

# Teoretické a pokročilé aspekty XML technologií

- (Ne)standardní jazyky pro popis schématu XML dat
- Metody odvozování XML schématu

Irena Mlýnková, Martin Nečaský,  
Jaroslav Pokorný



# Úvod

- XML data – obecně elementy + atributy + data + speciální prvky
  - Komentáře, CDATA, instrukce pro zpracování apod.
- Správně (semi)strukturovaný / zformovaný (well-formed) XML dokument
  - XML data jsou správně uzávorkována
- Správně strukturovaný / validní / platný (valid) XML dokument
  - XML data navíc odpovídají danému XML schématu



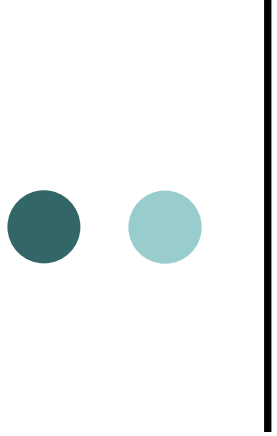
# Definice schématu XML dat

- Schéma XML dat
  - Popis přípustné struktury XML dokumentů
  - Popis elementů a atributů, které se smí v dokumentu vyskytovat a jejich vzájemných vztahů
- Nástroje pro definici struktury:
  - DTD (Document Type Definition)
  - XML Schema
  - Schematron, RELAX NG, ...
- Nástroje pro kontrolu validity XML dokumentů (tzv. XML validátory, XML procesory, ...)

# Zmatky v terminologii

- XML schéma = přípustná struktura XML dat popsaná v některém z existujících jazyků
  - DTD, XML Schema, RELAX NG, Schematron, ...
- XML Schema = jeden z jazyků pro definici přípustné struktury
  - „XML schéma v jazyce XML Schema“
- Existují i další varianty: XML-schema, XML-Schema, XML-schéma, XML schema, ...
- V angličtině: XML schema vs. XML Schema





DTD, XML Schema

# DTD – připomenutí (1)

- Součást Extensible Markup Language (XML) Recommendation – W3C
- Stále nejpoužívanější jazyk
  - Jednoduchý na pochopení, použití i zpracování
  - Postačující vyjadřovací síla (?)
    - Extrémisti: „Proč dělat schéma? Pak ho budu muset dodržovat...“
- Problém:
  - Aplikace jsou stále složitější
  - Rostou požadavky na přesnou specifikaci přípustné struktury
  - DTD přestává stačit



# DTD – připomenutí (2)

- V DTD lze definovat
  - Elementy a atributy
  - Vztahy element-podelement a element-atribut
  - Přípustný obsah elementů (prázdný, textový, elementový, smíšený, ...)
  - Pořadí ( , | ) a počet výskytů ( ? + \* ) elementů v rámci nadelementu
  - (V omezené míře) datové typy, povinnost výskytu a implicitní hodnoty atributů


# DTD – příklad

```
<?xml version="1.0" encoding="windows-1250"?>
<!ELEMENT zaměstnanci (osoba)+>
<!ELEMENT osoba (jméno, email*, vztahy?)>
  <!ATTLIST osoba id ID #REQUIRED>
  <!ATTLIST osoba poznámka CDATA #IMPLIED>
  <!ATTLIST osoba dovolená (ano|ne) "ne">
<!ELEMENT jméno ((křestní, příjmení)|(příjmení, křestní))>
<!ELEMENT křestní (#PCDATA)>
<!ELEMENT příjmení (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT vztahy EMPTY>
  <!ATTLIST vztahy nadřizený IDREF #IMPLIED>
  <!ATTLIST vztahy podřizení IDREFS #IMPLIED>
```



# XML Schema – připomenutí (1)

- Doporučení konsorcia W3C (3 části)
- Výhody:
  - Nevyžaduje speciální syntaxi
  - Silná podpora jednoduchých datových typů
  - Lze (snadno) vyjádřit přípustné počty výskytů elementů
  - Lze (snadno) vyjádřit libovolné pořadí elementů
  - Při modelování lze opakovaně využívat již definované prvky
  - Umožňuje přesněji specifikovat omezení identity
  - Lze definovat tutéž věc několika způsoby
  - Mnoho objektově-orientovaných prvků



# XML Schema – připomenutí (2)

- Nevýhody:
  - Nevyžaduje speciální syntaxi
  - Lze definovat tutéž věc několika způsoby
- Prvky jazyka XML Schema:
  - Základní – jednoduchý datový typ, složený datový typ, element, atribut, modelová skupina (skupina elementů), skupina atributů
  - Pokročilé – omezení identity, substituční skupiny, zástupci, externí schémata, notace, anotace

# XML Schema – příklad (1)

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="zaměstnanci">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="osoba" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="osoba">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="jméno"/>
        <xs:element ref="email" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="vztahy" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

...

# XML Schema – příklad (2)

```
<xs:attribute name="id" type="xs:ID" use="required"/>
<xs:attribute name="poznámka" type="xs:string"/>
<xs:attribute name="dovolená" default="ne">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="ano"/>
      <xs:enumeration value="ne"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>

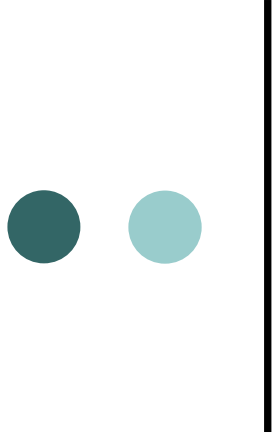
<xs:element name="křestní" type="xs:string"/>
<xs:element name="příjmení" type="xs:string"/>
<xs:element name="email" type="xs:string"/>
```

...

# XML Schema – příklad (3)

```
<xs:element name="jméno">
  <xs:complexType>
    <xs:all>
      <xs:element ref="křestní" minOccurs="0"/>
      <xs:element ref="příjmení"/>
    </xs:all>
  </xs:complexType>
</xs:element>

<xs:element name="vztahy">
  <xs:complexType>
    <xs:attribute name="vedoucí" type="xs:IDREF"/>
    <xs:attribute name="podřízení" type="xs:IDREFS"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```



RELAX NG



# RELAX NG

- Vznikl sloučením dvou jazyků:
  - TREX (Tree Regular Expressions for XML)
    - James Clark
    - <http://www.thaiopensource.com/trex/>
  - RELAX (Regular Language Description for XML)
    - Murata Makoto
    - <http://www.xml.gr.jp/relax/>
- ISO standard: ISO/IEC 19757-2:2002
- Založen na myšlence vzorů
  - RELAX NG schéma = vzor XML dokumentu
  - Poznámka: XML Schema je založeno na typech



# Obecné vlastnosti

- Je jednoduchý a snadno naučitelný
- Má dvě syntaxe: XML a kompaktní (ne-XML)
  - Vzájemně převoditelné
  - XML Schema kompaktní verzi nemá
- Podporuje jmenné prostory
- Má neomezenou podporu neuspořádaných sekvencí
  - XML Schema nemá
- Má neomezenou podporu pro smíšený obsah
  - DTD nemá
- Může být využíván se samostatnou množinou datových typů
  - např. z XML Schema



# XML vs. kompaktní syntaxe

```
<element xmlns="http://relaxng.org/ns/structure/1.0"
          name="zaměstnanec">
  <attribute name="id">
    <text/>
  </attribute>
  <element name="jméno">
    <text/>
  </element>
  <element name="příjmení">
    <text/>
  </element>
  <element name="plat">
    <text/>
  </element>
</element>
```

```
element zaměstnanec {
  attribute id { text },
  element jméno { text },
  element příjmení { text },
  element plat { text } }
```

```
<zaměstnanec id="101">
  <jméno>Irena</jméno>
  <příjmení>Mlýnková</příjmení>
  <plat>1000000</plat>
</zaměstnanec>
```

# Základní vzory

- Libovolný text:

```
<text/>
```

```
text
```

- Atribut:

```
<attribute name="poznámka">  
  <text/>  
</attribute>  
  
<attribute name="poznámka"/>
```

```
attribute poznámka { text }
```

- Element s textovým obsahem:

```
<element name="jméno">  
  <text/>  
</element>
```

```
element jméno { text }
```

# Základní vzory

- Element s prázdným obsahem:

```
<element name="hr">  
  <empty/>  
</element>
```

```
element hr { empty }
```

- Element s atributy (a textovým obsahem):

```
<element name="osoba">  
  <attribute name="id"/>  
  <text/>  
</element>  
<element name="osoba">  
  <text/>  
  <attribute name="id"/>  
</element>
```

```
element osoba {  
  attribute id { text },  
  text  
}  
element osoba {  
  text,  
  attribute id { text } }
```

Nezáleží na pořadí

# Základní vzory

- Element s podelementy (a atributem):

```
<element name="osoba">
  <element name="jméno">
    <text/>
  </element>
  <element name="příjmení">
    <text/>
  </element>
  <element name="plat">
    <text/>
  </element>
  <attribute name="id"/>
</element>
```

```
element osoba {
  element jméno { text },
  element příjmení { text },
  element plat { text },
  attribute id { text } }
```

# Základní vzory

- Nepovinný vzor:

```
<element name="osoba">
  <element name="jméno">
    <text/>
  </element>
  <optional>
    <element name="fax">
      <text/>
    </element>
  </optional>
  <optional>
    <attribute name="poznámka"/>
  </optional>
</element>
```

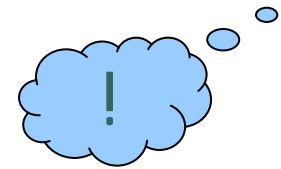
```
element osoba {
  element jméno { text },
  element fax { text }?,
  attribute poznámka { text }? }
```

...

# Základní vzory

```
<element name="osoba">
  <element name="jméno">
    <text/>
  </element>
  <element name="příjmení">
    <text/>
  </element>
  <optional>
    <element name="fax">
      <text/>
    </element>
    <attribute name="poznámka"/>
  </optional>
</element>
```

```
element osoba {
  element jméno { text },
  element příjmení { text },
  ( element fax { text },
    attribute poznámka { text } )? }
```



V DTD ani XML Schema  
1.0 nelze vyjádřit

- 1.1: assert, report

# Základní vzory

- Opakování vzorů:

```
<element name="osoba">
  <element name="jméno">
    <text/>
  </element>
  <oneOrMore>
    <element name="telefon">
      <text/>
    </element>
  </oneOrMore>
  <attribute name="id"/>
</element>
```

```
element osoba {
  element jméno { text },
  element telefon { text }+,
  attribute id { text } }
```

zeroOrMore  $\Leftrightarrow$  \*

Přesné určení počtu výskytů  
stejně jako v DTD

- Netriviálně

# Pokročilé vzory

- Uspořádaná posloupnost vzorů:

```
<element name="osoba">  
  <element name="jméno">  
    <text/>  
  </element>  
  <element name="příjmení">  
    <text/>  
  </element>  
  <element name="email">  
    <text/>  
  </element>  
</element>
```

```
<element name="osoba">  
  <group>  
    <element name="jméno">  
      <text/>  
    </element>  
    <element name="příjmení">  
      <text/>  
    </element>  
    <element name="email">  
      <text/>  
    </element>  
  </group>  
</element>
```



# Pokročilé vzory

- Výběr vzoru:

```
<element name="osoba">
  <element name="jméno">
    <text/>
  </element>
  <choice>
    <element name="email">
      <text/>
    </element>
    <element name="telefon">
      <text/>
    </element>
  </choice>
</element>
```

```
element osoba {
  element jméno { text },
  ( element email { text } |
    element telefon { text } ) }
```

# Pokročilé vzory

- Neuspořádaná posloupnost vzorů:

```
<element name="osoba">
  <element name="jméno">
    <text/>
  </element>
  <interleave>
    <element name="email">
      <text/>
    </element>
    <element name="telefon">
      <text/>
    </element>
  </interleave>
</element>
```

```
element osoba {
  element jméno { text },
  ( element email { text } &
    element telefon { text } ) }
```

Oproti XML Schema žádná  
omezení na obsah

# Pokročilé vzory

- Smíšený obsah:

```
<element name="odstavec">
  <interleave>
    <zeroOrMore>
      <element name="tučné">
        <text/>
      </element>
    </zeroOrMore>
    <zeroOrMore>
      <element name="italika">
        <text/>
      </element>
    </zeroOrMore>
    <text/>
  </interleave>
</element>
```

```
element odstavec {
  element pojem { tučné }* &
  element odkaz { italika }* &
  text }
```

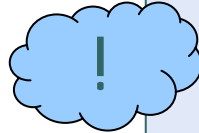
<text/> odpovídá libovolnému  
počtu textových uzlů

- Nepotřebuje + nebo \*

...

# Pokročilé vzory

```
<element name="odstavec">
  <mixed>
    <zeroOrMore>
      <element name="tučné">
        <text/>
      </element>
    </zeroOrMore>
    <zeroOrMore>
      <element name="italika">
        <text/>
      </element>
    </zeroOrMore>
  </mixed>
</element>
```



```
<element name="odstavec">
  <mixed>
    <group>
      <zeroOrMore>
        <element name="tučné">
          <text/>
        </element>
      </zeroOrMore>
      <zeroOrMore>
        <element name="italika">
          <text/>
        </element>
      </zeroOrMore>
    </group>
  </mixed>
</element>
```

group  $\Rightarrow$  interleave:

```
element odstavec {
  mixed {
    element tučné { text }* &
    element italika { text }* } }
```

# Datové typy

- Zabudované typy: string a token
- Typicky se využívají datové typy XML Schema

```
<element name="cena">  
  <data type="decimal"  
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>  
</element>
```

```
<element name="souřadnice"  
  datatypeLibrary="...">  
  <element name="x">  
    <data type="double"/>  
  </element>  
  <element name="y">  
    <data type="double"/>  
  </element>  
</element>
```

```
datatypes xs = "..."  
element cena { xs:decimal }
```

```
element cena { xsd:decimal }
```

xsd = default pro XML  
Schema

# Datové typy

- Parametry datových typů:

```
<element name="plat">  
  <data type="decimal">  
    <param name="minInclusive">100000</param>  
    <param name="maxExclusive">1000000</param>  
  </data>  
</element>
```

```
element plat {  
  xsd:decimal {  
    minInclusive = "100000"  
    maxExclusive = "1000000" } },
```

- Výčtové typy:

```
<attribute name="porty">  
  <choice>  
    <value>1001</value>  
    <value>1002</value>  
    <value>1003</value>  
  </choice>  
</attribute>
```

```
attribute porty {  
  "1001" | "1002" | "1003" }
```

# Datové typy

- Negativní výčet:

```
<element name="barva">  
  <data type="string">  
    <except>  
      <choice>  
        <value>černá</value>  
        <value>bílá</value>  
      </choice>  
    </except>  
  </data>  
</element>
```

```
element barva {  
  string – ( "černá" | "bílá" ) }
```

# Datové typy

- Vícehodnotový typ:

```
<element name="dopravníProstředek">
  <list>
    <oneOrMore>
      <data type="token"/>
    </oneOrMore>
  </list>
</element>
```

```
element dopravníProstředek {
  list { token+ }
```

```
<element name="dopravníProstředek">
  <list>
    <oneOrMore>
      <choice>
        <value>rikša</value>
        <value>velbloud</value>
        <value>slon</value>
      </choice>
    </oneOrMore>
  </list>
</element>
```

```
element dopravníProstředek {
  list { ("rikša" | "velbloud" | "slon" )+ } }
```

```
<dopravníProstředek>rikša velbloud</dopravníProstředek>
```



# Datové typy

```
<attribute name="rozměry">
  <list>
    <data type="decimal"/>
    <data type="decimal"/>
    <data type="decimal"/>
    <choice>
      <value>cm</value>
      <value>mm</value>
    </choice>
  </list>
</attribute>
```

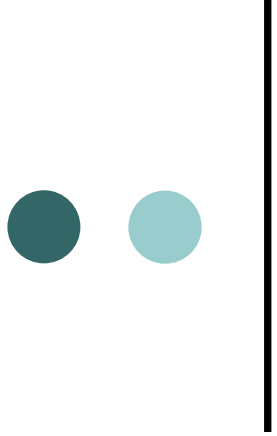
```
attribute rozměry {
  list { xsd:decimal,
         xsd:decimal,
         xsd:decimal,
         ("cm" | "mm" ) } }
```

```
rozměry="40 38.5 90 cm"
```



# Shrnutí

- Další prvky schématu:
  - Pojmenovávání částí schématu + opakované použití
  - Využití jmenných prostorů
  - Dokumentace a komentáře
- Další zdroje informací:
  - Jiří Kosek – XML schémata:  
<http://www.kosek.cz/xml/schema/rng.html>
  - RELAX NG Tutorial:  
<http://www.relaxng.org/tutorial-20011203.html>
  - RELAX NG Compact Syntax Tutorial:  
<http://www.relaxng.org/compact-tutorial-20030326.html>
  - <http://www.relaxng.org/>



# Schematron

# Schematron

- ISO standard: ISO/IEC 19757-3:2006
- Založen na myšlence vzorů
  - Podobně jako v XSLT
  - Nedefinuje gramatiku
    - Na rozdíl od DTD, XML Schema, RELAX NG
- Definuje sadu pravidel, která musí validní XML dokumenty splňovat
  - Vyjádřena pomocí XPath
  - Pro validaci lze použít XSLT procesor



# Struktura schématu

Kořenový element `<schema`

`xmlns="http://purl.oclc.org/dsdl/schematron">` obsahuje:

- `<title>` – název schématu (nepovinný)
- `<ns prefix="..." uri="..." />` – definice prefixů jmenných prostorů (libovolné množství)
- `<pattern>` – (alespoň jeden) vzor obsahující:
  - `<rule context="...">` – (alespoň jedno) pravidlo aplikované v kontextu context obsahující podelementy:
    - `<assert test="...">` – pokud XPath výraz v test není splněn, je výstupem validace obsah assert
    - `<report test="...">` – pokud XPath výraz v test je splněn, je výstupem validace obsah report

# Příklad

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.ascc.net/xml/schematron">
  <pattern name="Seznam zaměstnanců je neprázdný">
    <rule context="zaměstnanci">
      <assert test="zaměstnanec">V seznamu musí být alespoň jeden
        zaměstnanec</assert>
      <report test="sum(zaměstnanec/plat) > 500000">Součet platů
        nemůže být větší než 500.000</report>
    </rule>
  </pattern>
  <pattern name="Podmínky pro zaměstnance">
    <rule context="zaměstnanec">
      <assert test="jméno">Zaměstnanec musí mít jméno.</assert>
      <assert test="příjmení">Zaměstnanec musí mít příjmení.</assert>
      <assert test="email">Zaměstnanec musí mít e-mail.</assert>
      <assert test="@id">Zaměstnanec musí mít id.</assert>
    </rule>
  </pattern>
  ...
</schema>
```

# Příklad

```
...  
<report test="jméno[2]|příjmení[2]">Zaměstnanec nemůže mít  
více než jedno jméno.</report>  
</rule>  
</pattern>  
<pattern name="Duplicita osobních čísel">  
  <rule context="zamestnanec">  
    <report test="count(..//zaměstnanec[@id = current()/@id]) > 1">  
      Duplicitní osobní číslo <value-of select="@id"/> u elementu  
      <name/>.</report>  
    </rule>  
  </pattern>  
</schema>
```

- <name> – název aktuálního elementu
- <value-of select="..."> – výsledek XPath výrazu v select



# Validace

- Varianta A: Použití specifického SW/knihovny pro Schematron
- Varianta B: Použití libovolného XSLT procesoru
  - Existuje XSLT skript, který ze Schematronového schématu vygeneruje XSLT skript
  - XSLT skriptem transformujeme validovaný XML dokument
  - Výstupem je seznam chyb z assert a report elementů



# Kombinace s XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:sch="http://www.ascc.net/xml/schematron">
  <xs:element name="zaměstnanci">
    <xs:annotation>
      <xs:appinfo>
        <sch:pattern name="Máme dost na vyplacení platů">
          <sch:rule context="zaměstnanci">
            <sch:report test="sum(zaměstnanec/plat) > 50000">Součet platů
              nemůže být větší než 50.000.</sch:report>
          </sch:rule>
        </sch:pattern>
      </xs:appinfo>
    </xs:annotation>
    <xs:complexType>
      <!-- ... definice obsahu elementu v XSD ... -->
    </xs:complexType>
  </xs:element>
</xs:schema>
```



# Kombinace s XML Schema

- Validace:
  - Varianta A: Speciální validátor
  - Varianta B: Pomocí XSLT vyextrahujeme Schematronové schéma a to validujeme

# Kombinace s RELAX NG

```
<?xml version="1.0" encoding="utf-8"?>
<element xmlns="http://relaxng.org/ns/structure/1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
  xmlns:sch="http://www.ascc.net/xml/schematron"
  name="zamestnanci">
  <sch:pattern name="Máme dost na vyplacení platů">
    <sch:rule context="zamestnanci">
      <sch:report test="sum(zamestnanec/plat) > 50000">Součet platů
        nemůže být větší než 50.000.</sch:report>
    </sch:rule>
  </sch:pattern>
  <oneOrMore>
    <!-- ... definice obsahu elementu v XSD ... -->
  </oneOrMore>
</element>
```

# Kombinace s RELAX NG

- Existuje i kompaktní ne-XML varianta:

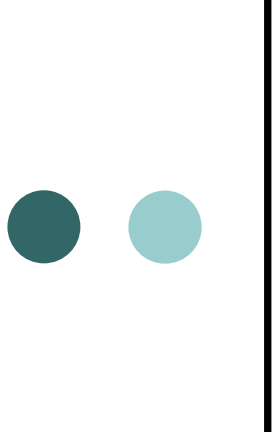
```
namespace sch = "http://www.ascc.net/xml/schematron"  
[ sch:pattern [ name = "Máme dost na vyplacení platů"  
  sch:rule [ context = "zamestnanci"  
    sch:report [ test = "sum(zamestnanec/plat) > 50000"  
      "Součet platů nemůže být větší než 50.000." ] ] ] ]
```

- Validace:
  - Varianta A: Speciální validátor
  - Varianta B: Pomocí XSLT vyextrahujeme Schematronové schéma



# Shrnutí

- Zcela jiný přístup ke specifikaci schématu
  - Nedefinuje gramatiku
- Využití síly jazyka XPath + kombinace s jinými jazyky
- Další zdroje informací:
  - Jiří Kosek – XML schémata:  
<http://www.kosek.cz/xml/schema/sch.html>
  - Oficiální stránka:  
<http://www.schematron.com/>



# Odvozování XML schématu

# Proč odvozovat XML schéma?

- XML schéma se používá pro optimalizaci XML zpracování
- Analýzy reálných XML dat:
  - 52% náhodně stažených XML dokumentů a 7.4% známých XML kolekcí nemá žádné XML schéma
  - Pouze 0.09% náhodně stažených XML dokumentů a 38% známých XML kolekcí využívá XML Schema
  - 85% XSD popisuje lokální stromové gramatiky
    - Stejná vyjadřovací síla jako DTD
- Ruční vytváření schémat lze použít pro jednoduché případy

⇒ Metody automatického odvozování XML schémat





# Specifikace problému

- Vstup: Množina XML dokumentů
- Výstup: XML schéma, vůči němuž jsou vstupní dokumenty validní
- Předpoklady:
  - XML dokumenty = pozitivní příklady
  - XML schéma – vhodný kompromis mezi triviálními extrémy
    - Schéma, kterému vyhovují pouze vstupní dokumenty
    - Schéma, kterému vyhovuje libovolný dokument
- Existující metody:
  - Odvozují obvykle DTD
  - Zaměřují se na regulární výrazy
    - Ignorují triviální případy: atributy, jednoduché datové typy, ...



# Trocha teorie

- XML schéma v jazyce DTD / XML Schema definuje tzv. rozšířenou bezkontextovou gramatiku.
- **Bezkontextová gramatika**  $G$  je čtveřice  $(N, T, P, S)$ , kde
  - $N$  je konečná množina neterminálů
  - $T$  je konečná množina terminálů
  - $S$  je počáteční neterminál
  - $P$  je konečná množina přepisovacích pravidel tvaru  $A \rightarrow \alpha$ , kde  $A \in N$  a  $\alpha$  je slovo nad abecedou  $N \cup T$
- **Rozšířená bezkontextová gramatika**  $G$  je čtveřice  $(N, T, P, S)$ , kde
  - ...
  - $P$  je konečná množina přepisovacích pravidel tvaru  $A \rightarrow \alpha$ , kde  $A \in N$  a  $\alpha$  je regulární výraz nad abecedou  $N \cup T$ 
    - Tj. reprezentuje více původních pravidel

# Trocha teorie

- **Regulární výraz** (RV) nad abecedou  $\Sigma$  je induktivně definován takto:
  - $\emptyset$  (prázdná množina) and  $\varepsilon$  (prázdný řetězec) jsou RV
  - $\forall a \in \Sigma$  je RV
  - Jsou-li **r** a **s** RV nad  $\Sigma$ , pak **(rs)** (konkatenace), **(r|s)** (alternace) a **(r\*)** (Kleeneho uzávěr) jsou RV
- Poznámky:
  - Jazyk DTD přidává zkrácené výrazy:
    - $(s|\varepsilon) = (s?)$
    - $(ss^*) = (s+)$
  - Konkatenace je v DTD vyjádřena operátorem ","
  - XML Schema přidává neuspořádanou sekvenci
    - **Alternace všech možných uspořádání dané sekvence**



# Trocha teorie

- Regulární výraz je rozpoznatelný **konečným automatem**  $A = (Q, \Sigma, \delta, S, F)$ , kde
  - $Q$  je konečná množina stavů
  - $\Sigma$  je konečná množina vstupních symbolů (abeceda)
  - $\delta : Q \times \Sigma^* \rightarrow Q$  je přechodová funkce
  - $S \in Q$  je počáteční symbol
  - $F \subseteq Q$  je konečná množina koncových stavů



# Klasifikace přístupů

- Obecně: Potřebujeme dané vstupy vhodně zobecnit
  - Nechceme schéma, které by odpovídalo pouze vstupním dokumentům
    - Triviální, prakticky nepoužitelné
  - Problém: Co je to vhodně?
- Heuristické metody
  - Heuristická pravidla odvozování
    - "Pokud se element vyskytuje více než 10x, může mít neomezený počet výskytů."
  - Výstup nepatří do žádné "rozumné" třídy jazyků
    - Nemůžeme nic říct o jeho vlastnostech

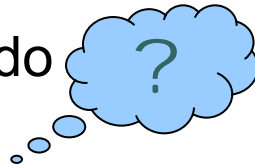
# Klasifikace přístupů

## Metody odvozující gramatiku

- Goldova věta: Třída jazyků, která obsahuje všechny konečné jazyky a alespoň jeden nekonečný, nemůže být limitně rozpoznatelná z pozitivních příkladů.
    - Metoda  $M$  rozpoznává jazyk  $L$  limitně, pokud po konečném počtu vstupních příkladů z  $L$  korektně určí  $L$  a toto určení s dalšími vstupy nezmění.
    - Třída jazyků je rozpoznatelná limitně pokud existuje metoda  $M$  taková, že libovolný jazyk  $L$  z této třídy při vhodné sekvenci vstupních příkladů limitně rozpozná.
- ⇒ Bezkontextové (ani regulární) jazyky nelze limitně rozpoznat z pozitivních příkladů
- ⇒ Metody odvozují zvolenou rozpoznatelnou podtřídu
- Problém: Která podtřída je smysluplná pro XML data?

# Obecný algoritmus

1. Pro každý výskyt elementu  $e$  a jeho podelementy  $e_1 e_2 \dots e_k$  vytvoříme odvozovací pravidlo  $e \rightarrow e_1 e_2 \dots e_k$ 
  - Tzv. počáteční gramatika
2. Pravidla rozdělíme do vhodných skupin
  - Typicky podle levé stany, popř. kontextu (tj. cesty od  $e$  do kořenového elementu)
3. Pravidla ve skupinách sloučíme, zjednodušíme a zobecníme
  - Heuristická pravidla vs. pravidla, která zaručí výstup v dané třídě jazyků
4. Výsledná pravidla zapíšeme v syntaxi zvoleného jazyka



# Kroky 1., 2.

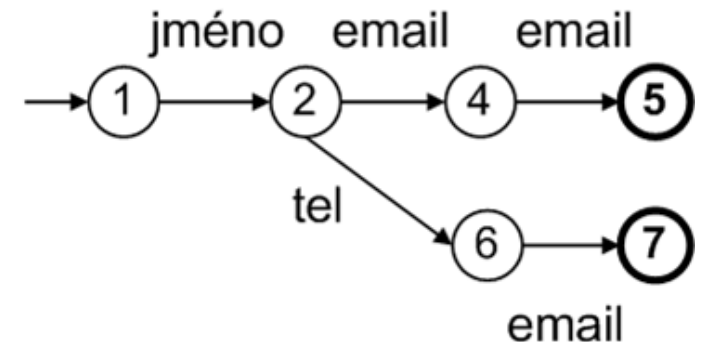
```
...  
<osoba id="123">  
  <jméno>  
    <křestní>Irena</křestní>  
    <příjmení>Mlýnkova</příjmení>  
  </jméno>  
  <email>irena.mlynkova@gmail.com</email>  
  <email>irena.mlynkova@mff.cuni.cz</email>  
</osoba>  
<osoba id="456" dovolená="ano">  
  <jméno>  
    <příjmení>Nečaský</příjmení>  
    <křestní>Martin</křestní>  
  </jméno>  
  <tel>123-456-789</tel>  
  <email>martin.necasky@mff.cuni.cz</email>  
</osoba>  
...
```

osoba → jméno email email  
osoba → jméno tel email

jméno → křestní příjmení  
jméno → příjmení křestní

křestní → PCDATA  
příjmení → PCDATA  
email → PCDATA  
tel → PCDATA

**osoba:**



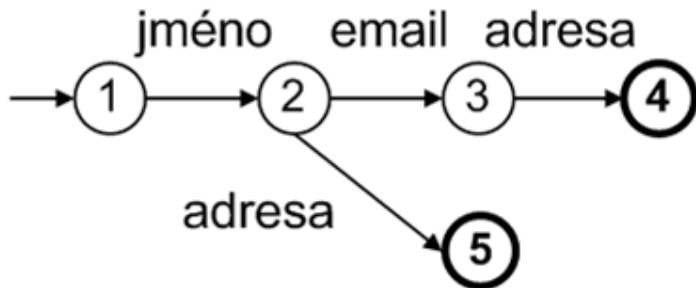
# Krok 3. Slévání stavů

- Triviální heuristiky
- Motivace: Nerodova ekvivalence stavů:
  - Dva stavy  $p$  a  $q$  jsou ekvivalentní, pokud množiny všech cest vedoucích z  $p$  a  $q$  do koncových stavů jsou ekvivalentní.
  - Obvykle se podmínka omezuje:
    - Na ekvivalenci množin cest délky  $k$  nebo kratších končících v koncových stavech
    - Na ekvivalenci množin  $s$  nejpravděpodobnějších cest délky max  $k$  (= ignorujeme speciální případy)
- $(k,h)$ -kontext:
  - Dva stavy  $p$  a  $q$  jsou ekvivalentní, pokud existují dvě identické cesty délky  $k$  vedoucí do  $p$  a  $q$ . Pak je i  $h$  předchozích stavů ekvivalentních.

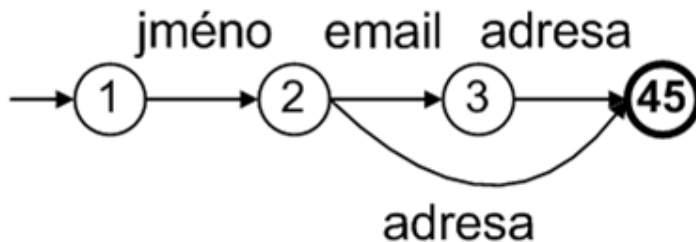
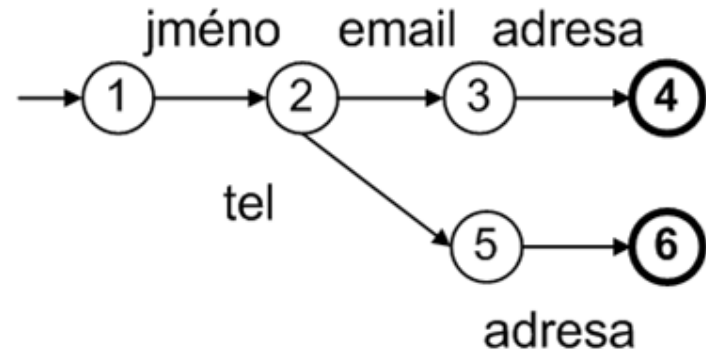


# Příklady slévání stavů

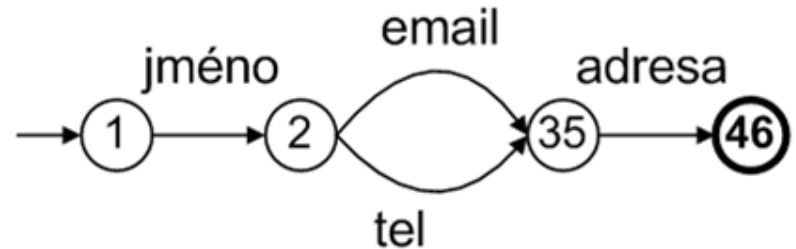
osoba → jméno email adresa  
osoba → jméno adresa



osoba → jméno email adresa  
osoba → jméno tel adresa



osoba → jméno email? adresa



osoba → jméno (email | tel) adresa



## Krok 4. Přepis do syntaxe jazyka

- DTD: přímý přepis
- XML Schema: více alternativ
  - Globální vs. lokální prvky, substituční skupiny, odvozování typů, ...
  - Existující metody obvykle neřeší



# Shrnutí

- Existující metody umí:
  - Odvozovat regulární výrazy
    - Heuristicky nebo dle vlastností zvolené třídy
  - Odvozovat triviální XSD konstrukty
    - Vybrané jednoduché datové typy, přesná omezení počtu výskytů
- Existující metody neumí:
  - Odvozovat složitější ne-DTD konstrukty
    - Pokročilé prvky XML Schema, Schematronové schéma, ...
  - Využívat interakci s uživatelem
    - Kroků jak zobecnit a zapsat schéma je hodně – který je správný?
  - Využívat další vstupní informace
    - Např. negativní příklady, původní neplatné schéma, dotazy, ...
  - Odvozovat integritní omezení
    - ID, IDREF(S), unique, key, keyref, assert, report, ...



# Další teoretické problémy

- Podobnost fragmentů XML schémat
  - Netriviální strukturální podobnost schémat
  - Podobnost XML schématu a XML dokumentu
- Vzájemný vztah schémat, tj. množin jejich instancí
  - Obecně těžký problém
- Evoluce XML schémat
  - Propagace změn ve schématu do instancí, dotazů, konceptuálních schémat...
- Verzování XML schémat
  - Udržování více verzí schémat  $\Rightarrow$  XML dat  $\Rightarrow$  překlady dotazů

**Konec**