XML Query Languages

Martin Steffen

IfI UiO

7. February 2007

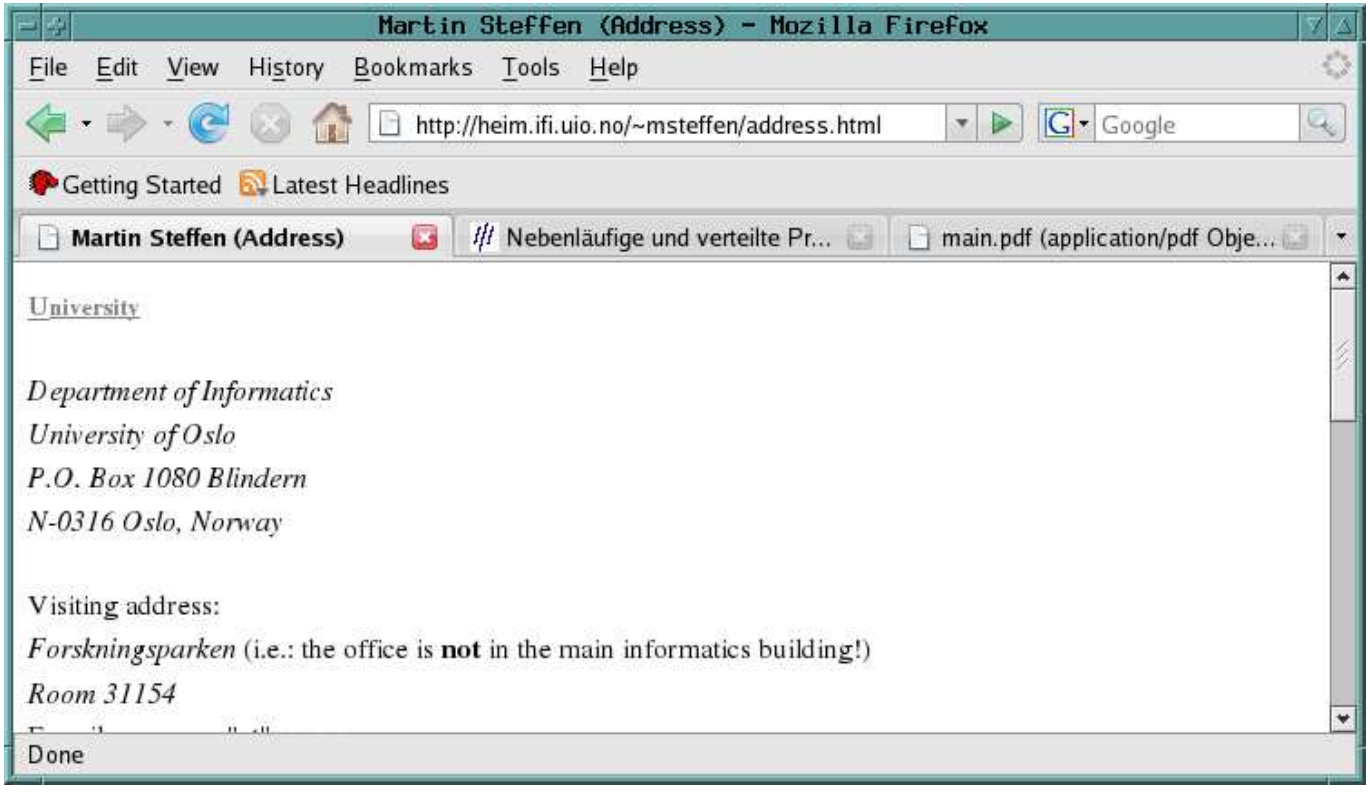
XML

- *extensible markup language*

Fixed mark-up

```
<HTML>
  <HEAD>
    <TITLE> Martin Steffen (Address) </TITLE>
  </HEAD>
<link rel=stylesheet
      type="text/css"
      href="styles/ms.css">
<body>
<H4>
   <a href="http://www.uio.no">University</a>
</H4>
<P>
<TABLE>
  <I>Department of Informatics </I>
    <br>
```

Fixed mark-up



History

- SGML: invented 1969, standardized 1980
- 1990: awareness for dynamic mark-up a la (S)GML at W3C
- 1996: first working draft of XML
- Feb. 1998: XML1.0 recommendation (later various versions)
- Feb. 2004: XML1.1
- 27. 1. 2007: official XQuery-semantics: [Draper et al., 2007]

XML-data = trees

```
<addrbook>
  <person>
    <name>Joe Doe</name>
    <email>doe@cba.org</email>
  </person>
  <person>
    <name>Jane Dow</name>
    <email>dow@egd.com</email>
    <tel>123-456-788</tel>
  </person>
</addrbook>
```

XML-data = trees

```
<addrbook>
  <person>
    <name>Joe Doe</name>
    <email>doe@cba.org</email>
  </person>
  <person>
    <name>Jane Dow</name>
    <email>dow@egd.com</email>
    <tel>123-456-788</tel>
  </person>
</addrbook>
```

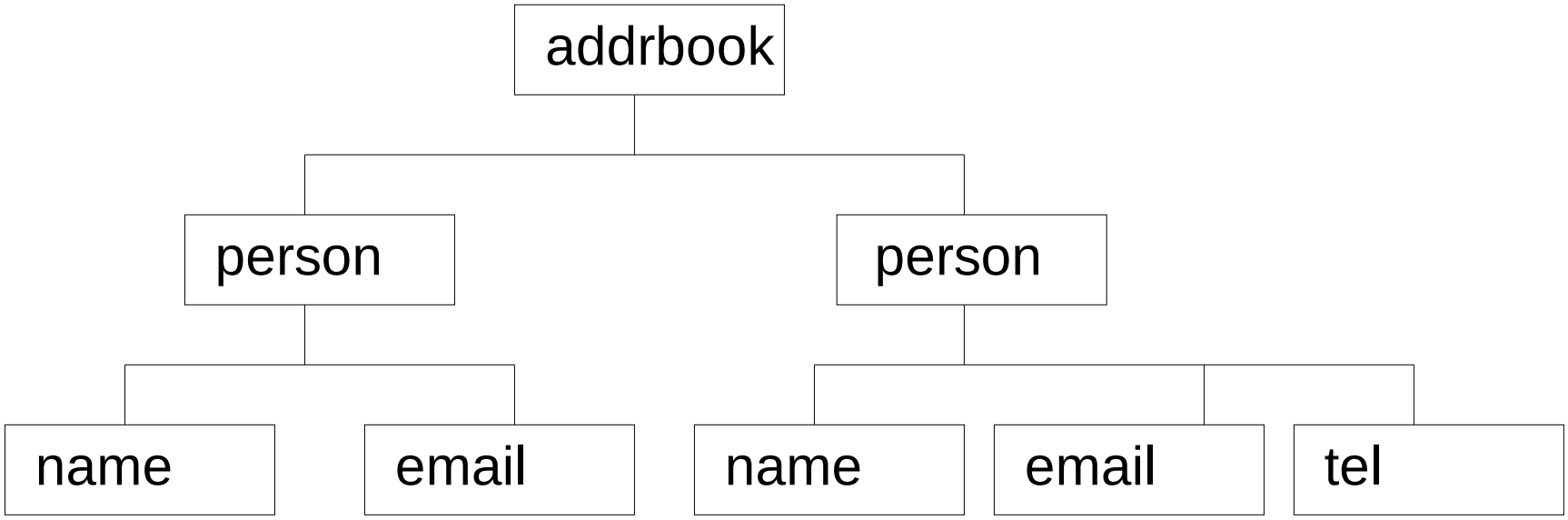
XML-data = trees

```
<addrbook>
  <person>
    <name>Joe Doe</name>
    <email>doe@cba.org</email>
  </person>
  <person>
    <name>Jane Dow</name>
    <email>dow@egd.com</email>
    <tel>123-456-788</tel>
  </person>
</addrbook>
```

XML-data = trees

```
<addrbook>
  <person>
    <name>Joe Doe</name>
    <email>doe@cba.org</email>
  </person>
  <person>
    <name>Jane Dow</name>
    <email>dow@egd.com</email>
    <tel>123-456-788</tel>
  </person>
</addrbook>
```

XML-data = trees



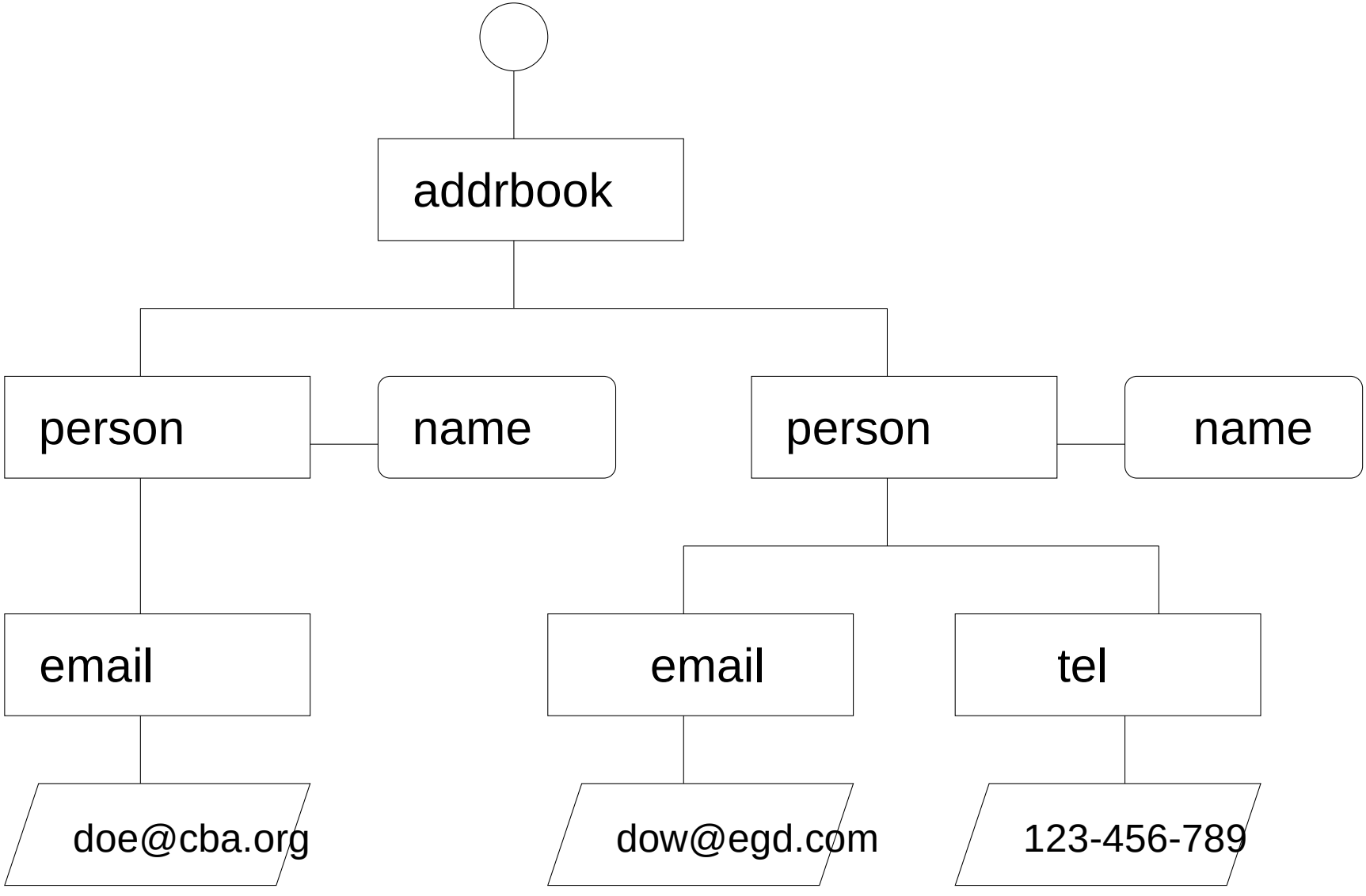
Trees with bells and whistles

- different data-models
- most important one: XPath data model
- complications:
  - different sorts of “nodes”:
    - element nodes (the “real” ones)
    - attribute nodes
    - text-nodes (= leaves)
    - comment nodes, processing instruction nodes, root node
- namespace mechanism [Bray et al., 2004a]

Trees with bells and whistles

```
<addrbook>
  <person name="Joe Doe">
    <email>doe@cba.org</email>
  </person>
  <person name="Jane Dow">
    <email>dow@egd.com</email>
    <tel>123-456-788</tel>
  </person>
</addrbook>
```

Trees with bells and whistles



Schema

- schema : specifying the allowed *form* of tree

```
<!ELEMENT addrbook person*>
<!ELEMENT person (name, email*, tel?>
<!ELEMENT name #PCDATA>
<!ELEMENT element #PCDATA>
<!ELEMENT tel  #PCDATA>
```

Schema

- schema : specifying the allowed *form* of tree
- valid XML-document: conforming to the given schema definition
- many different schema language (eg. XMLSchema

```
<!ELEMENT addrbook person*>
<!ELEMENT person (name, email*, tel?>
<!ELEMENT name #PCDATA>
<!ELEMENT element #PCDATA>
<!ELEMENT tel  #PCDATA>
```

Design goals of XQuery

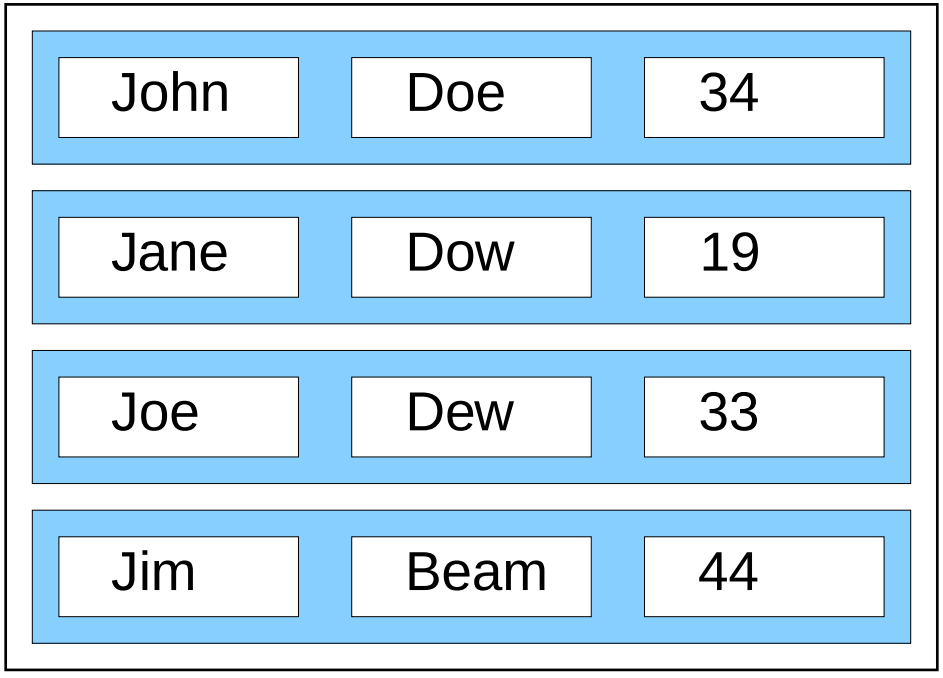
- XQuery: standard XML query language
- goals:
  - declarative
  - support of namespaces
  - “work together” with XMLSchema (support simple/complex datatypes)
  - support XML-syntax + human-readable syntax.
  - supports (static and dynamic) type checking

    [Boag et al., 2003, Section 5, Conformance]

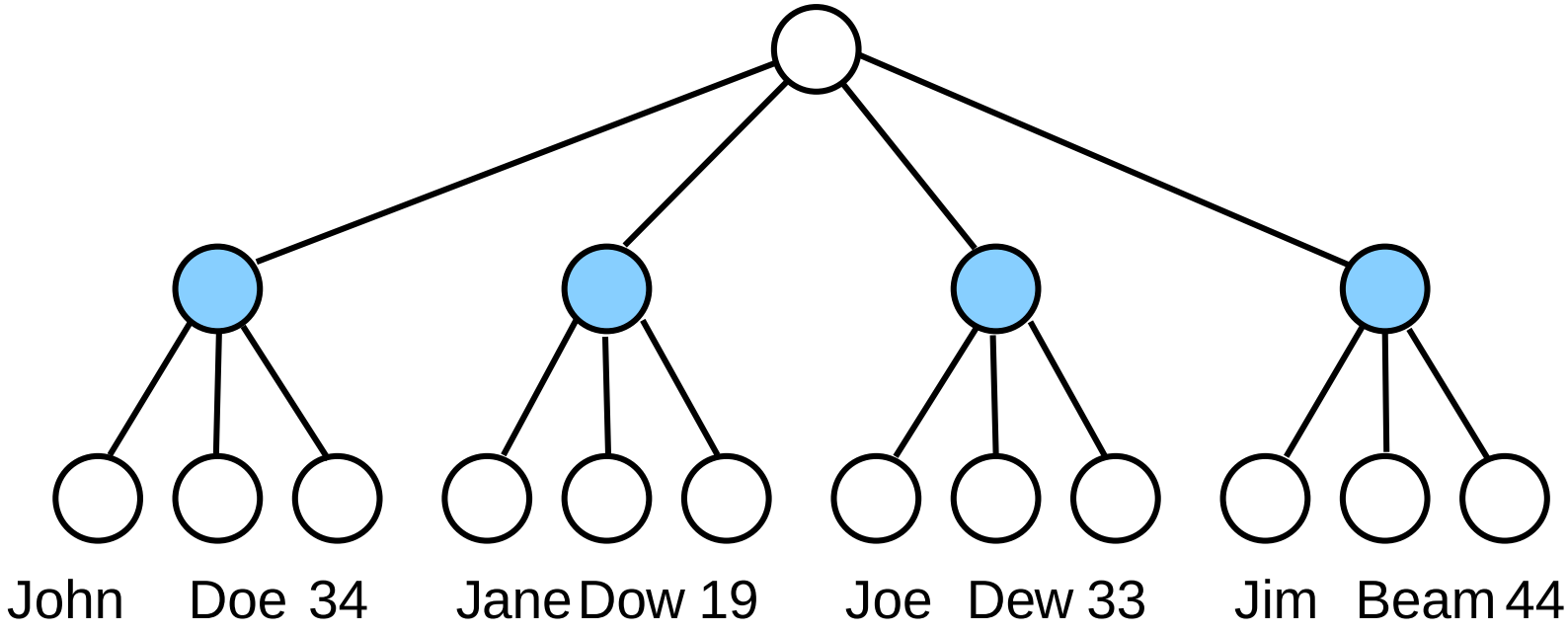
$\Rightarrow$ XQuery: designed to generalize SQL, as XML generalizes database tables

Relational tables & XML trees

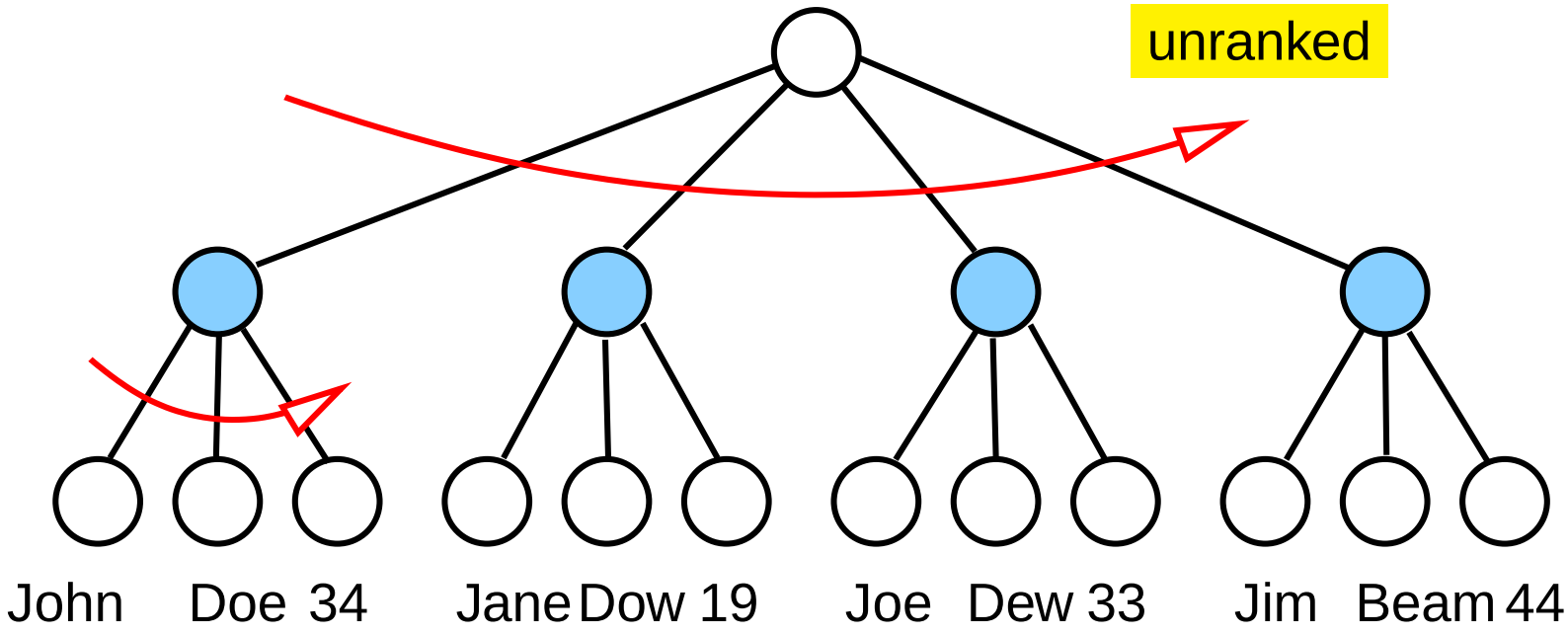
people(firstname,lastname,age)



Relational tables & XML trees



Relational tables & XML trees



From relations to trees

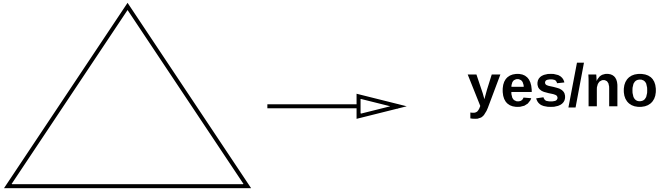
- obvious generalization of relational tables
  1. height two
  2. root with an unbounded number of child nodes
  3. nodes in the 2nd layer: fixed number of childs

Aspects of XML querying

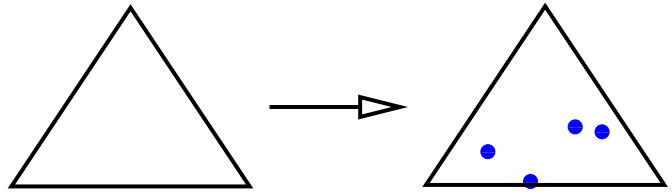
querying = matching/extraction/navigation
+ iteration
+ recombination/transformation
+ validation

Aspects of XML querying

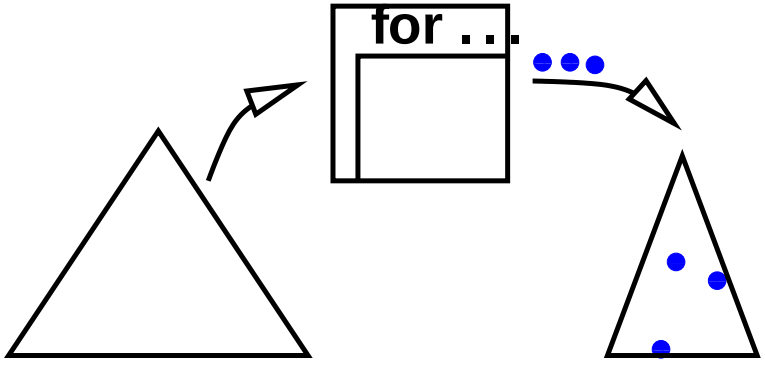
validation: schema



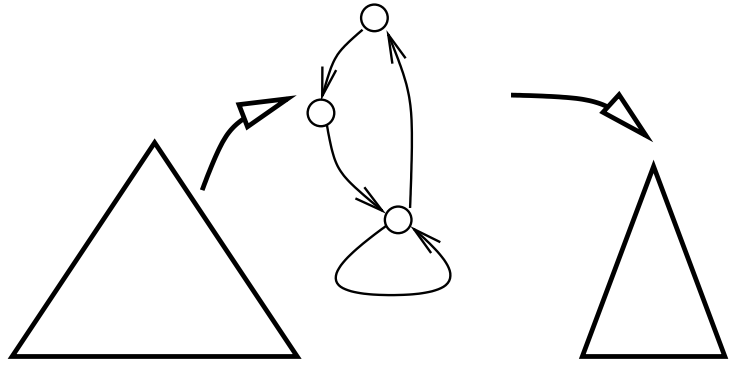
navigation: XPath



iteration: XQuery

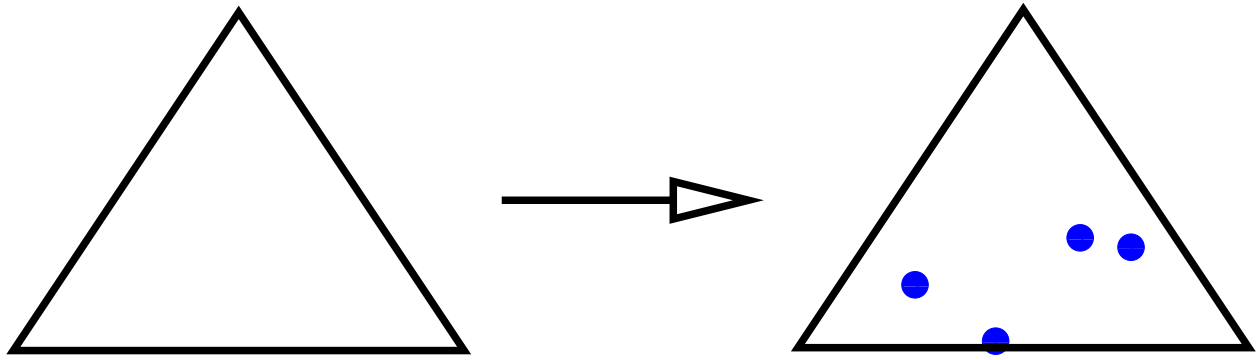


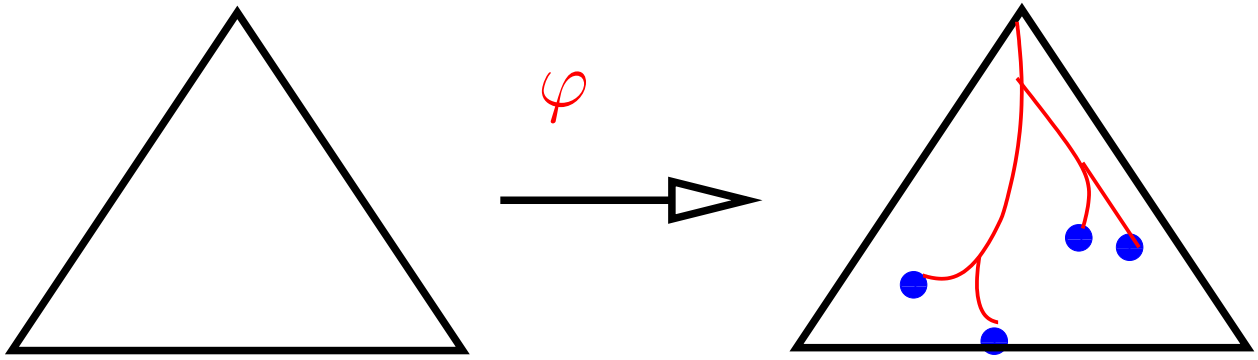
transformation: XLST



XPath [Clark and DeRose, 1999]

- typical path-based selection technology
- navigational primitives
- pinpoint where to capture data sub-components
- similar to path expressions in Lorel [Abiteboul et al., 1997] and Cω [Biermann et al., 2005]
- current version: XPath2.0
  - part of XQuery and XSLT
  - quite more expressive than XPath1.0





- φ = path expression = predicate on paths

XPath-navigation

- in principle: node selection by predicates on paths
- more concretely: sequence of location steps:

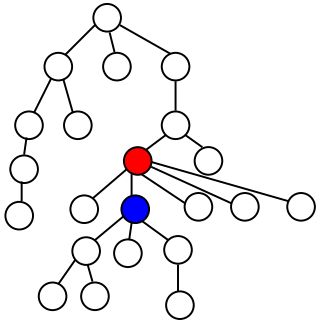
Node tests and predicates

1. axis : probe into the tree along paths
2. node test : check for the nature of the node (text, element etc)
3. predicates : general expressions (checking attributes, values, comparisons, variables . . .)

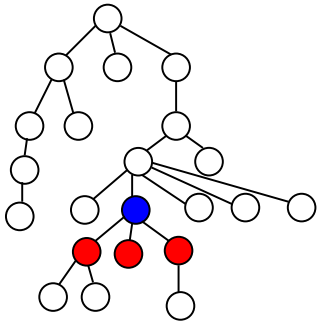
- evaluation
  - left to right
  - result(s): sequences of nodes.

Axes

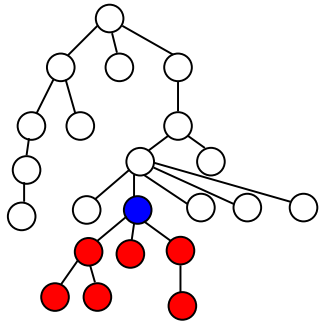
parent



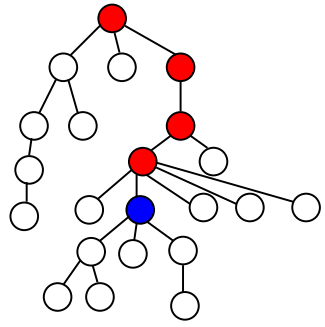
child



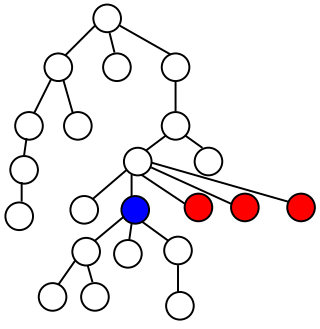
descendant



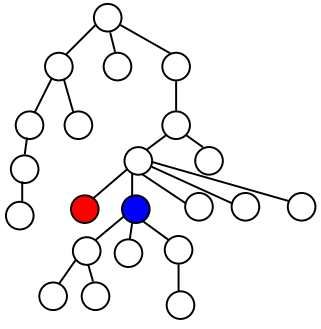
ancestor



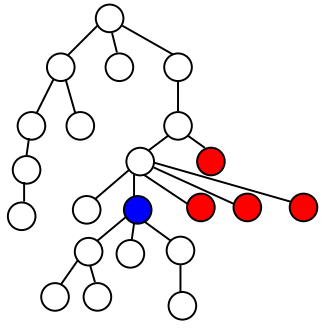
foll. sibling



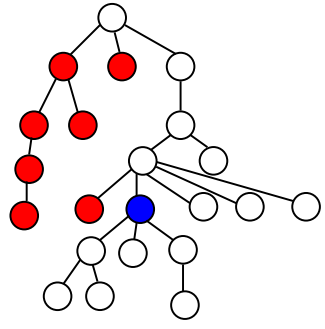
prec. sibling



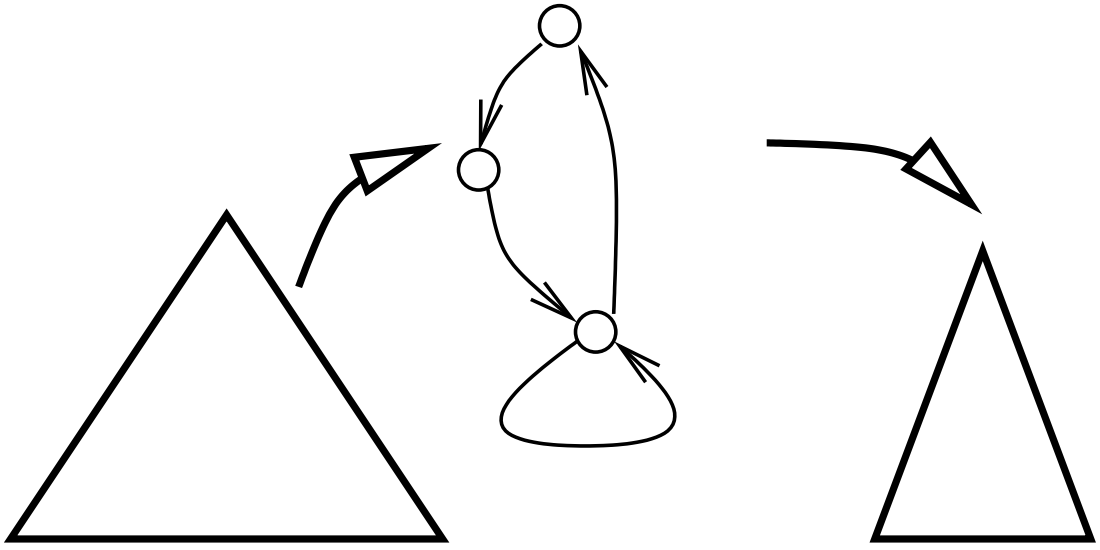
following



preceding



XSLT



XSLT

- XSL= *extensible stylesheet language*, XSLT= XSL transformations [1]
- original purpose: XML $\mapsto$ HTML, i.e., “generalized stylesheets”
- current version XSLT2.0: full-fledged query language [Clark, 1999]
- contains XPath2.0

[1] There is additionally XSL-FO for physical layout.

```
<xsl:template match="b:card">
  <html>
    <head>
      <title><xsl:value-of select="b:name/text()"/></title>
    </head>
    <body bgcolor="#ffffff">
      <table border="3">
        <tr>
          <td>
            <xsl:apply-templates select="b:name"/><br/>
            <xsl:apply-templates select="b:title"/><p/>
            <tt><xsl:apply-templates select="b:email"/></tt><br/>
            <xsl:if test="b:phone">
              Phone: <xsl:apply-templates select="b:phone"/><br/>
            </xsl:if>
          </td>
          <td>
            <xsl:if test="b:logo">
              <img src="business_card_files/uri.html"/>
            </xsl:if>
          </td>
        </tr>
      </table>
    </body>
  </html>
</xsl:template>

<xsl:template match="b:name|b:title|b:email|b:phone">
  <xsl:value-of select="text()"/>
</xsl:template>

</xsl:stylesheet>
```

```
<xsl:stylesheet version="2.0"
    xmlns:xsl="http ...." >

    <xsl:template match= "..." >
        ...       <xsl:apply-templates select= "..."/ >
                  ...
    </xsl:template match>
      ....

<xsl:stylesheet>
```

- set of **template rules**

```
<xsl:template match="...">
    ...      <xsl:apply-templates select="..."/>
             ...
</xsl:template match>
```

  - left-hand side: **pattern match**
  - right-hand side: "**sequence constructor**" = output + recursive descent

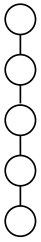
- **matching** strategy:
  1. select all matching left-hand sides
  2. select: most specific one
  3. evaluate: sequence constructor (by recursion)
- pattern: (restricted) XPath expression

Complications

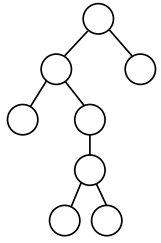
- modes: states of a tree transducer
- iteration: similar as in XQuery (see later)
- variables $\Rightarrow$ parameter passing
- of lesser impact: names and direct calls of templates, priorities, copying nodes, grouping, functions etc.

Strings / trees

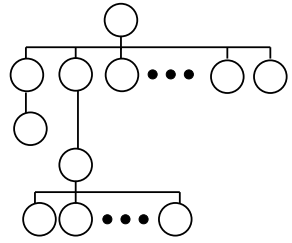
string ($n \leq 1$)



ranked tree ($n \leq 2$)



unranked tree (*)



ordered trees: sequence of children

From unranked to ranked

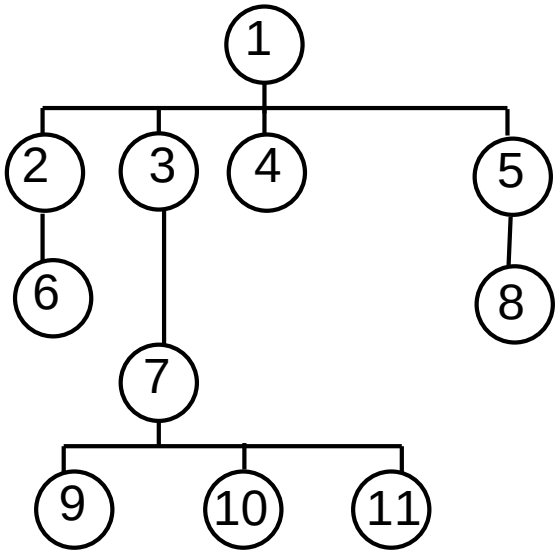
- many “classical” results on ranked trees
  [Rozenberg and Salomaa, 1996] [Thomas, 1990] [Comon et al., 2005]

$\Rightarrow$ two approaches

1. generalize theories
2. reduce to the unranked case

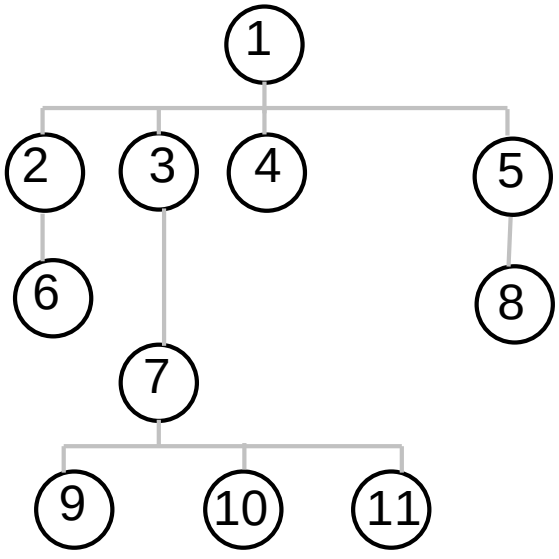
From unranked to ranked

unranked tree



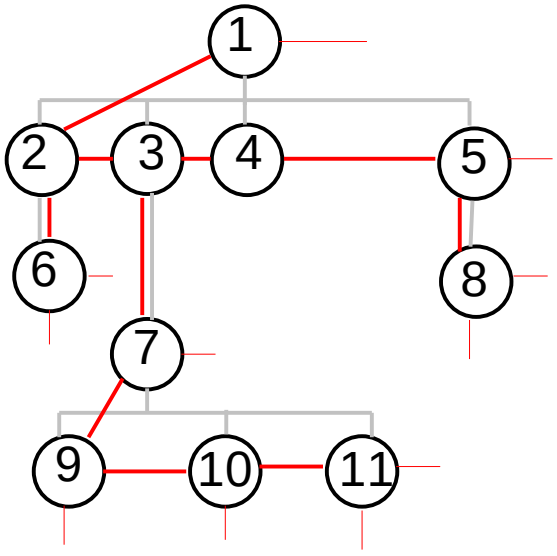
From unranked to ranked

unranked tree



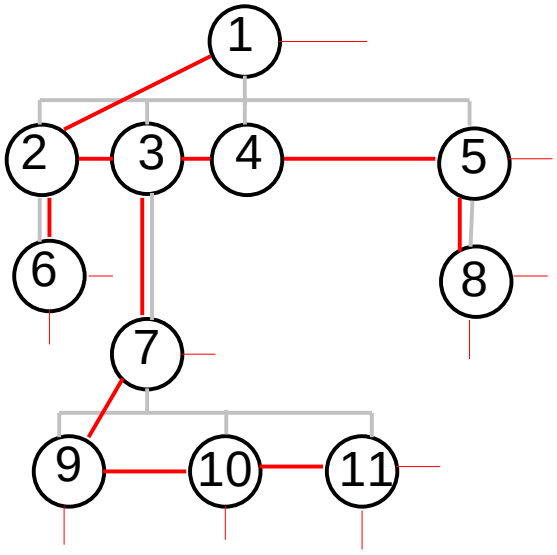
From unranked to ranked

unranked tree

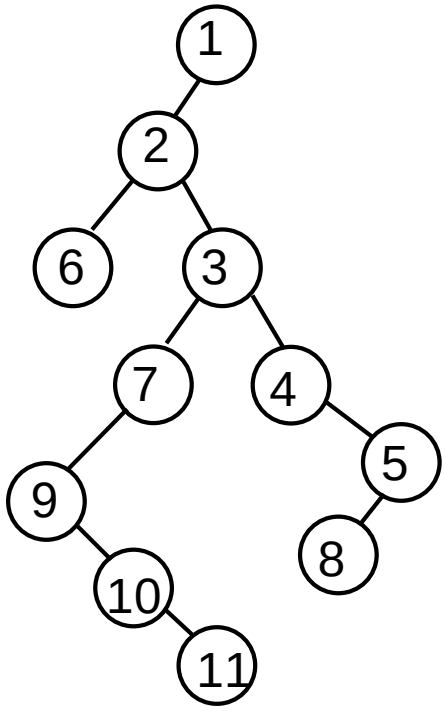


From unranked to ranked

unranked tree

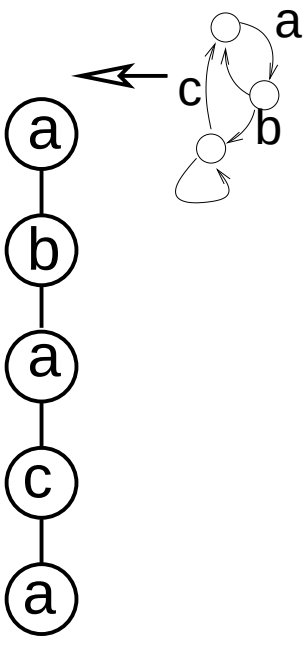


rank 2

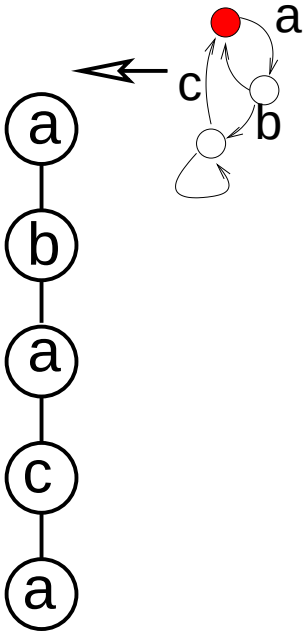


note: there are other encodings possible, too

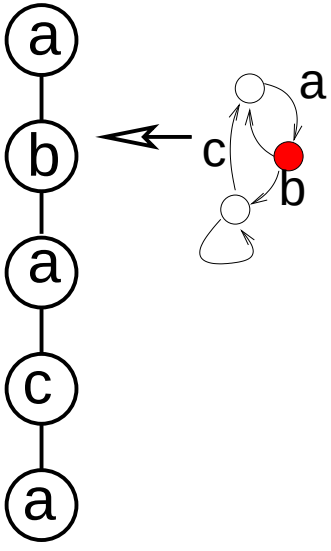
String automaton



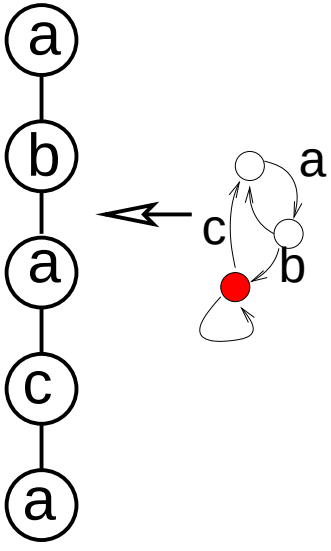
String automaton



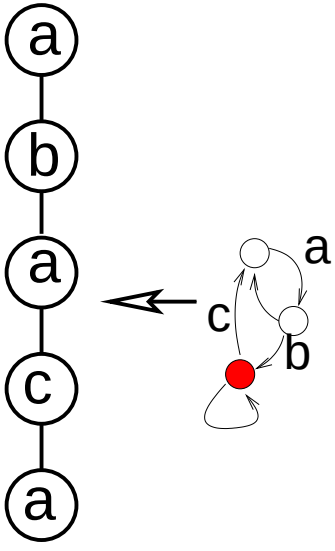
String automaton



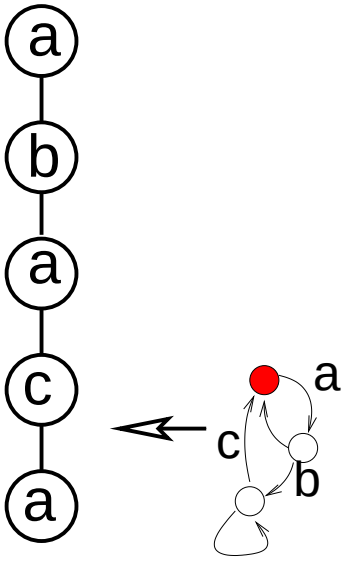
String automaton



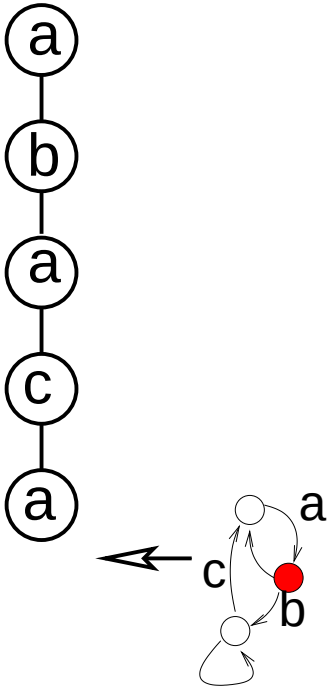
String automaton



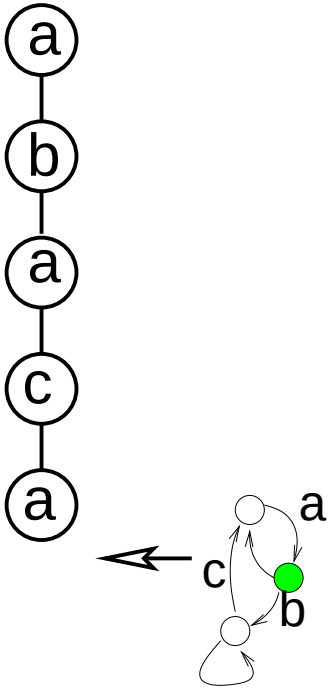
String automaton



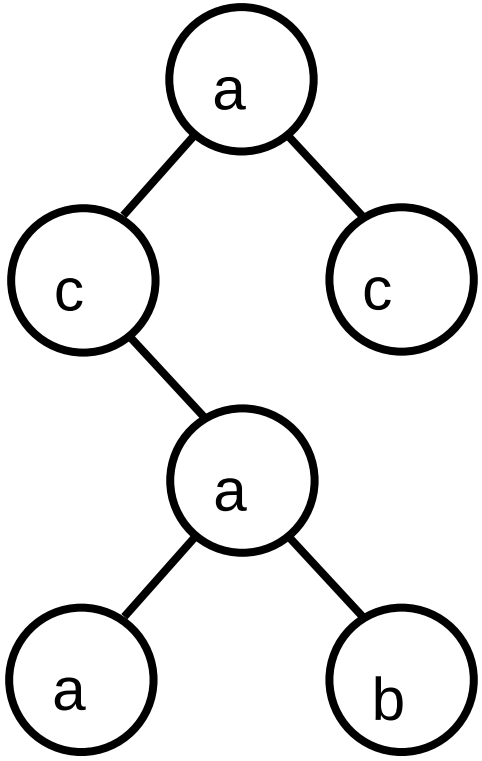
String automaton



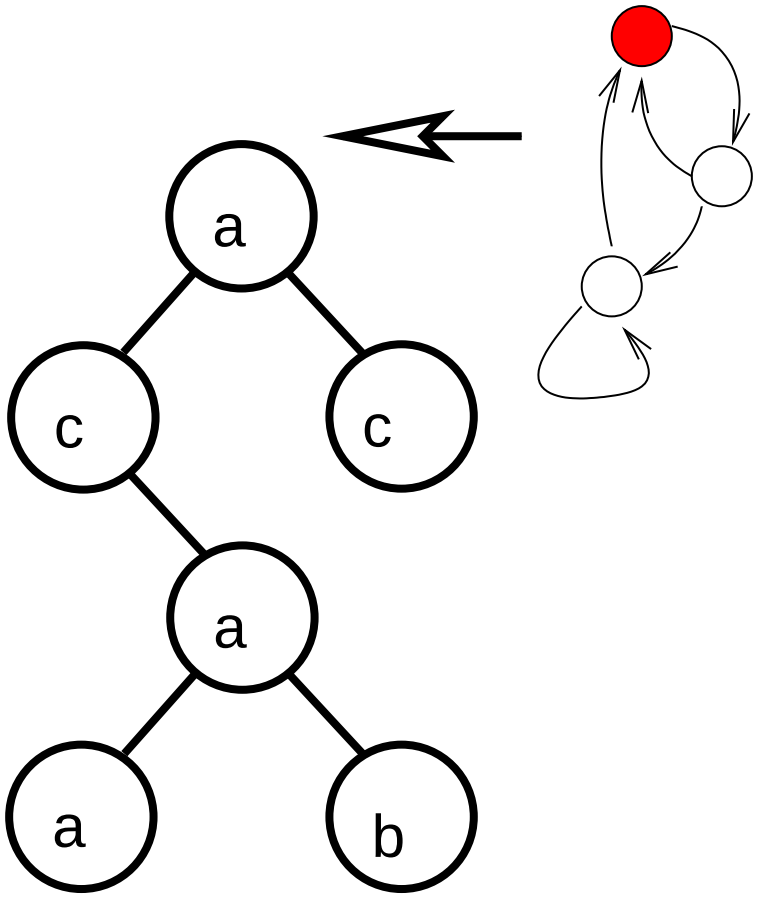
String automaton



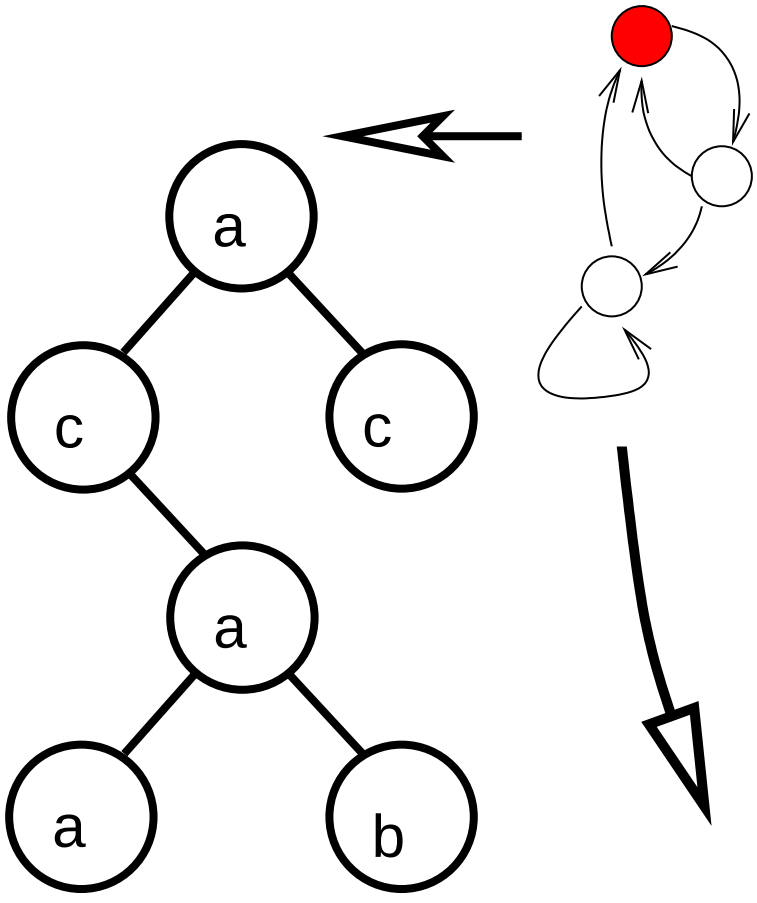
Automata on trees



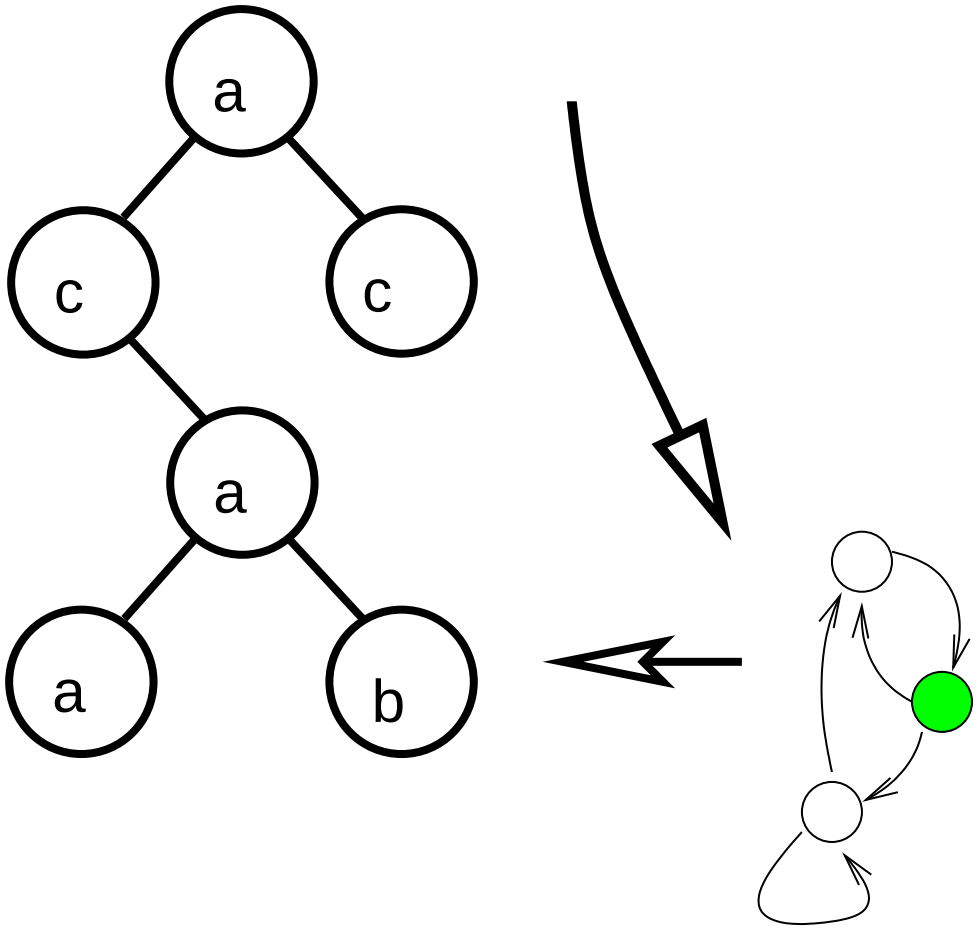
Automata on trees



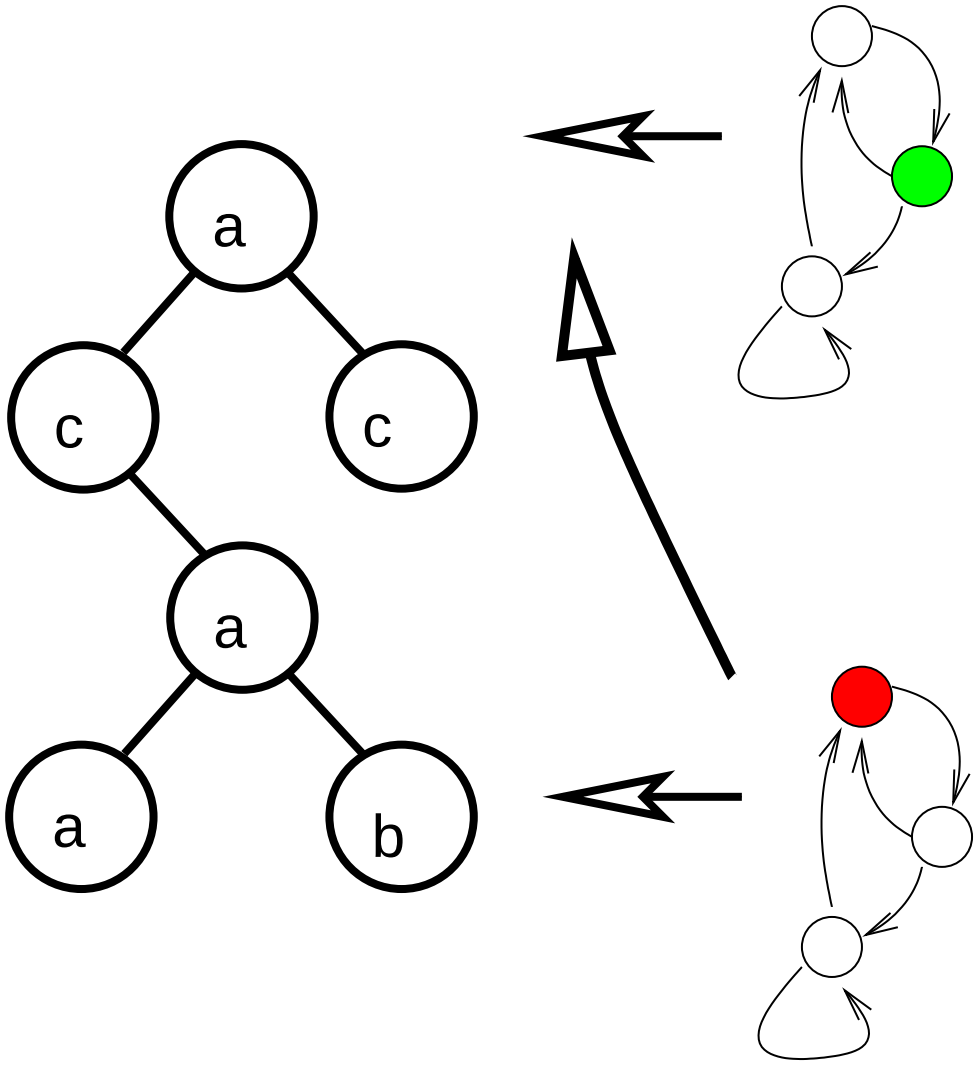
Automata on trees



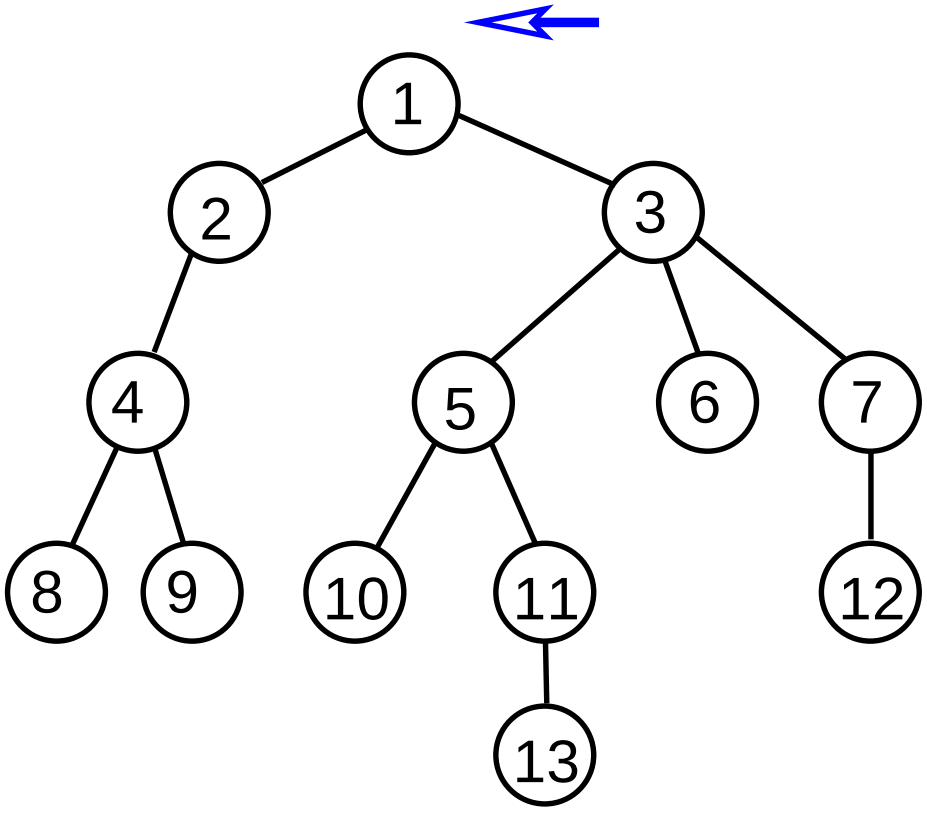
Automata on trees



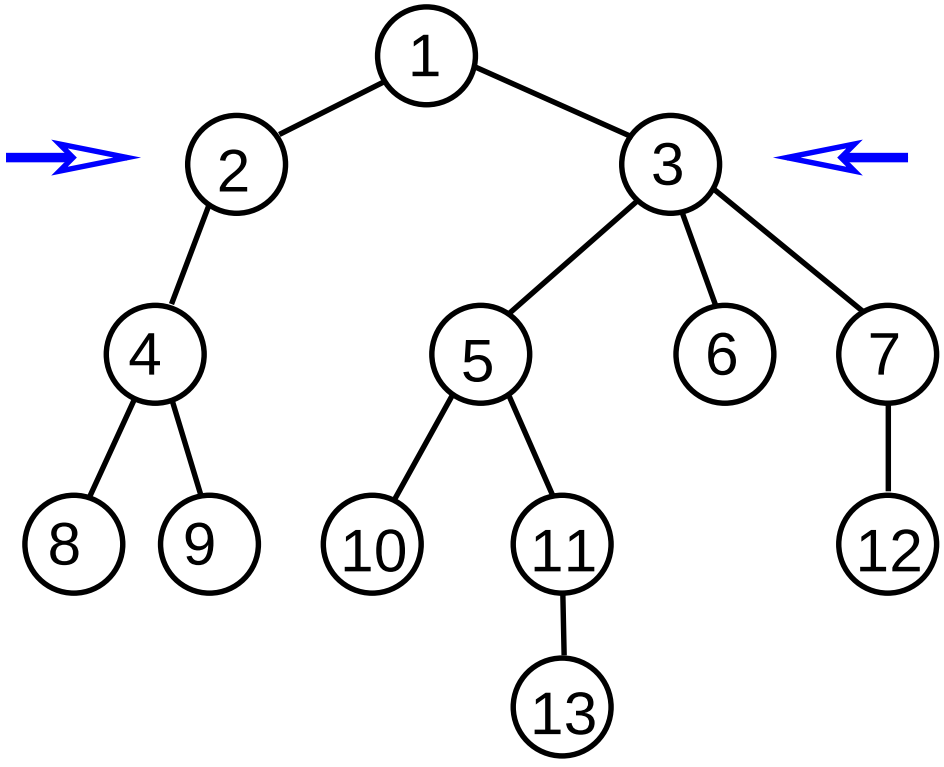
Automata on trees



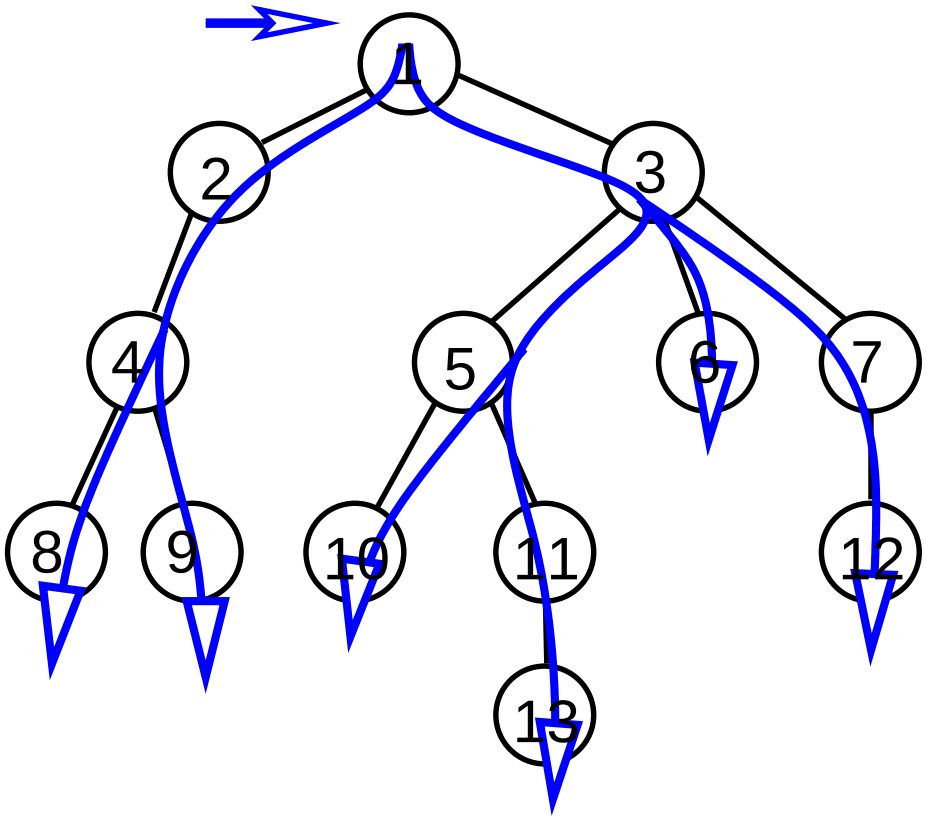
Parallel vs. sequential machines



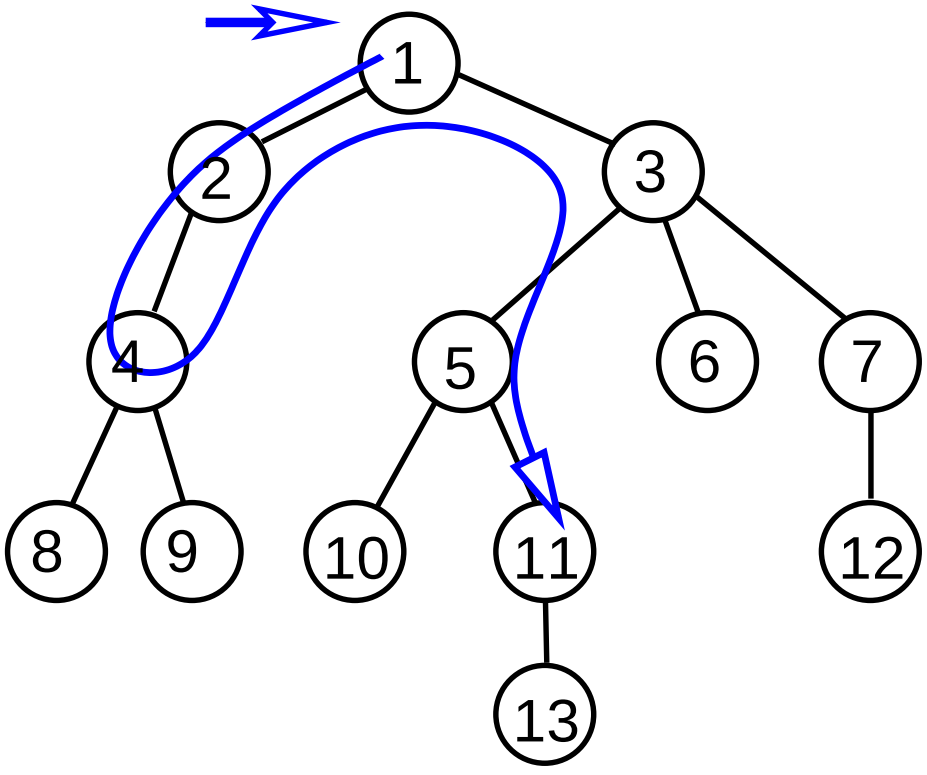
Parallel vs. sequential machines



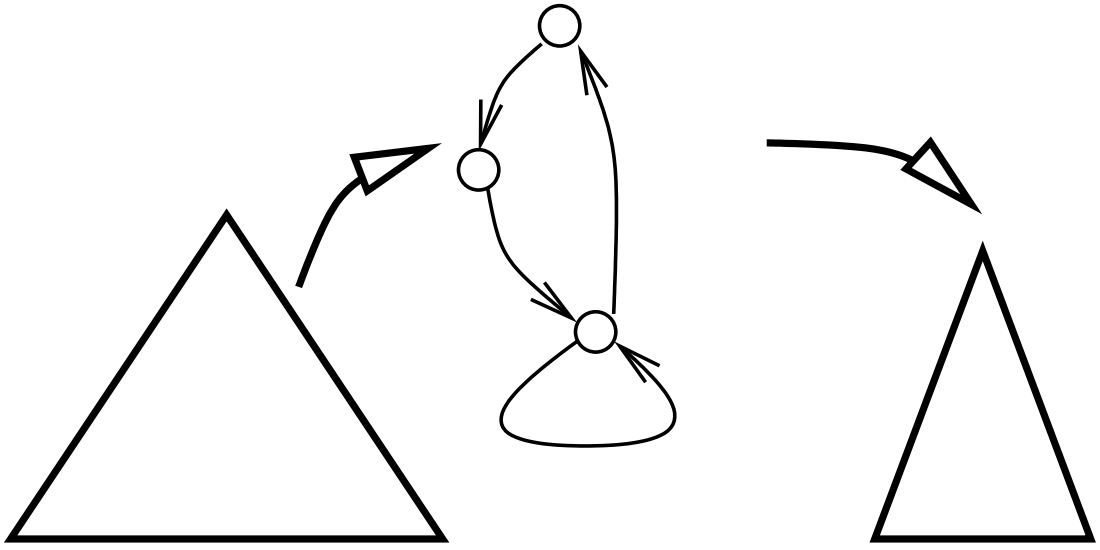
Parallel vs. sequential machines



Parallel vs. sequential machines: tree walking



XSLT



Transduction

```
<xsl:stylesheet ...>

   <xsl:template match= "a", mode = "q0">
      ...      <xsl:apply–templates select= "..." mode="q1"/>
                ...
    </xsl:template match>
     ....

<xsl:stylesheet>
```

Transduction

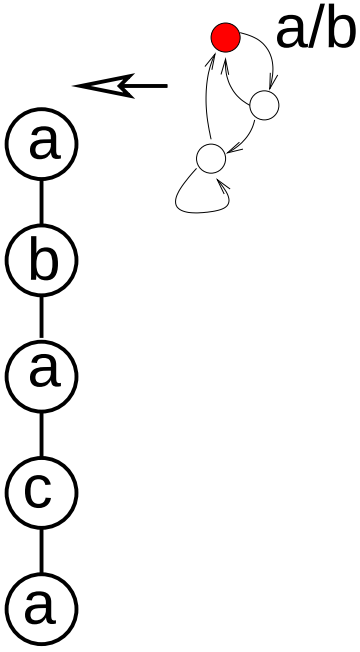
```
<xsl:stylesheet ...>

   <xsl:template match= "a", mode = "q0">
      <node> ...     <xsl:apply–templates select= "..." mode="q1"/>
      </node>            ...
    </xsl:template match>
     ....

<xsl:stylesheet>
```

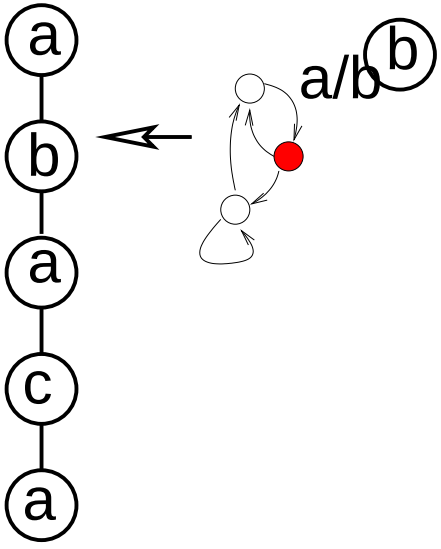
Transducers

- machines so far: acceptors: (yes/no)
- needed: transducers



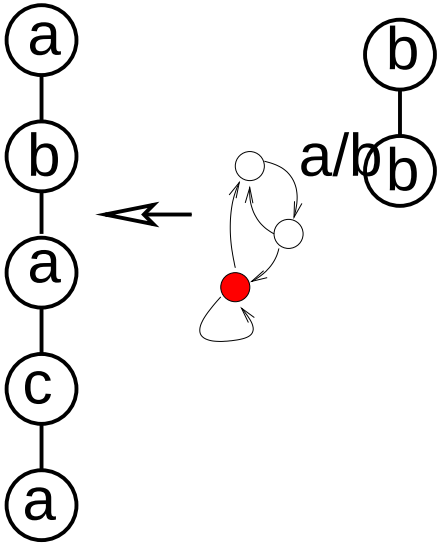
Transducers

- machines so far: acceptors: (yes/no)
- needed: transducers



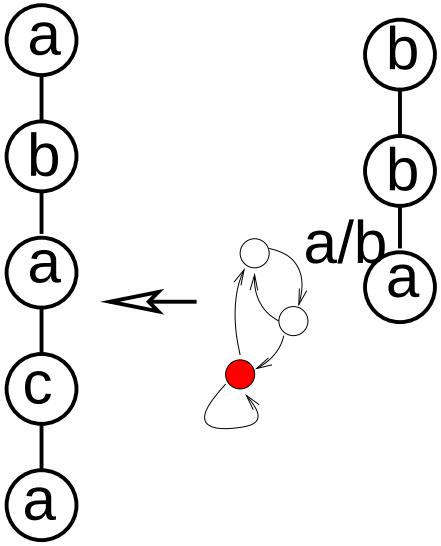
Transducers

- machines so far: acceptors: (yes/no)
- needed: transducers



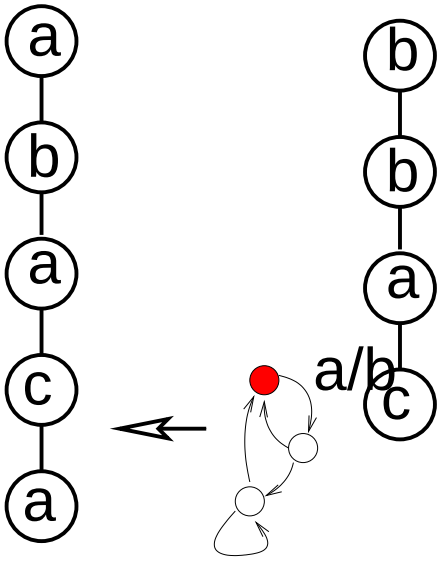
Transducers

- machines so far: acceptors: (yes/no)
- needed: transducers



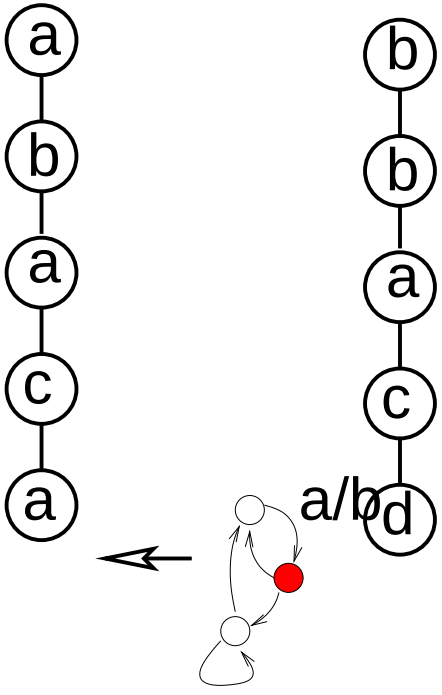
Transducers

- machines so far: acceptors: (yes/no)
- needed: transducers



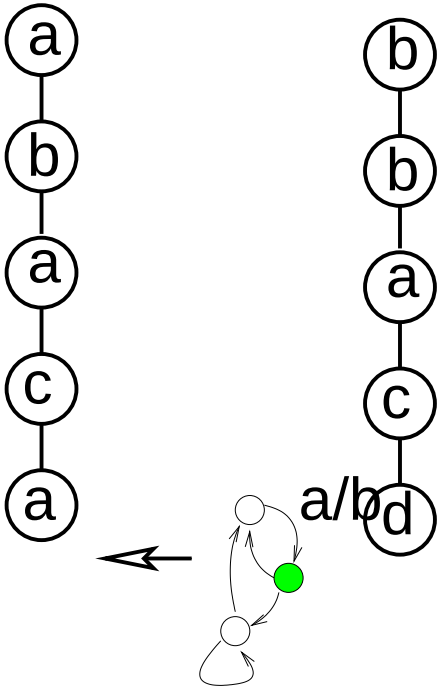
Transducers

- machines so far: acceptors: (yes/no)
- needed: transducers



Transducers

- machines so far: acceptors: (yes/no)
- needed: transducers



Generalization on trees

- analogous to the generalization from automata to trees.
- tree transducer:

$$q(f(x_1, \ldots, x_n)) \rightarrow h(q_1(x_1), \ldots, q_n(x_n))$$

Formal tree transducer models

- still no consensus
- many models
- tree walking
  - used to model XSLT [Bex et al., 2002], (and type checking [Milo et al., 2000]) [Neven, 2000]
  - [Bex et al., 2002] formal model based an tree walking transducer
  - expressive *pebble* automata/transducer, used for
    - XML-QL[Deutsch et al., 1999]
    - Lorel [Abiteboul et al., 1997]
    - UnQL[Buneman et al., 1996]
    - StruQL [Fernandez et al.,]
    - fragment of XSLT
  - still open: relationship to tree automata [Engelfriet et al., 1999] [Engelfrient and Hoogebook, 1999] [Neven and Schwentick, 2000]

XQuery2.0

- current W3C proposal
- in a nutshell (i.e., besides practical aspects) extension of XPath2.0 by
  - constructing of
  - joining of
  - iterating over

  XML-data

Expressions

- ”standard” data expression
- XPath expressions (navigation)
- XML-constructing expressions
- FLWOR -expressions
  - iteration
  - join

FLWOR

for:	iteration
let:	binding
where:	filter
order	
return:	construction

- cf. *SELECT* from SQL, also `for` in XSLT

```
<floury>
  { for $r in
      fn:doc("recipes.xml")//
      rcp:recipe[.//rcp:ingredient[@name="flour"]]
    return <dish>{$r/rcp:title/text()}</dish>
  }
</floury>
```

XSLT vs. XQuery

- different original intentions

$\Rightarrow$ different pragmatical strengths

- as full languages: both Turing complete [Kepser, 2004]

	XSLT	XQuery
origin	style sheet language	SQL . . .
status	W3C recommendation	W3C recommendation
abstraction level	lower level, procedural	declarative
strength:	XML-tree transformations by recursive descent	querying, joins
human readable syntax	no	yes

Querying, validation, & XML-processing

- schema validation (and static typing) optional in XQuery
- static vs. dynamic typing
- based on the analogy

schema	=	type
validation	=	type checking

Static typing and pattern matching for querying

- pionieered by XDuce [Hosoya and Pierce, 2003]
- two alternative ways of deconstructing XML-docs, from different “communities” [Castagna, 2005]
  - vertical: “path”-expressions (XPath most notably)
  - horizontal: regular expression pattern matching
- challenges
  - ML type system is not expressive/flexible enough for XML-data
  - adapt pattern matching + type inference
  - subtyping
  - advanced: parametric polymorphism

Pattern matching

```
let (x,y) = (1,"abc") in e'
```

- data value $(1, "abc")$, pattern (x, y),
- value
  - matched against pattern
  - result: substitution / binding: $x \mapsto 1, y \mapsto "abc"$
- pattern: value, some subterms replaced by (capture) variable
- match, using first match

```
match e with
  | (_,_) -> true
  | _     -> false
```

- wildcard _ (“don’t care”)

XDuce

- first-order domain-specific functional language [Hosoya et al., 2002]
- influential language design
- foundation for static typing of XML-processing
- static type-checking
  - often only data/dynamic type checking
  - mainstream of XML-processing: only data checking, validating parser
  - no systematic connection: program $\leftrightarrow$ type

$\Rightarrow$ regular expression type

Example

```
<addrbook>
  <person>
    <name>Joe Doe</name>
    <email>doe@cba.org</email>
  </person>
  <person>
    <name>Jane Dow</name>
    <email>dow@egd.com</email>
    <tel>123-456-788</tel>
  </person>
</addrbook>
```

Example

```
type Addrbook  = addrbook[Person*]
type Person    = person[Name, Email*,Tel?]
type Name      = name[String]
type Email     = email[String]
type Tel       = tel[String]
```

Regular expression types

$$
\begin{array}{rcll}
T & ::= & () & \text{empty sequence} \\
 & | & X & \text{type variables} \\
 & | & l[T] & \text{label} \\
 & | & T, T & \text{concatenation} \\
 & | & T \mid T & \text{union} \\
 & | & \emptyset & \text{empty set/type}
\end{array}
$$

- regular expressions as derived forms: $T*$, $T?$, $T+$.
- well-formedness restriction on the variables, ensuring regularity

Acceptance relation

$$\frac{t : M(X)}{t : X}\ \text{T-State} \qquad \frac{}{() : ()}\ \text{T-Empty}$$

$$\frac{v : T_1}{v : T_1 \mid T_2}\ \text{T-Or}_1 \qquad \frac{v : T_2}{v : T_1 \mid T_2}\ \text{T-Or}_2$$

$$\frac{t_1 : X_1 \qquad t_2 : X_2}{l[t_1, t_2] : l[X_1, X_2]}\ \text{T-Lab}$$

Pattern matching and type inference

- good/usual approach: type checking = type inference + subtyping
- avoid excesssive annotations
- local type inference, locally precise

assume input `Person = person[Name, Email*, Tel?]`
match expression:

```
match p with
  person[name[n],tel[t]]
    -> ...
| person[name[n],rest]
    -> ..
```

inferred types:

- *t*, *n*: strings
- **rest**: ?

Pattern matching and type inference

- good/usual approach: type checking = type inference + subtyping
- avoid excesssive annotations
- local type inference, locally precise

assume input `Person = person[Name, Email*, Tel?]`
match expression:

```
match p with
  person[name[n],tel[t]]
    -> ...
| person[name[n],rest]
    -> ..
```

inferred types:

- *t*, *n*: strings
- **rest**: `(Email*, Tel?)`

Pattern matching and type inference

- good/usual approach: type checking = type inference + subtyping
- avoid excesssive annotations
- local type inference, locally precise

assume input `Person = person[Name, Email*, Tel?]`
match expression:

```
match p with
  person[name[n],tel[t]]
    -> ...
| person[name[n],rest]
    -> ..
```

inferred types:

- *t*, *n*: strings
- **rest**: `(Email+, Tel?) | ()`

Core idea

- pattern and types are tree automata
  - patterns are types (or tree automata) with capture variables
  - core algorithmic question: given input type T and pattern P

> do they match ? if so: infer the most-precise types for the matching vars.

$$\frac{?}{\Pi \vdash T \triangleright P_1 \mid P_2 :: \Pi_2; (\Gamma_1 \mid \Gamma_2)}$$

- note: intersection and difference of tree automata (resp. tree languages)
- take care: recursion

Core idea

- pattern and types are tree automata
  - patterns are types (or tree automata) with capture variables
  - core algorithmic question: given input type T and pattern P

> do they match ? if so: infer the most-precise types for the matching vars.

$$\frac{\Pi \vdash (T \cap P_1) \triangleright P_1 :: \Pi_1; \Gamma_1 \qquad \Pi_1 \vdash (T \setminus P_1) \triangleright P_2 :: \Pi_2; \Gamma_2}{\Pi \vdash T \triangleright P_1 \mid P_2 :: \Pi_2; (\Gamma_1 \mid \Gamma_2)}$$

- note: intersection and difference of tree automata (resp. tree languages)
- take care: recursion

Type inference

$$\frac{\Pi \vdash (T \cap P_1) \triangleright P_1 :: \Pi_1; \Gamma_1 \qquad \Pi_1 \vdash (T \setminus P_1) \triangleright P_2 :: \Pi_2; \Gamma_2}{\Pi \vdash T \triangleright P_1 \mid P_2 :: \Pi_2; (\Gamma_1 \mid \Gamma_2)}\ \textsc{InfA-Or}_1$$

$$\frac{\Pi \vdash T_1 \triangleright P :: \Pi_1; \Gamma_1 \qquad \Pi_1 \vdash T_1 \triangleright P :: \Pi_2; \Gamma_2}{\Pi \vdash T_1 \mid T_2 \triangleright P :: \Pi_2; (\Gamma_1 \mid \Gamma_2)}\ \textsc{InfA-Or}_2$$

$$\frac{I(X_1, X_2) \not\leq \emptyset \qquad \Pi \vdash X_1 \triangleright Y_2 :: \Pi_1; \Gamma_1 \qquad \Pi_1 \vdash X_2 \triangleright Y_2 :: \Pi_2; \Gamma_2}{\Pi \vdash I(X_1, X_2) \triangleright I(Y_1, Y_2) :: \Pi_2; (\Gamma_1 \mid \Gamma_2)}\ \textsc{InfA-Lab}$$

Further properties

- type checking = type inference + subtyping
- semantic subtyping
  - type = set of values/tress
  - subtyping = inclusion of tree languages
- more advanced features:
  - parametric polymorphism = generic schemas
  - basis for pattern-matching querying optimization
- adopted widely: CDuce/CQL, Xtatic, Scala, XHaskell
- also: basis of type checking for XQuery

Trends

- robust theory
- algebra
- domain specific languages
- optimization
- non-exact matching

Left out

- XML-technologies:
  - name spaces
  - most of actual schema languages: XMLSchema, DTD (but there are many more)
  - parts of the technology covered has been simplified, especially XSLT

Material I

Primary sources for XML and related technologies is the web, especially the site of the W3C, where the recommendations and proposals are available. A general up-to date book about XML is [Møller and Schwartzbach, 2006]. The standard references for XQuery are [Boag et al., 2003]
[Frankhauser et al., 2001]. A formal description of XPath and XQuery is also available as official W3C recommendation since recently
[Draper et al., 2007]. XSLT is described in [Clark, 1999].
Concerning typing and pattern matching . . . , the original material is
[Hosoya and Pierce, 2003] [Hosoya et al., 2002] [Hosoya et al., 2005], and also the thesis [Hosoya, 2001].
Background literature on tree automata and related issues can be found in
[Rozenberg and Salomaa, 1996] [Comon et al., 2005] [Thomas, 1990].
Specifically in the context of XML, I consulted [Vianu, 2001], [Neven, 2002b] (or a bit deeper [Neven, 2002a]). A short comparison of various language designs for various XML query languages from various research groups present in the W3C XQuery working group (and thus influential on the design of XQuery) can be found [Fernandez et al., 1999].

References I

[Abiteboul et al., 1997] Abiteboul, S., Quass, D., McHugh, J., Widom, J., and Wiener, J. (1997).
The Lorel query language for semistructured data.
International Journal on Digital Libraries, 1(1):68–88.

[Bex et al., 2002] Bex, G. J., Maneth, S., and Neven, F. (2002).
A formal model for an expressive fragment of XSLT.
Information Systems, 27(1):21–39.

[Biermann et al., 2005] Biermann, G., Meijer, E., and Schulte, W. (2005).
The essence of data access in Cω.
In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages (ECOOP)*.

[Biron and Malhotra, 2004] Biron, P. V. and Malhotra, A. (2004).
XML Schema part 2: Datatypes Second Edition.
W3C Recommendation.

[Boag et al., 2003] Boag, S., Chamberlin, D., Fernandez, M. F., Florescu, D., Robie, J., Siméon, J., and Stefanescu, M. (2003).
XQuery 1.0: an XML Query Language.
W3C Working draft.

[Bray et al., 2004a] Bray, T., Hollaender, D., Layman, A., and Tobin, R. (2004a).
Namespaces in XML 1.1.
W3C Recommendation, available at `www.w3c.org/TR/xml11`.

[Bray et al., 2004b] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yeargeau, F., and Cowan, J. (2004b).
Extensible Markup Language (XML) 1.1.
W3C Recommendation.

[Buneman et al., 1996] Buneman, P., Davidson, S., Hillebrand, G. G., , and Suciu, D. (1996).
A query language and optimization for unstructured data.
In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, volume 25 of *SIGMOD Records*, pages 505–516. ACM Press.

References II

[Castagna, 2005] Castagna, G. (2005).
Patterns and types for querying XML documents.
Keynote talk DBPL - XSYM 2005.

[Clark, 1999] Clark, J. (1999).
XSLT Transformations (XSLT).
W3C.
W3C Recommendation.

[Clark and DeRose, 1999] Clark, J. and DeRose, S. (1999).
XML path language (XPath).
`http://www.w3c.org/TR/xpath`.
W3C Recommendation.

[Comon et al., 2005] Comon, H., Dauchet, M., Gilleron, R., Lugiez, D., Tison, S., and Tommasi, M. (2005).
Tree automata, techniques and application.
Available electronically.

[Deutsch et al., 1999] Deutsch, A., Fernandez, M. F., Florescu, D., Levy, A. Y., and Suciu, D. (1999).
XML-QL: A query language for XML.
In *WWW The Query Language Workshop (QL)*.

[Draper et al., 2007] Draper, D., Fankhauser, P., Fernández, M., Malhotra, A., Rose, K., Rys, M., Siméon, J., and Wadler, P. (2007).
XQuery 1.0 and XPath 2.0 formal semantics.
available at `http://www.w3.org/TR/xquery-semantics/`.
W3C Recommendation 23 January 2007.

[Engelfriient and Hoogebook, 1999] Engelfrient, J. and Hoogebook, H. (1999).
Tree-walking pebble automata.
In *Jewels are Forever. Contributions to Theoretical Computer Science in Honor of Arto Salomaa*, pages 72–83.
Springer-Verlag.

References III

[Engelfriet et al., 1999] Engelfriet, J., Hoogenboom, H., and Best, J. V. (1999).
Trips on trees.
Acta Cybernetica, 14:51–64.

[Fernandez et al.,] Fernandez, M., Siméon, J., Kang, J., Levy, A. Y., and Suciu, D.
Catching a boat with a strudel: Expreriences with a web-site management system.
In Haas, L. M. and Tiwary, A., editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 414–425.

[Fernandez et al., 1999] Fernandez, M. F., Siméon, J., and Wadler, P. L. (1999).
XML query languages: Experiences and exemplars.
Technical report, Bell Labs.
available at `www-db.research.bell-labds.com/users/simeon/xquery.html`.

[Frankhauser et al., 2001] Frankhauser, P., Fernández, M., Malhotra, A., Rys, M., Siméon, J., and Wadler, P. L. (2001).
Xquery 1.0. formal semantics.
`www.w3.org/TR/query-semantics`.

[Hosoya, 2001] Hosoya, H. (2001).
Regular Expression Types for XML.
PhD thesis, University of Tokyo.

[Hosoya et al., 2005] Hosoya, H., Frisch, A., and Castagna, G. (2005).
Parametric polymorphism for XML.
In *Proceedings of POPL '05*, pages 55–62. ACM.

[Hosoya and Pierce, 2003] Hosoya, H. and Pierce, B. C. (2003).
XDuce: A typed XML-processing language.
ACM Transactions on Internet Technology, 3(2):117–148.

References IV

[Hosoya et al., 2002] Hosoya, H., Vouillon, J., and Pierce, B. C. (2002).
Regular expression types for XML.
ACM Transactions on Programming Languages and Systems.
To appear. Short version in ICFP 2000.

[Kepser, 2004] Kepser, S. (2004).
A simple proof of the Turing-completeness of XSLT and XQuery.
In *Extreme Markup Languages*.

[Malhotra and Maloney, 1999] Malhotra, A. and Maloney, M. (1999).
XML Schema Requirements.
W3C Note.

[Milo et al., 2000] Milo, T., Suciu, D., and Vianu, V. (2000).
Typechecking for XML transformers.
In *Proceedings of the Ninetenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Databas Systems*, pages 11–22. ACM.

[Møller and Schwartzbach, 2006] Møller, A. and Schwartzbach, M. I. (2006).
An Introduction to XML and Web Technologies.
Addison-Wesley.

[Neven, 2000] Neven, F. (2000).
On the power of walking for querying tree-structured data.
In *Proceedings of the 21st Symposium on Principles of Database Systems (PODS 2000)*, pages 77–84. ACM Press.

[Neven, 2002a] Neven, F. (2002a).
Automata, logic, and XML.
In *Proceedings of CSL 2002*, volume 2471 of *Lecture Notes in Computer Science*, pages 2–26. Springer-Verlag.

[Neven, 2002b] Neven, F. (2002b).
Automata theory for XML researchers.
SIGMOD Record, 31(3).

References V

[Neven and Schwentick, 2000] Neven, F. and Schwentick, T. (2000).
On the power of tree-walking automata.
In Montanari, U., Rolim, J., and Welzl, E., editors, *Proceedings of ICALP 2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 547–560. Springer-Verlag.

[Rozenberg and Salomaa, 1996] Rozenberg, G. and Salomaa, A. (1996).
Handbook of Formal Languages.
Springer.

[Thomas, 1990] Thomas, W. (1990).
Automata on infinite objects.
In *Formal Models and Semantics*, chapter 4, pages 134–191. Elsevier Science.
Handbook of Theoretical Computer Science, Volume B.

[Thompson et al., 2004] Thompson, H. S., Beech, D., Maloney, M., and Mendelson, N. (2004).
XML Schema part 2: Structures Second Edition.
W3C Recommendation.

[Vianu, 2001] Vianu, V. (2001).
From Codd to XML.
In *ACM PODS, Santa Barbara, CA*.