Foundations of XML Data Manipulation

Giorgio Ghelli

Query Languages for SSD and XML
Some names

- UnQL: BunDavHil96
- Lorel: AbiQuaMcH97
- XSLT, XPath, XQuery: google:w3c xslt, w3c xpath, w3c xquery
- XML-QL: XMLQL, xquery99
- XDuce: HosPie03
- TQL: CarGhe03
- YATL, Strudel…

Path expressions

```xml
<bib>
<book year="1995">
  <author> <first>Serge</first> <last>Abiteboul</last> </author>
  <author> <first>Richard</first> <last>Hull</last> </author>
  <author> <first>Victor</first> <last>Vianu</last> </author>
  <publisher>Addison</publisher>
  <price>60</price>
</book>
<book year="1993">
  <title>Formal Semantics</title>
  <author> <first>Glynn</first> <last>Winskel</last> </author>
  <publisher>MIT Press</publisher>
  <price>42</price>
</book>
</bib>
```
Path expressions

- **Document** /bib/book/author/first:
  - `<first>Serge</first>`, `<first>Richard</first>`, `<first>Victor</first>`, `<first>Glynn</first>`

- Semantics:
  - All nodes you find starting from **Document** and walking down a /bib/book/author/first path
  - More generally: \([p] = \{ <n,m> | m \in n/p \}\)

- The interesting case:
  - Data is a graph
  - Paths is a regular expression

A formal definition

- Assume a graph \(G=(N,E)\) with \(E \subseteq N \times A \times N\)
- A word \(w\) in \(A^*\) determines a relation \(|w|\) on \(N \times N\):
  - \(n \mid w \mid n\)
  - \((n,a,n') \in E \Rightarrow n \mid a \mid n'\)
  - \(n \mid w \mid n' \text{ and } n' \mid w' \mid n'' \Rightarrow n \mid w.w' \mid n''\)

- A language \(L \subseteq A^*\) determines a relation:
  - \(n \mid L \mid n' \iff n \mid w \mid n' \text{ for some } w \in L\)

- A regexp \(r\) determines a relation:
  - \(n \mid r \mid n' \iff n \mid \text{Lang}(r) \mid n'\)

- We allow regexp on labels as well
Regular path expressions

- "book"."[T|t]itle":
  - {book.title, book.Title}
- book.(Title|title):
  - {book.title, book.Title}
- ".\*".title:
  - {book.title, author.title, name.title,...} i.e. {_.title}
- book.refs*.title:
- ("\*\")*.name:
  - {name, _._name, _._._name, _._._._name,...}

Syntax for path expressions

- a ::= ε | letter | . | a a | [a|a] | a*
- r ::= ε | a | _ | r.r | r|r | ε*
- Abbreviations:
  - r? = r|ε
  - r+ = r.r*
XPath syntax

• \( p, p' ::= / p \mid p/p' \mid \text{axis}::\text{nodetest} \)
• \( \text{nodetest} ::= * \mid \text{tag} \mid \text{text()} \mid \text{element(*)} \mid \text{element(tag)} \mid \text{attribute(*)} \mid \ldots \)
• \( \text{axis} ::= \text{child} \mid \text{parent} \mid \text{d-o-s} \mid \text{a-o-s} \mid \text{descendant} \mid \text{ancestor} \mid \text{following} \mid \text{preceding} \mid \text{following-sibling} \mid \text{preceding-sibling} \)
• \( p/\text{nodetest} = p/\text{child}::\text{nodetest} \)
• \( p//q = p/\text{d-o-s}::*//q \)
• \( .. = \text{parent}::\text{node()} \)

Tree Patterns

• Titles of books:
  \( $\text{doc} //\text{book}/\text{title} \)
• Titles of books by Buneman:
  \( $\text{doc} //\text{book}[/\text{last}/\text{text()} = 'Buneman']/\text{title} \)
Tree Patterns (Twigs)

- book[author]/title
- book/author/../title
- author/../../book/title

- Up pointers are slightly more expressive than twigs

Are path expressions a query language?
The structure of XML query languages

- FROM BindingExpression(X): generate a set of bindings for X
- WHERE Condition(X): filter some bindings out
- SELECT Result(X): evaluate Result(X) once for each binding, and find a way to merge the results
- Path expressions come handy in the FROM clause

MicroXQuery

- Reference: Colazzo et al., Types for Path Correctness of XML Queries, ICALP’04.

```xml
for $b in $doc /bib/book,
    $a in $b /author
where $b /@year > 2000
return <libro> $b/title, $a </libro>
```
MicroXQuery

- for $b in $doc /bib/book
  let $a = $b /author
  where $b /@year > 2000
  return <libro> $b/title, $a </libro>

- for $b in $doc /bib/book
  where $b /@year > 2000
  return <libro> $b/title, $b /author </libro>

Tree Patterns

- Titles of books by Buneman:
  for $x in $doc //book
  where $x/authors/last/text() = 'Buneman'
  return $x/title

- Same as:
  $doc //book[last/text() = 'Buneman']/title
Trees to relations to trees

let $authors = $doc /bib/book/author
for $a in distinct($authors)
return <booksByAuth> $a,
    for $bb in $doc /bib/book
        where $a isin $bb /author
        return $bb/title
</booksByAuth>

Result

• From
  – <bib>
  – (<book> (<author></author>)*
  – </book>)*
  – </bib>
• To
  – (<booksByAuth>
  – <author>…</author>
  – (<title>…</title>)*
  – </booksByAuth>)*
Yet another syntax

BBAS[
  let $authors = $doc /bib/book/author
  for $a in distinct($authors)
  return booksByAuth[
    $a,
    for $bb in $doc /bib/book
      where $a isin $bb /author
      return $bb/title
  ]
]

Type

• Type:
  BBAS[booksbyaut[ author[String],
    title[String]* ]*]

• If:
  $doc: bib[ book [ title[String],
    author[String]*
  ]*]
Tree manipulation

- Micro XQuery is able to do “nested relations”, i.e. trees with fixed depth
- It is unable to produce arbitrarily deeply nested trees:
  - Structural recursion
- It is unable to reverse a graph:
  - Skolem functions

Structural recursion

- $f(v) \Rightarrow v$
- $f(\text{author}[x]) \Rightarrow \text{autore}[f(x)]$
- $f(l[x]) \Rightarrow \text{if } l = \text{year} \text{ then } 0 \text{ else } l[f(x)]$
- $f(0) \Rightarrow 0$
- $f(x,y) \Rightarrow f(x),f(y)$
**Structural recursion**

- $f$: collect authors, if any
- $f(\text{author}[x], y) \Rightarrow \text{authors}[\text{author}[f(x)], g(y)], h(y)$
- $g(\text{author}[x]) \Rightarrow \text{author}[f(x)]$
- (else) $g(l[x]) \Rightarrow 0$
- $h(\text{author}[x]) \Rightarrow 0$
- (else) $h(l[x]) \Rightarrow l[f(x)]$
- Implicit: $f(a[x], y) \Rightarrow f(a[x]), f(y); f(a[x]) \Rightarrow a[f(x)]$

**Structural recursion vs. query languages**

- Paths go down
- for – where – selects iterates horizontally
- Structural recursion does both
- Supported by:
  - XDuce (SR only)
  - XSL (SR only)
  - XQuery (FWR + recursion)
XSL

```xml
<xsl:stylesheet version="1.0" ...>
  <xsl:template match="ul[parent::ul]">
    <li>
      <ul>
        <xsl:apply-templates select="@*|node()"/>
      </ul>
    </li>
  </xsl:template>
  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Skolem functions

- new(a,b,c,...): a (new) node
- n-lab-n: an edge

- for $x$-$lab$-$y$ in Edges
  - return new(node,$y$)-$lab$-new(node,$x$)

- for $x$-$lab$-$y$ in Edges
  - return new(node,$x$)-$lab$-new(node,$y$),
  - new(node,$y$)-home-new(root)
**Logical languages: TQL**

- Matching through ambient-logic formulas:

\[
\text{FROM \ $db \ |= \ .paper[ (.author[$X] \text{ or } .autore[$X])}
\text{ and not .editor[$X]}}
\]

\[
\text{SELECT author[$X]} \]

Returns:

\[
\text{author\{Cardelli\} | author\{Gordon\} | author\{Ghelli\}}
\]

**Find All Keys**

from $\text{Bib}$

\[
|= \ \text{bib}\{!book[.$k[T]]}
\]

And foreach $X$.

\[
\text{Not (.book.$k[$X] \text{ or } .book.$k[$X])}
\]

\[
\text{select key[$k]} \]
The Logic

- $F \vdash_\sigma 0$ iff $F = 0$
- $F \vdash_\sigma A \mid B$ iff $\exists F', F' = F' \mid F''$, $F' \vdash_\sigma A$, $F'' \vdash_\sigma B$
- $F \vdash_\sigma \eta[A]$ iff $F = \sigma(\eta)[F']$, $F' \vdash_\sigma A$
- $F \vdash_\sigma A \land B$ iff $F \vdash_\sigma A$ and $F \vdash_\sigma B$
- $F \vdash_\sigma \neg A$ iff not ($F \vdash_\sigma A$)
- $F \vdash_\sigma \eta = \eta'$ iff $\sigma(\eta) = \sigma(\eta')$
- $F \vdash_\sigma \exists x.A$ iff $\exists n. F \vdash_{\sigma[n/x]} A$
- $F \vdash_\sigma \forall X.A$ iff $\forall F'. F \vdash_{\sigma(F'/X)} A$
- $F \vdash_\sigma \mu \xi.A$ iff $F \vdash_\sigma A(\mu \xi.A/\xi)$ (is circular…)
- De Morgan duals: $\lor$, $\eta \Rightarrow A$, $F$, $\vee$, $\neq$, $\forall \xi.A$, $! \eta[A]$

Logical languages: monadic datalog

- doc //book/author X:
  - result(Y) :- desc(doc,X), name(X,book), child(X,Y), name(Y,author)
The full hybrid mu-calculus

• $A ::= i | \text{lab} | <s>A | A \land A | \neg A | \mu \xi.A | \xi$

• $E,L,w \vdash_i i$ if $L(i) = \{w\}$

• $E,L,w \vdash_{\text{lab}} \text{lab}$ if $w \in L(\text{lab})$ ($L(w) = \text{lab}$)

• $E,L,w \vdash_{<s>A} <s>A$ if $\exists w'. w.s.w' \in E$ and $E,L,w' \vdash \sigma A$

• $E,L,w \vdash_{<s>A} \neg<s>A$ if $\exists w'. w'.s.w \in E$ and $E,L,w' \vdash \sigma A$

• $E,L,w \vdash_{A \land B} A \land B$ if $E,L,w \vdash \sigma A$ and $E,L,w \vdash \sigma B$

• $E,L,w \vdash_{\neg A} \neg A$ if not $(E,L,w \vdash \sigma A)$

• $E,L,w \vdash_{\mu \xi.A} \mu \xi.A$ if $E,L,w \vdash \sigma A(\mu \xi.A/\xi)$ (is circular…)

• Over XML:
  -- Just two steps $s$: $\downarrow\uparrow$ (firstchild) and $\rightarrow\leftarrow$ (nextsibling)
  -- $E$ is a finite tree
  -- $L(\text{lab})$ is a partition of the nodes

• De Morgan duals: $[s]A, \lor, \forall \xi.A$

Encoding axes

• $\text{<child>} A = <\downarrow>(\mu \xi (A \lor <\rightarrow>\xi))$

• $\text{<parent>} A = \mu \xi ( <\uparrow> A \lor <\rightarrow>\xi)$

• $\text{<desc>} A = <\text{child}> (\mu \xi (A \lor <\text{child}>\xi))$

• $\text{<ancestor>}$

• $\text{<following-sibling>}$

• $[\text{everywhere}] A = \forall \xi (A \land [\downarrow] \xi \land [\rightarrow] \xi)$

• $\text{<somewhere>} A = \mu \xi (A \lor <\downarrow>\xi \lor <\rightarrow>\xi)$
Logical languages: modal mu calculus

- \(<m,n>\ in \[[p]]\ iff \ E,L,i\rightarrow{\{n\}}, \ m \models \langle\langle p\rangle\rangle\)
- //book/author:
  - \(<\text{desc}(\text{book} \land \langle\text{child}\rangle(\text{author} \land i))\>
- //book[title]/author
  - \(<\text{desc}(\text{book} \land (\langle\text{child}\rangle\text{title})
   \land (\langle\text{child}\rangle\text{author} \land i))\>
- //book[not title]/author
  - \(<\text{desc}(\text{book} \land (\text{not}\langle\text{child}\rangle\text{title})
   \land (\langle\text{child}\rangle\text{author} \land i))\>

Logical languages: MSO

- MTran: transformation language based on MSO [HinabaHosoya..]
- MSO: FO plus set quantification
- Able to express all regular tree query
MTran formulas

• //img:  x in <img>
• /*[date]:  ex1 y: x/y and y in <date>
• //a{x}/b{y}: x in <a> and x/y and y in <b>

MTran transformation

• Add an <li> around to any <ul> whose parent is an <ul>:
  – { visit x :: <ul>/x & x in <ul> :: li[x] }
• visit copies all unmatched nodes, gather does not:
  – <ul>{gather x :: x in <a> :: li[x]}<ul>
Nesting

• \{\text{gather } b :: b \in \text{book} :: \}
  \{\text{gather } a :: \text{b/a} \& a \in \text{author} :: \}
  \text{book-author}[b,a] \} \}

Evaluations

• Compile the formula to a tree automaton
  – Non-elementary in the worth case
  – MONA often works well

• Evaluate the automaton
  – Linear algorithm (non trivial, since the query is n-ary)
Tree Automata

• Morally, the same expressive power (over trees) as:
  – Monadic datalog
  – Tree patterns
  – MSO
  – Mu-calculus
  – XDuce type system
  – …

• Things are not so simple…

Readings

• Xquery99: comparison of XML-QL, YATL, Lorel, XQL
• BonCer00: comparison of Lorel, XML-QL, XML-GL, XSL, XQL
• ColGheAl06: MicroXQuery
• w3.org/TR/xquery|xpath|xslr: XQuery, XPath, XSLT
• BunDavHil96: UnQL
• AbiQuaMcH97: Lorel
• HosPie03: XDuce
• klarlundSchweintick: XPath, XQuery, XSLT
• CarGhe03: TQL
• Tocl-lics0203: Logics, Automata