Dati semistrutturati in XML

Massimo Franceschet
m.franceschet@unich.it
www.sci.unich.it/~francesc

Università "Gabriele D'Annunzio" di Chieti e Pescara

What XML is

XML is eXtensible Markup Language.

- XML is a formal language. This means it is defined by a set of formal rules (a grammar) that say exactly how to compose an XML document.

What XML is

XML is eXtensible Markup Language.

- XML is a formal language. This means it is defined by a set of formal rules (a grammar) that say exactly how to compose an XML document.
- XML is markup. Data is included in XML document as strings of text and is surrounded by text markup that describes the data.

What XML is

XML is eXtensible Markup Language.

- XML is a formal language. This means it is defined by a set of formal rules (a grammar) that say exactly how to compose an XML document.
- XML is markup. Data is included in XML document as strings of text and is surrounded by text markup that describes the data.
- XML is extensible. The language allows an extensible set of markup tags that can be adapted to meet many different needs.

What XML is

XML is eXtensible Markup Language.

- XML is a formal language. This means it is defined by a set of formal rules (a grammar) that say exactly how to compose an XML document.
- XML is markup. Data is included in XML document as strings of text and is surrounded by text markup that describes the data.
- XML is extensible. The language allows an extensible set of markup tags that can be adapted to meet many different needs.

An XML document is well-formed if it satisfies the XML grammar.

What XML is

- The markup permitted in a particular XML application can be documented in a schema.

What XML is

- The markup permitted in a particular XML application can be documented in a schema.
- The most broadly supported schema language is Document Type Definition (DTD).

What XML is

- The markup permitted in a particular XML application can be documented in a schema.
- The most broadly supported schema language is Document Type Definition (DTD).
- An XML document is said valid if it matches the schema.

XML for people

Common scenarios in which XML can be used by people include:

- Writing a book using DocBook. DocBook is nonproprietary, portable, modular, and easy to use with any text editor and you may format the final version according to your needs.

XML for people

Common scenarios in which XML can be used by people include:

- Writing a book using DocBook. DocBook is nonproprietary, portable, modular, and easy to use with any text editor and you may format the final version according to your needs.
- Write a web page in XHTML. XHTML has a well-defined syntax, you can work with any XML tool and web search engines eventually will understand your document and properly index it.

XML for machines

Common scenarios in which XML can be used by machines include:

- Data exchange. Information comes in different sources (relations, objects, documents, ...) and it needs to be exchanged between these sources. XML acts as the common dataspeak.

XML for machines

Common scenarios in which XML can be used by machines include:

- Data exchange. Information comes in different sources (relations, objects, documents, ...) and it needs to be exchanged between these sources. XML acts as the common dataspeak.
- Semistructured databases. These data has no regular schema and does not naturally fit into relational databases. XML has been proposed as the data model for semistructured data.

What XML is not

- XML is not a presentation language like HTML.
  XML defines the structure of the document and the semantics (meaning) of the data, but it doesn't tell how the data should look.

What XML is not

- XML is not a presentation language like HTML.
  XML defines the structure of the document and the semantics (meaning) of the data, but it doesn't tell how the data should look.
- XML is not a programming language like Java.
  An XML document by itself simply is. It does not do anything.

What XML is not

- XML is not a presentation language like HTML.
  XML defines the structure of the document and the semantics (meaning) of the data, but it doesn't tell how the data should look.
- XML is not a programming language like Java.
  An XML document by itself simply is. It does not do anything.
- XML is not a network transport protocol like HTTP.
  XML won't send data across the network.

What XML is not

- XML is not a presentation language like HTML.
  XML defines the structure of the document and the semantics (meaning) of the data, but it doesn't tell how the data should look.
- XML is not a programming language like Java.
  An XML document by itself simply is. It does not do anything.
- XML is not a network transport protocol like HTTP.
  XML won't send data across the network.
- XML is not a database management system like Oracle.
  XML does not store and retrieve data.

Example 1

1. Read the XML document `people.xml` with any browser;
2. watch the tree data model in `people.ps`;
3. check whether `people.xml` is well-formed by loading it with any browser;
4. read the DTD in `people.dtd` with any text editor;
5. check whether `people.xml` is valid by using STG XML Validation Form.

Example 2

1. Read the context description in `biblio.html`;
2. read the XML document `biblio.xml`;
3. watch the tree data model in `biblio.ps`;
4. read the DTD in `biblio.dtd`.

XML query languages

- A collection of related XML documents is called an XML database.

XML query languages

- A collection of related XML documents is called an XML database.
- The different data model of XML databases (trees) with respect to that of relational databases (tables) call for different query languages.

  The most popular XML query languages are:

XML query languages

- A collection of related XML documents is called an XML database.
- The different data model of XML databases (trees) with respect to that of relational databases (tables) call for different query languages.

  The most popular XML query languages are:
  - XML Path Language (XPath). It is a language to retrieve elements from a single XML document.

XML query languages

- A collection of related XML documents is called an XML database.
- The different data model of XML databases (trees) with respect to that of relational databases (tables) call for different query languages.

  The most popular XML query languages are:
  - XML Path Language (XPath). It is a language to retrieve elements from a single XML document.
  - XML Query Language (XQuery). It is a full query language for XML databases.

The structure of an XPath query

- An XPath query is a path, that is a sequence of steps separated by the slash sign:

$$/\mathtt{step}_1/\mathtt{step}_2/\ldots/\mathtt{step}_k$$

The structure of an XPath query

- An XPath query is a path, that is a sequence of steps separated by the slash sign:

$$/\mathtt{step}_1/\mathtt{step}_2/\ldots/\mathtt{step}_k$$
- each step has the form:

$$\mathtt{axis} :: \mathtt{test}[\mathtt{filter}]$$

The structure of an XPath query

- An XPath query is a path, that is a sequence of steps separated by the slash sign:

$$/\mathtt{step}_1/\mathtt{step}_2/\ldots/\mathtt{step}_k$$

- each step has the form:

$$\mathtt{axis} :: \mathtt{test}[\mathtt{filter}]$$

  - `axis` indicates how to navigate the XML tree;

The structure of an XPath query

- An XPath query is a path, that is a sequence of steps separated by the slash sign:

$$/\mathtt{step}_1/\mathtt{step}_2/\ldots/\mathtt{step}_k$$

- each step has the form:

$$\mathtt{axis} :: \mathtt{test}[\mathtt{filter}]$$

  - `axis` indicates how to navigate the XML tree;
  - `test` filters the result according to the nodes' type;

The structure of an XPath query

- An XPath query is a path, that is a sequence of steps separated by the slash sign:

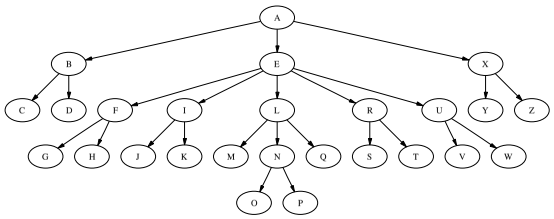
$$/\mathtt{step}_1/\mathtt{step}_2/\ldots/\mathtt{step}_k$$

- each step has the form:

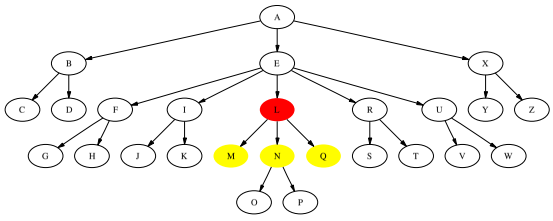
$$\mathtt{axis} :: \mathtt{test}[\mathtt{filter}]$$

  - `axis` indicates how to navigate the XML tree;
  - `test` filters the result according to the nodes' type;
  - `filter` is an optional Boolean path condition to further restrict the result.

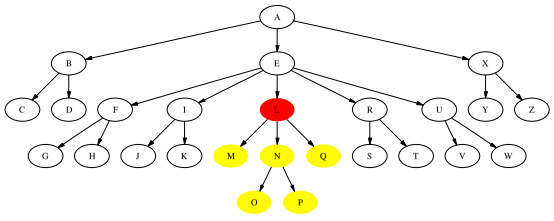
Learning the English alphabet...



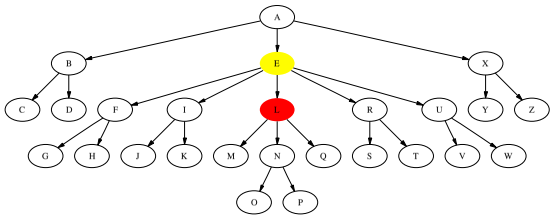
/descendant::L/child::*



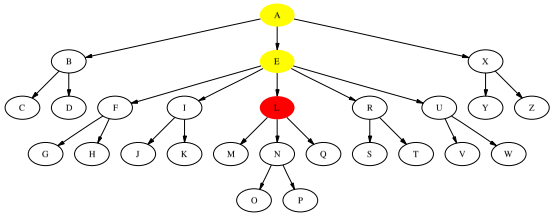
/descendant::L/descendant::*



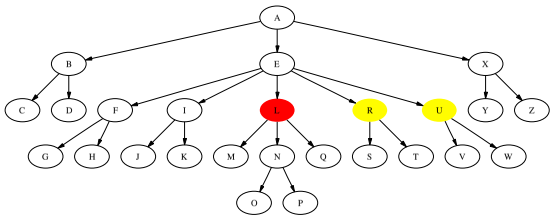
/descendant::L/parent::*



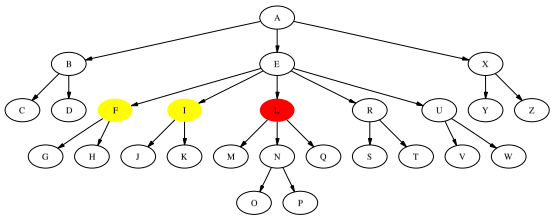
/descendant::L/ancestor::*



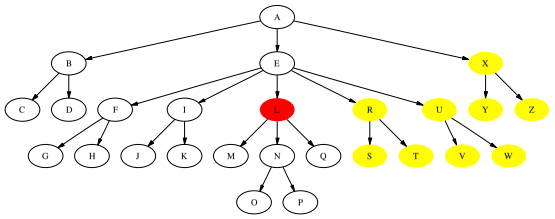
/descendant::L/following-sibling::*



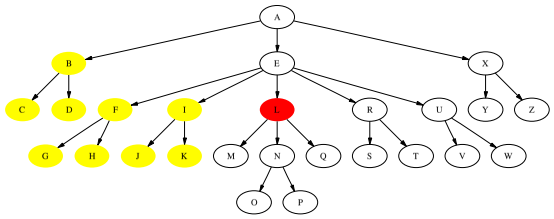
/descendant::L/preceding-sibling::*



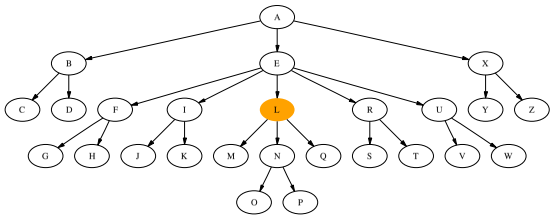
/descendant::L/following::*



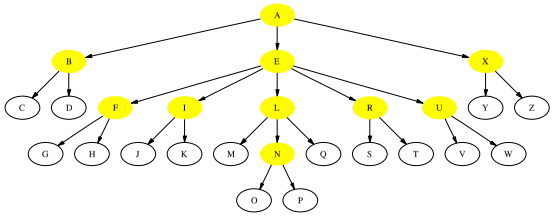
/descendant::L/preceding::*



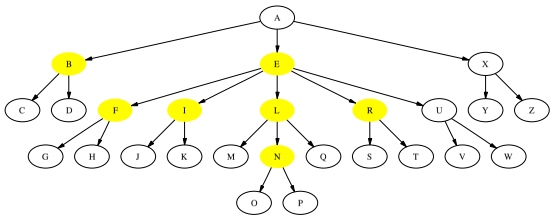
/descendant::L/self::*



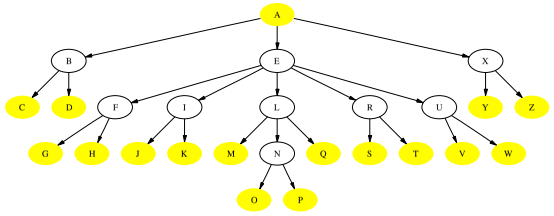
/descendant::*[child::*]



/descendant::*[child::* and following-sibling::*]



/descendant::*[not(child::*) or self::A]



Full XPath

Moreover, XPath offers:

- the use of node tests different form a tag name and *, for instance comment() and text();

Full XPath

Moreover, XPath offers:

- the use of node tests different form a tag name and *, for instance comment() and text();
- the use of comparison operators (like =, >, <) in filters;

Full XPath

Moreover, XPath offers:

- the use of node tests different form a tag name and *, for instance comment() and text();
- the use of comparison operators (like =, >, <) in filters;
- the use of functions (like contains(), position(), count(), id()) in filters.

Example

1. Read the XPath queries contained in `q1.xp`, `q2.xp`, `q3.xp`;
2. run them against `biblio.xml` by using Saxon

XQuery

- The XML query language (XQuery) is the counterpart of SQL for XML databases;

XQuery

- The XML query language (XQuery) is the counterpart of SQL for XML databases;
- XQuery inputs, processes, and outputs sequences (not sets of nodes like XPath);

XQuery

- The XML query language (XQuery) is the counterpart of SQL for XML databases;
- XQuery inputs, processes, and outputs sequences (not sets of nodes like XPath);
- each item of a sequence is either an XML element or an atomic value (like a string or a number);

XQuery

- The XML query language (XQuery) is the counterpart of SQL for XML databases;
- XQuery inputs, processes, and outputs sequences (not sets of nodes like XPath);
- each item of a sequence is either an XML element or an atomic value (like a string or a number);
- XPath queries are used in XQuery. Their results are converted into sorted sequences according to the document order.

Flowers on trees

FLWOR expressions are the most common expressions in XQuery. They are similar to select-from-where statements in SQL.

Flowers on trees

FLWOR expressions are the most common expressions in XQuery. They are similar to select-from-where statements in SQL.

The name FLWOR is an acronym, standing for the first letter of the clauses that may occur in such an expression:

Flowers on trees

FLWOR expressions are the most common expressions in XQuery. They are similar to select-from-where statements in SQL.

The name FLWOR is an acronym, standing for the first letter of the clauses that may occur in such an expression:

- For clauses iteratively bind variables to each value of the result of the corresponding expression.

Flowers on trees

FLWOR expressions are the most common expressions in XQuery. They are similar to select-from-where statements in SQL.

The name FLWOR is an acronym, standing for the first letter of the clauses that may occur in such an expression:

- For clauses iteratively bind variables to each value of the result of the corresponding expression.
- Let clauses bind variables to the entire result of an corresponding expression.

Flowers on trees

FLWOR expressions are the most common expressions in XQuery. They are similar to select-from-where statements in SQL.

The name FLWOR is an acronym, standing for the first letter of the clauses that may occur in such an expression:

- For clauses iteratively bind variables to each value of the result of the corresponding expression.
- Let clauses bind variables to the entire result of an corresponding expression.

A sequence of variable bindings created by the for and let clauses of a FLWOR expression is called a tuple.

Flowers on trees

- Where clauses filter tuples retaining only those that satisfy a condition;

Flowers on trees

- Where clauses filter tuples retaining only those that satisfy a condition;
- Order by clauses sort the tuples;

Flowers on trees

- Where clauses filter tuples retaining only those that satisfy a condition;
- Order by clauses sort the tuples;
- Return clauses build the result of the expression.

Example

1. Read the XQuery queries contained in `q4.xq`, `q5.xq`, `q6.xq`;
2. run them against `biblio.xml` by using Saxon

More information

http://www.sci.unich.it/~francesc/xml