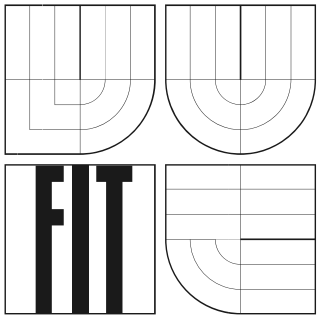
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Seminární práce do předmětu **VPD**

Gramatika pro popis syntaxe XML a její klasifikace v Chomského hierarchii

22. května 2007

Ing. Ondřej Nečas

Obsah

1 Úvod

Vzhledem k tomu, že v dnešní době se na světě využívá celá řada operačních systémů a aplikací, bylo by nepřijatelné přenášet dokumenty ve tvaru, který vyžaduje speciální software některé firmy. Je nezbytné používat otevřený formát, který není svázán s konkrétní platformou. Jako řešení této situace se přímo nabízí dokumenty formátu XML, které jsou založené na obyčejném textu a lze je tudíž číst a editovat prakticky v libovolném textovém editoru. Formát dokumentů XML je stanoven specifikací konsorcia W3C, ke které existuje volně dostupná dokumentace, což umožňuje implementaci XML v libovolné aplikaci.

Jednotlivé části textového obsahu jsou v XML dokumentech odděleny značkami, které určují jejich význam. Těmto značkám se říká *tagy*. Tagy nejsou předdefinované a pro konkrétní dokument si uživatel vždy definuje tagy vlastní. Máme-li definici značek použitých v XML dokumentu, můžeme snadno zkontrolovat, zda dokument této definici odpovídá, čímž odhalíme případné nesrovnalosti a chyby.

Historie schémat dokumentů

K vytvoření definice tagů se v minulosti hojně využívalo např. schéma dokumentu v souboru DTD. Velkými nevýhodami DTD jsou ale nemožnost kontroly datových typů a také absence podpory jmenných prostorů. Proto vznikají jiné alternativy, na jejichž základě bude možné vytvořit parser, který bude schopen validovat XML dokumenty lépe. Pro zápis DTD se používá jednoduchá syntaxe připomínající regulární výrazy. Tento způsob zápisu se ale již nikde jinde v XML nevyskytuje, a proto všechny novější jazyky pro popis schématu dokumentu používají pro jeho zápis rovnou syntaxi XML.

Za nejdůležitějšího následovníka DTD můžeme považovat jazyk XDR (XML-Data Reduced). Ten vznikl jako zjednodušená verze návrhu jazyka XML-Data, který nebyl ani tak vázán na XML, ale především umožňoval definovat charakteristiky tříd objektů. Obecně platným a přijatým standardem se však až později stal jazyk XML Schema konsorcia W3C, který se hned po DTD stal dodnes nejpoužívanějším. XML Schema se vyvinulo právě z XDR, ale ani ono není bez chyby. Opět to má na svědomí složitost a občasná nepřehlednost tohoto jazyka, což bylo některým z jeho autorů zřejmé dříve, než byl jazyk XML Schema uznán standardem.

Proto již v průběhu vývoje jazyka XML Schema vznikl poměrně odlišný jazyk Relax NG založený na zajímavé teorii alejových gramatik a automatů. Jeho systematický přístup vytváření gramatiky nového jazyka je podobný DTD, na rozdíl od XML Schema, jenž definuje datové typy, které musí dokument obsahovat.

Zcela odlišný přístup a poněkud jinou větev vývoje pak představuje jazyk Schematron, který umožňuje definovat sadu podmínek, které musí být v dokumentech splněny. Schematronové schéma je velmi silným nástrojem pro validaci, protože žádný z předchozích jazyků není např. schopen popsat taková omezení, kdy obsah jednoho elementu musí mít větší hodnotu, než obsah elementu jiného, nebo vztahy mezi daty uloženými v separátních

dokumentech. Schematron bývá tedy používán jako doplněk ke klasickým schématům vytvořeným v jazycích XML Schema nebo Relax NG, které oba umožňují vkládání výroků Schematronu přímo do schématu dokumentu.

Jak je vidět, mají dnes uživatelé k dispozici celou řadu různě vhodných nástrojů a mnohdy je těžké přesně určit, který z nich je nejvhodnější k řešení konkrétního problému. Volbu můžeme poměrně zjednodušit, když si uvědomíme, že DTD je vhodné pouze v případě, že v dokumentech nepotřebujeme podporu jmenných prostorů. Jinak budeme nejspíš volit mezi XML Schema a Relax NG. XML Schema si vybereme zejména tehdy, pokud jsme omezeni vývojovými nástroji komerčních firem. Naopak nejsme-li takto omezeni, poskytne nám Relax NG o něco větší jednoduchost a možnost kompaktního zápisu. Podívejme se nyní ale blíže na základ jazyka Relax NG, kterým jsou alejové gramatiky a automaty.

2 Teorie alejí

V komunitě okolo XML je teorie alejových automatů označována za jednoduchý, ale poměrně silný a účinný nástroj k vyjádření modelu XML schématu.

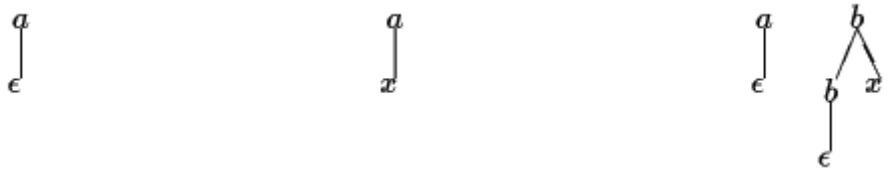
2.1 Aleje

Nejprve se podívejme na definici pojmu *aleje*. Neformálně řečeno je alej definována jako řada stromů (strom, který nemusí mít jeden vrchol). Z pohledu terminologie XML je pak alej sekvence elementů, které je možné vyjádřit pomocí znakových dat; v podstatě lze říci, že celý XML dokument je tedy alej.

Alej nad konečnou množinou symbolů **Σ** a konečnou množinou proměnných **X** je:

1. ε (prázdná alej),
2. *x*, kde *x* je proměnná z **X**,
3. *a* <*u*>, kde *a* je symbol z množiny **Σ** a *u* je alej (symbol *a* označujeme jako kořenový uzel), nebo
4. *uv*, kde *u* a *v* jsou aleje (konkatenace dvou alejí).

Na obr. 1 jsou zobrazeny tři aleje: *a* <ε> , *a* <*x* > , a *a* <ε > *b* < *b* <ε> *x* > . Přitom všechny nelistové uzly jsou zastoupeny elementy z množiny **Σ** (např. *a* a *b*), zatímco elementy z **X** (např. *x*) jsou použity pro uzly listové. Zápis *a* <ε> můžeme zjednodušit a zapsat jako *a*. Z toho plyne, že třetí příklad můžeme zapsisovat také jako *a b* <*b x* > .



Obr. 1.: Tři aleje: *a* <ε> , *a* <*x* > , a *a* <ε > *b* < *b* <ε> *x* >

Dále předpokládejme existenci XML dokumentu. Nechť **Σ** = {**doc**, **title**, **image**, **para**} a **X** = {**#PCDATA**}. Potom

doc < **title**<**#PCDATA**> **para**<**#PCDATA**> <**image/**> **para**<**#PCDATA**>>

je alej. Tato alej může být s pomocí syntaxe XML reprezentována následovně:

```
<doc>
  <title>#PCDATA</title>
  <para>#PCDATA</para>
  <image/>
  <para>#PCDATA</para>
</doc>
```

2.2 Regulární alejová gramatika

Definujme nyní *regulární alejové gramatiky* (RAG). RAG je mechanismus, který slouží pro generování alejí. Jinými slovy RAG popisuje množinu alejí.

Protože XML schéma primárně slouží k popisu množiny platných dokumentů, RAG může být považována za formální reprezentaci XML schématu.

Regulární alejová gramatika (RAG) je uspořádaná 5-tice <**Σ**, **X, N, P,** r_f >, kde:

- **Σ** je konečná množina terminálních symbolů,
- **X** je konečná množina proměnných,
- **N** je konečná množina non-terminálních symbolů,
- **P** je konečná množina *přepisovacích pravidel*, z nichž každé je právě jednoho ze dvou následujících tvarů:
  - $n \rightarrow x$, kde n je non-terminál z **N** a x je proměnná z **X**,
  - $n \rightarrow a$ <r>, kde n je non-terminál z **N**, a je symbol z **Σ** a r je regulární výraz obsahující non-terminální symboly,
- r_f je regulární výraz obsahující non-terminální symboly.

Nyní definujme *derivaci* RAG. Obecně lze říci, že se jedná o opakované nahrazení non-terminálních symbolů v dané sekvenci alejemi odpovídajícími pravým stranám přepisovacích pravidel.

Alej u lze *přímo derivovat* na alej v, když:

- pro některé přepisovací pravidlo tvaru $n \rightarrow x$, alej v získáme nahrazením non-terminálního symbolu n v aleji u prvkem z **X**, nebo
- pro některé přepisovací pravidlo tvaru $n \rightarrow a$ <r>, alej v získáme nahrazením non-terminálního symbolu n v aleji u nějakým a <w> takovým, že w je sekvence non-terminálů odpovídajících regulárnímu výrazu r.

Jazyk generovaný gramatikou **G** označujeme **L(G)** a jedná se o množinu alejí, které lze derivovat z nějaké non-terminální sekvence odpovídající regulárnímu výrazu r_f.

Nechť **G** = < *a, x*, {n_1 , n_2}, **P**, $n_1^?$> je RAG, kde:

P = {$n_1 \rightarrow$ a<n_2^+ >, $n_2 \rightarrow x$ }.

Potom

L(G) = {ε, *a*<*x* >, *a*<*xx*>, *a*<*xxx* >, ...}

Dále vytvoříme RAG, která bude odpovídat DTD. Jako příklad mějme následující DTD:

```
<!ELEMENT  doc    (title, (para|image)*)>
<!ELEMENT  title  (#PCDATA)>
<!ELEMENT  para   (#PCDATA)>
<!ELEMENT  image  EMPTY>
```

Toto DTD může být vyjádřeno RAG **G** = <**Σ**, **X, N, P**, n_d >, kde

- **Σ** = {**doc**, **title**, **image**, **para**},
- **X** = { **#PCDATA** },
- **N** = {n_d , n_t , n_p , n_i , $n_\#$ },
- **P** = {$n_d \rightarrow$ **doc** <n_t (n_p | n_i)* >,
  $n_t \rightarrow$ **title** <$n_\#$ >,
  $n_p \rightarrow$ **para** <$n_\#$ >,
  $n_i \rightarrow$ **image** <ε>,
  $n_\# \rightarrow$ **#PCDATA** }

V případě opačného postupu převodu RAG na DTD předpokládejme následující RAG **G** = <**Σ**, **X, N, P**, n_1>, kde:

- **Σ** = {**segment** , **para**} ,
- **X** = { **#PCDATA** } ,
- **N** = { n_1 , n_2 , n_p , $n_\#$ } ,
- **P** = {$n_1 \rightarrow$ **segment** <n_p^* n_2^* > ,
  $n_2 \rightarrow$ **segment** <n_p^* > ,
  $n_p \rightarrow$ **para** <$n_\#$ > ,
  $n_\# \rightarrow$ **#PCDATA** }

Obě pravidla pro non-terminál n_1 i pro n_2 mají **segment** na pravé straně. Nicméně první z nich má tvar pravé strany n_p^* n_2^*, kdežto druhé n_p^*. Takto lze vyjádřit, že segmenty nejvyšší úrovně mohou obsahovat vnořené segmenty, ale tyto vnořené segmenty již další vnořené podsegmenty obsahovat nemohou.

Syntaxí DTD nelze přesně vyjádřit tuto RAG, protože všechny výskyty segmentů musí být přepisovány na stejný tvar obsahu. Nejmenší DTD, které může odpovídat této RAG, je proto:

```
<!ELEMENT  segment (para*, segment*)>
<!ELEMENT  para    (#PCDATA)>
```

Toto DTD umožňuje nekonečné vnořování segmentů. Protože syntaxe DTD nepovoluje dva různé tvary obsahu segmentu, používá toto DTD pouze jeden.

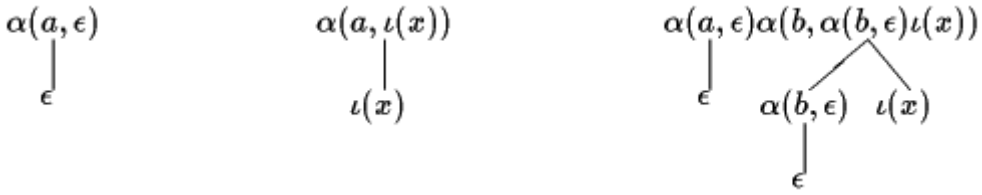
2.3 Alejový automat

V této části si ukážeme, co je to deterministický alejový automat a nedeterministický alejový automat.

Deterministický alejový automat (DAA) je uspořádaná 6-tice < **Σ**, **X, Q**, **α**, **ι**, **F**>, kde:

- **Σ** je konečná množina terminálních symbolů,
- **X** je konečná množina proměnných,
- **Q** je konečná množina stavů,
- **α** je funkce zobrazení $\mathbf{\Sigma} \times \mathbf{Q}^* \rightarrow \mathbf{Q}$ taková, že pro každé q z množiny **Q** a x z množiny **Σ** je $\{q_1\ q_2\ ...\ q_k \mid k >= 0, \mathbf{\alpha}(x, q_1\ q_2\ ...\ q_k) = q\}$ regulární množina,
- **ι** je funkce zobrazení $\mathbf{X} \rightarrow \mathbf{Q}$,
- **F** je regulární množina nad **Q**.

Obr. 2 zobrazuje posloupnost provádění DAA pro alej z obr. 1.



Obr. 2.: Výpočet deterministického alejového automatu

Dále si ukážeme, jak DAA přijímá první příklad jazyka v části popisu RAG. Nechť

M = <a , x , { q_0, q_1, q_2 }, **α**, **ι**, $q_1^?$ }, kde:

- **α**(a, u) =
  - q_1 (pokud u patří do jazyka $\mathbf{L}(q_2^+)$),
  - q_0 (jinak),
- **ι**(x) = q_2 .

Potom

L(G) ={ε , a<x> , a<xx> , a<xxx> , ... }

Nedeterministický alejový automat (NDAA) je uspořádaná 6-tice < **Σ**, **X, Q**, **α**, **ι**, **F**>, kde:

- **Q**, **Σ**, a **F** mají shodný význam jako u DAA,
- **α** je relace (nazývaná *přechodová funkce*) $\mathbf{\Sigma} \times \mathbf{Q}^* \rightarrow \mathbf{Q}$ (nebo funkce $\mathbf{\Sigma} \times \mathbf{Q}^* \rightarrow 2^{\mathbf{Q}}$) taková, že pro každé *q* z množiny **Q** a *x* z množiny **Σ** je $\{q_0q_1...q_k \mid k >= 0, \mathbf{\alpha}(x, q_0q_1...q_k, q)\}$ regulární jazyk řetězců,
- **ι** je relace $\mathbf{X} \rightarrow \mathbf{Q}$ (nebo funkce $\mathbf{X} \rightarrow 2^{\mathbf{Q}}$).

Podle definice je každý DAA zároveň také NDAA. Tudíž také předchozí příklad DAA je zároveň příkladem NDAA. Jen je potřeba dávat pozor na záměnu stavu a jednoprvkové množiny obsahující stav.

Poslední příklad RAG v části popisu RAG může být snadno vyjádřen pomocí NDAA <**Σ**, **X**, **Q**, **α**, **ι**, **F** >, kde

- **Σ** = {**segment**, **para** },
- **X** = { **#PCDATA** },
- **Q** = < q_1, q_2, q_p, $q_{\#}$ >,
- **α**(*a*, *u*) obsahuje q_1, když *a* = **segment** a zároveň *u* patří do jazyka **L** ($q_p^*\, q_2^*$)),
- **α**(*a*, *u*) obsahuje q_2, když *a* = **segment** a zároveň *u* patří do jazyka **L** (q_p^*),
- **α**(*a*, *u*) obsahuje q_p, když *a* = **para** a *u* patří do jazyka **L**($q_{\#}$)),
- **ι**(*x*) = $q_{\#}$, když *x* = **#PCDATA**,
- **F** = q_1 .

2.4 Vlastnosti regulárních alejových jazyků

Ekvivalence

Následující podmínky jsou ekvivalentní.

1. **L** je generován RAG,
2. **L** je přijímán DAA,
3. **L** je přijímán NDAA.

Důkaz, že z tvrzení (3) vyplývá tvrzení (2), nám dává teorie množin a podmnožin.

Booleovská uzavřenost

Předpokládejme, že množina $\mathbf{L}_1$, resp. $\mathbf{L}_2$ je přijímána DAA automatem $\mathbf{M}_1$, resp. NDAA automatem $\mathbf{M}_2$. Pak lze jednoduše vytvořit (N)DAA automaty takové, že přijímají následující jazyky:

1. průnik jazyků $\mathbf{L}_1$ a $\mathbf{L}_2$,
2. sjednocení jazyků $\mathbf{L}_1$ a $\mathbf{L}_2$,
3. doplněk jazyka $\mathbf{L}_1$ (množina všech alejí, které nejsou součástí $\mathbf{L}_1$)

Syntaktické stromy bezkontextových gramatik

Množině syntaktických stromů bezkontextové gramatiky říkáme *lokální* stromový jazyk. O vztazích mezi lokálními stromovými jazyky a regulárními jazyky je mnoho známo. Připomeňme dvě skutečnosti, které se přímo týkají XML:

1. lokální stromový jazyk je také regulární (alejový) jazyk (jinými slovy, pro libovolnou rozšířenou bezkontextovou gramatiku lze zkonstruovat DAA),
2. pro libovolný regulární alejový jazyk obsahující pouze stromy existuje jednoznačný minimální lokální stromový jazyk, který zahrnuje také tento regulární alejový jazyk.

Z tvrzení (1) vyplývá, že RAG jsou silnější než DTD, kdežto tvrzení (2) říká, že pro libovolnou danou RAG můžeme vytvořit přijatelné DTD.

3 Shrnutí

Klasifikace alejových gramatik

Protože každou gramatiku lze klasifikovat podle Chomského hierarchie, pokusme se do ní zařadit také gramatiky alejové.

Zanořená struktura značek v dokumentech XML, stejně jako popis syntaxe mnoha dalších programovacích jazyků, ukazují na to, že alejové gramatiky jsou v zásadě bezkontextové. Koneckonců konsorcium W3C pro svůj jazyk XML Schema použilo také bezkontextovou gramatiku v rozšířené Backus-Naurově formě (EBNF).

Jak je ale vidět v definici alejové gramatiky, její množina přepisovacích pravidel **P** obsahuje mimo jiné pravidla tvaru $n \rightarrow a \langle r \rangle$, kde n je non-terminál z **N**, a je symbol z **Σ** a r je regulární výraz obsahující non-terminální symboly. Uveďme si nyní znovu příklad alejové gramatiky:

$\mathbf{G} = \langle \mathbf{\Sigma}, \mathbf{X}, \mathbf{N}, \mathbf{P}, n_d \rangle$

- $\mathbf{\Sigma} = \{\mathbf{doc}, \mathbf{title}, \mathbf{image}, \mathbf{para}\}$
- $\mathbf{X} = \{ \mathbf{\#PCDATA} \}$
- $\mathbf{N} = \{n_d, n_t, n_p, n_i, n_\# \}$
- $\mathbf{P} = \{ n_d \rightarrow \mathbf{doc} \langle n_t (n_p \mid n_i)^* \rangle,$
  $n_t \rightarrow \mathbf{title} \langle n_\# \rangle,$
  $n_p \rightarrow \mathbf{para} \langle n_\# \rangle,$
  $n_i \rightarrow \mathbf{image} \langle \varepsilon \rangle,$
  $n_\# \rightarrow \mathbf{\#PCDATA} \}$

Inkriminovaným pravidlem je zde $n_d \rightarrow$ **doc** $< n_t\ (n_p \mid n_i)^* >$. Regulární výraz $n_t\ (n_p \mid n_i)^*$, který toto pravidlo obsahuje, můžeme převést na samostatnou gramatiku:

$\mathbf{G_R} = < \{ S, A \}, \{ n_t, n_p, n_i \}, \mathbf{P_R}, S >$

$\mathbf{P_R} = \{ S \rightarrow n_t A,$
$A \rightarrow n_p A,$
$A \rightarrow n_i A,$
$A \rightarrow \varepsilon \}$

Takto vzniklá gramatika $\mathbf{G_R}$ spadá podle Chomského hierarchie mezi gramatiky regulární. Pokusme se nyní začlenit gramatiku $\mathbf{G_R}$ přímo do alejové gramatiky **G**:

$\mathbf{G} = <\mathbf{\Sigma}, \mathbf{X}, \mathbf{N}, \mathbf{P}, n_d >$

- $\mathbf{\Sigma} = \{$**doc**, **title**, **image**, **para**$\}$
- $\mathbf{X} = \{$ **#PCDATA** $\}$
- $\mathbf{N} = \{n_d, n_t, n_p, n_i, n_\#, S, A \}$
- $\mathbf{P} = \{ n_d \rightarrow$ **doc** $< S >,$
  $S \rightarrow n_t A,$
  $A \rightarrow n_p A,$
  $A \rightarrow n_i A,$
  $A \rightarrow \varepsilon,$
  $n_t \rightarrow$ **title** $< n_\# >,$
  $n_p \rightarrow$ **para** $< n_\# >,$
  $n_i \rightarrow$ **image** $< \varepsilon >,$
  $n_\# \rightarrow$ **#PCDATA** $\}$

Zajímavé je, že regulární výrazy v alejové gramatice obsahují výhradně non-terminální symboly, které ale v dílčí regulární gramatice $\mathbf{G_R}$ představují symboly abecedy. Odstranění regulárního výrazu z alejové gramatiky **G** jsme pak dosáhli pouhým přidáním všech non-terminálních symbolů a přepisovacích pravidel z $\mathbf{G_R}$ do **G**. Takto vzniklou gramatiku lze ještě v tomto případě snadno zjednodušit o jeden non-terminální symbol S a jedno pravidlo $S \rightarrow n_t A$:

$\mathbf{G} = <\mathbf{\Sigma}, \mathbf{X}, \mathbf{N}, \mathbf{P}, n_d >$

- $\mathbf{\Sigma} = \{$**doc**, **title**, **image**, **para**$\}$
- $\mathbf{X} = \{$ **#PCDATA** $\}$
- $\mathbf{N} = \{n_d, n_t, n_p, n_i, n_\#, A \}$
- $\mathbf{P} = \{ n_d \rightarrow$ **doc** $< n_t A >,$
  $A \rightarrow n_p A,$
  $A \rightarrow n_i A,$
  $A \rightarrow \varepsilon,$
  $n_t \rightarrow$ **title** $< n_\# >,$
  $n_p \rightarrow$ **para** $< n_\# >,$
  $n_i \rightarrow$ **image** $< \varepsilon >,$
  $n_\# \rightarrow$ **#PCDATA** $\}$

Tím jsme do původní bezkontextové gramatiky obsahující pravidlo s regulárním výrazem dostali pravidla z gramatiky regulární. A protože množina regulárních gramatik a jazyků je podmnožinou gramatik a jazyků bezkontextových, můžeme nově vzniklou gramatiku považovat rovněž za bezkontextovou.

Pro úplnost uveďme definici *bezkontextové gramatiky podle Chomského hierarchie*. Jedná se o uspořádanou čtveřici G = <**N**, **Σ**, **P**, S>, kde:

- **N** je konečná množina nonterminálních symbolů,
- **Σ** je konečná množina terminálních symbolů,
- **P** je konečná množina *přepisovacích pravidel*, které jsou podmnožinou kartézského součinu (**N** ∪ **Σ**) * N (**N** ∪ **Σ**)* × (**N** ∪ **Σ**)* a každé z nich je následujícího tvaru:
    - $A \rightarrow \gamma$
      kde $A \in \mathbf{N}$ a $\gamma \in (\mathbf{N} \cup \mathbf{\Sigma})^*$,
- S startovací nonterminál.

Důležitou odlišností alejových gramatik je ovšem to, že na rozdíl od bezkontextových gramatik, které obsahují jediný startovací nonterminál, součástí alejových gramatik je startovací regulární výraz obsahující nonterminální symboly.

Když můžeme regulární alejovou gramatiku převést na běžnou bezkontextovou, nabízí se otázka, proč vlastně používat regulární výrazy? Regulární výrazy jsou mocný nástroj pro práci s textem, který lze použít kromě hledání a editace zejména pro ověření vstupního řetězce. Dalším dobrým důvodem může být stručnost a přehlednost zápisu. Síla regulárních výrazů je pak také v tom, že s nimi lze pokrýt širokou paletu gramatik a jsou poměrně rychlé.

DTD, XML Schema, Relax NG

Celou dobu se bavíme o popisu schématu XML dokumentu různými způsoby. Na počátku stálo DTD, které popisovalo strukturu dokumentu regulárními výrazy, ale trpělo řadou nedostatků. Následoval tedy vývoj jazyka XML Schema, který ovšem našel své odpůrce ještě dříve, než-li byl uznán standardem. Je pravdou, že pro jednoduchou manuální editaci bez použití sofistikovaných nástrojů je jazyk XML Schema příliš složitý a nepřehledný. Započal se tedy vývoj jazyka, který spojuje přednosti obou předchozích – Relax NG. Definice datových typů a zápis v syntaxi XML přejímá z XML Schema, regulární výrazy naopak z DTD.

Podívejme se nyní na konkrétní rozdíly v zápisu shodného dokumentu různými metodami. Mějme jednoduchý příklad XML dokumentu, který má následující podobu:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<knihovna>
  <kniha id="1">
    <nazev>Název první knihy</nazev>
    <isbn>123-456-789-X</isbn>
    <jmeno_autora>Jan</jmeno_autora>
    <prijmeni_autora>Novák</prijmeni_autora>
    <rok>1995</rok>
    <cena>250,00</cena>
  </kniha>
  <kniha id="2">
    <nazev>Název druhé knihy</nazev>
    <isbn>987-654-321-Z</isbn>
    <jmeno_autora>Petr</jmeno_autora>
    <prijmeni_autora>Procházka</prijmeni_autora>
    <rok>1997</rok>
    <cena>520,00</cena>
  </kniha>
</knihovna>
```

Jak již bylo zmíněno, základním nástrojem pro tvorbu schématu dokumentu bylo v dávných dobách DTD. Jeho hlavními nevýhodami jsou především to, že nepodporuje jmenné prostory a definici datových typů. Kromě toho se v DTD využívá zápisu pomocí regulárních výrazů, což nemá nikde jinde v XML obdoby. Příklad DTD popisu uvedeného XML dokumentu by vypadal takto:

```
<!ELEMENT knihovna (kniha+)>
<!ELEMENT kniha (nazev, isbn, jmeno_autora, prijmeni_autora, rok, cena)>
<!ELEMENT nazev              (#PCDATA)>
<!ELEMENT isbn               (#PCDATA)>
<!ELEMENT jmeno_autora       (#PCDATA)>
<!ELEMENT prijmeni_autora    (#PCDATA)>
<!ELEMENT rok                (#PCDATA)>
<!ELEMENT cena               (#PCDATA)>
<!ATTLIST kniha
          id CDATA #REQUIRED>
```

XML Schema konsorcia W3C již využívá zápisu přímo v XML. Odstraňuje výše zmíněné nedostatky DTD. Ve stávající podobě je však XML Schema poměrně složitý a místy nepřehledný jazyk, není příliš pohodlné vytvářet schémata tohoto formátu bez pomoci specializovaného editoru. Schémata mají také poměrně košatý zápis a jsou proto prostorově náročná. XSD pro uvedený příklad XML dokumentu:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="knihovna">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="kniha"
                    maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nazev" type="xs:string"/>
              <xs:element name="isbn" type="xs:string"/>
              <xs:element name="jmeno_autora" type="xs:string"/>
              <xs:element name="prijmeni_autora" type="xs:string"/>
              <xs:element name="rok" type="xs:int"/>
              <xs:element name="cena" type="xs:decimal" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:int"
                          use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Jako alternativa k W3C XML Schema vznikl tedy jazyk Relax NG. Jazyk má prostorově úspornější zápis než XML Schema, nabízí podobu velmi jednoduchého a přehledného textu:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<element xmlns="http://relaxng.org/ns/structure/1.0"
         name="knihovna">
  <oneOrMore>
    <element name="kniha">
      <attribute name="id">
        <text/>
      </attribute>
      <element name="nazev">
        <text/>
      </element>
      <element name="isbn">
        <text/>
      </element>
      <element name="jmeno_autora">
        <text/>
      </element>
      <element name="prijmeni_autora">
        <text/>
      </element>
      <element name="rok">
        <text/>
      </element>
      <optional>
        <element name="cena">
          <text/>
        </element>
      </optional>
    </element>
  </oneOrMore>
</element>
```

Navíc poskytuje možnost zápisu v kompaktní syntaxi:

```
element knihovna {
  element kniha {
    attribute id { text },
    element nazev { text },
    element isbn { text },
    element jmeno_autora { text },
    element prijmeni_autora { text },
    element rok { text }
    element cena { text }?,
  }+
}
```

Ve výchozí podobě neobsahuje jazyk Relax NG podporu datových typů, což lze ale jednoduše doplnit:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<element xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
         name="knihovna">
  <oneOrMore>
    <element name="kniha">
      <attribute name="id">
        <data type="int"/>
      </attribute>
      <element name="nazev">
        <data type="string"/>
      </element>
      <element name="isbn">
        <data type="string"/>
      </element>
      <element name="jmeno_autora">
        <data type="string"/>
      </element>
      <element name="prijmeni_autora">
        <data type="string"/>
      </element>
      <element name="rok">
        <data type="int"/>
      </element>
      <optional>
        <element name="cena">
          <data type="decimal"/>
        </element>
      </optional>
    </element>
  </oneOrMore>
</element>
```

A opět schéma v úsporné kompaktní syntaxi, tentokrát s definicemi datových typů:

```
element knihovna {
  element kniha {
    attribute id { xsd:int },
    element nazev { xsd:string },
    element isbn { xsd:string },
    element jmeno_autora { xsd:string },
    element prijmeni_autora { xsd:string },
    element rok { xsd:int }
    element cena { xsd:decimal }?,
  }+
}
```

Schémata Relax NG lze v dnešní době již s pomocí k tomu určených nástrojů pohodlně převádět jak do jazyka XML Schema, tak i do DTD.

4 Závěr

Alejové gramatiky a automaty nejsou jen výkřikem do neznáma a snad ani slepou větví vývoje schémat dokumentů, ale mají také praktické použití. Na základě alejových gramatik je například postaven jazyk Relax NG společnosti OASIS určený ke specifikaci syntaxe dokumentů XML. O jeho úspěchu jasně vypovídá mimo jiné to, že získává stále větší oblibu nejen u technických výborů OASIS pro DocBook a OpenDocument, ale rovněž mezi pracovními skupinami společnosti W3C, tvůrce a propagátora konkurenčního jazyka XML Schema. A právě XML Schema bylo již od dob svého vzniku stále více kritizováno, zejména z důvodu své složitosti a nepřehlednosti. Naproti tomu Relax NG má dnes status specifikace OASIS a mezinárodního standardu ISO/IEC.

Velmi zajímavý je v této souvislosti příklad jazyka WSDL (Web Services Description Language), který popisuje služby založené na výměně zpráv XML pomocí protokolu SOAP. Velmi úzce spolupracuje s dalším jazykem, který má v sobě přímo zakomponován a který slouží k vytvoření syntaxe těchto zpráv pomocí popisu jejich schématu. Primárním obsaženým jazykem sice zůstalo XML Schema coby standard W3C, ale od začátku je pamatováno také na použití dalších jazyků, zejména Relax NG.

Samozřejmě nelze tvrdit, že by XML Schema bylo na ústupu převálcováno novou technologií, která zvítězí na celé čáře. Naopak pozice tohoto jazyka je stále velmi pevná, o čemž svědčí připravovaná nová verze XML Schema 1.1 nebo skutečnost, že tento jazyk je součástí standardů XPath/XSLT 2.0 a XQuery 1.0 a je také základem zmíněného WSDL. Rovněž dosud neexistuje reálná konkurence k datovým standardům jazyka XML Schema.

Relax NG je nutno chápat jako další nástroj a prostředek, který si kdokoliv může vybrat pro realizaci svých aplikací. Při výměně vytvořených XML dokumentů si pak bude možné zvolit, podle jakého standardního schématu chceme dokumenty validovat. Potěšující je také skutečnost, že již existují nástroje (například Trang), s jejichž pomocí lze schémata převádět nejen mezi syntaxemi XML Schema a Relax NG, ale třeba také DTD. Relax NG však již nelze označit jen za pouhou alternativu.

5 Literatura

[1] Murata, M.: Hedge automata: a formal model for XML schemata [online]. Dostupné na URL: http://www.horobi.com/Projects/RELAX/Archive/hedge_nice.html (květen 2007).

[2] Cimprich, P.: Akta X: Relax NG se prosazuje [online]. Poslední revize: 18. 12. 2006. Dostupné na URL: http://www.root.cz/clanky/akta-x-relax-ng-se-prosazuje/ (květen 2007).

[3] Kosek, J.: XML schémata [online]. Dostupné na URL: http://www.kosek.cz/xml/schema/uvod.html (květen 2006).