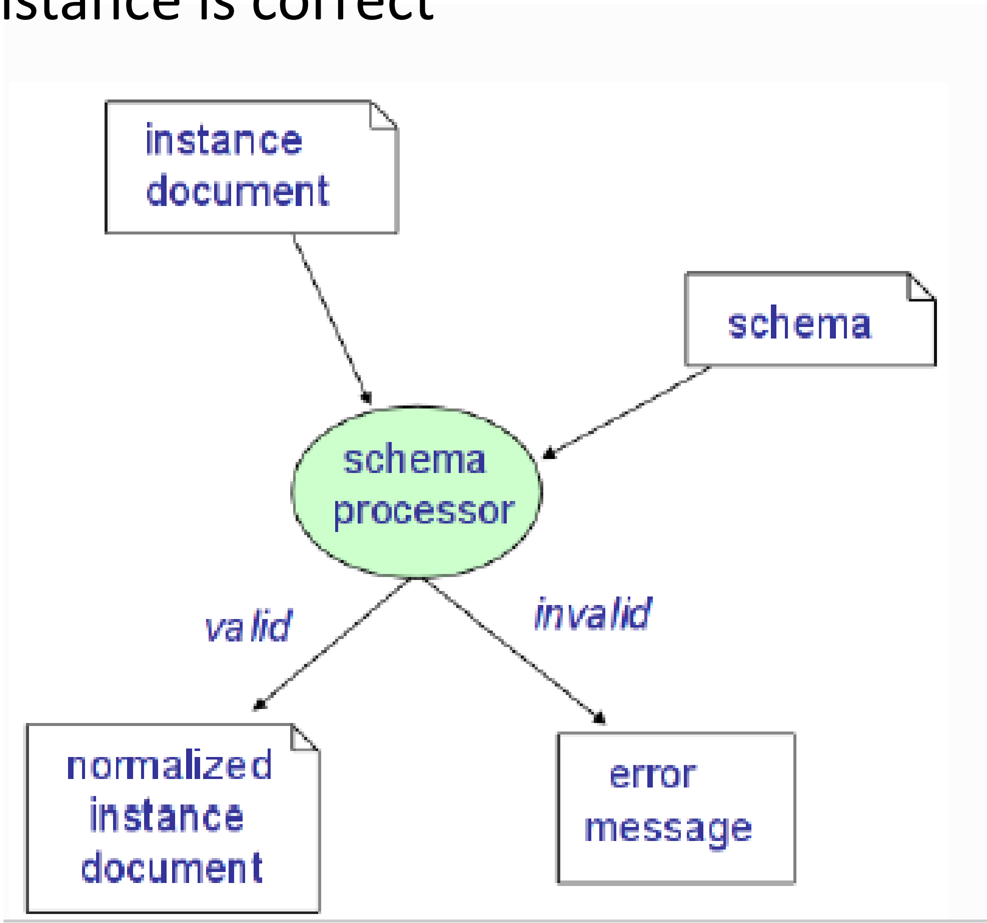
INFO-H-509
Exercises 2

XML Schema

XML Schema

Idea: check if an instance is correct



XML Schema

`Everything is in a root element called schema:`

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
    - simple type definitions
    - complex type definitions
    - element declaration
    - attribute declaration
</schema>
```

XML Schema

Element declaration assigns a element to a simple or complex type:

```
<element name=“student” type=“mycomplextype”/>
<element name=“course” type=“mysimpletype1”/>
```

Attribute declaration assigns a attribute to a simple type:

```
<attribute name=“age” type=“mysimpletype2”/>
```

Once declared, these can also be referenced:

```
<element ref=“student”/>
<attribute ref=“age”/>
```

XML Schema – Simple Type

There exist built in simple types : string, boolean, integer, float, …

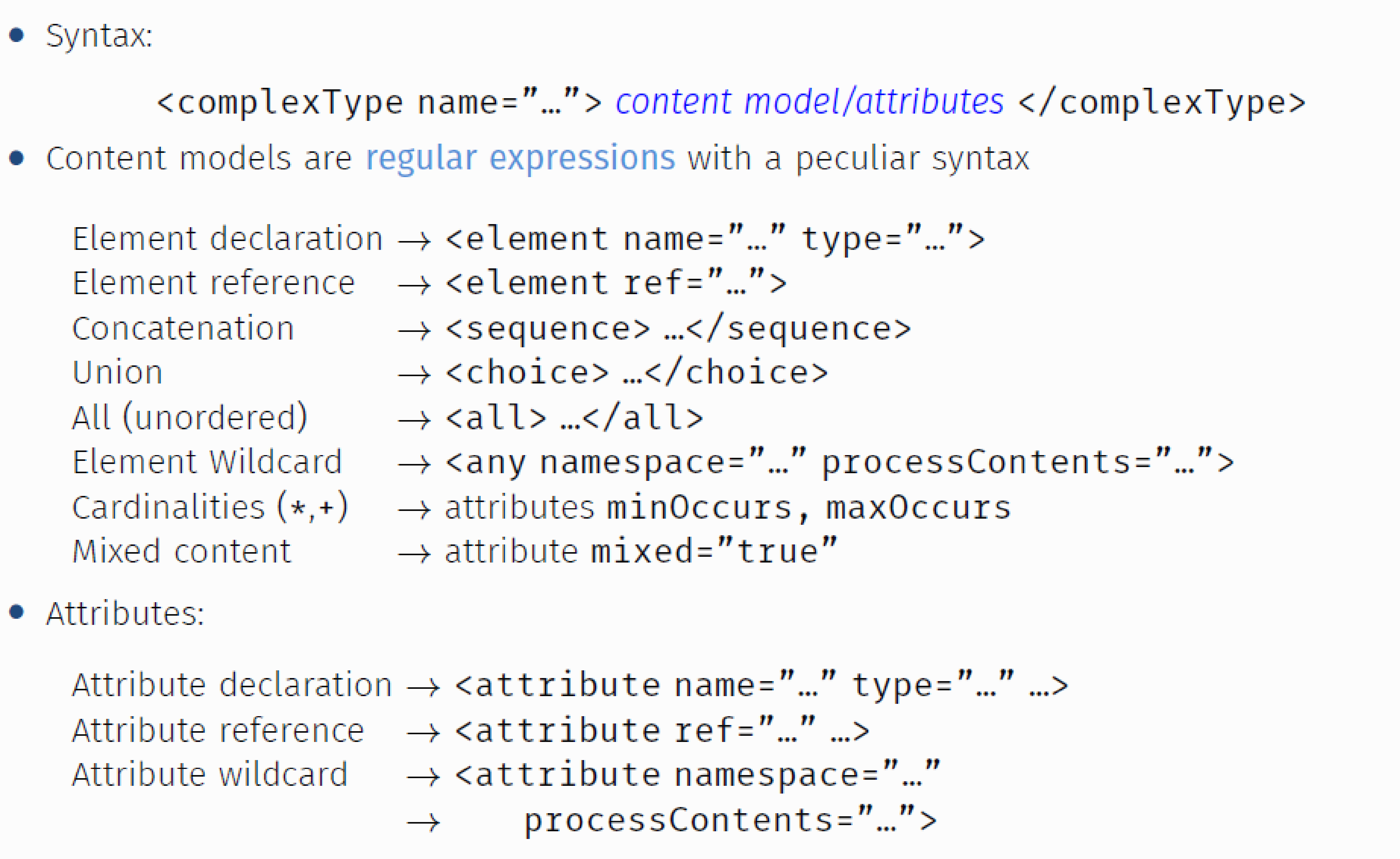
```
<element name=“course” type=“string”/>
```

We can create new simple types by performing restrictions:

```
<attribute name=“age” type=“mysimpletype”/>

<simpleType name=“mysimpletype”>
  <restriction base=“integer”>
      <minInclusive value=“18”>
  </restriction>
</simpleType>
```

XML Schema – Complex Type



- Syntax:

  `<complexType name="…">` *content model/attributes* `</complexType>`

- Content models are regular expressions with a peculiar syntax

  Element declaration → `<element name="…" type="…">`
  Element reference → `<element ref="…">`
  Concatenation → `<sequence> …</sequence>`
  Union → `<choice> …</choice>`
  All (unordered) → `<all> …</all>`
  Element Wildcard → `<any namespace="…" processContents="…">`
  Cardinalities (*,+) → attributes `minOccurs, maxOccurs`
  Mixed content → attribute `mixed="true"`

- Attributes:

  Attribute declaration → `<attribute name="…" type="…" …>`
  Attribute reference → `<attribute ref="…" …>`
  Attribute wildcard → `<attribute namespace="…"`
  → `processContents="…">`

XML Schema – Complex Type

Cardinality

```
<element name=“a”
    minOccurs=“4”
    maxOccurs=“6”
 />
```

- Element a must appear between 4 and 6 times.

```
<sequence
    minOccurs=“0”
    maxOccurs=“unbounded”>
   ...
</sequence>
```

- The whole sequence is optional. The number of occurences is however not restricted.

XML Schema – Complex Type

Sequence

```
<sequence>
  <element name=“a” />
  <element name=“b” />
  <element name=“c” />
</sequence>
```

- a, b, and c in this precise order

XML Schema – Complex Type

Choice

```
<choice>
   <element name=“a” />
   <element name=“b” />
   <element name=“c” />
</choice>
```

- One of a, b, or c

XML Schema – Complex Type

All

```
<all>
  <element name=“a” />
  <element name=“b” />
  <element name=“c” />
</all>
```

- a, b, and c, in any order

maxOccurs must be ≤ 1

XML Schema – Complex Type

We can extend complex types:

```
<complexType name=“a”>
  <sequence>
   <element ref=“s”/>
  </sequence>
</complexType>
```

```
<complexType name=“b”>
  <complexContent>
   <extension base=“a”>
     <sequence> … </sequence>
   </extension>
  </complexContent>
</complexType>
```

DTD

Binding XML to DTD

<!DOCTYPE *{root-element}*
SYSTEM ‘*{uri}*’ [
{definitions}
]>

Element

<!ELEMENT *{name}* *{content-model}*>

Content models:

(#PCDATA | *{e1}* | *{e2}* | …)	**Mixed**
EMPTY	
ANY	
(*{e1}*, *{e2}*, *{e3}*, …)	**Sequence**
(*{e1}* | *{e2}* | *{e3}* | …)	**Choice**

Cardinality

?	0-1
*	0-inf
+	1-inf

Elements of sequence and choice can in turn be sequences or choices, with cardinality specifiers.

Mixed can be reduced to (#PCDATA) to only accept text.

Attribute list

<!ATTLIST {element}
{att1} {type1} {opt1}
{att2} {type2} {opt2} {def2}
…
{attn} {typen} {optn}
>

Type

CDATA	Any text
({v1}|{v2}|{v3})	List of values
NMTOKEN, NMTOKENS	(List of) XML names
ID	Unique identifier
IDREF	Reference to an ID
ENTITY, ENTITIES	

Options

#REQUIRED	
#IMPLIED	Optional
#FIXED	Cannot be changed

Exercices

- Validation tools

```
java -jar DTDValidation.jar <xmldoc>
java -jar XSDValidation.jar <schema> <xmldoc>
```

Deterministic Regular Expressions

1. For a regular expression *a*, define a' to be the regular expression obtained by replacing the i-th occurrence of symbol *s* by s_i
   - $a = (a \mid b)^+ cba^* (a \mid c)$
   - $a' = (a_1 \mid b_1)^+ c_1 b_2 a_2^* (a_3 \mid c_2)$
2. The regular expression is deterministic if there are no two strings wb_iv and wb_jz ($i \neq j$) in the regular language
   - Consider $a_1c_1b_2a_2a_3$ and $a_1c_1b_2a_3c_2$