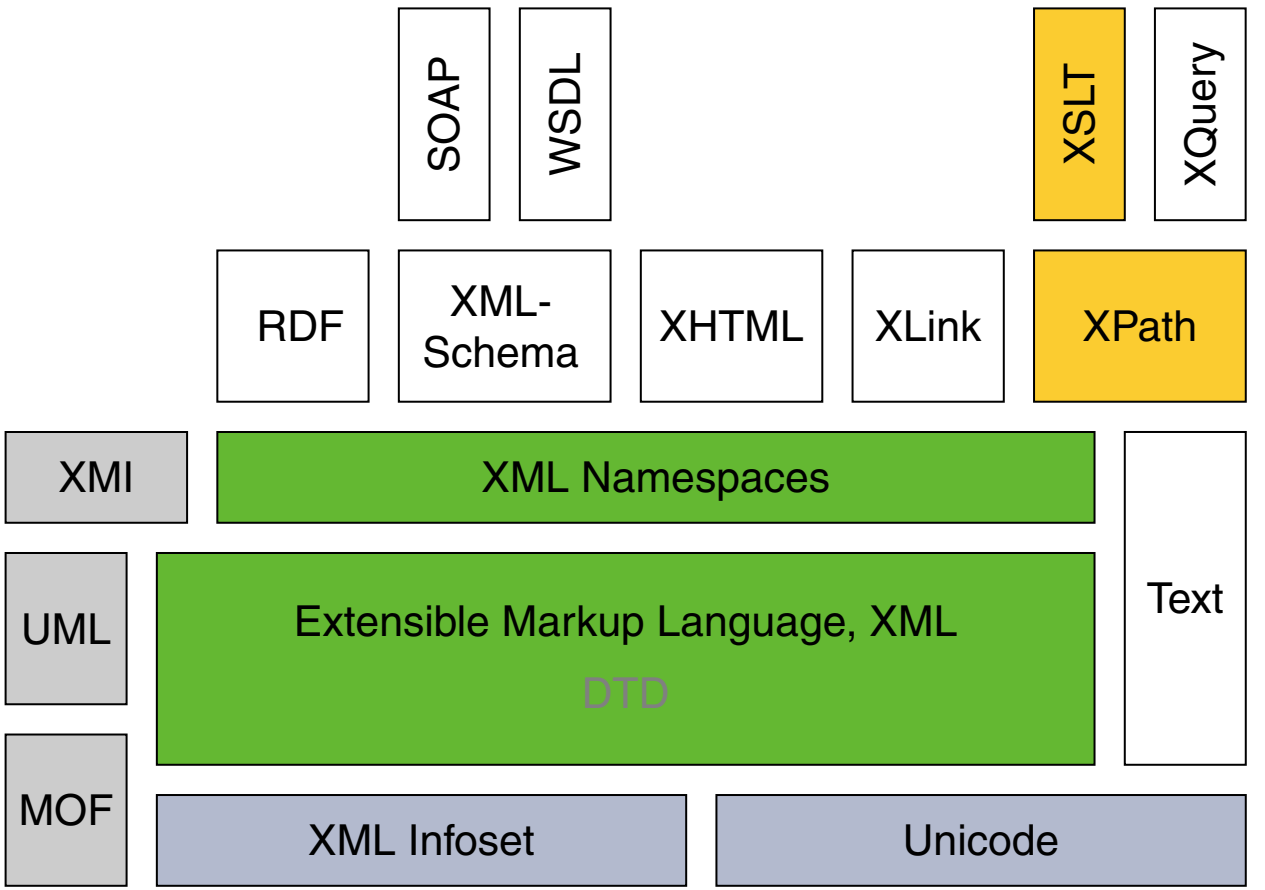
Kapitel WT:III (Fortsetzung)

III. Dokumentsprachen

- Auszeichnungssprachen
- HTML
- Cascading Stylesheets CSS
- XML-Grundlagen
- XML-Schema
- Die XSL-Familie
- APIs für XML-Dokumente

Die XSL-Familie

Einordnung [Jeckle 2004]



Die XSL-Familie

Historie: zentrale XML-Spezifikationen

2006 Extensible Markup Language (XML) 1.1. Recommendation. [W3C REC, status]

2004 XML-Schema Part 0: Primer. Recommendation. [W3C REC]

2012 XML-Schema (XSD) 1.1 Part 1: Structures. [W3C REC, status]

2012 XML-Schema (XSD) 1.1 Part 2: Datatypes. [W3C REC]

2017 XSL Transformations (XSLT) 3.0. Prop. Recommendation. [W3C REC, status]

2017 XML Path Language (XPath) 3.1. Recommendation. [W3C REC, status]

2017 XML Query Language (XQuery) 3.1. Recommendation. [W3C REC, status]

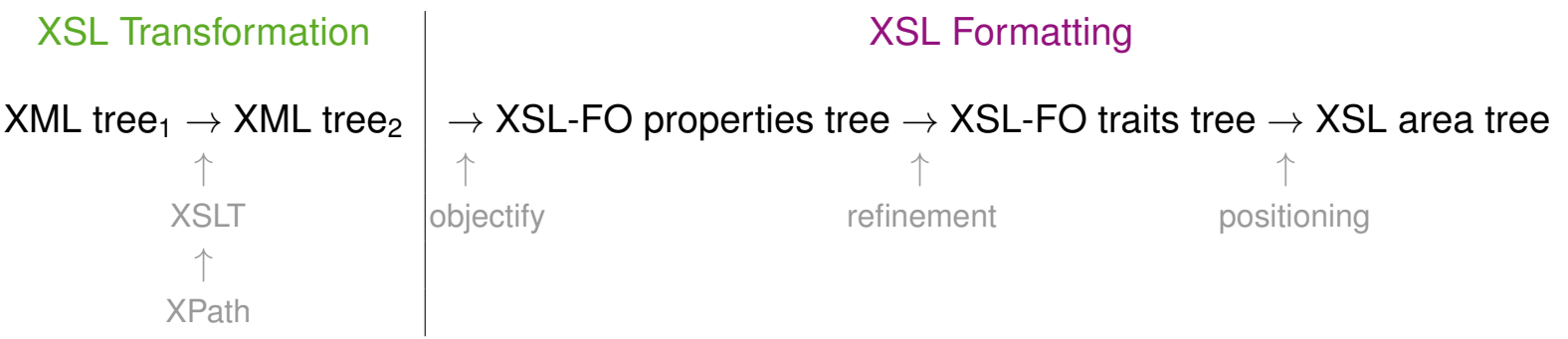
2012 XSL Formatting Objects (XSL-FO) 2.0. Working Draft. [W3C WD, status]

Bemerkungen:

- “[The extensible stylesheet language family] XSL is a family of recommendations for defining XML document transformation and presentation. It consists of three parts:” XSLT, XPath, XSL-FO. [W3C]

- CSS versus XSL. Why two Style Sheet languages? [W3C 1, 2]

- Schritte eines XSL-Verarbeitungsprozesses [W3C]:



- Die Formatierungsmöglichkeiten von XSL-FO orientieren sich an den Anforderungen von Print-Medien. Bei der Verarbeitung (dem Formatieren) von HTML-Seiten geschieht anstelle des XSL-Formatting-Prozess üblicherweise ein CSS-Formatting-Prozess.

Die XSL-Familie

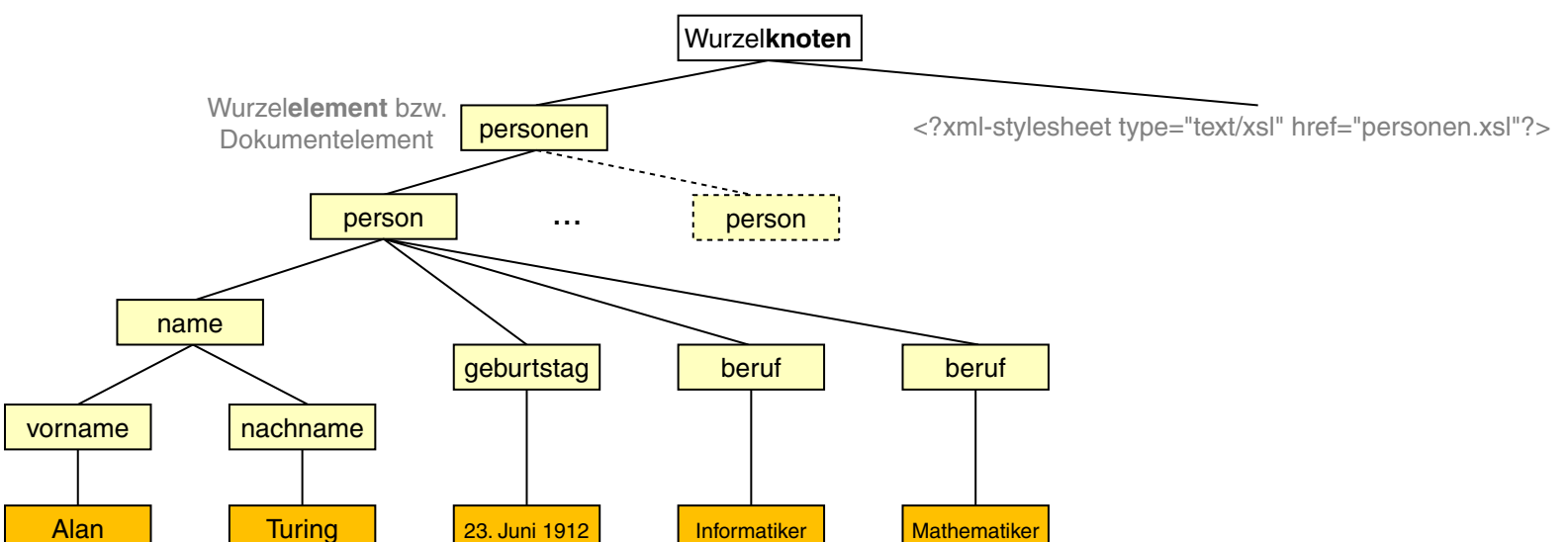
Verwendung von XPath

XSLT	Finden und Auswählen von Elementen im Eingabedokument, die in das Ausgabedokument kopiert/transformiert werden.
XQuery	Finden und Auswählen von Elementen.
XPointer	Identifikation einer Stelle im XML-Dokument, auf die ein XLink verweist.
XML-Schema	Formulierung von Constraints hinsichtlich der Eindeutigkeit oder der Identität von Elementen.
XForm	Bindung von Formularsteuerungen an Instanzdaten; Formulierung von Werte-Constraints und Berechnungen.

Die XSL-Familie

XML-Knotentypen unter dem XPath-Modell

1. Wurzelknoten
2. Elementknoten
3. Textknoten
4. Attributknoten
5. Kommentarknoten
6. Verarbeitungsanweisungsknoten
7. Namensraumknoten



Bemerkungen:

- Der Wurzel*knoten* eines XML-Dokuments ist nicht identisch mit dem Wurzel*element*: Der Wurzelknoten entspricht dem *Document Information Item* des XML Information Sets. Das Wurzelelement hingegen ist das erste benannte Element des Dokuments und wird durch ein *Element Information Item* dargestellt.
- XPath dient zur Navigation in Dokumenten und der Auswahl von Dokumentbestandteilen; XPath ist keine Datenmanipulationssprache.
- XPath-Ausdrücke können zu einzelnen Knoten (XML-Element, XML-Attribut), zu Knotenmengen, zu Zeichenketten, zu Zahlen und zu Bool'schen Werten evaluieren. XPath stellt deshalb Funktionen zum Zugriff auf Knotenmengen und zur Manipulation verschiedener Datentypen zur Verfügung.
- Wiederholung. Das W3C hat mittlerweile drei Datenmodelle für XML-Dokumente definiert: XML Information Set, XPath, Document Object Model (DOM). Das XPath-Datenmodell basiert auf einer Baumstruktur, die bei der Abfrage eines XML-Dokuments durchlaufen wird und ist dem XML Information Set ähnlich; DOM ist der Vorläufer beider Datenmodelle. DOM und das XPath-Datenmodell können als Interpretationen des XML Information Sets betrachtet werden. [MSDN]

Die XSL-Familie

XPath-Lokalisierungspfade

- Ein Lokalisierungspfad spezifiziert eine eventuell leere Menge von Knoten in einem XML-Dokument.
- Ein Lokalisierungspfad setzt sich aus aufeinander folgenden Lokalisierungsschritten *(Location steps)* zusammen.
- Jeder Lokalisierungsschritt wird relativ zu einem bestimmten Knoten des XML-Dokuments ausgewertet, der dann als aktueller Knoten *(Current node)* oder Kontextknoten bezeichnet wird.

Die XSL-Familie

XPath-Lokalisierungspfade

- Ein Lokalisierungspfad spezifiziert eine eventuell leere Menge von Knoten in einem XML-Dokument.
- Ein Lokalisierungspfad setzt sich aus aufeinander folgenden Lokalisierungsschritten *(Location steps)* zusammen.
- Jeder Lokalisierungsschritt wird relativ zu einem bestimmten Knoten des XML-Dokuments ausgewertet, der dann als aktueller Knoten *(Current node)* oder Kontextknoten bezeichnet wird.

- Lokalisierungsschritte werden durch Schrägstriche *(Slashes)* getrennt:

  . . . / *Schritt_i* / *Schritt_i+1* / . . .

- Beginnt ein Lokalisierungspfad mit einem Schrägstrich, bezeichnet dieser den Wurzelknoten. Der Wurzelknoten ist dann Kontextknoten zum ersten Lokalisierungsschritt:

  / *Schritt_1*

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

(b) `/personen/person[1]/name/vorname`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

(b) `/personen/person[1]/name/vorname`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

(b) `/personen/person[1]/name/vorname`

(c) `/personen/person[1]/beruf`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `/personen/person`

(b) `/personen/person[1]/name/vorname`

(c) `/personen/person[1]/beruf`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Allgemeine Form eines Lokalisierungsschritts:

. . . / *Achse* : : *Knotentest* [*Prädikat*] / . . .

1. Achse. Spezifiziert Knotenmenge relativ zum Kontextknoten. Es werden 13 Achsen unterschieden, `child::` ist die Defaultachse.

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Allgemeine Form eines Lokalisierungsschritts:

. . . / *Achse* : : *Knotentest* [*Prädikat*] / . . .

1. Achse. Spezifiziert Knotenmenge relativ zum Kontextknoten. Es werden 13 Achsen unterschieden, `child::` ist die Defaultachse.

2. Knotentest. Filtert die durch eine Achse (1) spezifizierte Knotenmenge weiter. Hierzu gibt es für jeden Knotentyp ein Testschema.

   Beispiele:

   - ein qualifizierender Name (wie „Person“) ∼ Test auf Knoten mit diesem Namen
   - die Funktion `text()` ∼ Test auf Textknoten

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Allgemeine Form eines Lokalisierungsschritts:

. . . / *Achse* : : *Knotentest* [*Prädikat*] / . . .

1. Achse. Spezifiziert Knotenmenge relativ zum Kontextknoten. Es werden 13 Achsen unterschieden, `child::` ist die Defaultachse.

2. Knotentest. Filtert die durch eine Achse (1) spezifizierte Knotenmenge weiter. Hierzu gibt es für jeden Knotentyp ein Testschema.

   Beispiele:

   - ein qualifizierender Name (wie „Person“) ∼ Test auf Knoten mit diesem Namen
   - die Funktion `text()` ∼ Test auf Textknoten

3. Prädikat. Filtert die durch Achse (1) und Knotentest (2) spezifizierte Knotenmenge weiter. Jeder gültige XPath-Ausdruck kann Prädikat sein.

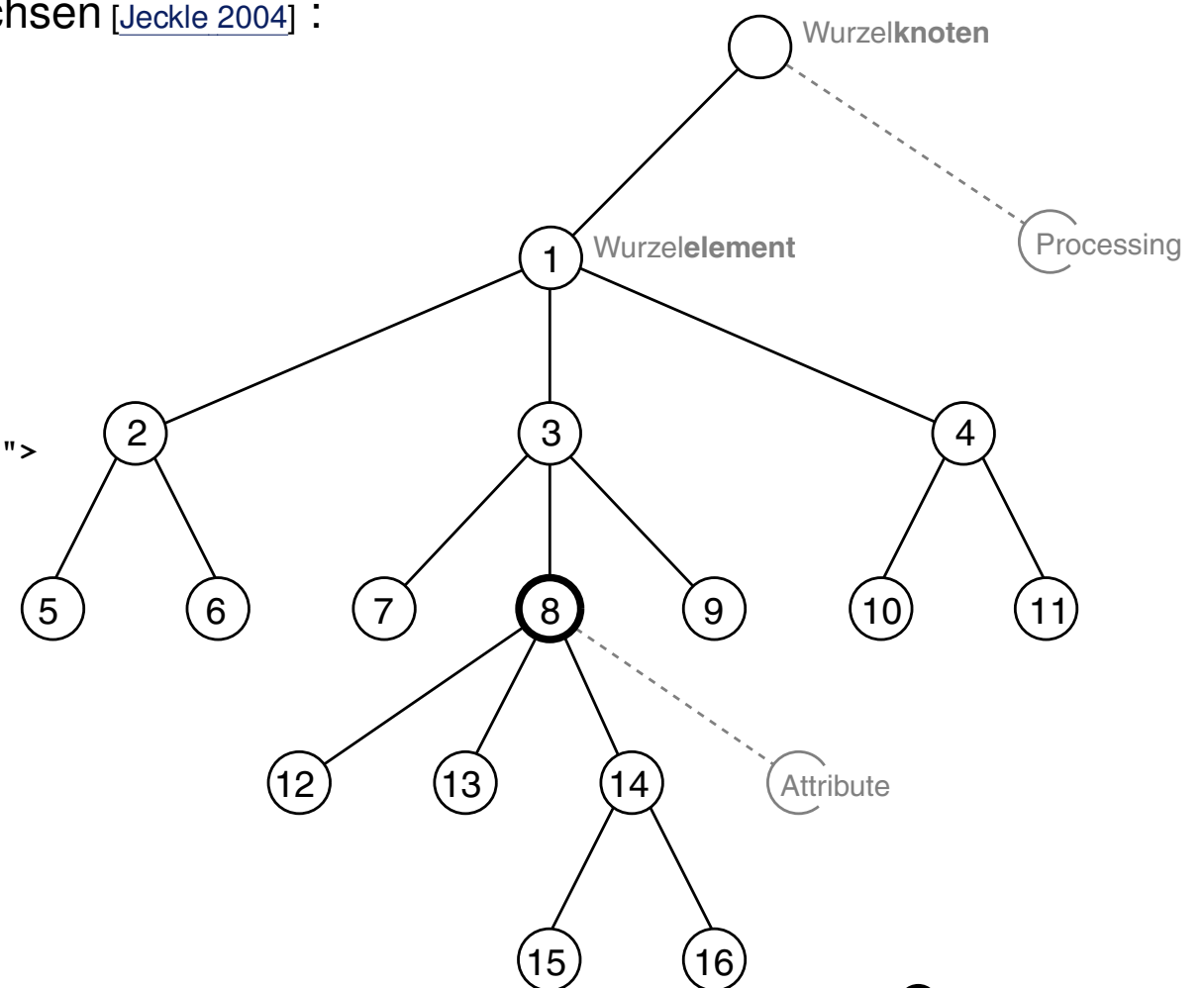
   Beispiele: Test auf Kindknoten, Position bzw. Index

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Illustration wichtiger Achsen [Jeckle 2004] :

```
<?xml version="1.0">
<node1>
  <node2>
    <node5/>
    <node6/>
  </node2>
  <node3>
    <node7/>
    <node8> Attribute="42">
      <node12/>
      <node13/>
      <node14>
        <node15/>
        <node16/>
      </node14>
    </node8>
    <node9/>
  </node3>
  <node4>
    <node10/>
    <node11/>
  </node4>
</node1>
```

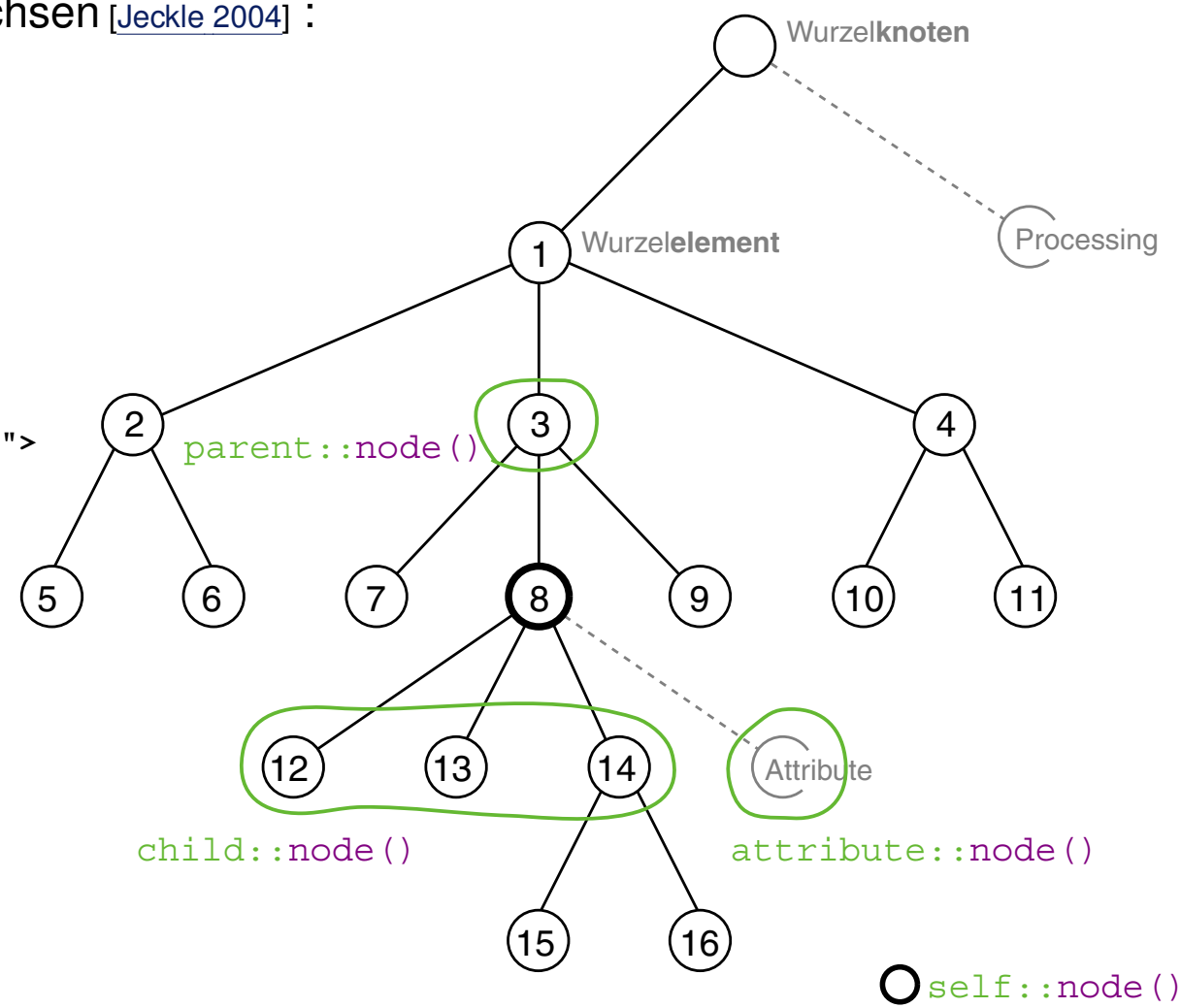


Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Illustration wichtiger Achsen [Jeckle 2004] :

```
<?xml version="1.0">
<node1>
  <node2>
    <node5/>
    <node6/>
  </node2>
  <node3>
    <node7/>
    <node8> Attribute="42">
      <node12/>
      <node13/>
      <node14>
        <node15/>
        <node16/>
      </node14>
    </node8>
    <node9/>
  </node3>
  <node4>
    <node10/>
    <node11/>
  </node4>
</node1>
```

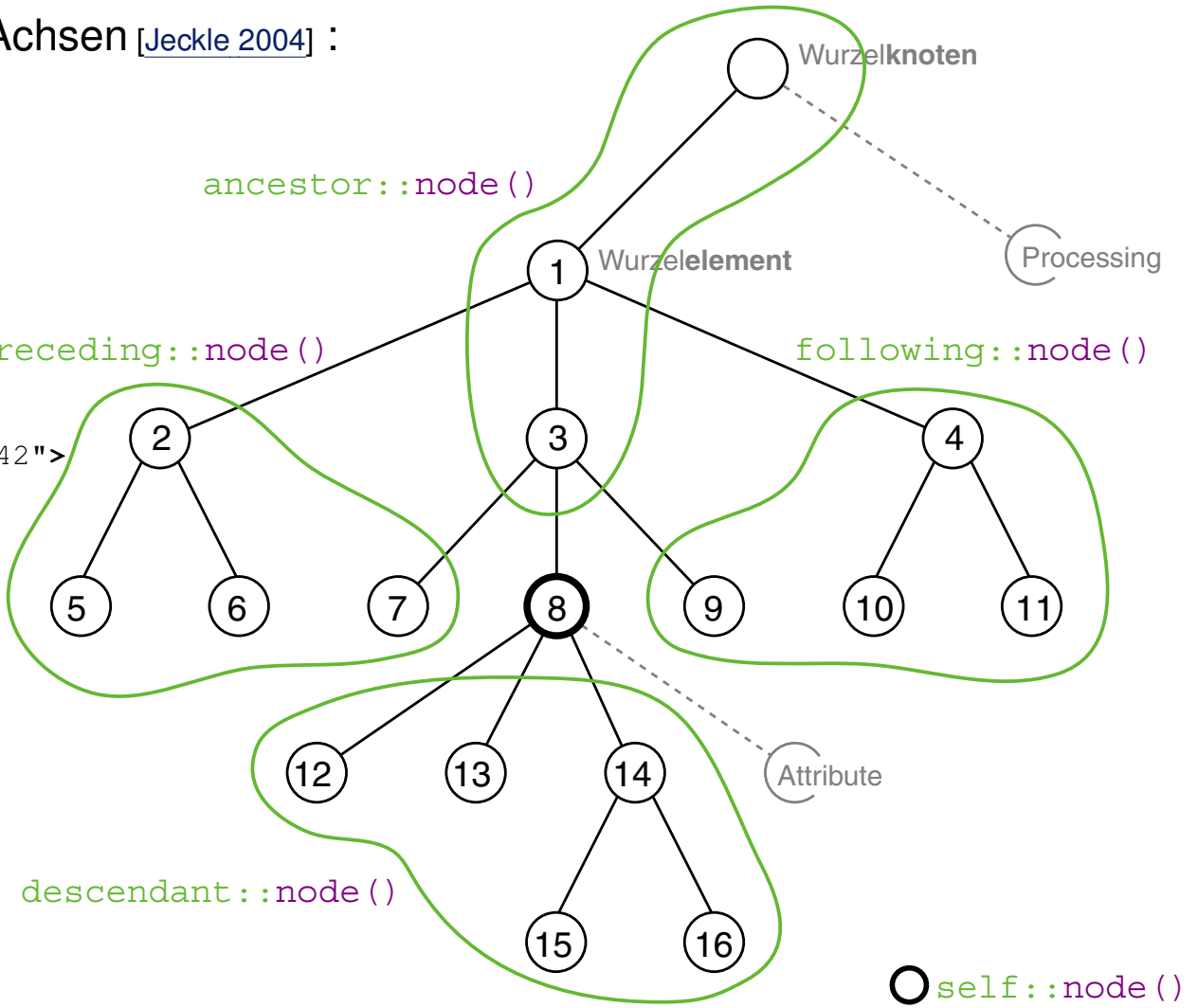


Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Illustration wichtiger Achsen [Jeckle 2004] :

```
<?xml version="1.0">
<node1>
  <node2>
    <node5/>
    <node6/>
  </node2>
  <node3>
    <node7/>
    <node8> Attribute="42">
      <node12/>
      <node13/>
      <node14>
        <node15/>
        <node16/>
      </node14>
    </node8>
    <node9/>
  </node3>
  <node4>
    <node10/>
    <node11/>
  </node4>
</node1>
```



Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

- Schreibweisen häufig verwendeter Lokalisierungsschritte:

kurz	lang	Semantik
`.`	`self::node()`	Kontextknoten
`..`	`parent::node()`	Elternknoten des Kontextknotens
`//`	`/descendant-or-self::node()/`	Kontextknoten einschließlich aller seiner Nachkommen
`@*`	`attribute::*`	alle Attributknoten

- Wildcards für Knotentests:

`node()`	Knoten jeden Typs
`*`	achsensabhängig: Element- oder Attributknoten
`text()`	Textknoten
`comment()`	Kommentarknoten
`processing-instruction()`	Verarbeitungsanweisungsknoten

- Spezifikation von alternativen Lokalisierungspfaden:

Pfad_1 | *Pfad_2* | ... | *Pfad_n*

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

- Schreibweisen häufig verwendeter Lokalisierungsschritte:

kurz	lang	Semantik
`.`	`self::node()`	Kontextknoten
`..`	`parent::node()`	Elternknoten des Kontextknotens
`//`	`/descendant-or-self::node()/`	Kontextknoten einschließlich aller seiner Nachkommen
`@*`	`attribute::*`	alle Attributknoten

- Wildcards für Knotentests:

`node()`	Knoten jeden Typs
`*`	achensabhängig: Element- oder Attributknoten
`text()`	Textknoten
`comment()`	Kommentarknoten
`processing-instruction()`	Verarbeitungsanweisungsknoten

- Spezifikation von alternativen Lokalisierungspfaden:

Pfad_1 | *Pfad_2* | ... | *Pfad_n*

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

- Schreibweisen häufig verwendeter Lokalisierungsschritte:

kurz	lang	Semantik
`.`	`self::node()`	Kontextknoten
`..`	`parent::node()`	Elternknoten des Kontextknotens
`//`	`/descendant-or-self::node()/`	Kontextknoten einschließlich aller seiner Nachkommen
`@*`	`attribute::*`	alle Attributknoten

- Wildcards für Knotentests:

`node()`	Knoten jeden Typs
`*`	achensabhängig: Element- oder Attributknoten
`text()`	Textknoten
`comment()`	Kommentarknoten
`processing-instruction()`	Verarbeitungsanweisungsknoten

- Spezifikation von alternativen Lokalisierungspfaden:

  Pfad_1 | *Pfad_2* | ... | *Pfad_n*

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

(d) `/personen/descendant::name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

(d) `/personen/descendant::name`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

(d) `/personen/descendant::name`

(e) `//person[geburtstag!='unknown']`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" ...?>

<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Beispiele für Lokalisierungspfade:

(a) `//person/name/descendant::*`

(b) `//geburtstag/parent::*/name`

(c) `/personen/child::name`

(d) `/personen/descendant::name`

(e) `//person[geburtstag!='unknown']`

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Algorithmus zur Auswertung eines Lokalisierungspfades:

. . . / *Schritt_i* / *Schritt_i+1* / . . .

1. Die Auswertung der Lokalisierungsschritte geschieht von links nach rechts.
2. Jeder Lokalisierungsschritt spezifiziert eine Knotenmenge M.

Die XSL-Familie

XPath-Lokalisierungspfade (Fortsetzung)

Algorithmus zur Auswertung eines Lokalisierungspfades:

. . . / *Schritt_i* / *Schritt_i+1* / . . .

1. Die Auswertung der Lokalisierungsschritte geschieht von links nach rechts.
2. Jeder Lokalisierungsschritt spezifiziert eine Knotenmenge M.

3. Jeder Knoten n der Knotenmenge M_i des Lokalisierungsschritts i wird als Kontextknoten hinsichtlich des Lokalisierungsschritts $i+1$ interpretiert und spezifiziert im Lokalisierungsschritt $i+1$ die Knotenmenge M_{i_n}.
4. Die Vereinigung der Mengen M_{i_n}, $n \in M_i$, bildet die Knotenmenge M_{i+1} des Lokalisierungsschritts $i+1$.

Bemerkungen:

- Jeder der in irgendeinem Schritt spezifizierten Knoten kommt im Laufe der Auswertung in die Rolle des Kontextknotens.

- Vergleich verschiedener Lokalisierungspfade am Beispiel:
  - Alle `<beruf>`-Elemente:
    `/descendant-or-self::node()/beruf` (bzw. `//beruf`)
    $\equiv$
    `/descendant-or-self::beruf`

  - Von jedem Elementknoten das jeweils dritte `<beruf>`-Kindelement:
    `/descendant-or-self::node()/beruf[3]` (bzw. `//beruf[3]`)
    $\not\equiv$
    `/descendant-or-self::beruf[3]`
    (das dritte `<beruf>`-Element im gesamten Dokument)

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
1
2  <?xml version="1.0" ?>
3  <?xml-stylesheet type="text/xsl" ...?>

4  <personen>
5    <person>
6      <name>
7        <vorname>Alan</vorname>
8        <nachname>Turing</nachname>
6      </name>
9      <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
5    </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
4  </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
1
2  <?xml version="1.0" ?>
3  <?xml-stylesheet type="text/xsl" ...?>

4  <personen>
5    <person>
6      <name>
7        <vorname>Alan</vorname>
8        <nachname>Turing</nachname>
6      </name>
9      <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
5    </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
4  </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
/person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
 1
 2 <?xml version="1.0" ?>
 3 <?xml-stylesheet type="text/xsl" ...?>

 4 <personen>
 5   <person>
 6     <name>
 7       <vorname>Alan</vorname>
 8       <nachname>Turing</nachname>
 6     </name>
 9     <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
 5   </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
 4 </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
 1
 2 <?xml version="1.0" ?>
 3 <?xml-stylesheet type="text/xsl" ...?>

 4 <personen>
 5   <person>
 6     <name>
 7       <vorname>Alan</vorname>
 8       <nachname>Turing</nachname>
 6     </name>
 9     <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
 5   </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
 4 </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
 1
 2 <?xml version="1.0" ?>
 3 <?xml-stylesheet type="text/xsl" ...?>

 4 <personen>
 5   <person>
 6     <name>
 7       <vorname>Alan</vorname>
 8       <nachname>Turing</nachname>
 6     </name>
 9     <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
 5   </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
 4 </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

Lokalisierungspfade (Fortsetzung)

Knoten

```
 1
 2 <?xml version="1.0" ?>
 3 <?xml-stylesheet type="text/xsl" ...?>

 4 <personen>
 5   <person>
 6     <name>
 7       <vorname>Alan</vorname>
 8       <nachname>Turing</nachname>
 6     </name>
 9     <geburtstag>23. Juni 1912</geburtstag>
10     <beruf>Mathematiker</beruf>
11     <beruf>Informatiker</beruf>
 5   </person>

12   <person>
13     <name>
14       <vorname>Judea</vorname>
15       <nachname>Pearl</nachname>
13     </name>
16     <geburtstag>unknown</geburtstag>
17     <beruf>Informatiker</beruf>
12   </person>
 4 </personen>
```

Illustration des Algorithmus:

```
//person/name/descendant::*
//person/name/descendant::*
```

$M_1 = \{1\}$

$M_2 = \{1, \ldots, 17\}$

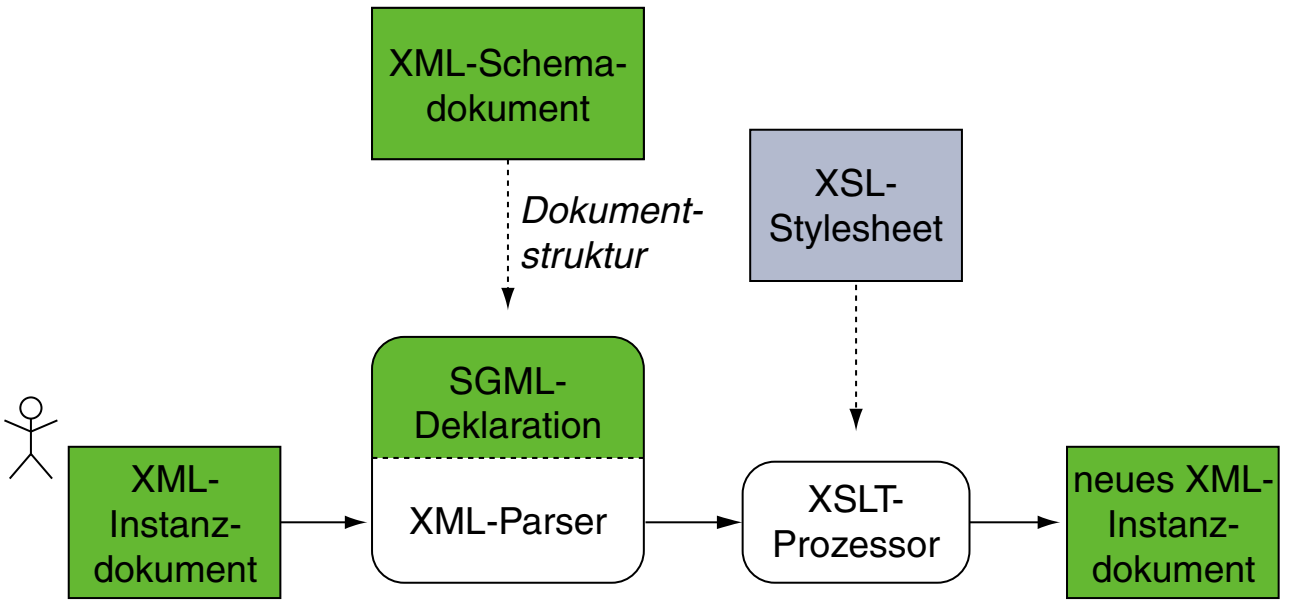
$M_3 = \{5, 12\}$

$M_4 = \{6, 13\}$

$M_5 = \{7, 8, 14, 15\}$

Die XSL-Familie

XSL Transformation



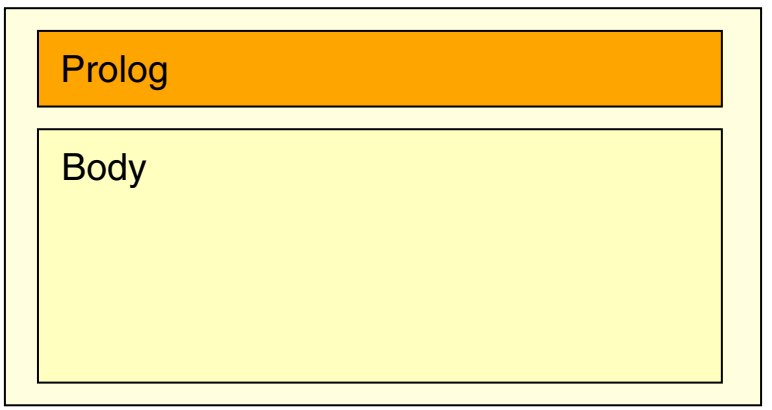
XSLT ist eine Turing-vollständige Programmiersprache zur Transformation wohlgeformter XML-Dokumente in andere XML-Dokumente. Ein XSLT-Programm liegt üblicherweise als XSL-Stylesheet vor.

Die Transformation umfasst die Selektion von Teilen des Eingabedokuments, deren Umordnung sowie die Generierung neuer Inhalte aus den bestehenden.

Die XSL-Familie

Aufbau eines XSL-Stylesheets

XSL-Stylesheets sind XML-Dokumente:




```
<?xml version="1.0"?>


<xsl:stylesheet xmlns:xsl=...>
  <xsl:template match=...>
      ...
  </xsl:template>
  ...
</xsl:stylesheet>
```

- Wurzelelement jedes XSL-Schemas ist das Element `<xsl:stylesheet>` oder synonym `<xsl:transform>`.
- Die Kindelemente von `<xsl:stylesheet>` bzw. `<xsl:transform>` definieren Transformationsvorschriften in Form von Template-Regeln.
- Vergleiche hierzu die XML-Dokumentstruktur und die XML-Schema-Dokumentstruktur.

Bemerkungen:

- Das Vokabular zur Definition von XSL-Stylesheets gehört zum Namensraum http://www.w3.org/1999/XSL/Transform. Das übliche Präfix bei der Namensraumdeklaration ist `xsl:`, es kann aber beliebig gewählt werden. Wird der offizielle Namensraum gebunden, ist auch das Attribut `version="1.0"` anzugeben.
- Die Dateiendung einer XSL-Stylesheet-Datei ist `.xsl`.
- Aufbau einer realen Turingmaschine.

Die XSL-Familie

XML-Beispieldokument

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="personen2html.xsl"?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Bemerkungen:

- Die Verknüpfung von XML-Dokument und XSL-Stylesheet kann explizit, in Form von Parametern für den XSLT-Prozessor, aber auch implizit geschehen:
  Die Zeile `<?xml-stylesheet type="text/xsl" href="..."?>` im Prolog eines XML-Dokuments deklariert ein Stylesheet. Vergleiche hierzu die Stylesheet-Deklaration in HTML-Dokumenten.
- Beispiel: Verknüpfung von personen.xml mit einem Stylesheet zu einem HTML-Dokument.
- Aufruf des XSLT-Prozessors Xalanjava über die Kommandozeile:
  ```
  java org.apache.xalan.xslt.Process -in personen.xml -xsl tiny.xsl
  ```
  Hierfür muss der Ort der Xalan-Bibliothek `xalan.jar` im Classpath spezifiziert sein. Alternativ der Aufruf mit expliziter Angabe der Xalan-Bibliothek:
  ```
  java -cp /usr/share/java/xalan.jar ...
  ```

Die XSL-Familie

Elemente eines XSL-Stylesheets

Das einfachste (= leere) Stylesheet:

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

Die XSL-Familie

Elemente eines XSL-Stylesheets

Das einfachste (= leere) Stylesheet:

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

Angewandt auf das Beispieldokument:

```
<?xml version="1.0"?>

     Alan
     Turing

   23. Juni 1912
   Mathematiker
   Informatiker

     Judea
     Pearl

   unknown
   Informatiker
```

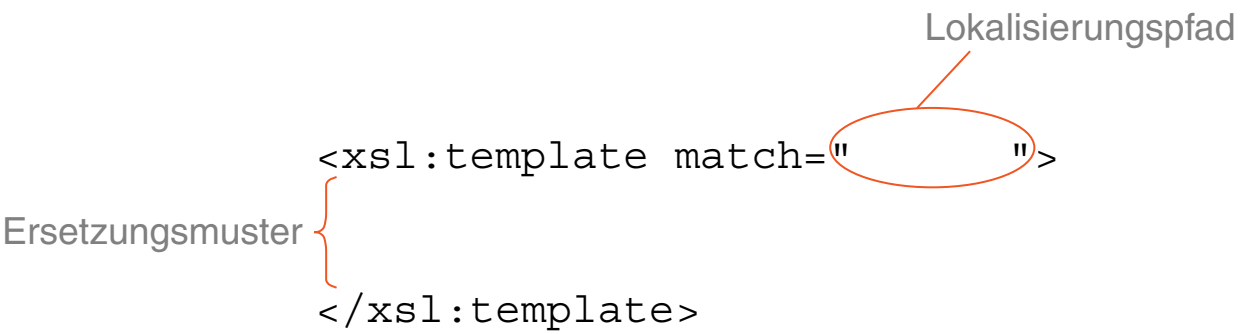
Bemerkungen:

- In diesem Beispiel enthält das Stylesheet keine matchende Template-Regel. Die Ausgabe entsteht, weil in einer solchen Situation vom XSLT-Prozessor das Built-in-Template zur Ausgabe von Text- und Attributknoten angewandt wird.
- Diejenigen Konstrukte eines XML-Dokuments, die nicht zu einem der sieben Knotentypen des XPath-Modells gehören, werden unverändert übernommen. Hierzu zählt u.a. die `<?xml ...?>`-Deklaration.

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Wichtigstes Stylesheet-Element ist die Template-Regel *(Template)*:

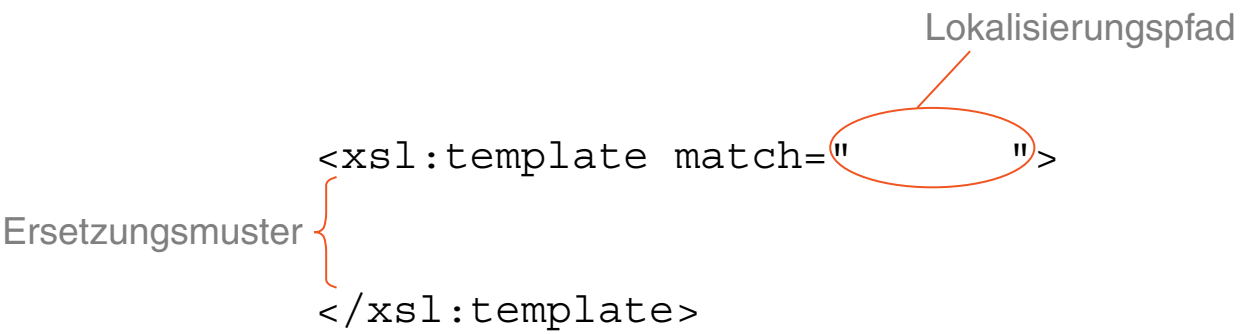


- Der Lokalisierungspfad des `match`-Attributs spezifiziert – ausgehend von dem Kontextknoten – eine Knotenmenge M.
- Wird während der Verarbeitung eines XML-Dokuments ein Knoten n mit $n \in M$ erreicht, dann matched die Template-Regel diesen Knoten.

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Wichtigstes Stylesheet-Element ist die Template-Regel *(Template)*:



- Der Lokalisierungspfad des `match`-Attributs spezifiziert – ausgehend von dem Kontextknoten – eine Knotenmenge M.
- Wird während der Verarbeitung eines XML-Dokuments ein Knoten n mit $n \in M$ erreicht, dann matched die Template-Regel diesen Knoten.
- Matched eine Template-Regel einen Knoten n, behandelt das Ersetzungsmuster den gesamten Teilbaum des XML-Dokuments, der Knoten n als Wurzel hat. Dieser Teilbaum gilt als abgearbeitet.

Bemerkungen:

- Der Wert des `match`-Attributes im `<xsl:template>`-Element ist ein Lokalisierungspfad in eingeschränkter XPath-Syntax.
- Um die Knotenmenge M zu spezifizieren für deren Elemente eine Template-Regel matched, sind alternative Pfadangaben möglich. Beispielsweise spezifizieren die Ausdrücke `match`="`//`*Elementname*" und `match`="*Elementname*" dieselbe Knotenmenge.
  D.h., ein relativer Lokalisierungspfad des `<xsl:template>`-Elements kann wie der entsprechende absolute durch "`//`" eingeleitete Lokalisierungspfad aufgefasst werden – und umgekehrt.

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet mit literaler Ausgabe:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:text>Person found!</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet mit literaler Ausgabe:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:text>Person found!</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
Person found!

Person found!
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zum Kopieren der Elemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:copy-of select="self::*"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zum Kopieren der Elemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:copy-of select="self::*"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
  <geburtstag>23. Juni 1912</geburtstag>
  <beruf>Mathematiker</beruf>
  <beruf>Informatiker</beruf>
</person>
...
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zum Kopieren der Elementinhalte:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:value-of select="self::*"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zum Kopieren der Elementinhalte:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="person">
    <xsl:value-of select="self::*"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
    Alan
    Turing

  23. Juni 1912
  Mathematiker
  Informatiker

  ...
```

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zur Elementselektion mittels leerer Template-Regeln:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

  <xsl:template match="geburtstag"/>
  <xsl:template match="beruf"/>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

Die XSL-Familie

Elemente eines XSL-Stylesheets (Fortsetzung)

Stylesheet zur Elementselektion mittels leerer Template-Regeln:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

  <xsl:template match="geburtstag"/>
  <xsl:template match="beruf"/>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
Turing, Alan
Pearl, Judea
```

Vergleiche die Elementselektion durch explizite Verarbeitungssteuerung.

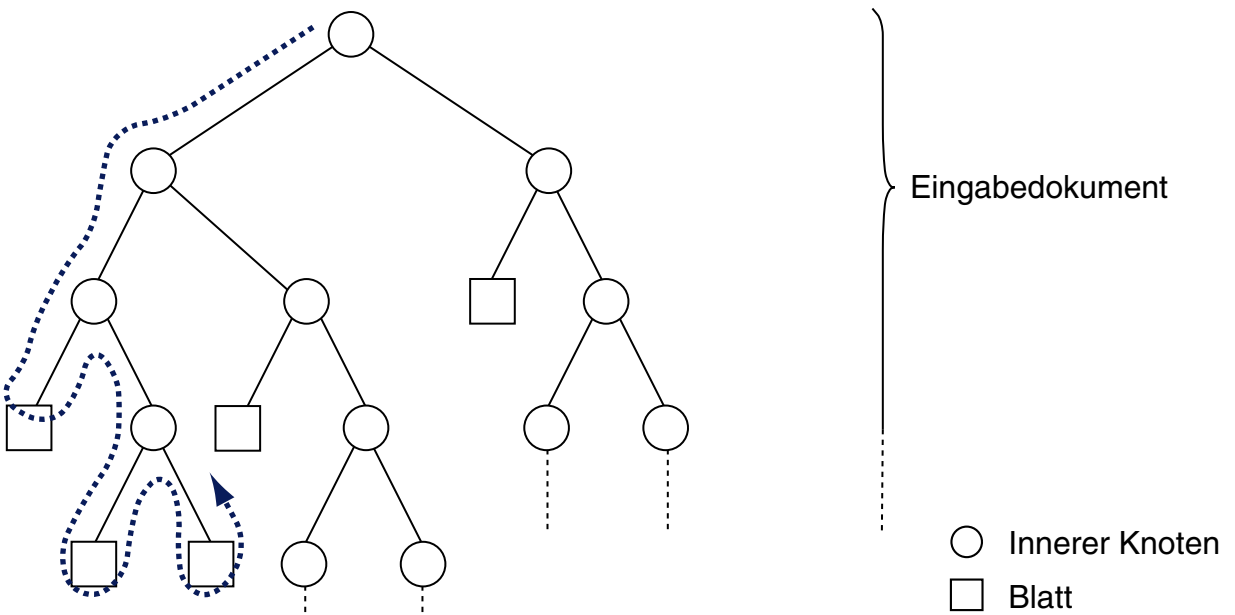
Bemerkungen (Wiederholung):

- Matched eine Template-Regel einen Knoten im XML-Dokument, so gilt der Knoten einschließlich des zugehörigen Teilbaums als abgearbeitet.
- Mit leeren Template-Regeln kann man Knoten und Teilbäume filtern, die nicht in der Ausgabe erscheinen sollen.
- Matched keine Template-Regel des Stylesheets einen Knoten im XML-Dokument, wird vom XSLT-Prozessor das Built-in-Template zur Ausgabe von Text- und Attributknoten angewandt.

Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie

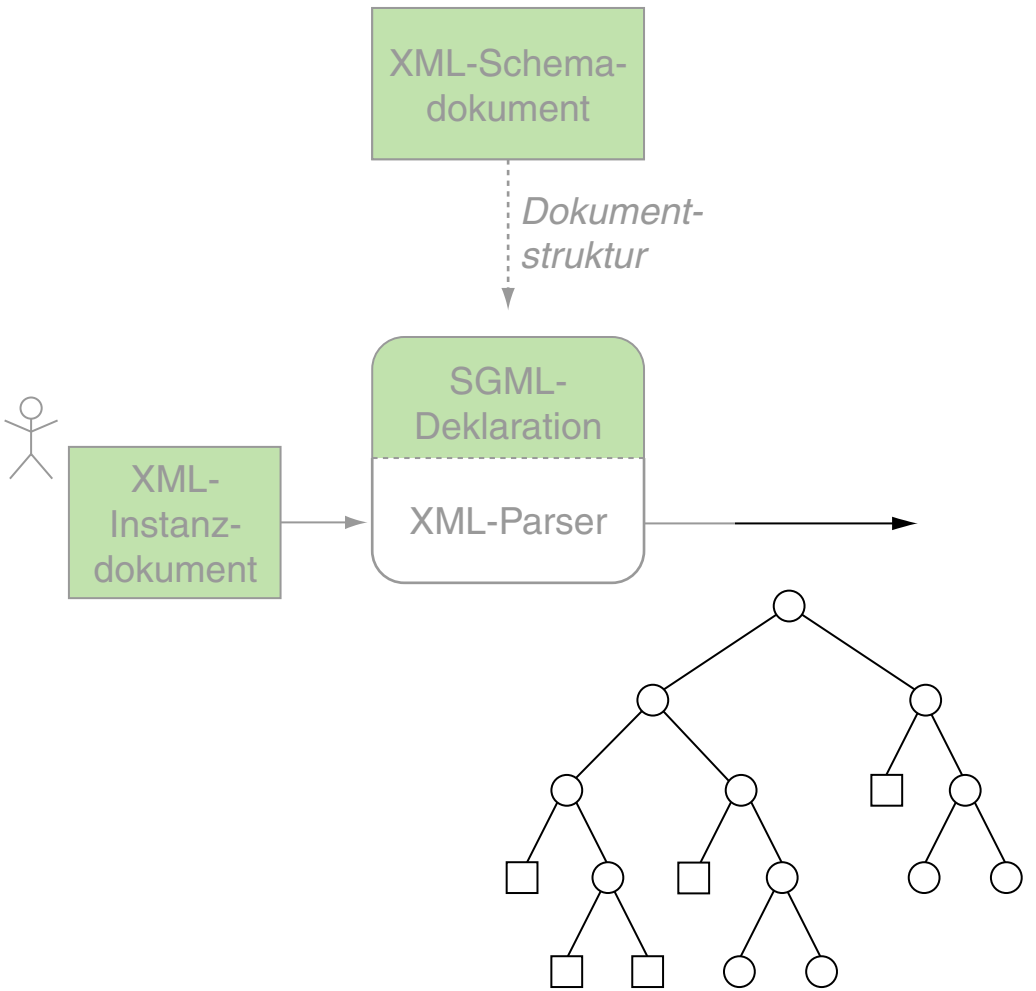
Standardmäßig durchläuft der XSLT-Prozessor den aus dem Eingabedokument erzeugten Baum ausgehend vom Wurzelknoten in Preorder-Reihenfolge.



Während des Traversierungsvorgangs wird für jeden besuchten Knoten das speziellste, matchende Template gesucht und angewandt. So transformiert der XSLT-Prozessor einen XML-Quellbaum in einen XML-Zielbaum.

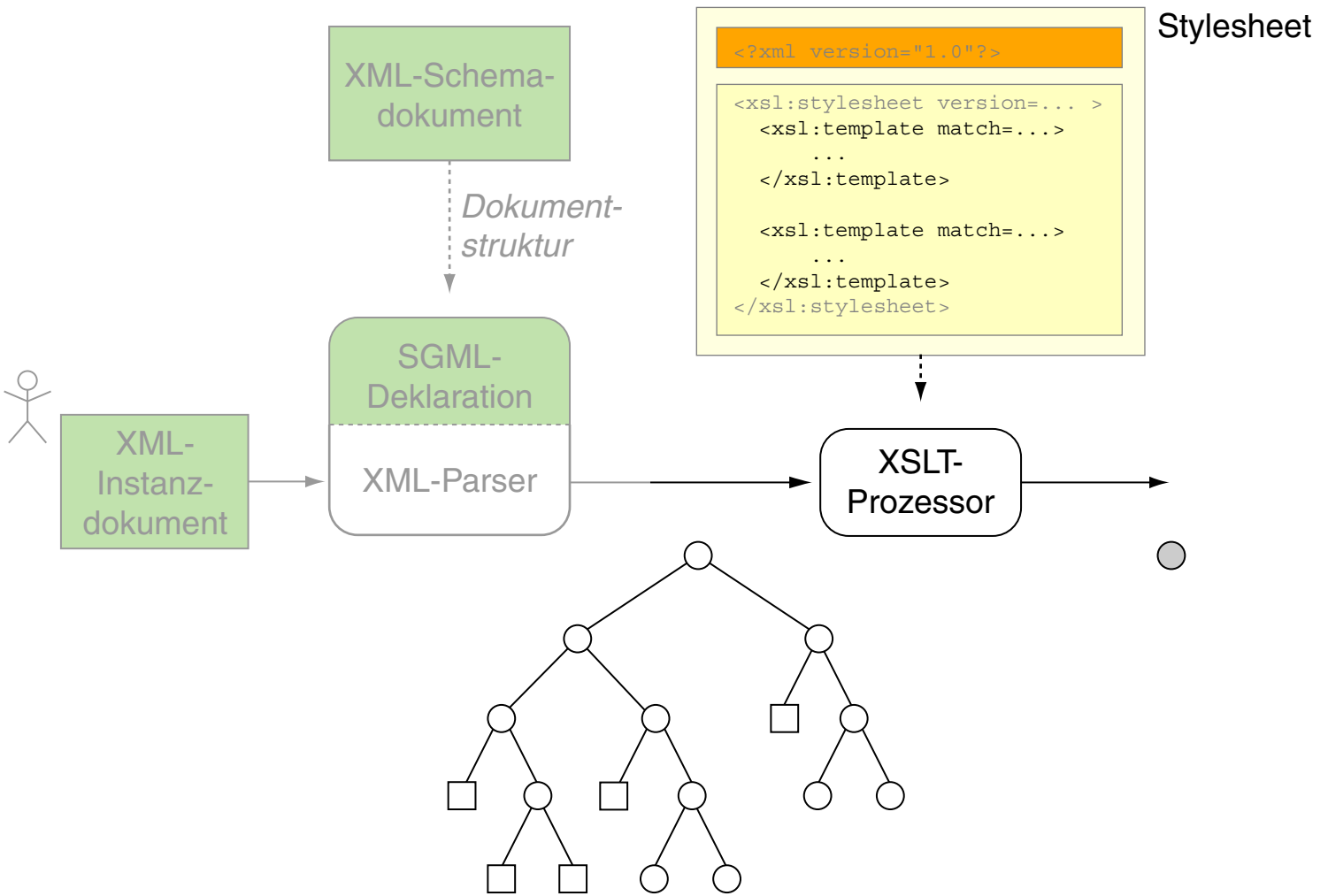
Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)



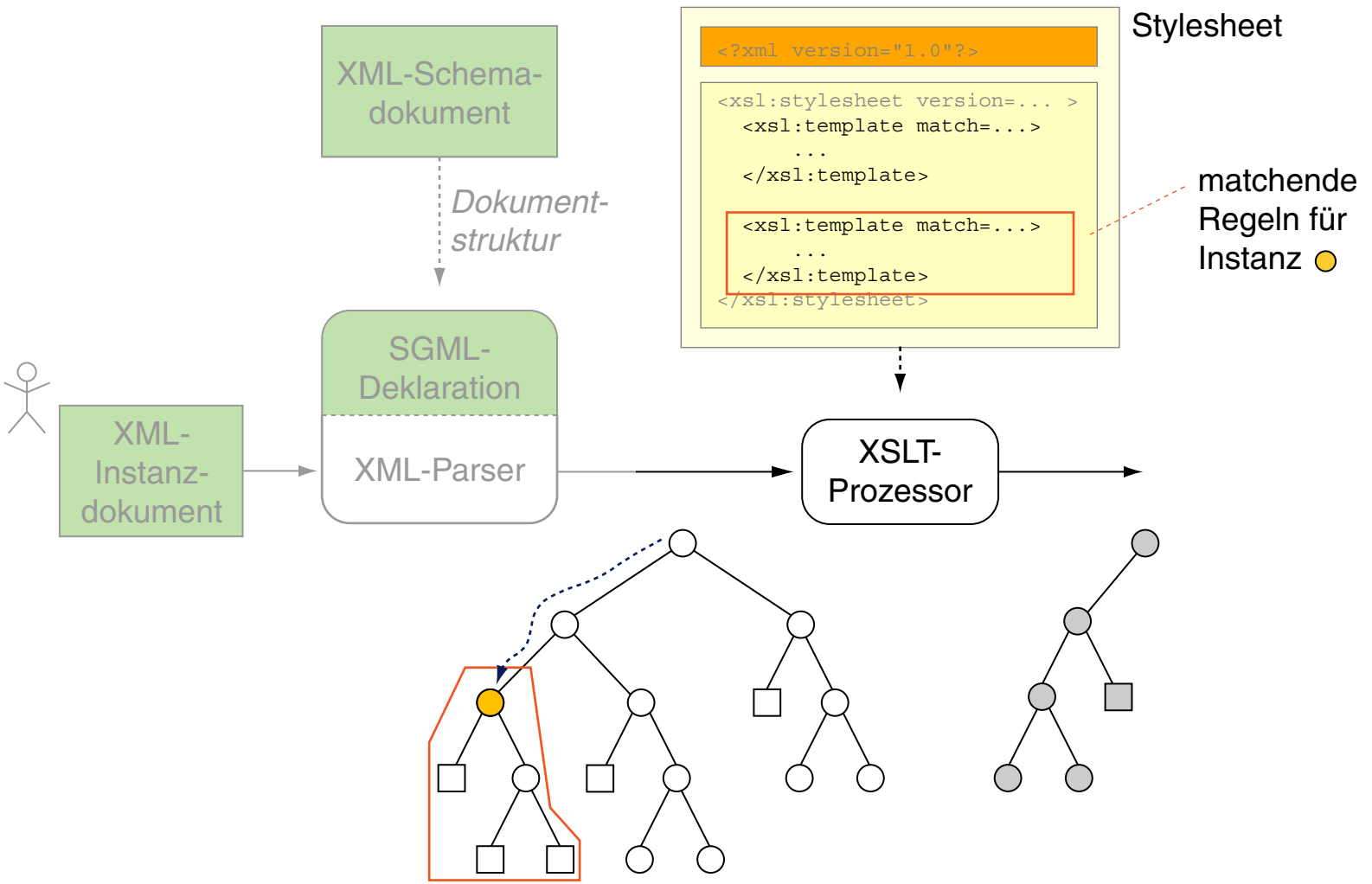
Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)



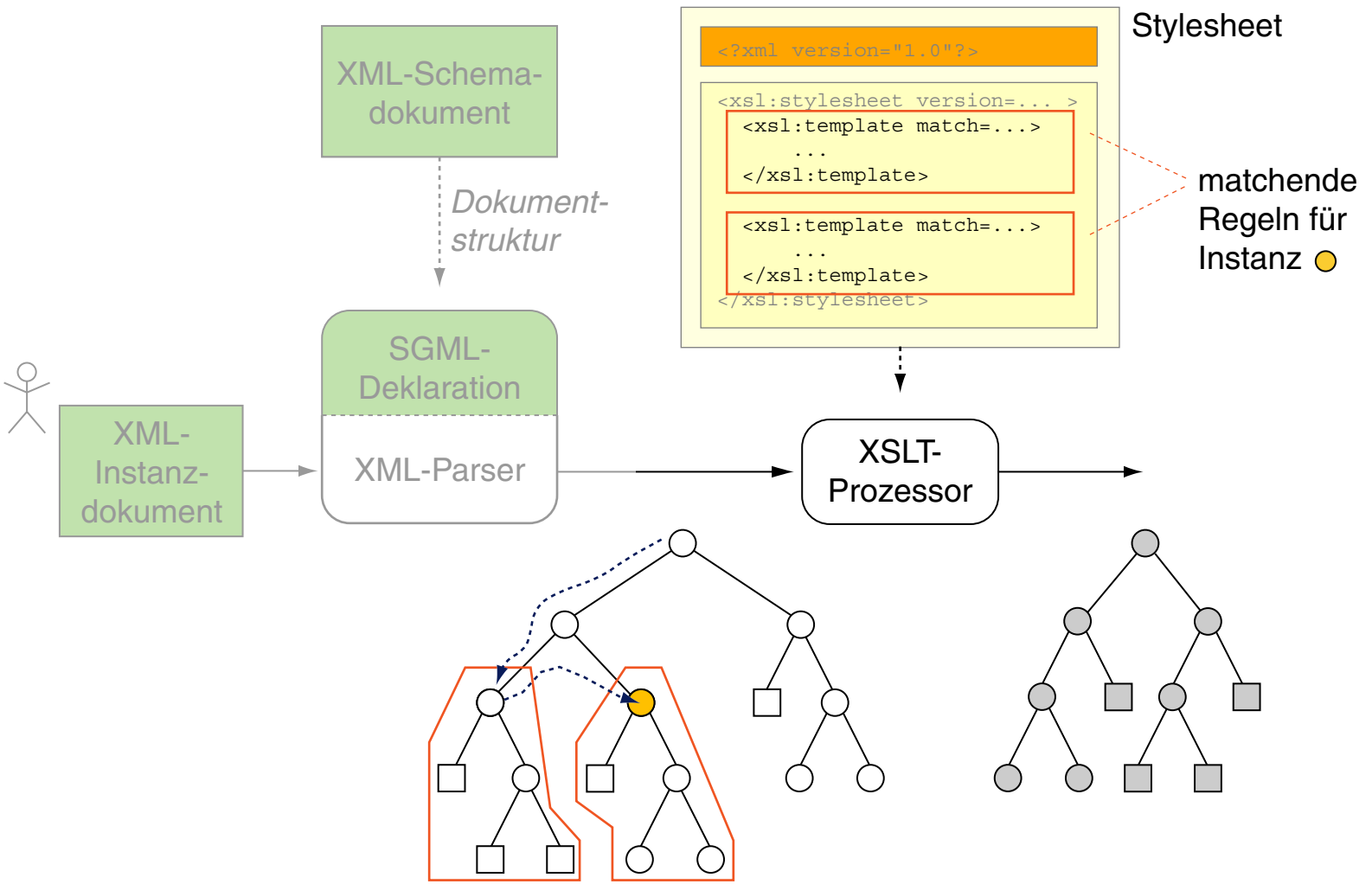
Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)



Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung) [WT:III CSS-Verarbeitung]



Bemerkungen:

- Aus Verarbeitungssicht spielt somit die Reihenfolge der Template-Regeln in einem XSL-Stylesheet keine Rolle: die Verarbeitung wird ausschließlich durch die *Reihenfolge der Elemente im Eingabedokument* bestimmt.
- Ein Anwendungskonflikt liegt vor, wenn Lokalisierungspfade von verschiedenen Template-Regeln t_1, t_2 einen Knoten n in ihrer spezifizierten Knotenmengen M_{t_1}, M_{t_2} enthalten. In diesem Fall kommt das Template t_x, $x \in \{1, 2\}$, mit dem speziellsten Pfad im `match`-Attribut zur Anwendung: $|M_{t_x}| \leq \min\{|M_{t_1}|, |M_{t_2}|\}$

Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)

Stylesheet zur Elementselektion mit expliziter Verarbeitungssteuerung:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

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

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)

Stylesheet zur Elementselektion mit expliziter Verarbeitungssteuerung:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

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

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
Turing, Alan
Pearl, Judea
```

Vergleiche die Elementselektion mittels leerer Template-Regeln.

Bemerkungen:

- Das `<xsl:apply-templates>`-Element startet für die mit dem `select`-Attribut spezifizierte Knotenmenge erneut einen Preorder-Durchlauf zur Anwendung der Template-Regeln des Stylesheets.
- Der Wert des `select`-Attributes im `<xsl:apply-templates>`-Element ist ein Lokalisierungspfad in eingeschränkter XPath-Syntax. Weil sich so beliebige Knoten im Dokument spezifizieren lassen, ermöglicht das `<xsl:apply-templates>`-Element die mehrmalige Verarbeitung von Knoten, also auch die Erzeugung von Endlosschleifen.
- Falls keine andere Achse angegeben ist, setzt der Lokalisierungspfad des `<xsl:apply-templates>`-Elements den Pfad des matchenden Knoten fort. Das heißt, die Ausdrücke `select=`“`./`*Elementname*” und `select=`“*Elementname*” spezifizieren dieselbe Knotenmenge.
- Enthält das `<xsl:apply-templates>`-Element kein `select`-Attribut, so gelten per Default die Kindknoten (`child::`-Achse) des matchenden Knoten als spezifiziert.

Die XSL-Familie

XSLT-Prozessor: Verarbeitungsstrategie (Fortsetzung)

Algorithm: xsl:apply-templates

Input: `select`. XPath expression or empty string.
n_c. Context node.
T. XSL stylesheet with templates.

Output: –

`xsl:apply-templates(select`, n_c, T)

1. `nodes` = *evalXpath*(`select`, n_c)
2. `LOOP`
3. `IF nodes` = $\emptyset$ `THEN RETURN`
4. n = *pop*(`nodes`)
5. t = *mostSpecificTemplate*(T, n)
6. `IF` $t \neq$ `Null`
   `THEN` *executeTemplate*(t, n)
   `ELSE` *executeBuiltInTemplate*(n)
7. `ENDLOOP`

Bemerkungen:

- Die Funktionen *executeTemplate*(t, n) und *executeBuiltInTemplate*(n) wenden das Ersetzungsmuster eines `<xsl:template>`-Elements auf den Knoten n an.
- Der Preorder-Durchlauf entsteht durch den rekursiven Aufruf von `xsl:apply-templates()` in Schritt 6 – entweder durch benutzerdefinierte `<xsl:apply-templates>`-Elemente in t oder durch Anwendung eines Built-in-Templates.
- Der XSLT-Prozessor verwaltet intern das XML Information Set des zu verarbeitenden XML-Dokuments und stellt der Funktion `xsl:apply-templates()` den Kontextknoten n_c und das Stylesheet T zur Verfügung.

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie

Stylesheet mit zweifacher Verarbeitung der `<name>`-Kindelemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

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

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie

Stylesheet mit zweifacher Verarbeitung der `<name>`-Kindelemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

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

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
  Turing, AlanTuring, Alan

  Pearl, JudeaPearl, Judea
```

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie (Fortsetzung)

Stylesheet zur wiederholten Verarbeitung aller `<name>`-Elemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="https://www.w3.org/...">

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

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie (Fortsetzung)

Stylesheet zur wiederholten Verarbeitung aller `<name>`-Elemente:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="https://www.w3.org/...">

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

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
  Turing, AlanPearl, Judea
  Turing, AlanPearl, Judea
```

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie (Fortsetzung)

Stylesheet, dessen matchende Template-Regel die leere Knotenmenge liefert:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

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

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie (Fortsetzung)

Stylesheet, dessen Verarbeitung in eine Endlosschleife führt:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="https://www.w3.org/...">

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

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

Die XSL-Familie

XSLT-Prozessor: Beispiele zur Verarbeitungsstrategie (Fortsetzung)

Stylesheet, dessen Verarbeitung in eine Endlosschleife führt:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="https://www.w3.org/...">

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

  <xsl:template match="name">
    <xsl:value-of select="nachname"/>
    <xsl:text>, </xsl:text>
    <xsl:value-of select="vorname"/>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
(Location of error unknown)XSLT Error (java.lang.StackOverflowError):
null
```

Die XSL-Familie

XSLT-Prozessor: Built-in-Templates [xpath notation]

1. Built-in-Template, das die rekursive Verarbeitung garantiert, falls kein matchendes Template im Stylesheet existiert:

```
<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

2. Built-in-Template zur Ausgabe von Text- und Attributknoten:

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

3. Built-in-Template, das die Kommentare matched und ignoriert:

```
<xsl:template match="processing-instruction()|comment()"/>
```

   Vergleiche die Elementselektion mittels leerer Template-Regeln.

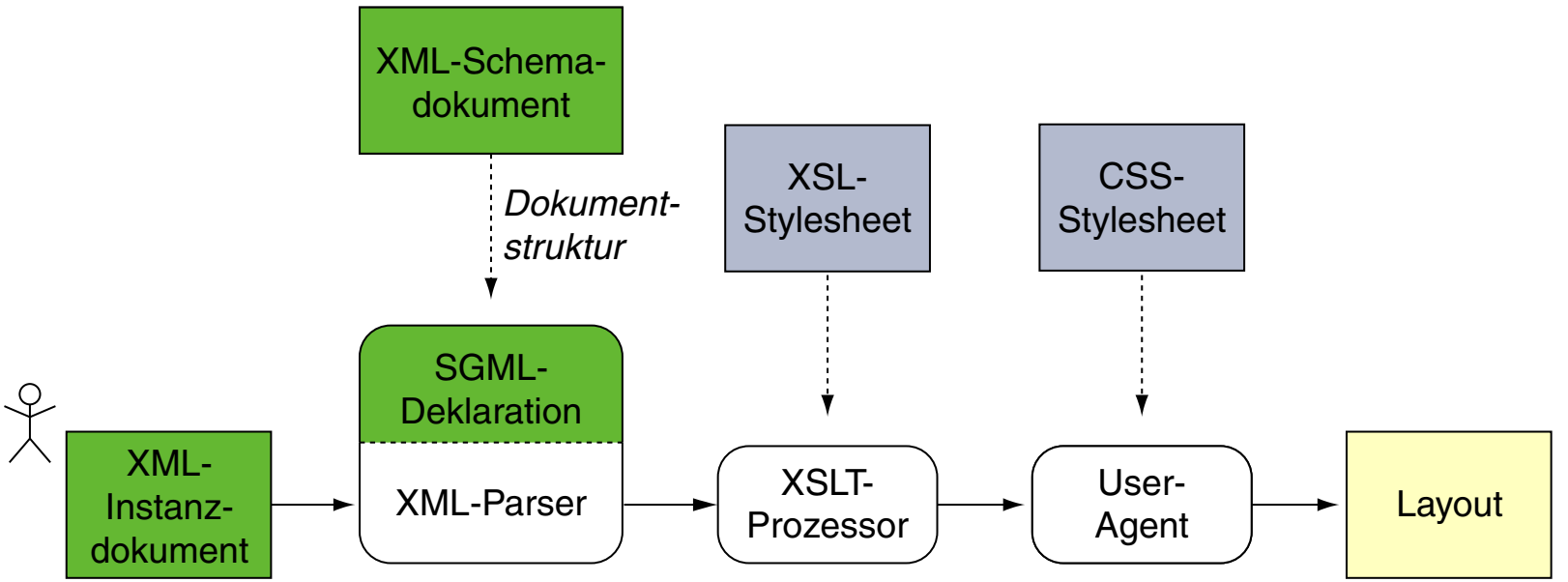
Die XSL-Familie

Weitere XSLT-Konzepte

- Template-Modi zur Charakterisierung von Verarbeitungsphasen
- benannte Templates zur Realisierung direkter Aufrufe
- Nummerierung und Sortierung von Ausgabeelementen
- bedingte Verarbeitung und Schleifen
- Import anderer Stylesheets

Die XSL-Familie

XML-Dokumentenverarbeitung: Erzeugung von HTML-Dokumenten



Vergleiche hierzu den Standardprozess der XSL Transformation.

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Zusammenspiel mit CSS

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="personen2html.xsl"?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Zusammenspiel mit CSS (Fortsetzung)

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Zusammenspiel mit CSS (Fortsetzung)

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Zusammenspiel mit CSS (Fortsetzung)

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Zusammenspiel mit CSS (Fortsetzung)

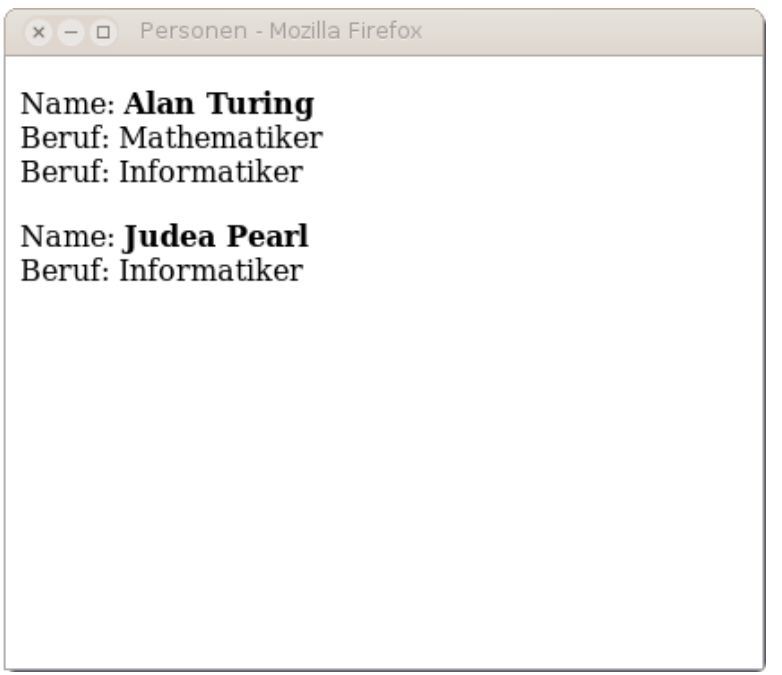
```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="name">
  <p/>
  <div>
    <xsl:text>Name: </xsl:text>
    <span style="font-weight:bold">
      <xsl:value-of select="self::*"/>
    </span>
  </div>
</xsl:template>
...
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Zusammenspiel mit CSS (Fortsetzung)

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="name">
  <p/>
  <div>
    <xsl:text>Name: </xsl:text>
    <span style="font-weight:bold">
      <xsl:value-of select="self::*"/>
    </span>
  </div>
</xsl:template>
...
```

[ohne / mit Stylesheet]



Die XSL-Familie

Erzeugung von HTML-Dokumenten: Zusammenspiel mit CSS (Fortsetzung)

```
<xsl:template match="personen">
  <html>
    <head>
      <title>
        <xsl:text>Personen</xsl:text>
      </title>
      <link rel="stylesheet" type="text/css" href="personen.css"/>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="name">
  <p/>
  <div>
    <xsl:text>Name: </xsl:text>
      <xsl:value-of select="self::*"/>
  </div>
</xsl:template>
...
```

Bemerkungen:

- Eine Anwendung nach diesem Prinzip sind die FAQs des W3C:
  Aus der XML-Source faq.xml gemäß der DTD faq.dtd wird mittels des Stylesheets faqxsl.xsl das HTML-Dokument faq.html erzeugt.

  Weil in faq.xml das Stylesheet faq.css verlinkt ist, zeigt der Browser nicht den XML-Dokumentenbaum an:

```
<?xml version="1.0"?>
<!DOCTYPE faq SYSTEM "faq.dtd">
<?xml-stylesheet href="faq.css" type="text/css"?>
<faq>
  <head>
    <title>Document Object Model FAQ</title>
    ...
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung

CD-Datenbank als XML-Beispieldokument [w3schools] :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  ...
  <cd>
    <title>Unchain my heart</title>
    <artist>Joe Cocker</artist>
    <company>EMI</company>
    <price>8.20</price>
    <year>1987</year>
  </cd>
</catalog>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/c
      <tr>
        <td><xsl:value-of select="titl
        <td><xsl:value-of select="arti
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

[w3schools xml, xsl]

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Filtern mit XPath:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd[artist='Bob Dylan']">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Filtern mit XPath:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd[artist='Bob Dylan']">
      <tr>
        <td><xsl:value-of select="tit
        <td><xsl:value-of select="art
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

[w3schools]

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Sortieren:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:sort select="artist"/>
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Sortieren:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/c
      <xsl:sort select="artist"/>
      <tr>
        <td><xsl:value-of select="titl
        <td><xsl:value-of select="arti
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Verwenden von Bedingungen:

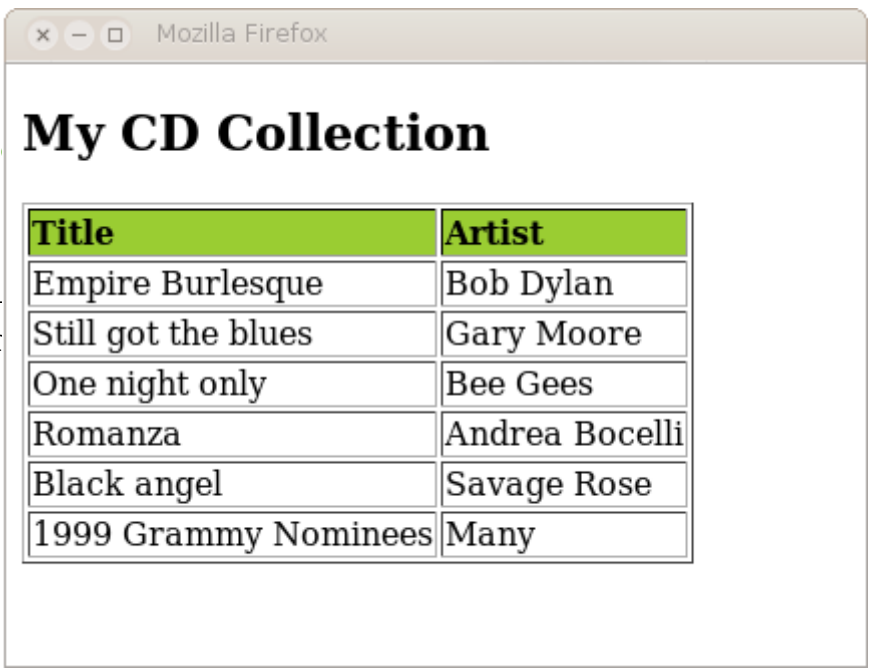
```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:if test="price &gt; 10">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
      </xsl:if>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Verwenden von Bedingungen:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/c
      <xsl:if test="price &gt; 10">
        <tr>
          <td><xsl:value-of select="ti
          <td><xsl:value-of select="ar
        </tr>
      </xsl:if>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```



Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Verwenden von Bedingungen:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      ...
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <xsl:choose>
          <xsl:when test="price &gt; 10">
            <td bgcolor="#ff00ff"><xsl:value-of select="artist"/></td>
          </xsl:when>
          <xsl:otherwise>
            <td><xsl:value-of select="artist"/></td>
          </xsl:otherwise>
        </xsl:choose>
      </tr>
      </xsl:for-each>
    </table>
  ...
```

Die XSL-Familie

Erzeugung von HTML-Dokumenten: Datenaufbereitung (Fortsetzung)

Verwenden von Bedingungen:

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      ...
      <xsl:for-each select="catalog/c
      <tr>
        <td><xsl:value-of select="titl
        <xsl:choose>
          <xsl:when test="price &gt; 1
            <td bgcolor="#ff00ff"><xsl
          </xsl:when>
          <xsl:otherwise>
            <td><xsl:value-of select="
          </xsl:otherwise>
        </xsl:choose>
      </tr>
      </xsl:for-each>
    </table>
  ...
```

Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen [WT:III DOM-API]

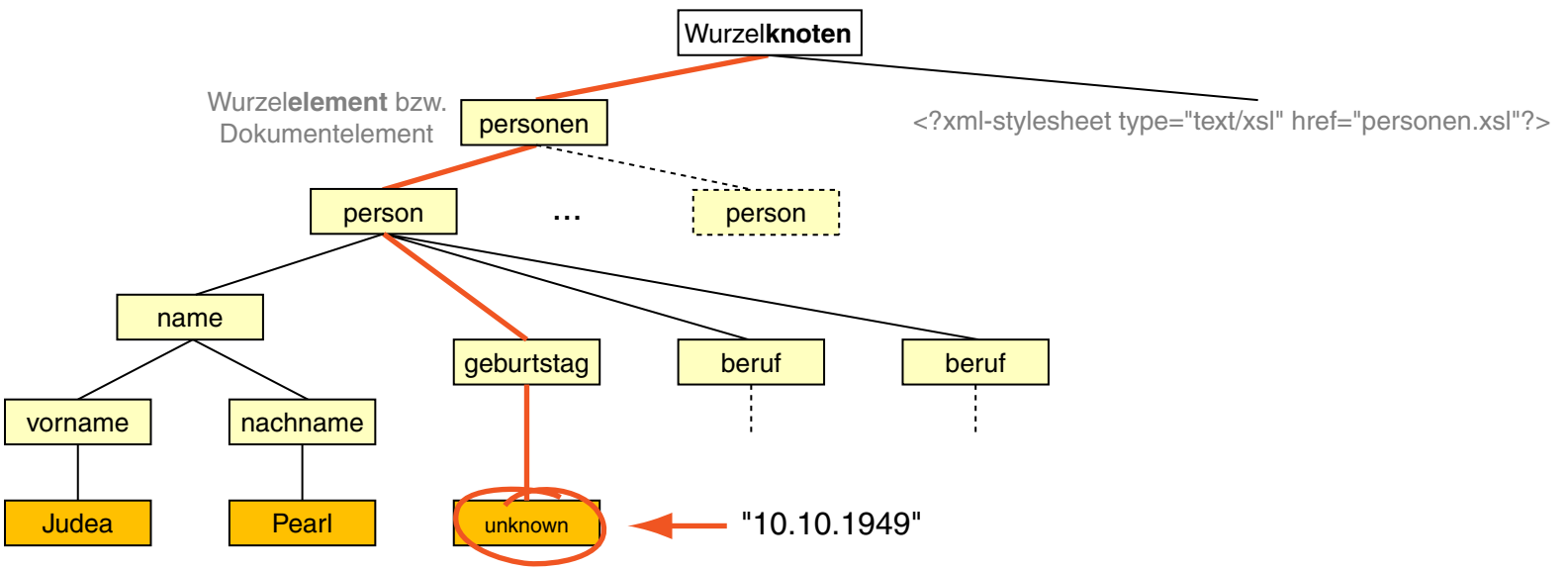
```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="personen2html.xsl"?>
<personen>
  <person>
    <name>
      <vorname>Alan</vorname>
      <nachname>Turing</nachname>
    </name>
    <geburtstag>23. Juni 1912</geburtstag>
    <beruf>Mathematiker</beruf>
    <beruf>Informatiker</beruf>
  </person>

  <person>
    <name>
      <vorname>Judea</vorname>
      <nachname>Pearl</nachname>
    </name>
    <geburtstag>unknown</geburtstag>
    <beruf>Informatiker</beruf>
  </person>
</personen>
```

Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen (Fortsetzung)

Aufgabe:

1. Die Person „Judea Pearl“ finden.
2. Seinen Geburtstag auf einen bestimmten Wert setzen.



Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen (Fortsetzung)

Stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="@*|node()"> [xpath notation]
    <xsl:copy><xsl:apply-templates select="@*|node()|"/></xsl:copy> [W3C]
  </xsl:template>




</xsl:stylesheet>
```

Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen (Fortsetzung)

Stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="@*|node()"> [xpath notation]
    <xsl:copy><xsl:apply-templates select="@*|node()|"/></xsl:copy> [W3C]
  </xsl:template>

  <xsl:template match= "person[name/nachname='Pearl' and
       name/vorname='Judea']/geburtstag/text()">
    <xsl:text>10.10.1949</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

Die XSL-Familie

XML-Dokumentenverarbeitung: Elementinhalte anpassen (Fortsetzung)

Stylesheet:

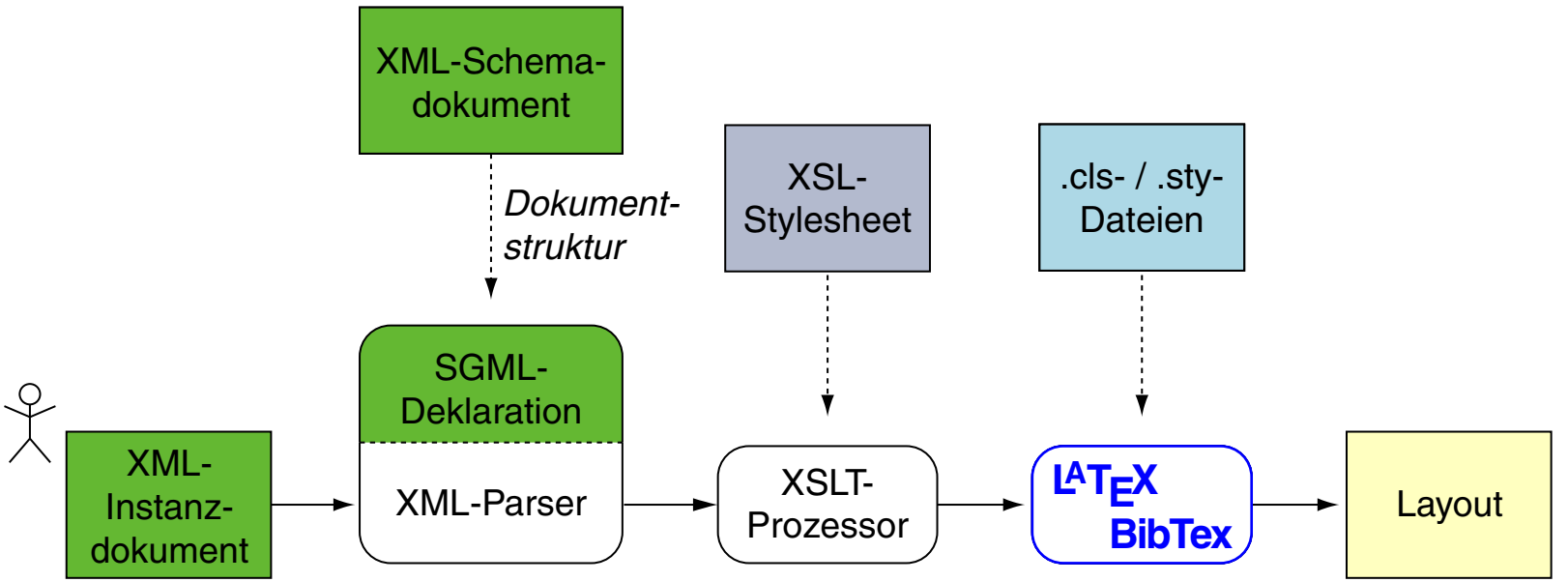
```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/...">

  <xsl:template match="@*|node()"> [xpath notation]
    <xsl:copy><xsl:apply-templates select="@*|node()|"/></xsl:copy> [W3C]
  </xsl:template>

  <xsl:template match= "person[name/nachname='Pearl' and
       name/vorname='Judea']/geburtstag/text()">
    <xsl:text>10.10.1949</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

Angewandt auf das Beispieldokument:

```
...
  <name>
    <vorname>Judea</vorname>
    <nachname>Pearl</nachname>
  </name>
  <geburtstag>10.10.1949</geburtstag>
...
```

Die XSL-Familie

XML-Dokumentenverarbeitung: Prozesskette für Printmedien



Vergleiche hierzu

- den Standardprozess der XSL Transformation
- und die HTML-Prozesskette.

Die XSL-Familie

Prozesskette für Printmedien: Erzeugung von Latex-Dokumenten

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="xml2latex.xsl"?>

<book>

<section>
  <title>Eine Überschrift</title>

  Hier ist der Fließtext ...
</section>

</book>
```

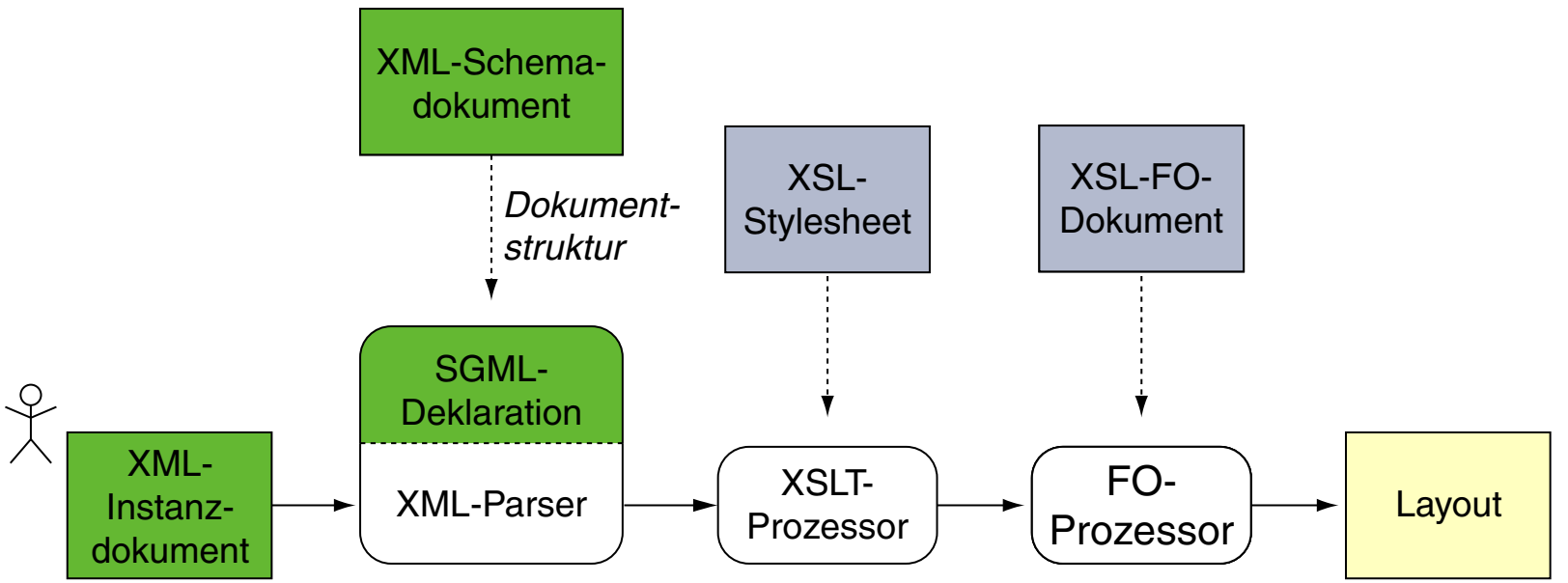
Die XSL-Familie

Prozesskette für Printmedien: Erzeugung von Latex-Dokumenten (Fortsetzung)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="https://www.w3.org/...">

<xsl:template match="/">
  \documentclass{article}
  \usepackage[T1]{fontenc}
  \usepackage[english,german]{babel}
  \begin{document}
  <xsl:apply-templates/>
  \end{document}
</xsl:template>

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

<xsl:template match="title">
  \section{<xsl:value-of select="self::*"/>}
</xsl:template>

...

</xsl:stylesheet>
```

Die XSL-Familie

XML-Dokumentenverarbeitung: Erzeugung beliebiger Formate mit XSL-FO



Vergleiche hierzu

- den Standardprozess der XSL Transformation,
- die HMTL-Prozesskette
- und die Latex-Prozesskette.

Die XSL-Familie

Quellen zum Nachlernen und Nachschlagen im Web: Referenz

- W3C. *XSL Transformations (XSLT) 3.0.*
  www.w3.org/TR/xslt-30
- W3C *XML Path Language (XPath) 3.0.*
  www.w3.org/TR/xpath-30
- W3C *XML Query Language (XQuery) 3.0.*
  www.w3.org/TR/xquery-30
- W3C *XSL Formatting Objects (XSL-FO) 2.0.*
  www.w3.org/TR/xslfo20

Die XSL-Familie

Quellen zum Nachlernen und Nachschlagen im Web: Usage

- Cover Pages. *Extensible Stylesheet Language.*
  xml.coverpages.org/xsl.html
- W3 Schools. *XSLT.*
  https://www.w3schools.com/xml/xsl_intro.asp
- Apache. *Xalan Project.*
  xalan.apache.org
- Saxonica.com. *XSLT and XQuery Processing.*
  www.saxonica.com