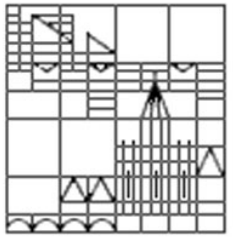


XML
eXtensible Markup Language

. . . and its usefulness for linguists

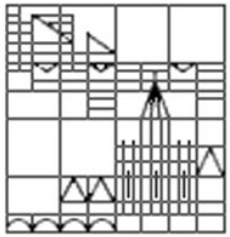
Thomas Mayer

thomas.mayer@uni-konstanz.de

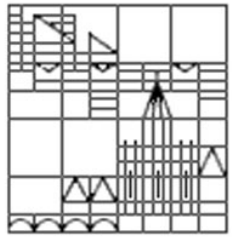
Fachbereich Sprachwissenschaft, Universität Konstanz

Seminar Computerlinguistik II (Miriam Butt)

23. Juli 2004

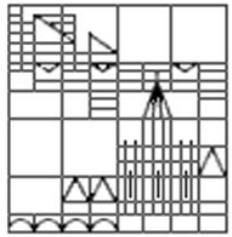


0 Outline



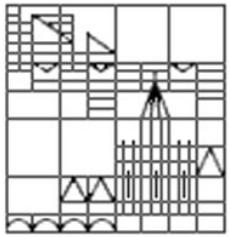
0. Outline

- Preliminaries



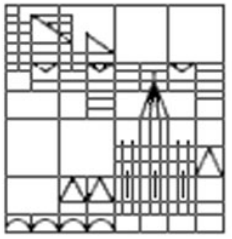
0. Outline

- Preliminaries
- Introduction to XML



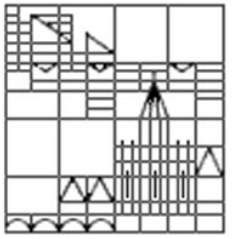
0 Outline

- Preliminaries
- Introduction to XML
- Using XML for grammar writing (with XLE)



1 Preliminaries

At least two types of applications can be distinguished

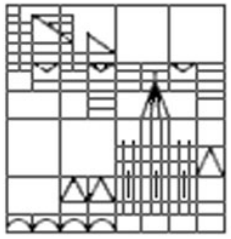


1 Preliminaries

At least two types of applications can be distinguished

- **Document-centric** applications (e.g. Information Retrieval)

  mainly text documents with little structure (chapters, sections, . . .)



1 Preliminaries

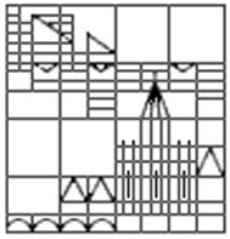
At least two types of applications can be distinguished

- **Document-centric** applications (e.g. Information Retrieval)

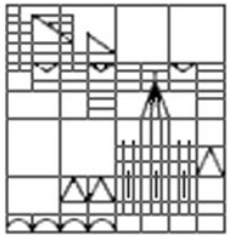
  mainly text documents with little structure (chapters, sections, . . .)

- **Data-centric** applications (”traditional” database applications)

  very regular data; sometimes, however, we might want some flexibility

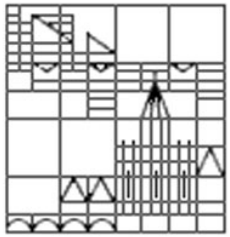


XML was designed to satisfy the needs of both worlds



XML was designed to satisfy the needs of both worlds

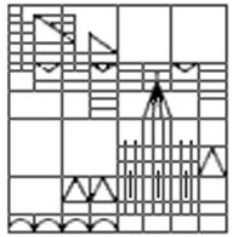
- It was designed to be simple, generic and extensible



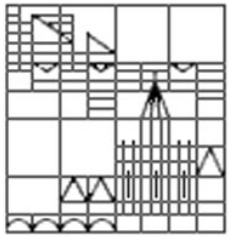
XML was designed to satisfy the needs of both worlds

- It was designed to be simple, generic and extensible
- XML documents are pure ASCII files

$\Longrightarrow$ Humans can read them

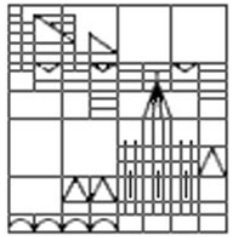


2. Introduction to XML



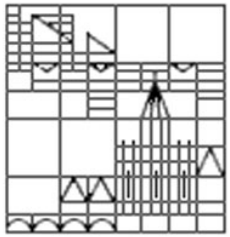
2. Introduction to XML

1. XML – a new HTML?



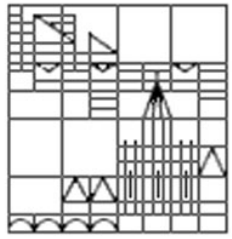
2. Introduction to XML

1. XML – a new HTML?
2. The structure of an XML document



2. Introduction to XML

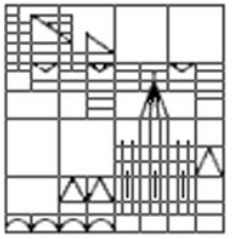
1. XML – a new HTML?
2. The structure of an XML document
3. XML data query and retrieval



2.1 XML – a new HTML?

. . . definitely not!

- HTML (HyperText Markup Language) is for displaying web contents on the browser
- HTML is a subset of XML (HTML $\subset$ XML $\subset$ SGML)
- XML could be used to determine the layout of a web page, but it is much more powerful



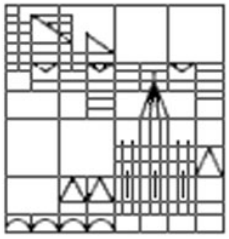
2.2 The structure of an XML document

head

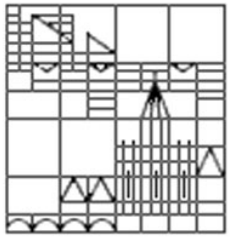
```
<?xml version="1.0"? encoding="ISO-8859-1"?>
...
```

body

```
<note>
<to>Tove</to>
<from>Jani</from>
<subject>Reminder</subject>
<message>Take a break!</message>
</note>
```

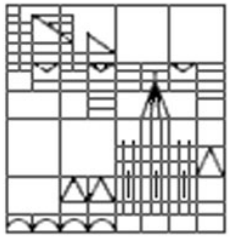


The basic element of an XML document is a tag



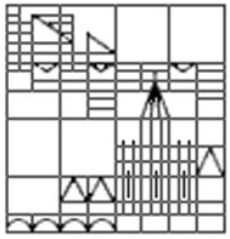
The basic element of an XML document is a tag

- A tag usually consists of a start tag (e.g. `<foo>`) and its corresponding end tag (e.g. `</foo>`)



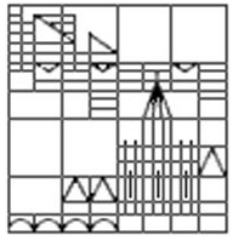
The basic element of an XML document is a tag

- A tag usually consists of a start tag (e.g. `<foo>`) and its corresponding end tag (e.g. `</foo>`)
- **Exception:** empty elements (e.g. `<foo />`)

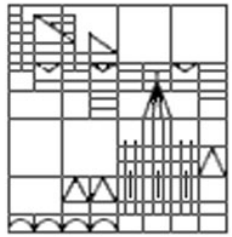


The basic element of an XML document is a tag

- A tag usually consists of a start tag (e.g. `<foo>`) and its corresponding end tag (e.g. `</foo>`)
- **Exception:** empty elements (e.g. `<foo />`)
- Elements can have attributes, i.e. pairs of attributes and values separated by an equation sign:

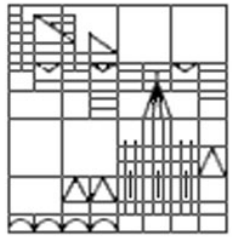


```
<foo attribute="value">
```

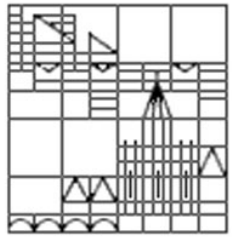


```
<foo attribute="value">
```

- Element names may contain the following characters: letters (capital and small), numbers, underscores (‘_’), hyphens (‘-’), dots (‘.’) and colons (‘:’).



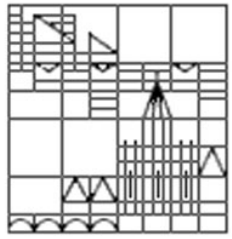
However, there are special elements . . .



However, there are special elements . . .

- **Processing Instructions (PIs):** information for programs that work on the XML document

```
<?xml-stylesheet href="headlines.css" type="te
```

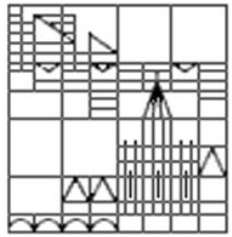


However, there are special elements . . .

- **Processing Instructions (PIs):** information for programs that work on the XML document

```
<?xml-stylesheet href="headlines.css" type="te
```

- Special elements starting with '!':

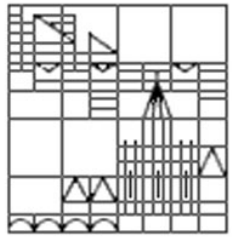


However, there are special elements . . .

- **Processing Instructions (PIs):** information for programs that work on the XML document

```
<?xml-stylesheet href="headlines.css" type="te
```

- Special elements starting with '!':
  - `<!DOCTYPE ...>`

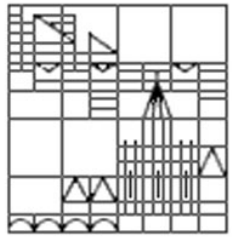


However, there are special elements . . .

- **Processing Instructions (PIs):** information for programs that work on the XML document

```
<?xml-stylesheet href="headlines.css" type="te
```

- Special elements starting with '!':
  - `<!DOCTYPE ...>`
  - `<![CDATA[ ...]]>`

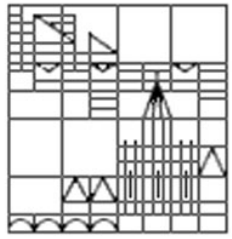


However, there are special elements . . .

- **Processing Instructions (PIs):** information for programs that work on the XML document

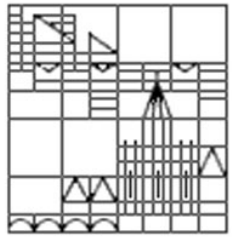
```
<?xml-stylesheet href="headlines.css" type="te
```

- Special elements starting with '!':
  - `<!DOCTYPE ...>`
  - `<![CDATA[ ...]]>`
  - `<!-- ... -->` comments as in HTML

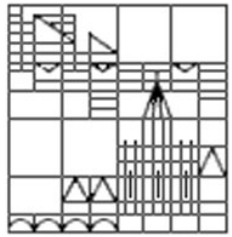


The building blocks of XML documents

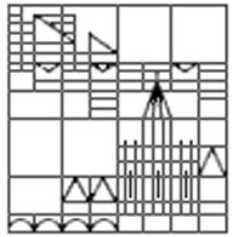
- Elements
- Tags
- Attributes
- PCDATA
- CDATA
- Processing Instructions



- Comments
- Entities



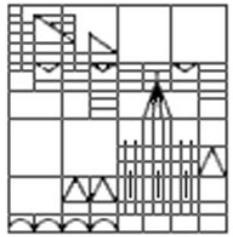
PCDATA sections



PCDATA sections

- character strings between a start and an end tag

```
<foo> ...  whatever ... </foo>
```

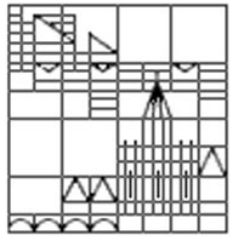


PCDATA sections

- character strings between a start and an end tag

  `<foo> ... whatever ... </foo>`

- Escape sequences for special characters `<`, `>`, `&quot;`, `&apos;`
  They begin with an ampersand and end with a semi-colon (e.g. `&lt;` for `<`)

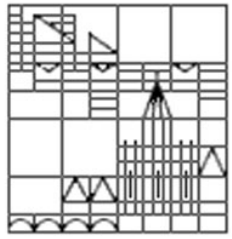


PCDATA sections

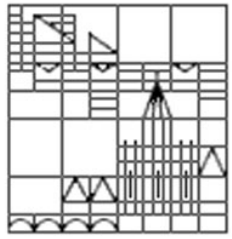
- character strings between a start and an end tag

  `<foo> ... whatever ... </foo>`

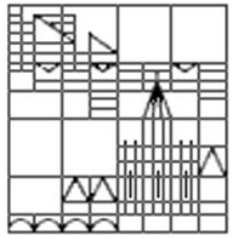
- Escape sequences for special characters `<`, `>`, `&quot;`, `&apos;`
  They begin with an ampersand and end with a semi-colon (e.g. `&lt;` for `<`)



- You can define your own entities in the same way (e.g. `&myEntity;`)

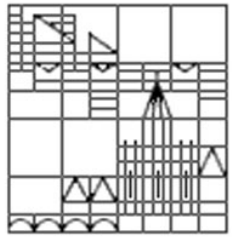


Well-formed vs. valid XML documents



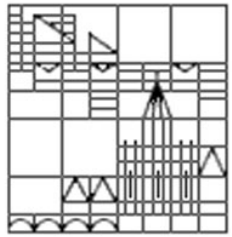
Well-formed vs. valid XML documents

- An XML document is **well-formed** if it observes the rules of the XML specification (i.e. the restrictions on element names, proper nesting, etc.)

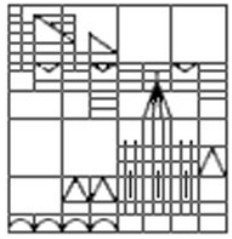


Well-formed vs. valid XML documents

- An XML document is **well-formed** if it observes the rules of the XML specification (i.e. the restrictions on element names, proper nesting, etc.)
- An XML document is **valid** if it is well-formed and conforms to its Document Type Definition (DTD) (either inline in the XML document or as an external reference), which specifies some rules

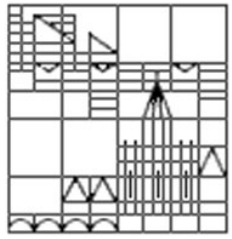


regarding the nesting of certain elements, etc.

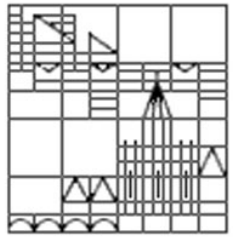


regarding the nesting of certain elements, etc.

- What does a DTD look like?



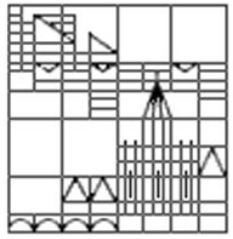
Document Type Definition (DTD)



Document Type Definition (DTD)

- **Internal DTD:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note [
...
]>
```



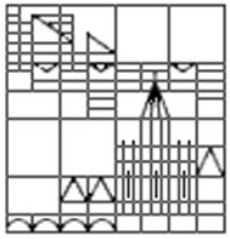
Document Type Definition (DTD)

- **Internal DTD:**

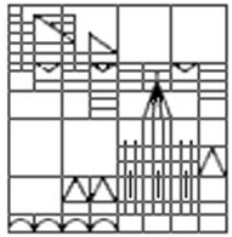
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note [
...
]>
```

- **External DTD:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

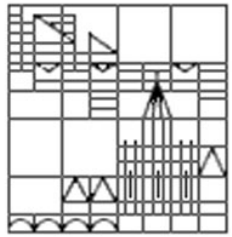


```
<!DOCTYPE note SYSTEM "note.dtd">
...
```

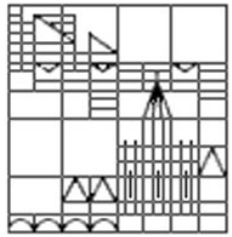


An example DTD ("note.dtd"):

```
<?xml version="1.0"?>
<!ELEMENT note (to,from,subject,message)>
<!ELEMENT to(#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT message (#PCDATA)>
```

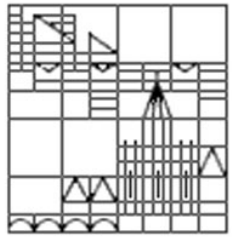


2.3 XML related standards



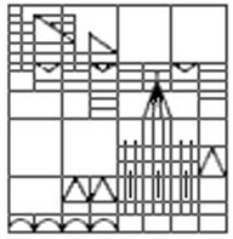
2.3 XML related standards

- XPath



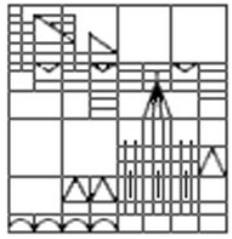
2.3 XML related standards

- XPath
- XQuery

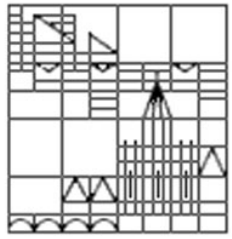


2.3 XML related standards

- XPath
- XQuery
- XSLT

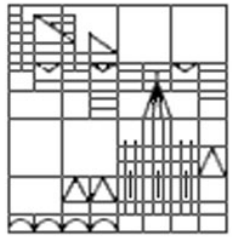


XPath



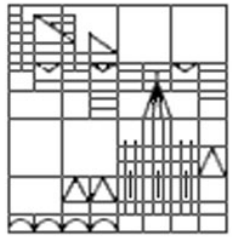
XPath

- "Querying XML data" essentially means to **identify (or address) nodes**, and to test further properties of these nodes.



XPath

- "Querying XML data" essentially means to **identify (or address) nodes**, and to test further properties of these nodes.
- There are certain relationships between nodes (**axes**)

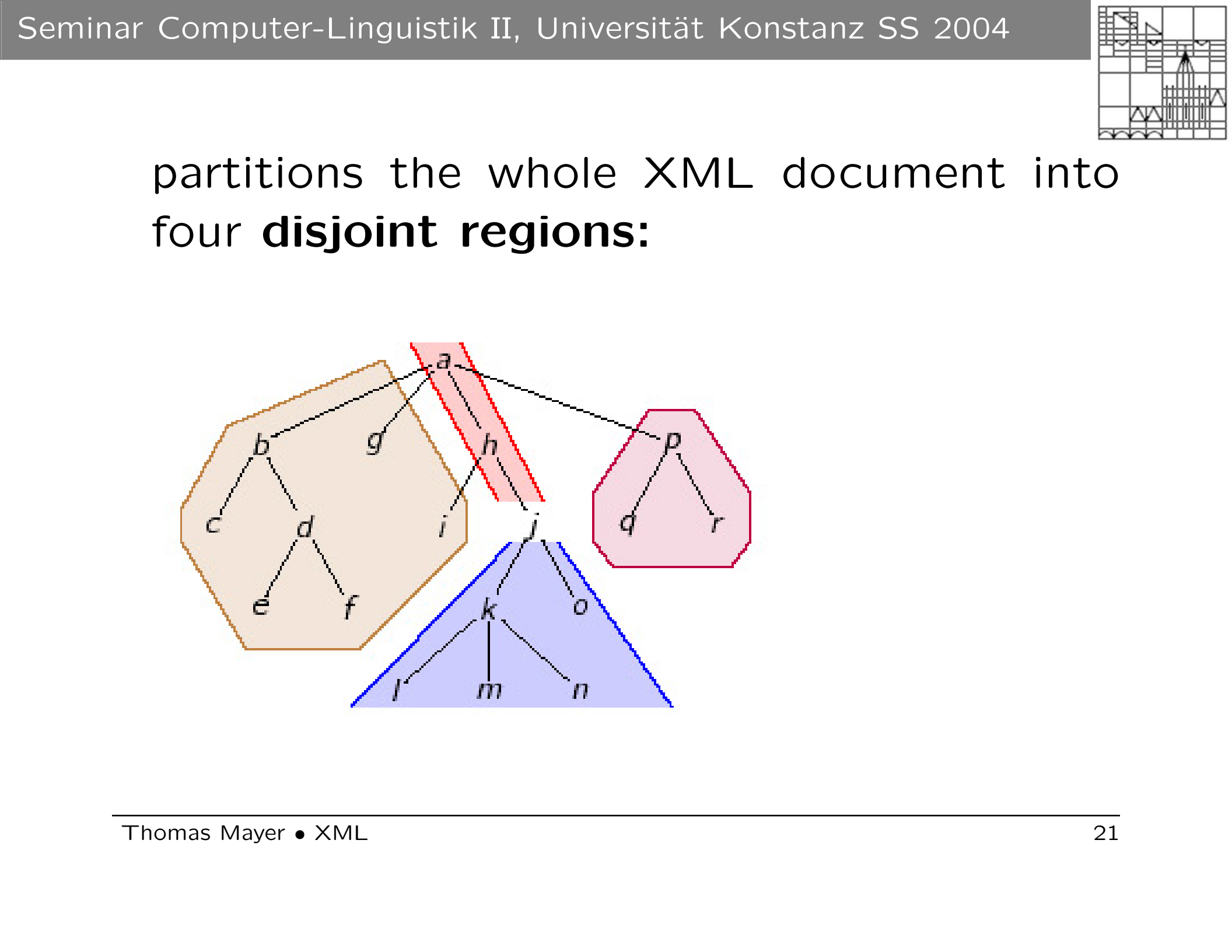


XPath

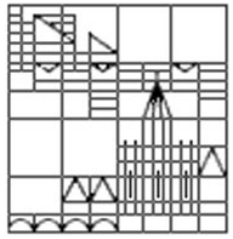
- "Querying XML data" essentially means to **identify (or address) nodes**, and to test further properties of these nodes.

- There are certain relationships between nodes (**axes**)

- Given the relationships **preceding/following** and **ancestor/descendant** each node

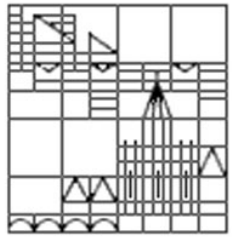


partitions the whole XML document into four **disjoint regions:**

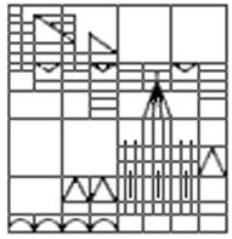


Thirteen axes are defined by the XPath standard:

Axis α	**Nodes reachable from context node c**
`ancestor`	all nodes in the ancestor region of c
`ancestor-or-self`	like `ancestor`, plus c
`child`	children of c
`descendant`	all nodes in the descendant region of c
`descendant-or-self`	like `descendant`, plus c
`following`	all nodes in the following region of c
`following-sibling`	siblings of c in the following region of c
`parent`	parent of c
`preceding`	all nodes in the preceding region of c
`preceding-sibling`	siblings of c in the preceding region of c
`self`	c
`attribute`	attributes of c
`namespace`	namespace nodes of c (not discussed in this course)

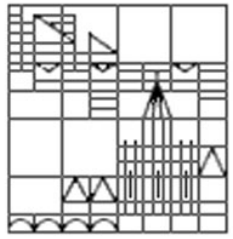


The syntax of XPath



The syntax of XPath

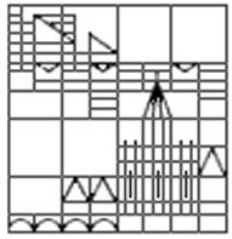
- The path expression is the core construct of XPath



The syntax of XPath

- The path expression is the core construct of XPath
- Each path consists of one or more **steps**, syntactically separated by /.

$$s_0/s_1/\ldots/s_n$$

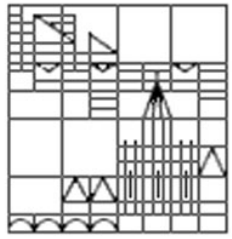


The syntax of XPath

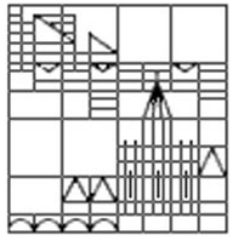
- The path expression is the core construct of XPath
- Each path consists of one or more **steps**, syntactically separated by /.

$$s_0/s_1/\ldots/s_n$$

- The entire XPath expression will always return a **sequence of nodes** (duplicate free and in document order)

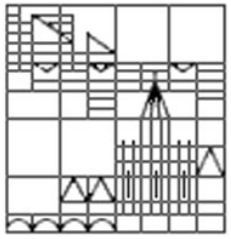


XQuery



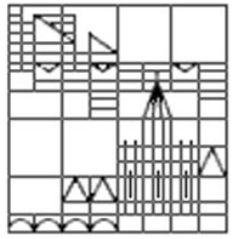
XQuery

- XQuery is a standard means to query XML databases

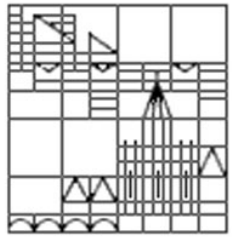


XQuery

- XQuery is a standard means to query XML databases
- It extends XPath by adding some powerful new operators and an elaborate type system

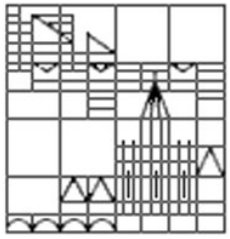


XSLT



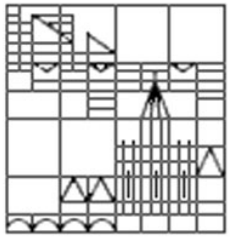
XSLT

- XSL (eXtensible Stylesheet Language) consists of two important components:



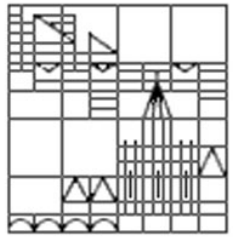
XSLT

- XSL (eXtensible Stylesheet Language) consists of two important components:
  - one for formatting XML data



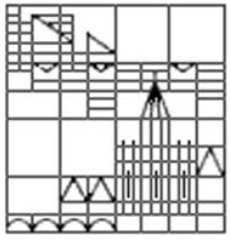
XSLT

- XSL (eXtensible Stylesheet Language) consists of two important components:
  - one for formatting XML data
  - one for transforming XML data into other XML data



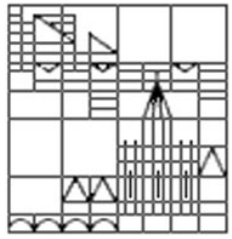
XSLT

- XSL (eXtensible Stylesheet Language) consists of two important components:
  - one for formatting XML data
  - one for transforming XML data into other XML data
- The second component is often abbreviated to XSLT (XSL Transformation)



3. Grammar writing with XML

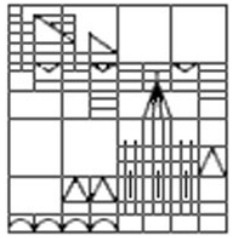
XML can be used for . . .



3. Grammar writing with XML

XML can be used for . . .

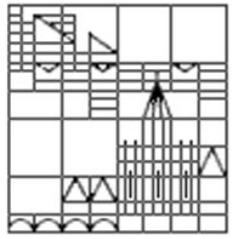
- documenting grammars



3. Grammar writing with XML

XML can be used for . . .

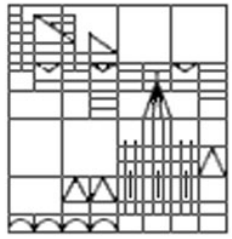
- documenting grammars
- annotating corpora



3. Grammar writing with XML

XML can be used for . . .

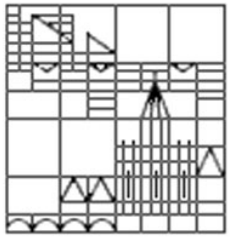
- documenting grammars
- annotating corpora
- encoding grammars



3. Grammar writing with XML

XML can be used for . . .

- documenting grammars
- annotating corpora
- encoding grammars
- . . .



3.1 Documenting grammars

Idea: The source grammar (XLE) is annotated with XML tags for documentation purposes.

Together with a source documentation you can extract the information from the grammar and build a complete (output) documentation for the grammar.

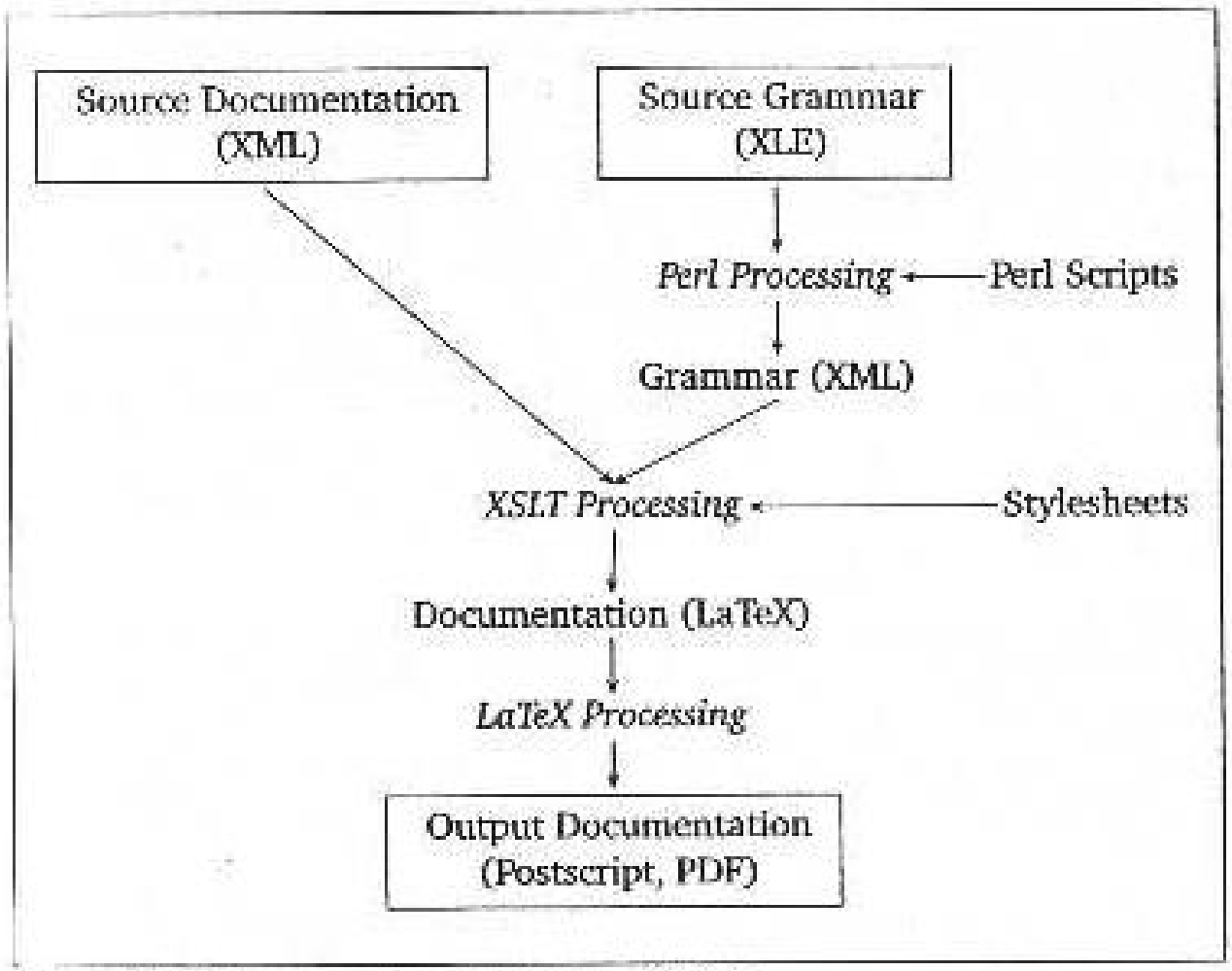
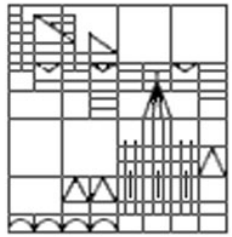
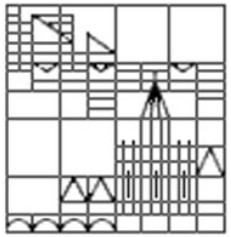


Figure 5.2: Documenting via XML and XSLT



3.2 Annotating corpora

We already discussed that . . .

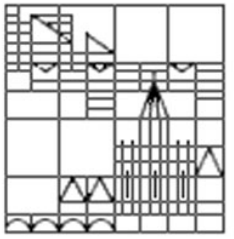


3.3 Encoding grammars

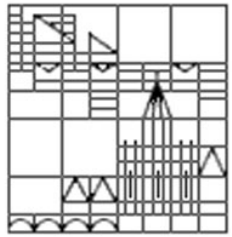
A rule such as

```
NP -->
      D: ^=!;
      N: ^=!;
```

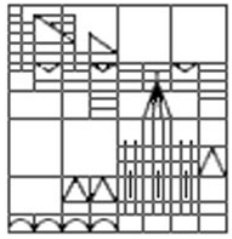
can be encoded as . . .



```
<rule>
   <mother>NP</mother>
   <daughters>
      <daughter>
         <c_struc>D</c_struc>
         <f_struc>^=!</f_struc>
      </daughter>
      <daughter>
         <c_struc>N</c_struc>
         <f_struc>^=!</f_struc>
      </daughter>
   </daughters>
</rule>
```

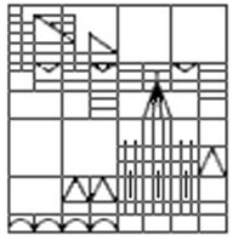


Several output files could be generated



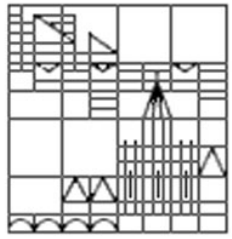
Several output files could be generated

- grammar code



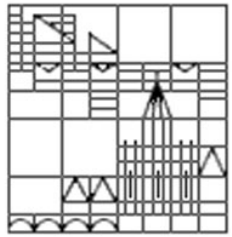
Several output files could be generated

- grammar code
- pdf-file with tree structure



Several output files could be generated

- grammar code
- pdf-file with tree structure
- html-file with links to the rules



Have a nice term break!