Extensible Markup Language

General Advantages of XML for KR

XML offers new general possibilities, from which AI knowledge representation (KR) can profit:

(1) Definition of self-describing data in worldwide standardized, non-proprietary format

(2) Structured data and knowledge exchange for enterprises in various industries

(3) Integration of information from different sources (into uniform documents)

Specific Advantages of XML for KR

XML provides the most suitable infrastructure for knowledge bases on the Web (incl. for W3C languages such as RDF)

Additional special KR uses of XML are:

- Uniform storage of knowledge bases
- Interchange of knowledge bases between different AI languages
- Exchange between knowledge bases and databases, application systems, etc.

Even transformation/compilation of AI source programs using XML markup and annotations is possible

Address Example: External to HTML

External Presentation:

Xaver M. Linde
Wikingerufer 7
10555 Berlin

HTML tags are still presentation-oriented

HTML Markup:

```
<em>Xaver M. Linde</em>
<br>
Wikingerufer 7
<br>
<strong>10555 Berlin</strong>
```

Address Example: HTML to XML

HTML Markup:

```
<em>Xaver M. Linde</em>
<br>
Wikingerufer 7
<br>
<strong>10555 Berlin</strong>
```

While not conveying any formal semantics:

XML tags are chosen for content-structuring *needs*

XML Markup:

```
<address>
  <name>Xaver M. Linde</name>
  <street>Wikingerufer 7</street>
  <town>10555 Berlin</town>
</address>
```

Address Example: XML to External

XML Markup:

```
<address>
  <name>Xaver M. Linde</name>
  <street>Wikingerufer 7</street>
  <town>10555 Berlin</town>
</address>
```

XML stylesheets are, e.g., usable to generate different presentations

External Presentations:

Xaver M. Linde
Wikingerufer 7
10555 Berlin

. . .

Xaver **M**. **L**inde
Wikingerufer 7
10555 Berlin

XML to HTML: XSLT Example – Input

```
<addresses>

  <address>
    <name>Xaver M. Linde</name>
    <street>Wikingerufer 7</street>
    <town>10555 Berlin</town>
  </address>

  <address>
    <name>John Doe</name>
    <street>42 Gary Cooper Street</street>
    <town>Stanwyck City</town>
  </address>

</addresses>
```

XML to HTML: XSLT Example – Stylesheet

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/addresses">
 <html>
  <head><title>Addresses</title></head>
  <body bgcolor="white">
   <xsl:apply-templates/>
  </body>
 </html>
</xsl:template>

<xsl:template match="address">
 <p>
  <i><xsl:value-of select="name"/></i><br/>
  <xsl:value-of select="street"/><br/>
  <b><xsl:value-of select="town"/></b>
 </p>
</xsl:template>

</xsl:stylesheet>
```

template for document root (of addresses)

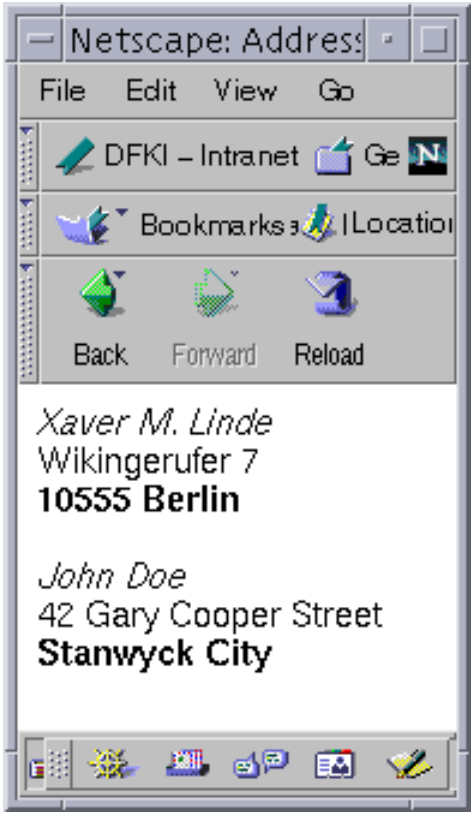
template for address elements

XML to HTML: XSLT Example – Output

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
  "http://www.w3.org/TR/REC-html40/strict.dtd">

<html>
  <head><title>Addresses</title></head>
  <body bgcolor="white">
    <p><i>Xaver M. Linde</i><br>
          Wikingerufer 7<br>
          <b>10555 Berlin</b>
    </p>
    <p><i>John Doe</i><br>
          42 Gary Cooper Street<br>
          <b>Stanwyck City</b>
    </p>
  </body>
</html>
```

*(The HTML code was produced with Apache´s Cocoon in HTML mode, hence
 became
)*



Address Example: XML to XML

XML Markup 1:

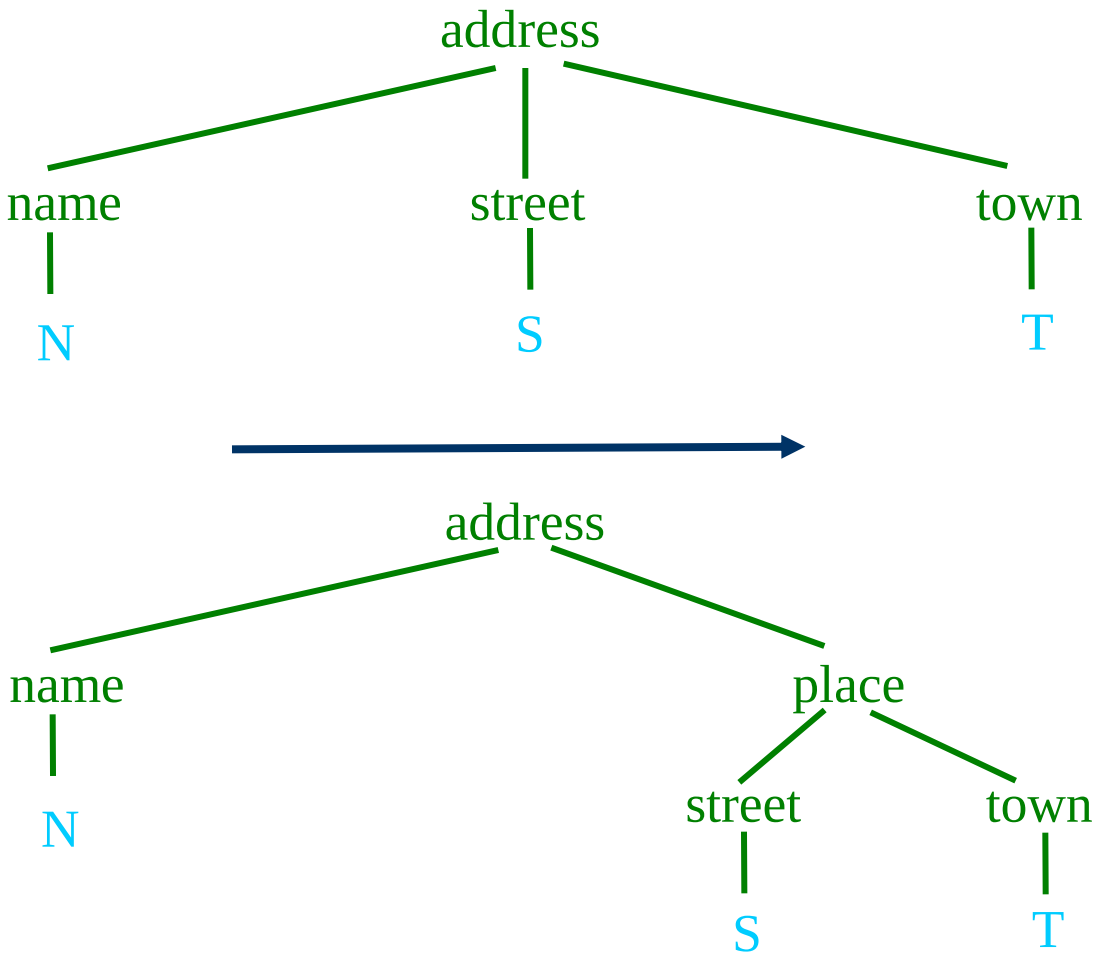
```
<address>
  <name>Xaver M. Linde</name>
  <street>Wikingerufer 7</street>
  <town>10555 Berlin</town>
</address>
```

XML Markup 2:

```
<address>
  <name>Xaver M. Linde</name>
  <place>
    <street>Wikingerufer 7</street>
    <town>10555 Berlin</town>
  </place>
</address>
```

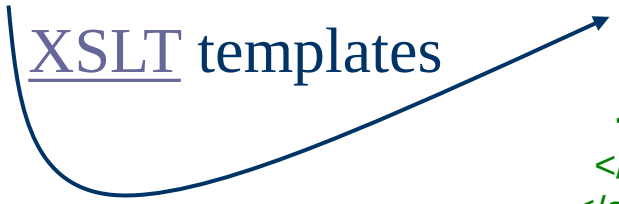
XML stylesheets are also usable to transform XML representations

Address Example: Some Stylesheets Will Contain Term-(Tree-)Rewriting Rules



XML to XML: Transformational Semantics via an XSLT Stylesheet

```
<addresses>
 <address>
  <name>Me2XML</name>
  <street>96 Hyper Road</street>
  <town>Boston</town>
 </address>
 <address>
  <name>RDF4All</name>
  <street>2001 Broadway</street>
  <town>New York</town>
 </address>
 <address>
  <name>XML4You</name>
  <street>96 Hyper Road</street>
  <town>Boston</town>
 </address>
</addresses>
```



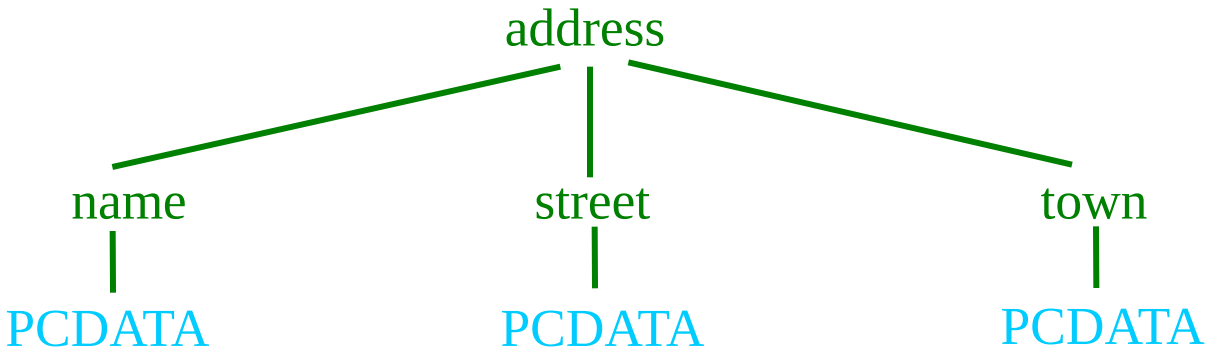

```
<addresses>
 <address>
  <name>Me2XML</name>
  <place>
   <street>96 Hyper Road</street>
   <town>Boston</town>
  </place>
 </address>
 <address>
  <name>RDF4All</name>
  <place>
   <street>2001 Broadway</street>
   <town>New York</town>
  </place>
 </address>
 <address>
  <name>XML4You</name>
  <place>
   <street>96 Hyper Road</street>
   <town>Boston</town>
  </place>
 </address>
</addresses>
```

```
% start fact base for addresses
 address(
   name("Me2XML"),
   place(
     street("96 Hyper Road"),
     town("Boston")
   )
 ).
 address(
   name("RDF4All"),
   place(
     street("2001 Broadway"),
     town("New York")
   )
 ).
 address(
   name("XML4You"),
   place(
     street("96 Hyper Road"),
     town("Boston")
   )
 ).
% end fact base for addresses
```

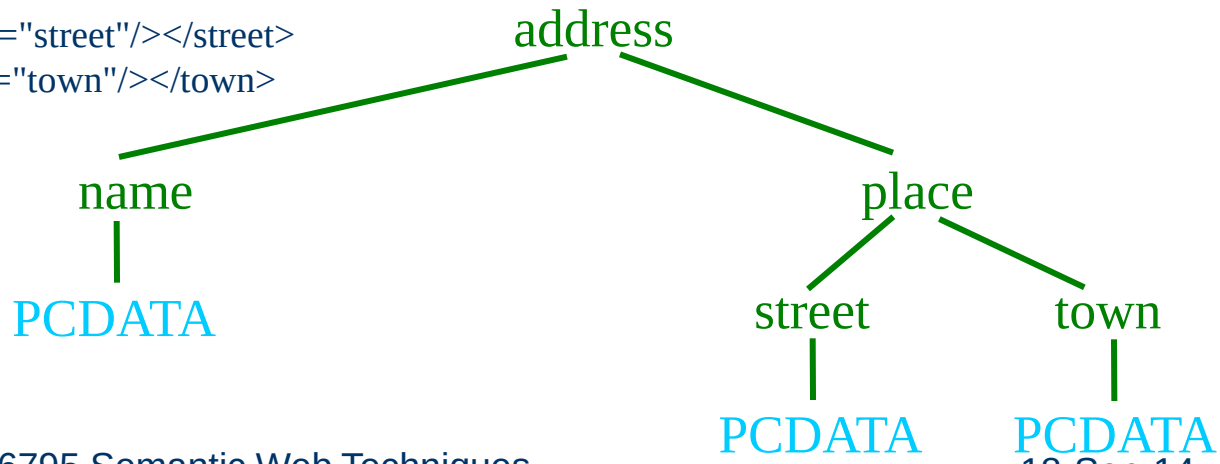
XML to XML: XSLT Stylesheet with a Tree-Transforming Template 1


```
<!-- process addresses and position address nester -->
 <xsl:template match="/addresses">
   <addresses>
     <xsl:apply-templates/>
   </addresses>
 </xsl:template>
```

XML to XML: XSLT Stylesheet with a Tree-Transforming Template 2




```
<!-- apply address nester to each flat address -->
 <xsl:template match="address">
   <address>
     <name><xsl:value-of select="name"/></name>
     <place>
       <street><xsl:value-of select="street"/></street>
       <town><xsl:value-of select="town"/></town>
     </place>
   </address>
 </xsl:template>
```



XML to XML: XSLT Example – Stylesheet

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!-- process addresses and position address nester -->
 <xsl:template match="/addresses">
  <addresses>
   <xsl:apply-templates/>
  </addresses>
 </xsl:template>

<!-- apply address nester to each flat address -->
 <xsl:template match="address">
  <address>
   <name><xsl:value-of select="name"/></name>
   <place>
    <street><xsl:value-of select="street"/></street>
    <town><xsl:value-of select="town"/></town>
   </place>
  </address>
 </xsl:template>
</xsl:stylesheet>
```

Address Example: XML Normalization for Recovering Canonical Ordering

XML Markups (Non-Normalized):

```
<address>
  <name>Xaver M. Linde</name>
  <town>10555 Berlin</town>
  <street>Wikingerufer 7</street>
</address>
```

. . .

```
<address>
  <town>10555 Berlin</town>
  <street>Wikingerufer 7</street>
  <name>Xaver M. Linde</name>
</address>
```

XSLT transformations can perform XML normalizations

XML Markup (Normalized):

```
<address>
  <name>Xaver M. Linde</name>
  <street>Wikingerufer 7</street>
  <town>10555 Berlin</town>
</address>
```

An address element in normal form has 3 ordered subelements

<xsl:value-of select="name"/> etc. find subelements in any position, so earlier stylesheets can be readily adapted for this

Shortlist Example: Irrecoverable Ordering (Used for Ranking)

XML Markup (Given Positional Information):

A shortlist element has 3 ordered subelements

```
<shortlist>
  <name>Mary Poppins</name>
  <name>Xaver M. Linde</name>
  <name>John Doe</name>
</shortlist>
```

XML Markup (Lost Positional Information):

```
<shortlist>
  <name>Mary Poppins</name>
  <name>John Doe</name>
  <name>Xaver M. Linde</name>
</shortlist>
```

. . .

```
<shortlist>
  <name>John Doe</name>
  <name>Xaver M. Linde</name>
  <name>Mary Poppins</name>
</shortlist>
```

Address Example: XML Queries

XML Markup:

element

subelements

```
<address>
  <name>Xaver M. Linde</name>
  <street>Wikingerufer 7</street>
  <town>10555 Berlin</town>
</address>
```

XML Query (XML-QL, which influenced XQuery standard):

```
WHERE
  <address>
    <name>Xaver M. Linde</name>
    <street>$s</street>
    <town>$t</town>
  </address>
CONSTRUCT
  <binding>
    <s>$s</s>
    <t>$t</t>
  </binding>
```

XML queries can select subelements of XML elements

```
<binding>
  <s>Wikingerufer 7</s>
  <t>10555 Berlin</t>
</binding>
```

Address Example: Prolog Queries

Prolog Term:

structure

```
address(
   name("Xaver M. Linde"),
   street("Wikingerufer 7"),
   town("10555 Berlin")
)
```

substructures

Prolog Query:

```
address(
   name("Xaver M. Linde"),
   street(S),
   town(T)
)
```

Prolog queries can select substructures of Prolog structures

S = "Wikingerufer 7"
T = "10555 Berlin"

Address Example: The Element Tree

XML Markup:

element / subelements

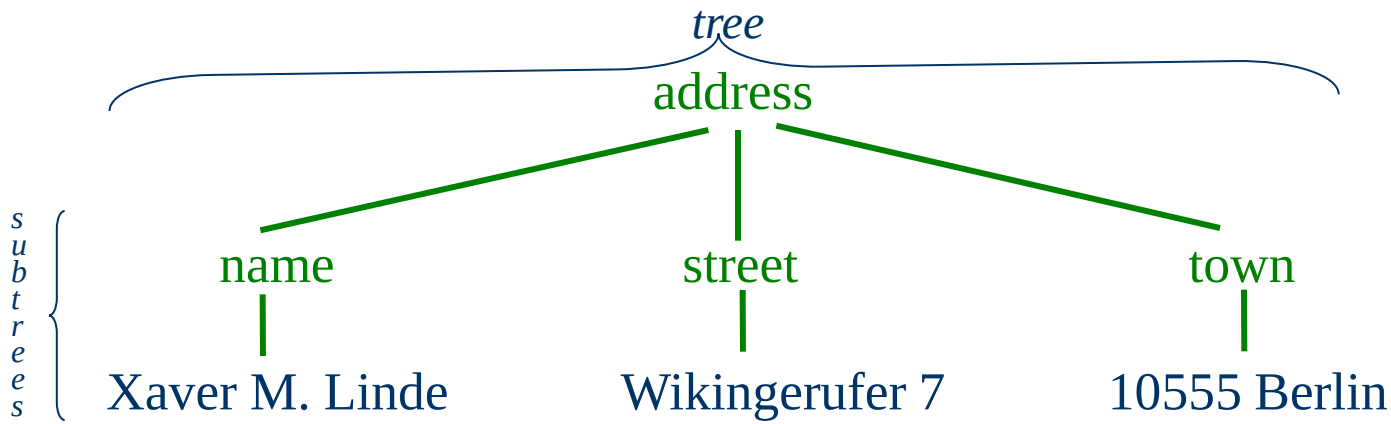
```
<address>
  <name>Xaver M. Linde</name>
  <street>Wikingerufer 7</street>
  <town>10555 Berlin</town>
</address>
```

Prolog Term:

structure / substructures

```
address(
  name("Xaver M. Linde"),
  street("Wikingerufer 7"),
  town("10555 Berlin")
)
```

Node-Labeled, (Left-to-Right-)Ordered Element Tree:



Address Example: Document Type Definition and Tree (1)

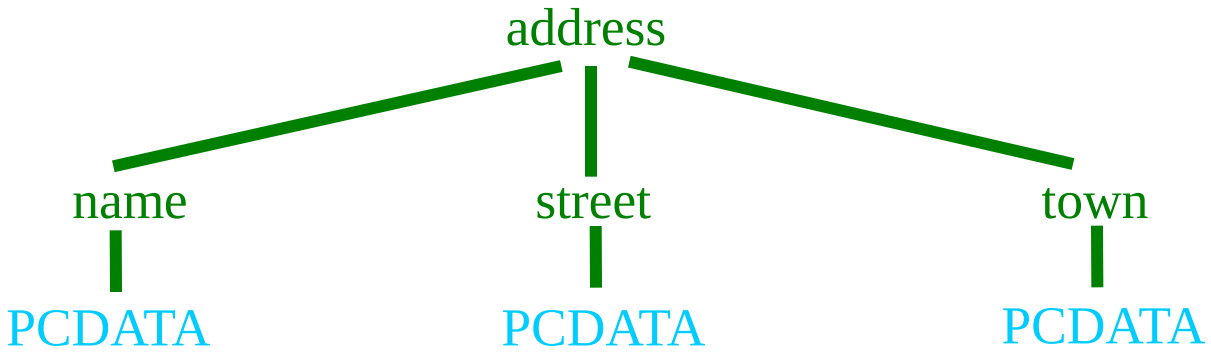
Document Type Definition (DTD):

```
<!ELEMENT address (name, street, town) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT street (#PCDATA) >
<!ELEMENT town (#PCDATA) >
```

Extended Backus-Naur Form (EBNF):

```
address ::= name street town
name ::= PCDATA
street ::= PCDATA
town ::= PCDATA
```

Document Type Tree:

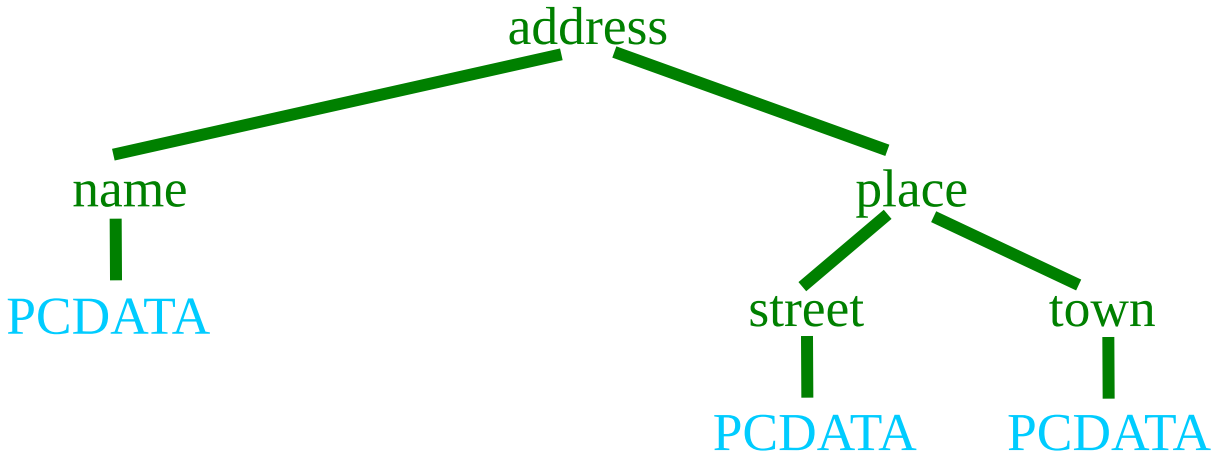


Address Example: Document Type Definition and Tree (2)

Document Type Definition (DTD):

```
<!ELEMENT address   (name, place) >
<!ELEMENT place     (street, town) >
<!ELEMENT name      (#PCDATA) >
<!ELEMENT street    (#PCDATA) >
<!ELEMENT town      (#PCDATA) >
```

Document Type Tree:



Well-Formedness and Validity

XML principles for a document being *well-formed*:

- Open and close all tags
- Empty tags end with />
- There is a unique root element
- Elements may not overlap
- Attribute values are quoted
- < and & are only used to start tags and entities
- Only the five predefined entity references are used

XML principle for a document being *valid with respect to (w.r.t.) a DTD* :

- Match the constraints listed in the DTD (or, generate from DTD as linearized derivation tree, as shown later)

Checked by **validators** such as http://validator.w3.org/

Mail-Box Example: Address Variant

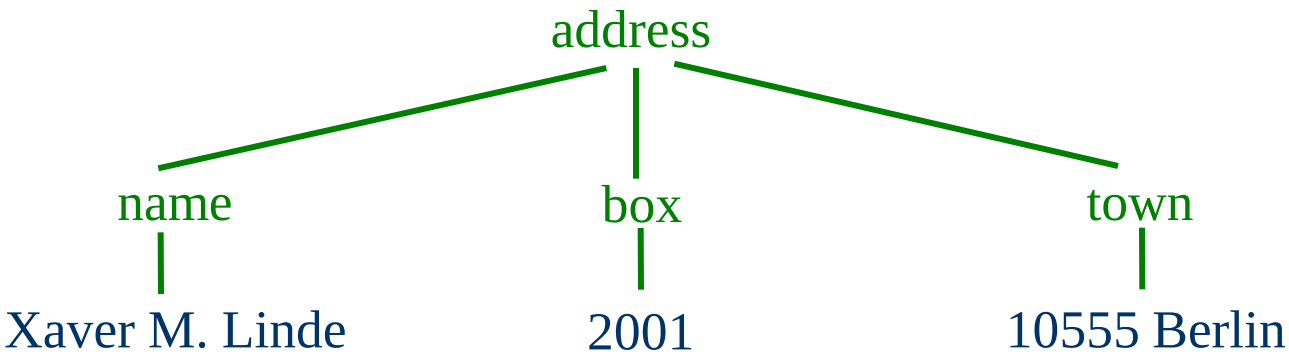
XML Markup:

```
<address>
  <name>Xaver M. Linde</name>
  <box>2001</box>
  <town>10555 Berlin</town>
</address>
```

Prolog Term:

```
address(
  name("Xaver M. Linde"),
  box("2001"),
  town("10555 Berlin")
)
```

Node-Labeled, (Left-to-Right-)Ordered Element Tree:



"|"-Disjoined Street/Mail-Box Example: Document Type Definition and Tree

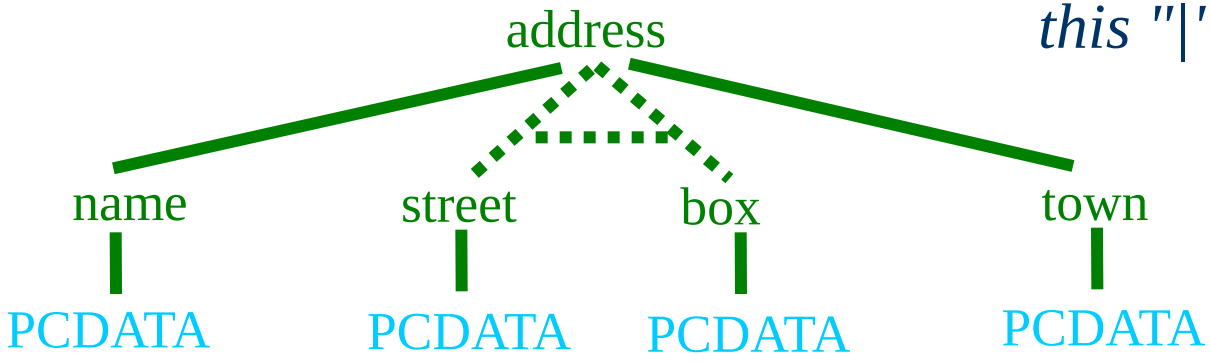
Document Type Definition (DTD):

```
<!ELEMENT address   (name, (street | box), town) >
<!ELEMENT name      (#PCDATA) >
<!ELEMENT street    (#PCDATA) >
<!ELEMENT box       (#PCDATA) >
<!ELEMENT town      (#PCDATA) >
```

"|": Choice

The above box address and the original street address are valid w.r.t. this "|"-DTD

Document Type Tree:



Phone & Fax Example: Address Variant

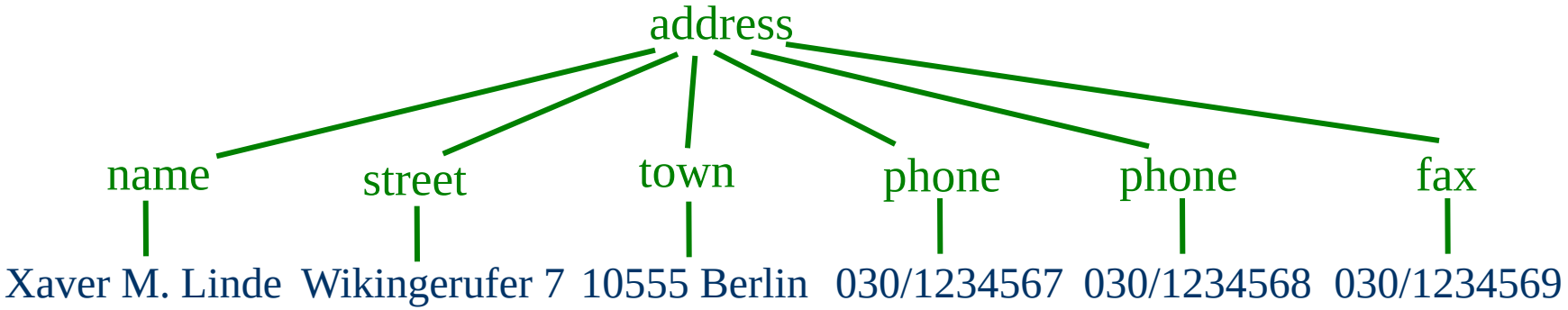
XML Markup:

```
<address>
  <name>Xaver M. Linde</name>
  <street>Wikingerufer 7</street>
  <town>10555 Berlin</town>
  <phone>030/1234567</phone>
  <phone>030/1234568</phone>
  <fax>030/1234569</fax>
</address>
```

Prolog Term:

```
address(
  name("Xaver M. Linde"),
  street("Wikingerufer 7"),
  town("10555 Berlin"),
  phone("030/1234567"),
  phone("030/1234568"),
  fax("030/1234569")
)
```

Node-Labeled, (Left-to-Right-)Ordered Element Tree:



"+"/"*"-Repetitive-Phone & -Fax Example: Document Type Definition and Tree

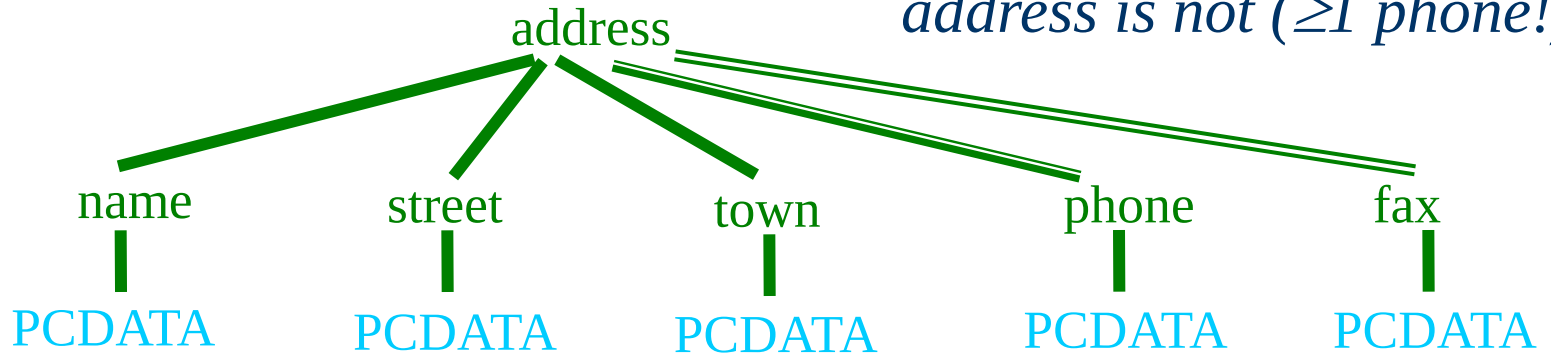
Document Type Definition (DTD):

```
<!ELEMENT address   (name, street, town, phone+, fax*) >
<!ELEMENT name      (#PCDATA) >
<!ELEMENT street    (#PCDATA) >
<!ELEMENT town      (#PCDATA) >
<!ELEMENT phone     (#PCDATA) >
<!ELEMENT fax       (#PCDATA) >
```

"+"/"*": One/Zero or More

The above two-phone/one-fax address is valid w.r.t. this "+"/""-DTD but the original no-phone/no-fax address is not (≥1 phone!)*

Document Type Tree:



Country Example: Address Variant

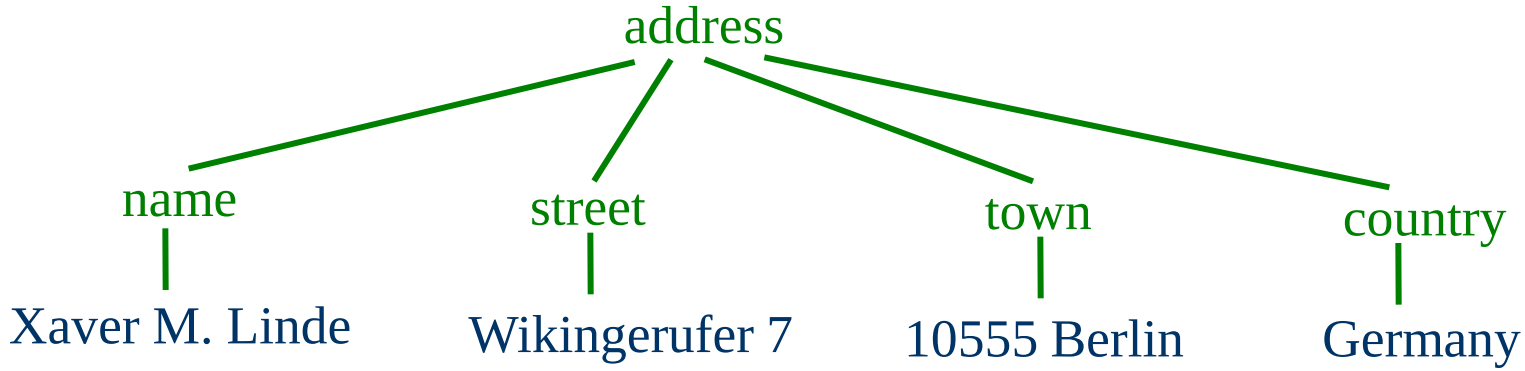
XML Markup:

```
<address>
  <name>Xaver M. Linde</name>
  <street>Wikingerufer 7</street>
  <town>10555 Berlin</town>
  <country>Germany</country>
</address>
```

Prolog Term:

```
address(
  name("Xaver M. Linde"),
  street("Wikingerufer 7"),
  town("10555 Berlin"),
  country("Germany")
)
```

Node-Labeled, (Left-to-Right-)Ordered Element Tree:



"?"-Optional-Country Example: Document Type Definition and Tree

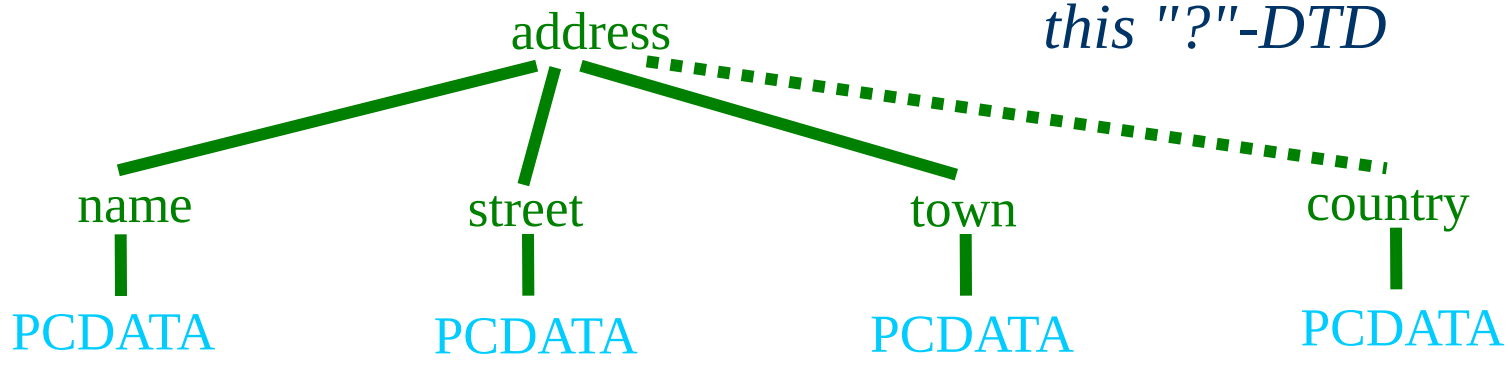
Document Type Definition (DTD):

```
<!ELEMENT address   (name, street, town, country?) >
<!ELEMENT name      (#PCDATA) >
<!ELEMENT street    (#PCDATA) >
<!ELEMENT town      (#PCDATA) >
<!ELEMENT country   (#PCDATA) >
```

"?": One or Zero

The above country address and the original countriless address are valid w.r.t. this "?"-DTD

Document Type Tree:



Country Address: A Complete XML Document Referring to an External DTD

XML Instance Document Referring to DTD (via root element):

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE address SYSTEM "country-address.dtd">
<address>
   <name>Xaver M. Linde</name>
   <street>Wikingerufer 7</street>
   <town>10555 Berlin</town>
   <country>Germany</country>
</address>
```

The *XML declaration* uses standalone attribute with "no" value: DTD import

The *DOCument TYPE declaration* names the *root element* address and, after the *SYSTEM keyword*, refers to an *external DTD at "country-address.dtd"* (or, at absolute URL "http://www.cs.unb.ca/~boley/cs6795swt/country-address.dtd")

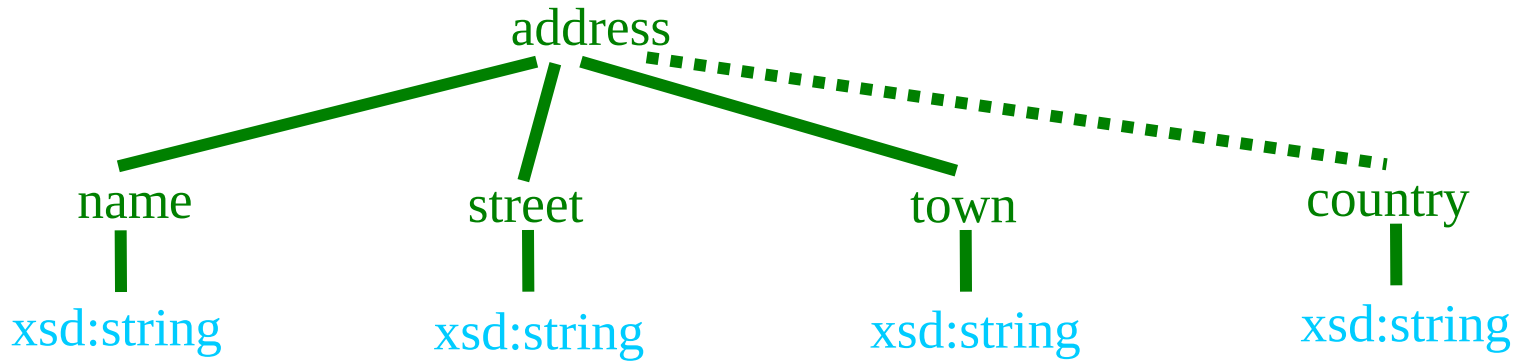
"minOccurs"-Optional-Country Example: XML Schema Definition

Equivalent XML Schema Definition (XSD):

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="address">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="street" type="xsd:string"/>
        <xsd:element name="town" type="xsd:string"/>
        <xsd:element name="country" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

"minOccurs"-Optional-Country Example: XML Schema Tree

Equivalent Document Schema Tree:



Country Address: A Complete XML Document Referring to an External XSD

XML Instance Document Referring to XSD (via root element):

```
<?xml version="1.0"?>
<address
    xsi:noNamespaceSchemaLocation="country-address.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>
   <name>Xaver M. Linde</name>
   <street>Wikingerufer 7</street>
   <town>10555 Berlin</town>
   <country>Germany</country>
</address>
```

The *xsi:noNamespaceSchemaLocation attribute* is embedded in the *root element* address and refers to an *external XSD* at "country-address.xsd" (or, at absolute URL "http://www.cs.unb.ca/~boley/cs6795swt/country-address.xsd")

"?"-Optional-Country Example: Relax NG Compact Syntax (RNC)

Equivalent Relax NG in Compact Syntax (RNC):

```
default namespace = ""

start = address
address = element address { name, street, town, country? }
name = element name { xsd:string }
street = element street { xsd:string }
town = element town { xsd:string }
country = element country { xsd:string }
```

"optional"-Country Example: Relax NG XML Syntax (RNG)

Equivalent Relax NG in XML Syntax (RNG):

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
 <start>
  <ref name="address"/>
 </start>
 <define name="address">
  <element name="address" ns="">
   <group>
    <ref name="name"/>
    <ref name="street"/>
    <ref name="town"/>
    <optional>
     <ref name="country"/>
    </optional>
   </group>
  </element>
 </define>
 <define name="name">
  <element name="name" ns="">
   <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
  </element>
 </define>
 <define name="street">
  <element name="street" ns="">
   <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
  </element>
 </define>
 <define name="town">
  <element name="town" ns="">
   <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
  </element>
 </define>
 <define name="country">
  <element name="country" ns="">
   <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
  </element>
 </define>
</grammar>
```

Country Address: A Complete XML Document Referring to an External RNG

XML Instance Document Referring to RNG (via xml-model line):

```
<?xml version="1.0"?>
<?xml-model
        href="country-address.rng"
        type="application/xml"
        schematypens="http://relaxng.org/ns/structure/1.0"?>
<address>
  <name>Xaver M. Linde</name>
  <street>Wikingerufer 7</street>
  <town>10555 Berlin</town>
  <country>Germany</country>
</address>
```

The *href attribute* refers to an *external RNG* at "country-address.rng"

(or, at absolute URL "http://www.cs.unb.ca/~boley/cs6795swt/country-address.rng")