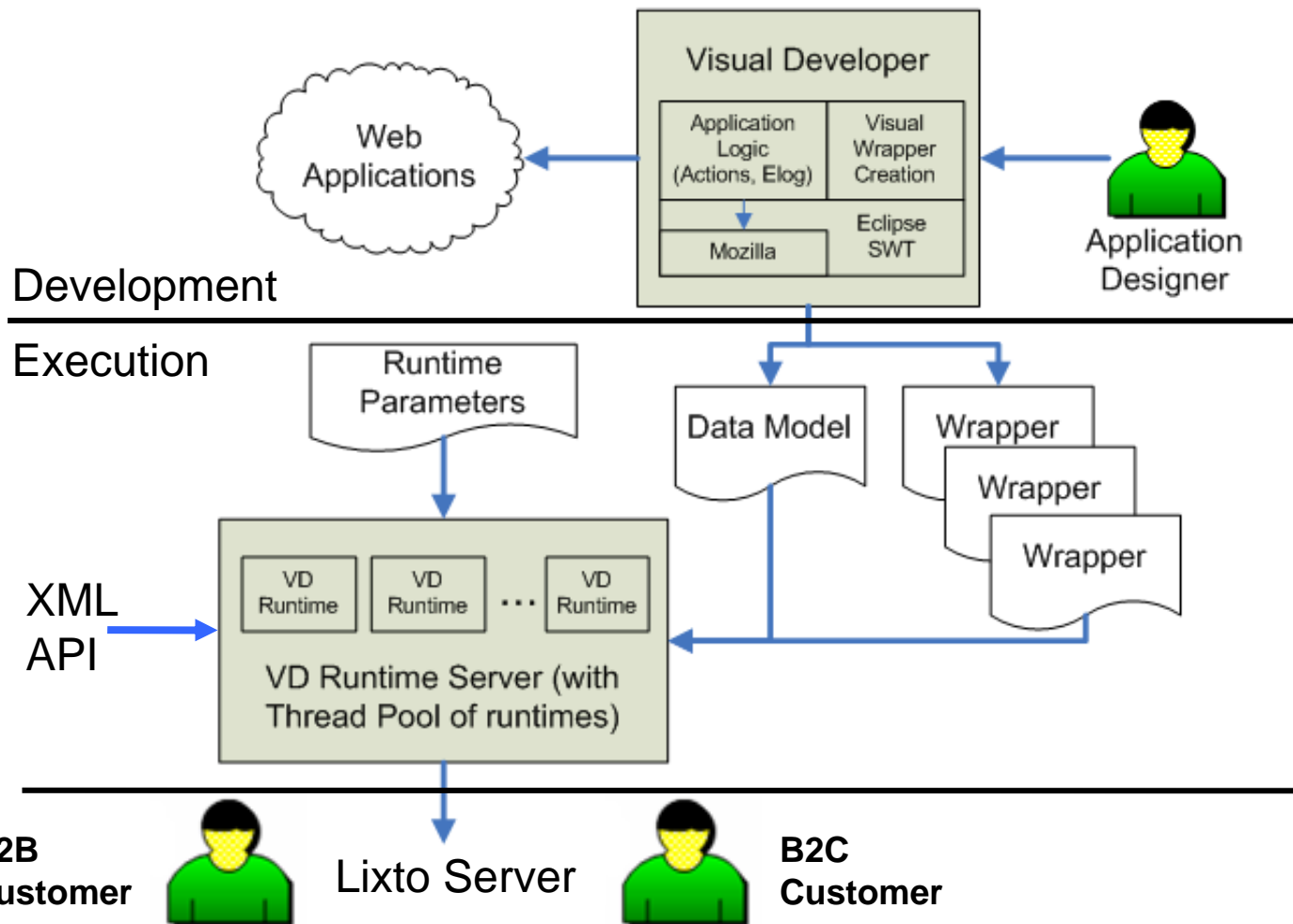
Wrapper as Function f : HTML DOM Tree \Rightarrow Subtrees
 Leaves of subtrees are among leaves of original tree



\Rightarrow „Visually“ define such functions

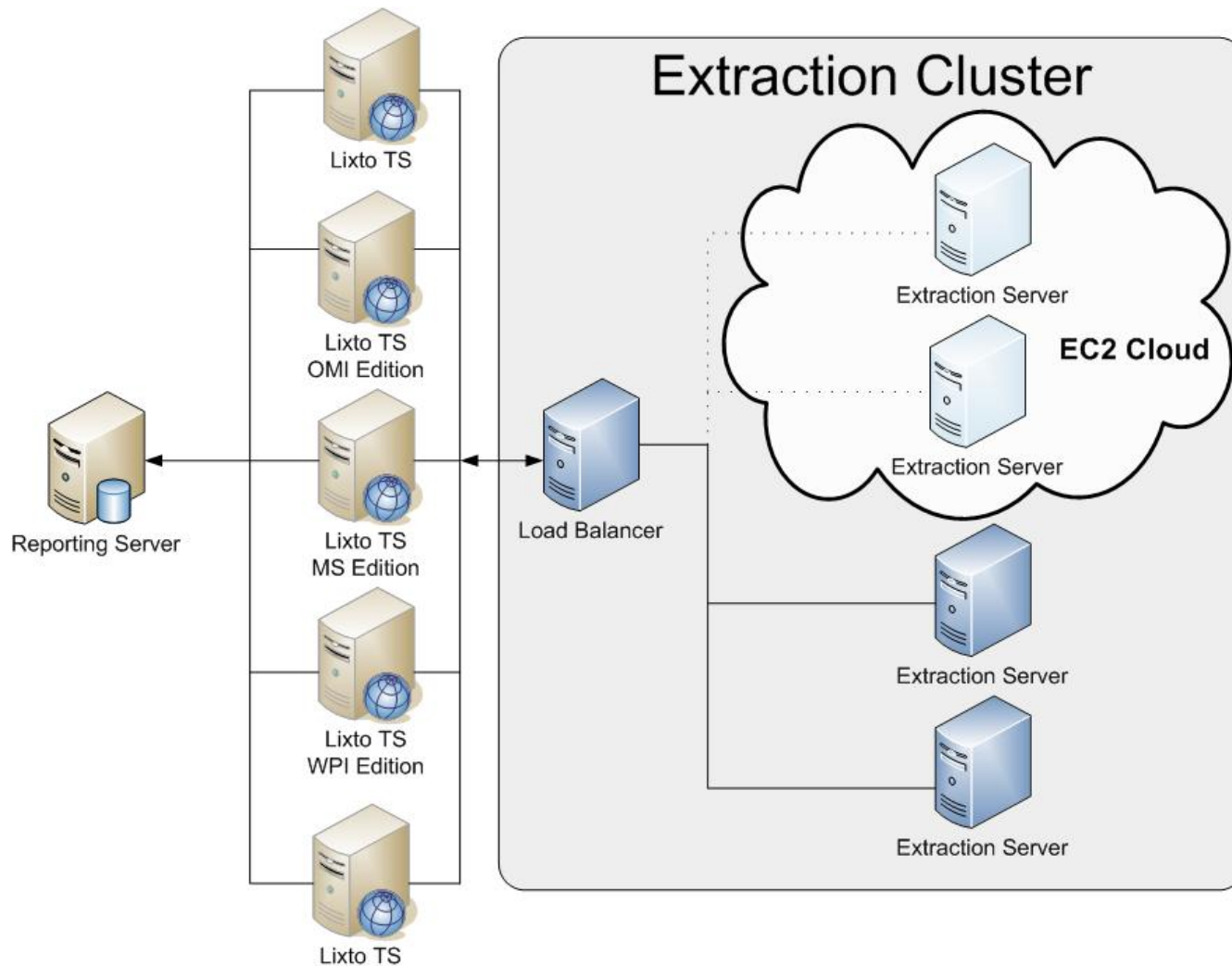


Lixto Data Extraction Environment



Lixto Enterprise Server Landscape

System Presentation
on Thursday



Lixto Visual Developer

- **Visual and Interactive definition of Web data extraction program (on top of Eclipse)**
- **Recording of user actions on the user interface level, e.g. mouse and key events**
- **Replay of recordings simulates human browsing behavior**
- **Compliant with Web 2.0 style dynamically changing pages of e.g. dynamic HTML and Ajax-based applications**
- **Support for complex navigation patterns including “detail” pages and “next” pages**
- **Embedding state-of-the-art browser (Mozilla)**
- **Parameterization of all kind of actions**

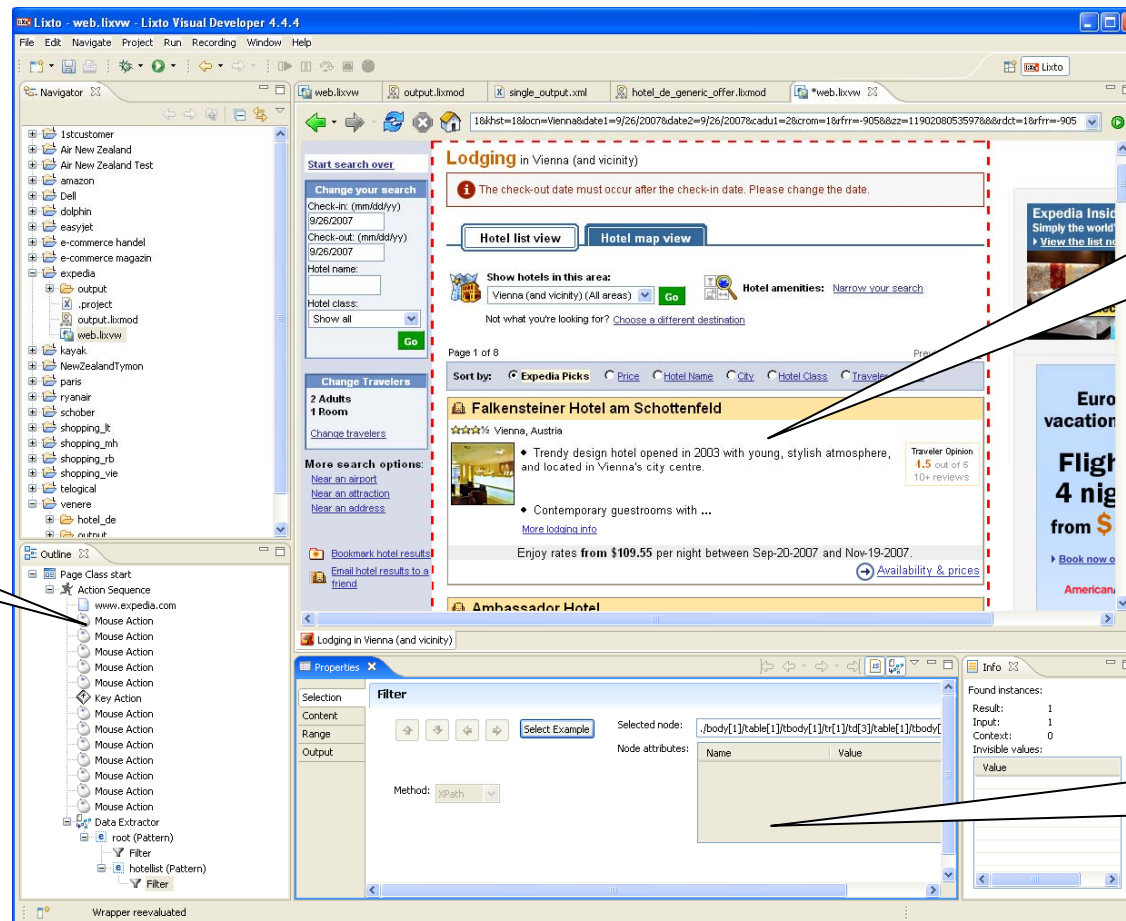


- Handling of pop-up windows and authentication requests
- Expressive declarative logic-based extraction language
- Robustness to structural changes of the Web page
- Full XPath 2 support to access the HTML document model
- Integrated data modeler to build XML-based data models for the output document
- Data extraction from additional data formats such as PDF, MS Office





Navigation Steps



The screenshot displays the Lixto Visual Developer 4.4.4 interface. The main window shows a Mozilla Web Browser displaying a hotel search page for "Lodging in Vienna (and vicinity)". The browser window includes a search form with fields for check-in and check-out dates, a "Go" button, and a list of hotel results. The first result is "Falkensteiner Hotel am Schottenfeld". The browser's address bar shows a complex URL with various parameters.

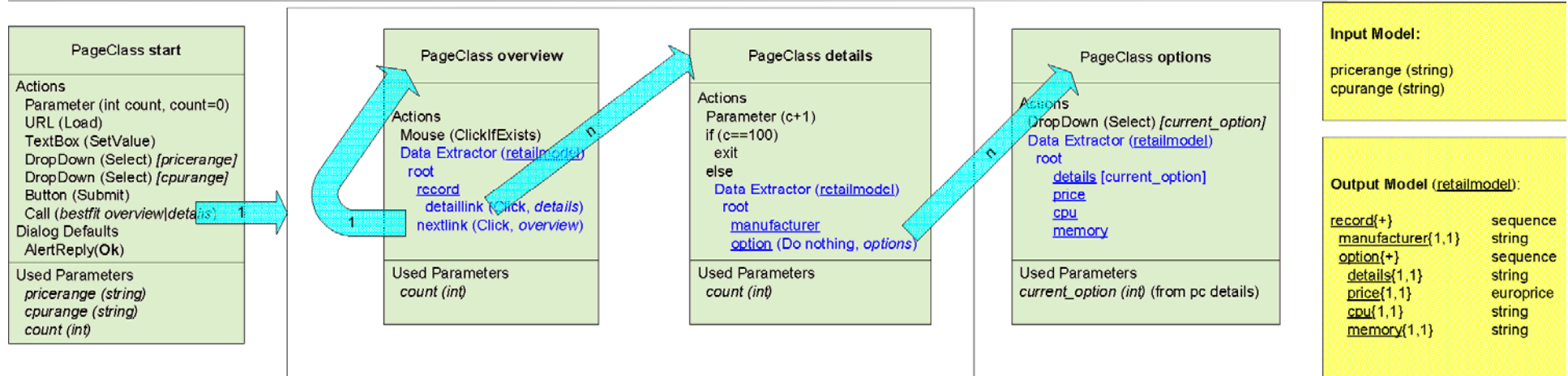
On the left side of the Lixto interface, there is a "Navigator" pane showing a project tree with folders like "1stcustomer", "amazon", "expedia", and "hotel_de". Below the Navigator is the "Outline" pane, which shows a hierarchical view of the page structure, including "Action Sequence", "Filter", and "Data Extractor".

At the bottom of the Lixto interface, there is an "Extraction Configuration" pane. It shows a "Filter" configuration with a "Method" set to "XPath". The "Selected node" is displayed as `./body[1]/table[1]/tbody[1]/tr[1]/td[3]/table[1]/tbody[1]`. The "Node attributes" section shows a table with "Name" and "Value" columns. The "Found instances" section shows a table with "Result", "Input", "Context", and "Invisible values" columns.

Mozilla
Web
Browser

Extraction
Configuration





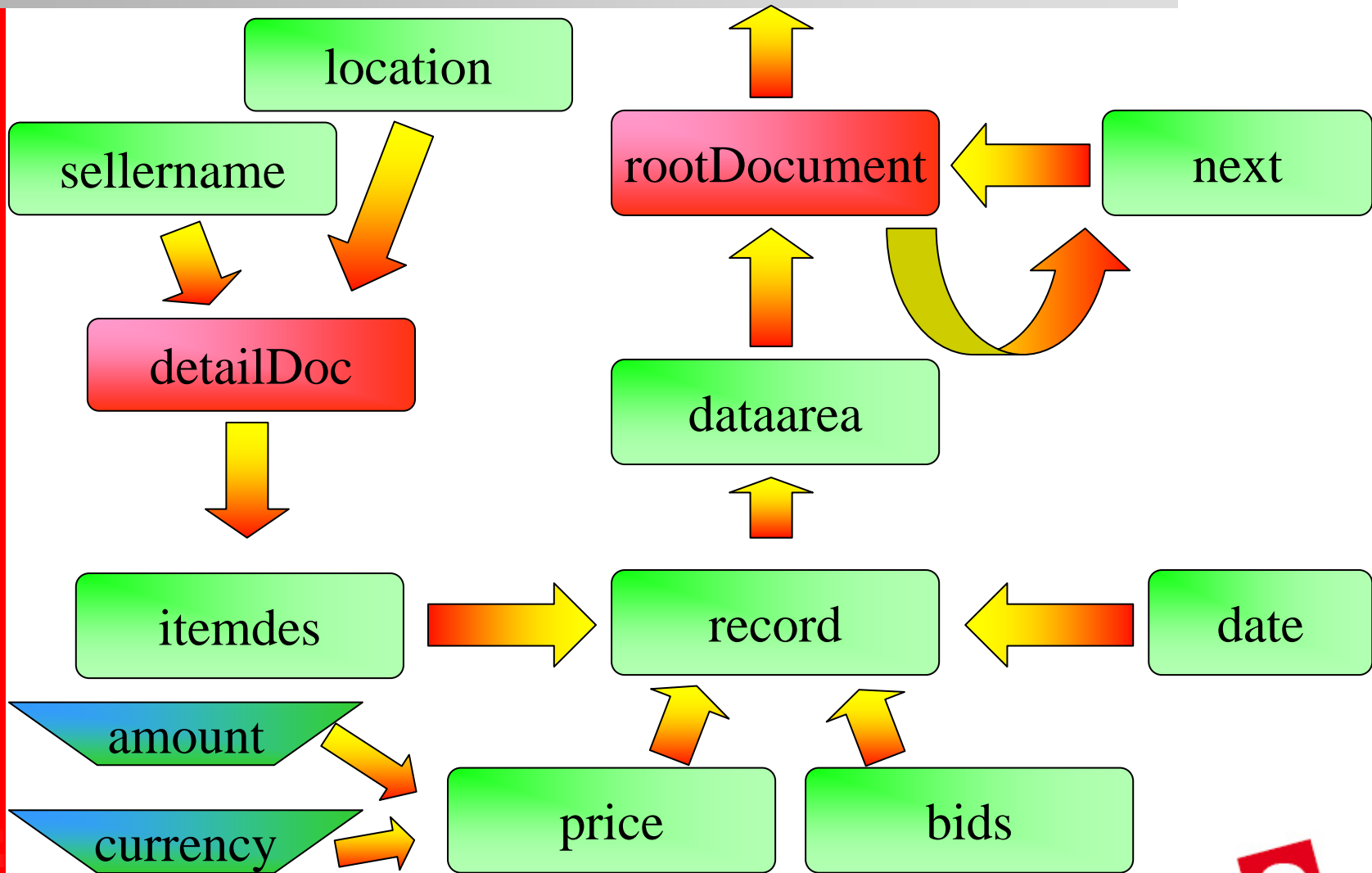
“PageClass” Template Concept

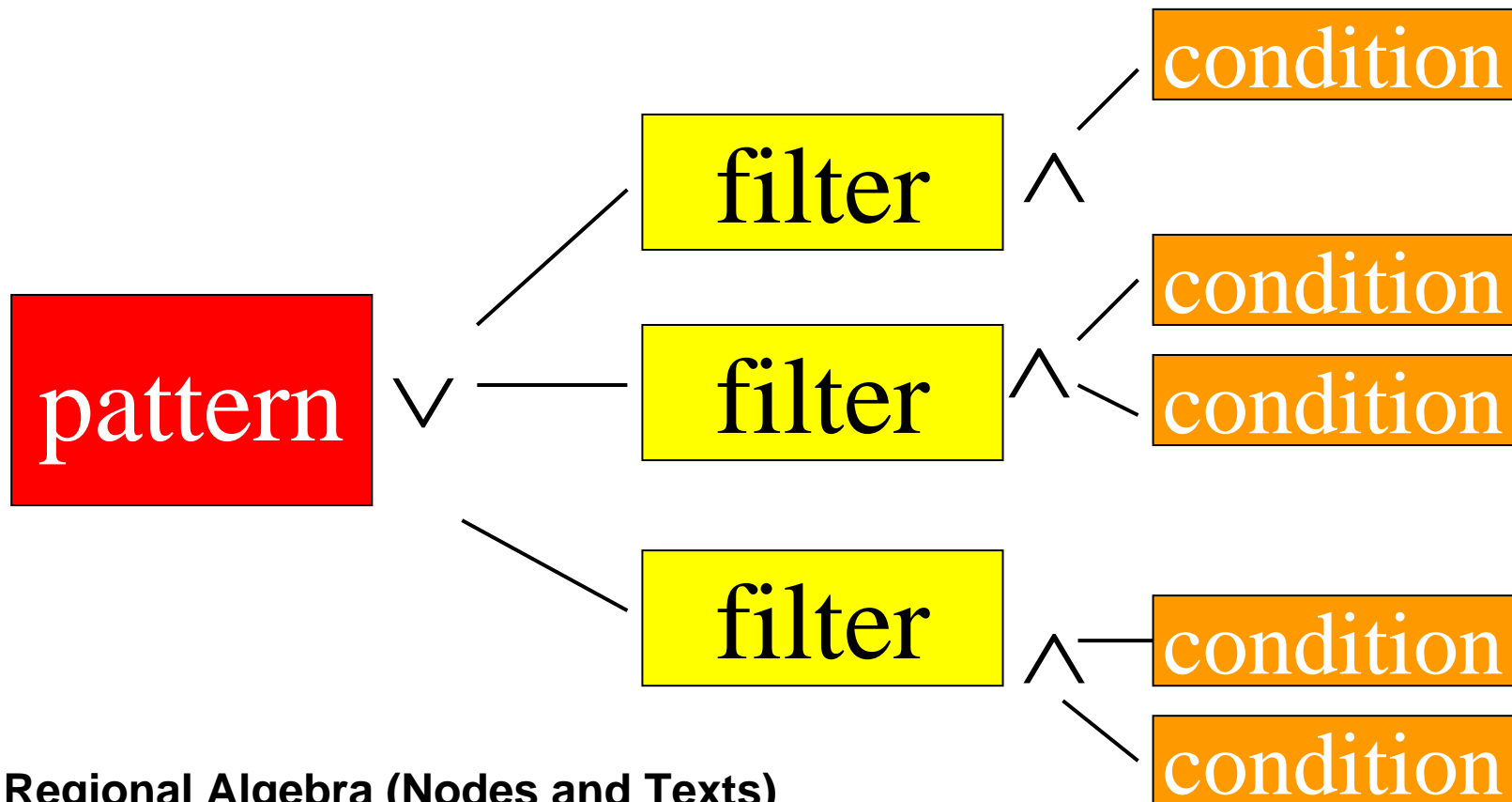
- Procedural Navigation Actions (e.g. form fillout)
 - Record Macros
- Declarative Extraction Program (e.g. find prices)
 - Example-based Creation





Data Extractor: Pattern Graph and Pattern Recursion



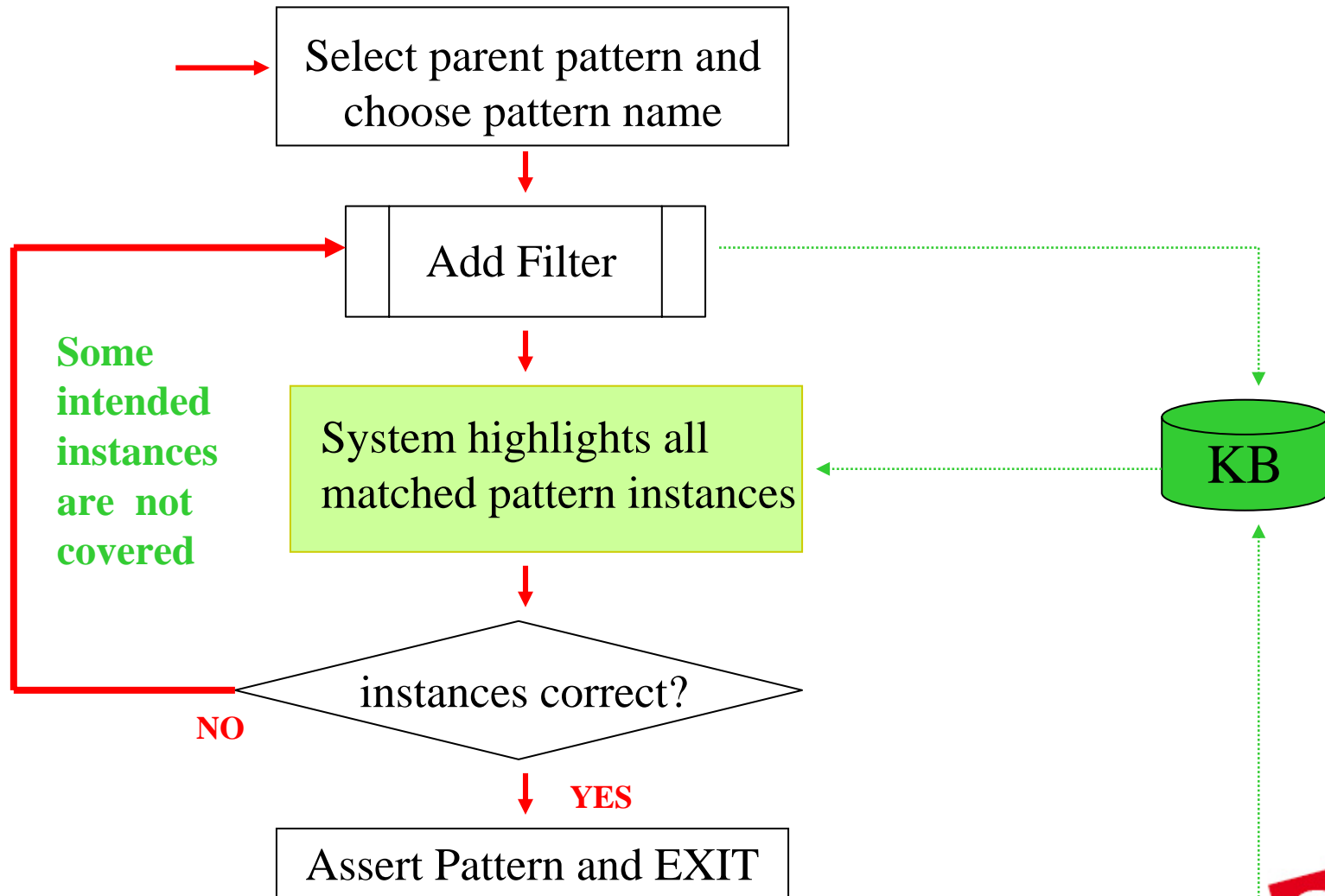


Regional Algebra (Nodes and Texts)

Basic conditions before, after, contains, n-th child



Interactive Pattern Generation



Interactive Filter Creation

System highlights an instance of parent pattern

User marks characteristic element therein
System identifies corresponding tree path
System generalizes to a **robust path**

System highlights all matched instances of filter

instances correct?

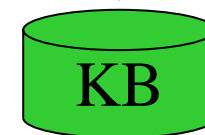
Assert Filter and EXIT

YES

One or more undesired
elements were displayed

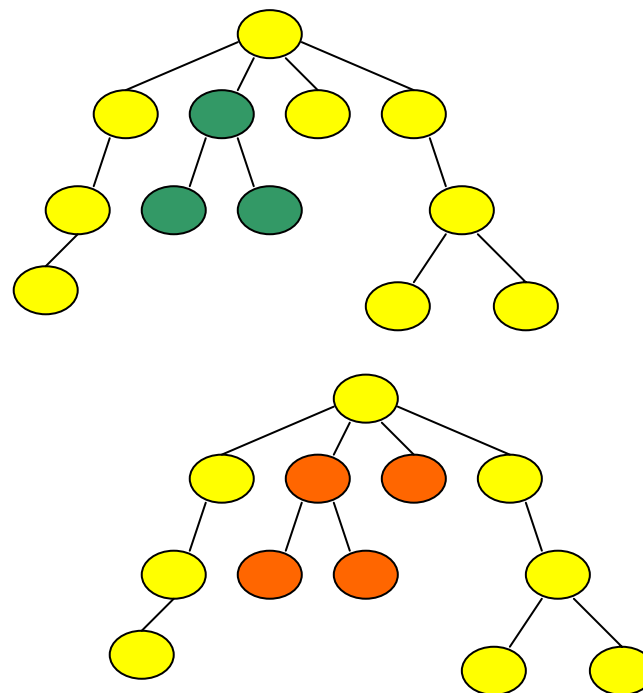
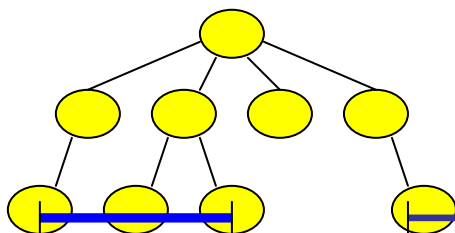
NO

Add Condition



Object Model and Objects

- **Labeled unranked tree**
 - Tree Nodes
 - Complex Tree Regions
- **String Representation**
 - Strings
 - Attribute Values
 - Output Transformation
- **CSS Box Model with spatial relations**



Phosphorus (yellow or white)	7732-51-9	Only if it is a yellow or white form
Sulfuric acid (acid aerosols including mists, vapors, gas, fog, and other airborne forms of any particle size)	7664-93-9	Only if it is an aerosol form as defined
Vanadium (except when contained in an alloy)	7440-62-2	Example if it is contained in an alloy
Zinc (fume or dust)	7440-66-6	Only if it is in a fume or dust form

The qualifier for the following three chemicals is based on the chemical activity rather than the form of the chemical. These chemicals are subject to EPCRA section 313 reporting requirements only when the indicated activity is performed:

Chemical/Chemical Category	CAN Number	Qualifier
Disks and disks-like compounds (manufacturing, and the processing or otherwise use of disks and disks-like compounds if the disks and disks-like compounds are processed as components in a chemical and if they were created during the manufacture of that chemical.)	NA	Only if they are manufactured at the facility, or are processed or otherwise used when present as contaminants in a chemical that only if they were created during the manufacture of that chemical.
Isopropyl alcohol (only persons who manufacture by the strong acid process are subject, see supplier notification)	67-63-9	Only if it is being manufactured by the strong acid process. Facilities that process or otherwise use isopropyl alcohol see 226 above.
Sulfuric acid (only persons who manufacture are subject, see supplier notification)	7664-93-9	Only if it is being manufactured.

There are no supplier notification requirements for isopropyl alcohol and sulfuric acid the processes and uses of these chemicals are not required to report. Manufacturers of these chemicals do not need to notify their customers that these are reportable EPCRA



Demo Example Elog Program Fragment

```
overview(X0,X1) :-
    getDocumentFromNavigation(X0=$1,X1).

root(X0,X1) :-
    overview(_,X0),
    subelem(X0,(.,[]),X1).

hotel(X0,X1) :-
    root(_,X0),
    subelem(X0,(...lixto:nearest("tr"),[]),X1),
    contains(X1,(./td[1]/a[1],[]),X2),
    before(X0,X1,(...strong[1],(['text','Sortieren',CONTAINS])),0,-1,X3,X4).

detail(X0,X1) :-
    link(_,X0), getDocumentbyClick(X0,X1).

price(X0,X1) :-
    hotel(_,X0),
    subelem(X0,(...lixto:nearest("div")/lixto:nearest("strong"),[]),X1)
    [1,1].

next(X0,X1) :-
    root(_,X0),
    subelem(X0,(...lixto:nearest("a"),(['text','Nächste Seite',CONTAINS])),X1).

overview(X0,X1) :-
    next(_,X0), getDocumentbyClick(X0,X1).
```



Internal Knowledge Representation

ELOG: a datalog like language

Wrapper : Elog Program

Pattern : IDB-Predicate

Filter : Rule

Condition : Atom of rule body

Parent Pattern : Special body atom

Element Path : Special body atom

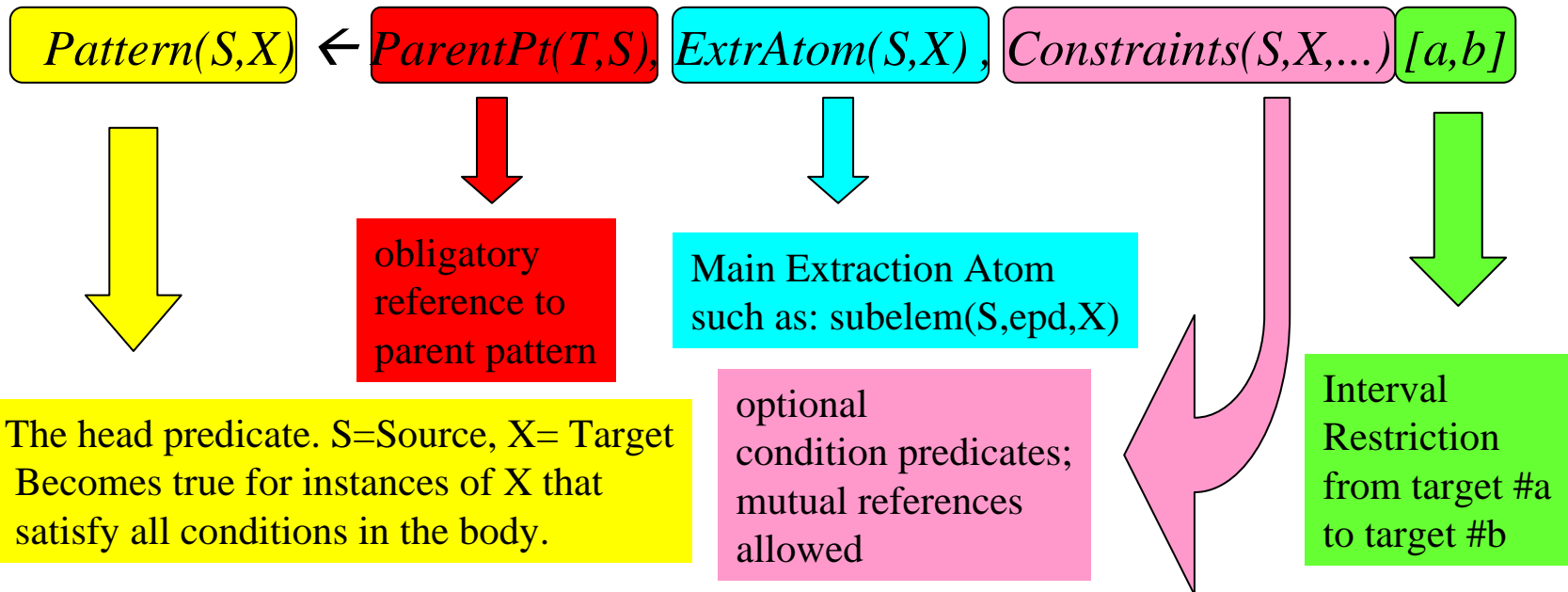
The declarative and/or semantics of datalog nicely matches the narrowing and broadening steps of the pattern generation process.

Maintenance in Elog Wrappers is local to the changed criteria and hence simple.

The Elog program itself is invisible to the wrapper designer.



Basic Elog Rule Structure



- The extracted set of targets contains all targets (tree- and string regions), in which the head predicate is true. Targets are minimized for complex regions.
- Rules with the same head predicate form a pattern. The target set of a pattern is minimized.
- Range is special to Elog, see mapping to Datalog rules below



○ Homogeneous Programs

- All Filters that define the same IDB predicate refer to the same parent pattern
- Pattern Graph is a tree

○ Heterogeneous Programs

- At least two filter that define the same IDB predicate refer to different parent patterns
- Pattern Graph can be cyclic
- Still, locally stratified if not mixing range and pattern references
- Fixed-point evaluation – stop if no new instance can be derived

```
document(S, X) ← getDocument($1, X)
table(S, X) ← document(_, S), subelem(S, . * .table, X)
table(S, X) ← table(_, S), subelem(S, . * .table, X)
```

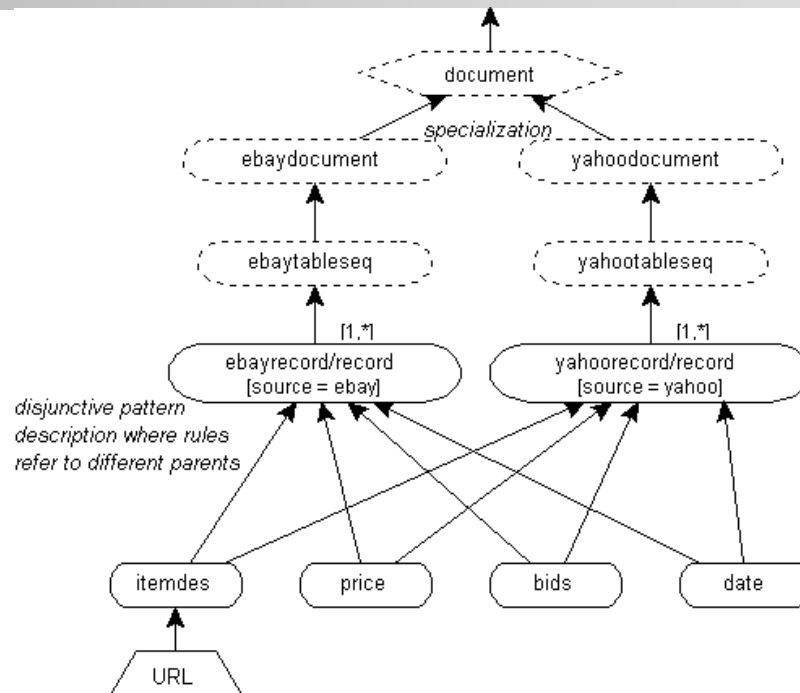


Built-In Predicates

- **Built-In predicates with binding schema**
- **subelem(S, path, X)**: `subelem(s, path, x)` evaluates to true iff `s` is a node, `path` is a relative XPath and `x` is a node contained in `s` where `x` matches `path`.
- **On textual level:** `subtext`, `subatt`
- **On document level:** `getDocument(X, Y), ...` (`x` is a URL)
- **Contextual:** `before(S, X, path, d1, d2, Y, D)`, `after`, `notbefore`, `notafter`
- **Internal:** `contains`, `nthchild`, ...
- **Concepts:** `isDate(X, Y)`, `isNumber(X)`, `isCity`, ...
- **Comparisons:** `<`, `>`, ... (e.g. compare distances or dates)
- **Pattern References:** refer to any IDB predicate (like: `before` there is an instance of `price`)



Specialization and Filter Disjunction

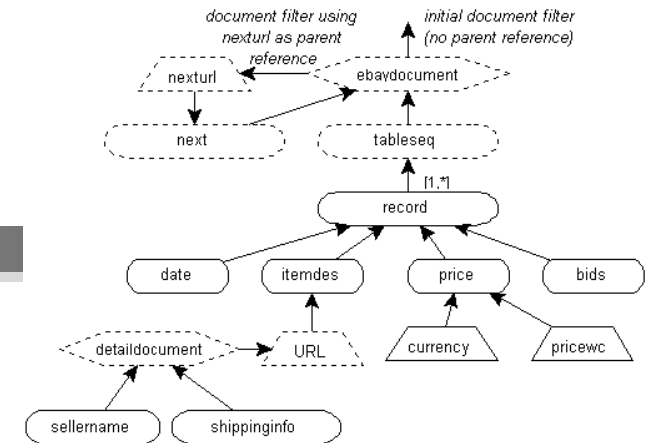


```

document(S, X) ← getDocument(S = $1, X)
ebaydocument(S, X) ← document(S, X),
    contains(X, (*.body, [(elementtext, eBay, substr)]), _)
yahoodocument(S, X) ← document(S, X),
    contains(X, (*.body, [(elementtext, Yahoo, substr)]), _)

```

Recursive Wrapping



```

next(S, X) ← ebaydocument(⊥, S),
             subelem(S, (*.content, [(href, , substr),
                                     (elementtext, (next page), exact)]), X)
nexturl(S, X) ← next(⊥, S), subatt(S, href, X)
ebaydocument(S, X) ← getDocument(S = $1, X)
ebaydocument(S, X) ← nexturl(⊥, S), getDocument(S, X)

```

Or, alternatively:

```

ebaydocument(S, X) ← ebaydocument(⊥, S),
                     subelem(S, (*.content, [(href, , substr),
                                             (elementtext, (next page), exact)]), Y),
                     subatt(Y, href, Z), getDocument(Z, X)

```



Expressing instance minimization

- using stratified negation
- and built-in predicate `contained_in`

```
p(S,X) :- par1(_,S), Ex1(S,X), Co1(S,X,...).  
p(S,X) :- par2(_,S), Ex2(S,X), Co2(S,X,...).  
p(S,X) :- par3(_,S), Ex3(S,X), Co3(S,X,...).
```

```
p'(S,X) :- par1(_,S), Ex1(S,X), Co1(S,X,...).  
p'(S,X) :- par2(_,S), Ex2(S,X), Co2(S,X,...).  
p'(S,X) :- par3(_,S), Ex3(S,X), Co3(S,X,...).  
p''(S,X) :- p'(S,X1), p'(S,X), contained_in(X1,X).  
p(S,X) :- p'(S,X), ¬ p''(S,X).
```



Express Range in Datalog using stratified negation

```
NewPat :- filterbody [a,b].
```

```
NewPat'(S,X) :- filterbody.
```

```
NewPat(S,X) :- NewPat'(S,X), Solpos(S,X,P),  
a <= P <= b.
```

```
Solpos(S,X,1) :- NewPat'(S,X), FirstSol(S,X).
```

```
Solpos(S,X,P) :- SolPos(S,X',P'), NewPat'(S,X),  
succ(S,X',X), P=P'+1
```

... and in the definition of FirstSol and succ predicate NewPat is negated



Pattern References and Range

```
p(S,X) :- par(_,S), subelem(S,epd,X), before(S,X,...Y), q(S,Y) [a,b]
q(S,X) :- par(_,S), subelem(S,epd,X), before(S,X,...Y), p(S,Y) [c,d]
p(S,X) :- ...
q(S,X) :- ...
```

Define p via q and q via p and applying range restrictions.

Translation results in unstratified negation and results in non-monotonicity.

Generates multiple models with stable model semantics.



Elog⁻ Core Fragment

Expressing built-in predicates with child and nextsibling relations on labeled trees. Ignoring features of Elog on textual structures, distances, etc.

Signature: $t_{ur} = \langle \text{dom}, \text{root}, \text{leaf}, (\text{label}_a)_{a \in \Sigma}, \text{firstchild}, \text{nextsibling}, \text{lastsibling} \rangle$

Basic Elog relations can be derived:

$$\begin{aligned} \text{subelem}_\epsilon(X, Y) &:= X = Y. \\ \text{subelem}_{\text{path}}(X, Y) &:= \text{child}(X, Z), \\ &\quad \text{subelem}_{\text{path}}(Z, Y). \\ \text{subelem}_{a.\text{path}}(X, Y) &:= \text{child}(X, Z), \text{label}_a(Z), \\ &\quad \text{subelem}_{\text{path}}(Z, Y). \\ \\ \text{contains}_{\text{path}}(X, Y) &:= \text{subelem}_{\text{path}}(X, Y). \\ \text{before}_{\text{path}}(S, X, Y) &:= \text{subelem}_{\text{path}}(S, Y), \\ &\quad \text{nextsibling}(X, Y). \\ \text{after}_{\text{path}}(S, X, Y) &:= \text{subelem}_{\text{path}}(S, Y), \\ &\quad \text{nextsibling}(Y, X). \\ \text{firstson}(S, X) &:= \text{firstchild}(S, X). \\ \text{lastson}(S, X) &:= \text{lastchild}(S, X). \end{aligned}$$

Elog- Monadic Kernel of Elog:
Make all IDB predicates unary by rewriting rules. Elog⁻ and Elog⁻₂ characterize the same tree languages.



- **Unary queries in MSO over trees as expressiveness yardstick for information extraction functions**
- **MSO hard to use as wrapping language due to lack of operational semantics**

Theorem [Gottlob & Koch, PODS 2002]

Monadic Datalog over τ_U has
combined complexity:

$$O(|\text{dom}| * |\text{program}|)$$



Theorems [Gottlob & Koch, PODS 2002]:

Over τ_U , Monadic Datalog = MSO

A unary query is definable in MSO iff it is definable via a monadic datalog program.

ELOG⁻ expresses monadic datalog + child

All of ELOG⁻ is graphically programmable via *LiXto*

LiXto expresses all MSO wrapping tasks.



Current/Future Research Directions

- **Declarative Data Extraction from PDF/Visual Structures**
- **Robustness**
- **Focused Spidering**
- **Wrapper Adaptation**
- **Product Matching**
- **Execution Planning**
- **Sampling Algorithms**



Selected Lixto Publications

[Baumgartner, Flesca & Gottlob 2001]

Visual Web Information Extraction with Lixto. *VLDB 2001*.

[Herzog & Gottlob 2001]

InfoPipes. A flexible framework for M-Commerce Applications. *TES 2001*.

[Gottlob & Koch 2002]

Monadic Datalog and the Expressive Power of Languages for Web Information Extraction. *PODS 2002*.

[Baumgartner, Gottlob & Herzog 2009]

Scalable Web Data Extraction for Online Market Intelligence. *VLDB 2009*.

[Baumgartner, Campi, Gottlob & Herzog 2010]

Web Data Extraction for Service Creation. *In Search Computing Challenges, Springer LNCS*.

[Gatterbauer, Herzog et al. 2007]

Towards domain-independent information extraction from web tables. *WWW 2007*.

[Hassan & Baumgartner 2007]

Table Recognition and Understanding from PDF Files. *ICDAR 2007*.





Thank you for your time!

Lixto Software GmbH
Favoritenstrasse 16/DG
A-1040 Vienna, Austria

robert.baumgartner@lixto.com

