情報学科CSコース

情報システム(3年後期)
第14回
(田島担当分第6回)

田島 敬史

2013 年 1 月 23 日

XMLのためのスキーマ記述言語

HTML: 規格で使用するタグ・セットを規定

⇕

XML: アプリケーション毎に使用するタグ・セットを規定

⟶ 各アプリケーションにおいて，使用するタグ・セットとそれらの間の関係(スキーマ)を明示的に規定する枠組みが欲しい

- データがそのアプリケーションの規定に合っているかのチェック
- 規定に合ったデータを前提とした処理プログラムの作成
- 検索処理等における規定に合ったデータを前提とした最適化

DTD

- Document Type Declaration
- SGML 時代からの標準スキーマ言語
- XML 文書中での DTD の指定：`<!DOCTYPE ...>`
  - univ → 文書型の名前．document root の名．
  - SYSTEM "univ.dtd" → DTD ファイルのURI.
  - PUBLIC ..."univ.dtd" → まず公開されている ...を探し，なければ univ.dtd
  - <!DOCTYPE univ [...]> と [ ] 内に直接 DTD を書いても良い．両方も可．
- XML 文書の二つのクラス
  - well-formed — XML としての文法にあっている
  - valid — DTD が付けられていて，DTD の型にあっている

DTD

DTD を指定している XML データの例

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE univ SYSTEM "univ.dtd">
<univ>
  <dept>
    <name>CIS</name>
    <staff staffId="253">
      <name>Smith</name>
      <email>smith@cis.univ.edu</email></staff>
          :
    <student advisor="253"><name>Mich&eacute;l</name></student>
          :
  </dept>
  <dept>
    <name>BIO</name>
          :
  </dept>
</univ>
```

DTD

DTD の例

```
<!ELEMENT univ (dept)+>
<!ELEMENT dept (name,staff*,student*)>
<!ELEMENT staff (name,email*)>
<!ATTLIST staff staffId ID #REQUIRED>
<!ELEMENT student (name,email*)>
<!ATTLIST student advisor IDREF #IPLIED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```


DTD

DTD の構成要素

- 要素定義: <!ELEMENT 要素名 コンテンツの定義>
  - ANY — 制限なし
  - EMPTY — 空の要素
  - #PCDATA
  - コンテンツモデル
    - *... , ... , ... — 並び
    - *(... | ... | ...) — 選択
    - *?, *, +, — 繰り返し (0回または1回，0回以上, 1回以上)
  - mixed content (PCDATA と子要素両方を持つ) は (#PCDATA | 子要素名 | 子要素名 | ...)* の形のみ

DTD

- 属性リスト定義: <!ATTLIST 要素名 属性名 型 省略の扱い>
  - 型:
    - CDATA — 一般の文字列
    - ID — そのXMLデータの，全てのID属性の中で一意
    - IDREF — そのXMLデータ内のID属性の値への参照
    - IDREFS — 複数のID属性の値への参照
    - その他 (略)
  - 省略時の扱い
    - #REQUIRED — 必須．必ず定義しないといけない
    - #IMPLIED — 必須ではない
    - 値 — 定義されなければこの値になる
    - #FIXED 値 — この値に固定

DTD

DTD の構成要素

- 実体定義: <!ENTITY 実体名 値>
  - 例: <!ENTITY amp "&">
- notation 定義: (略)

XML のためのスキーマ記述言語の比較

DTD

- 局所木文法 (Local Tree Grammar)
  - 同じ要素名は必ず同じコンテンツモデルを持つ
  - 和，差に閉じていない (積については閉じている)
- データ型がない
- XML形式ではない独自の記法

XML のためのスキーマ記述言語の比較

XML Schema

- DTDの欠点を解消すべく W3C が規格化
  (しかし規格が巨大化)
- (ほぼ) 単一型木文法 (Single-Type Tree Grammar)
  - 出現位置によってコンテントモデルが変わる要素名を表現可能
  - データの「解釈」が一意に決まる
  - 和，差に閉じていない (積については閉じている)
- データ型を導入
- XML 形式で記述

XML のためのスキーマ記述言語の比較

RELAX NG

- Regular Language Description for XML-New Generation
- RELAX Core (村田真) と TREX (James Clark) の後継
- 正規木文法 (Regular Tree (Hedge) Grammar)
  - 出現位置によってコンテントモデルが変わる要素名を表現可能
  - 和，差，積について閉じている
  - incremental validation は DTD より難しい
- データ型に関しては，独立に定義されたデータ型ライブラリを取り込む形をとり，XML Schema のものもそのまま使用可能
- XML 形式での記法も用意されている

和について閉じていない DTD の例

DTD 1:

```
<!ELEMENT memberList (member)*)>
<!ELEMENT member (name, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
```

DTD 2:

```
<!ELEMENT memberList (member)*)>
<!ELEMENT member (name, email)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

和について閉じていない DTD の例

以下の DTD は「DTD 1 ∪ DTD 2」より大きい

```
<!ELEMENT memberList (member)*)>
<!ELEMENT member (name, (address|email))>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

例:

```
<memberList>
  <member><name>A</name><address>Uji</address></member>
  <member><name>B</name><email>b@a.org</email></member>
</memberList>
```


RELAX NG

Hedge

- a hedge = a sequence of trees
- (有限) シンボル集合 Σ, (有限) 変数集合 X 上の hedge:
  - ϵ は hedge (空の hedge)
  - x は hedge ($x \in X$)
  - $a\langle u \rangle$ は hedge ($a \in \Sigma, u$ は hedge)
  - uv は hedge (u, v は hedge)
- hedge の例: $a\langle\epsilon\rangle b\langle b\langle\epsilon\rangle x\rangle$
  (XML の記法なら `<a/><b><b/>x</b>`)

RELAX NG

Regular Hedge Grammar (RHG)

RHG $G = \langle \Sigma, X, N, P, r_{\mathrm{f}} \rangle$

- Σ — シンボルの (有限) 集合
- X — 変数の (有限) 集合
- N — 非終端記号の (有限) 集合
- P — 以下のいずれかの形式の生成規則の (有限) 集合.
  $n \to x$ $\quad (n \in N, x \in X)$
  $n \to a(u)$ $\quad (n \in N, a \in \Sigma, u$ は N 上の正規表現)
- r_{f} — N 上の正規表現 (開始記号の集合 S なら正規「木」言語)

r_{f} にマッチする非終端記号の列から始めて，生成規則を繰り返し適用して得られる Σ, X 上の hedge が G の言語

RELAX NG

RHG の例 (1)

$\Sigma = \{$`doc, title, para, image`$\}$
$X = \{$`#PCDATA`$\}$
$N = \{n_d,\ n_t,\ n_p,\ n_i,\ n_{\#}\}$
$P = \{n_d \to$ `doc`$\langle n_t(n_p|n_i)^*\rangle,\ \ n_t \to$ `title`$\langle n_{\#}\rangle,$
$\quad n_p \to$ `para`$\langle n_{\#}\rangle,\ n_i \to$ `image`$\langle\epsilon\rangle,\ \ n_{\#} \to$ `#PCDATA`$\}$
$r_{\mathrm{f}} = n_d$

等価な DTD (`doc` が root と指定されているとする)

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

RELAX NG

RHG の例(2)

$\Sigma = \{\texttt{segment}, \texttt{para}\}$
$X = \{\texttt{\#PCDATA}\}$
$N = \{n_1,\ n_2,\ n_p,\ n_\#\}$
$P = \{n_1 \to \texttt{segment}\langle(n_p|n_2)^*\rangle,\ \ n_2 \to \texttt{segment}\langle {n_p}^*\rangle,$
$\quad n_p \to \texttt{para}\langle n_\#\rangle,\ \ n_\# \to \texttt{\#PCDATA}\}$
$r_{\mathrm{f}} = n_1$

等価な DTD はない(segment のコンテンツが深さによって違う)
上の RHG で定義される言語を含む DTD の例:

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


RELAX NG

RHG の例(3)

$\Sigma = \{\texttt{doc}, \texttt{section}, \texttt{para}, \texttt{footnote}\}$
$X = \{\texttt{\#PCDATA}\}$
$N = \{n_d,\ n_s,\ n_1,\ n_2,\ n_f,\ n_\#\}$
$P = \{n_d \to \texttt{doc}\langle {n_1}^*(n_s {n_2}^*)^*\rangle,\ \ n_s \to \texttt{section}\langle n_\#\rangle,$
$\quad n_\# \to \texttt{\#PCDATA},\ \ n_1 \to \texttt{para}\langle n_\#\rangle,$
$\quad n_2 \to \texttt{para}\langle(n_\#|n_f)^*\rangle,\ \ n_f \to \texttt{footnote}\langle n_\#\rangle\}$
$r_{\mathrm{f}} = n_d$

等価な DTD はない(para のコンテンツが位置によって違う)
上の RHG で定義される言語を含む DTD:

```
<!ELEMENT doc (section|para)*)>
<!ELEMENT section (#PCDATA)>
<!ELEMENT para (#PCDATA|footnote)*>
<!ELEMENT footnote (#PCDATA)>
```


RELAX NG

DTD と RHG の違い

- どちらもコンテントモデルは正規表現
- DTD では，タグ名と非終端記号が一対一対応
- DTD = 局所木文法 ($\subseteq$ RHG)

RELAX NG

Deterministic Hedge Automaton (DHA)

DHA $A = \langle \Sigma, X, Q, \alpha, \iota, F\rangle$

- Σ — シンボルの(有限)集合
- X — 変数の(有限)集合
- Q — 状態の(有限)集合
- α — $\Sigma \times Q^*$ から Q への関数．ただし
  $\forall s \forall q(\{q_1 \ldots q_k \mid k \geq 0, \alpha(s, q_1, \ldots, q_k) = q\}$ は正規集合)
- ι — X から Q への関数
- F — Q 上の regular set

> α に従って bottom-up に状態遷移していき，F に含まれる $q_1 \ldots q_i (\in Q^*)$ になって終われば受理

RELAX NG

Non-Deterministic Hedge Automaton (NDHA)

NDHA $A = \langle \Sigma, X, Q, \alpha, \iota, F \rangle$

- Σ, X, Q, F — DHA と同様
- α — $\Sigma \times Q^*$ から 2^Q への関数．ただし $(\forall x \forall q)(\{q_1 \dots q_k \mid k \geq 0, \alpha(x, q_1, \dots, q_k) = S, q \in S\}$ は正規集合)
- ι — X から 2^Q への関数

α と ι が集合への関数になっており
遷移規則が非決定性になっている

RELAX NG

NDHA の例

「RHG の例(2)」に対応する NDHA

$$\begin{aligned}
\Sigma &= \{\texttt{segment},\ \texttt{para}\} \\
X &= \{\texttt{\#PCDATA}\} \\
Q &= \{q_1,\ q_2,\ q_p,\ q_\#\} \\
\alpha(\texttt{segment}, u) &\ni q_1 \quad (u \in L((q_p|q_2)^*)) \\
\alpha(\texttt{segment}, u) &\ni q_2 \quad (u \in L({q_p}^*)) \\
\alpha(\texttt{para}, u) &\ni q_p \quad (u \in L({q_\#}^*)) \\
\iota(\texttt{\#PCDATA}) &= \{q_\#\} \\
F &= \{q_1\}
\end{aligned}$$

通常の automaton 同様，NDHA は DHA に変換できる

RELAX NG

RELAX NG による具体的な記述例

```
<grammar xmlns="http://relaxng.org/ns/structure/0.9">
<start><ref name="segment1"></start>

<define name="segment1">
  <element name="segment">
    <oneOrMore><choice>
      <ref name="segment2">
      <ref name="paragraph">
    </choice></oneOreMore>
  </element>
</define>
```



```
<define name="segment2">
  <element name="segment">
    <oneOrMore><ref name="paragraph"></oneOreMore>
  </element>
</define>

<define name="paragraph">
   <element name="para"><text/></element>
</define>

</grammar>
```

XML Schema

- RELAX NG 同様，型名 (=非終端記号) とタグ名を分離
- 単一型木文法 = 「解釈 (or 型)」が一意になるように制限
  - 異なる開始記号が互いに「競合」しない
  - 一つの規則の右辺の中に「競合」する非終端記号が現われない
- XML Schema では以下に対応
  - 根の型は一つ
  - コンテントモデルに対する制限

XML Schema

解釈

木文法 G における木 t の解釈 I とは，木のノード e から G の非終端記号 $I(e)$ への写像で以下を満たすもの

- e が t の根なら $I(e)$ は開始記号
- 各ノード e とその子ノード列 $e_1, e_2, \ldots, e_n$ に対して，以下を満たす生成規則 $n \to a(u)$ を G が持つ:
  - $I(e)$ が n
  - e のタグ名が a
  - $I(e_1), I(e_2), \ldots, I(e_n)$ が u にマッチ

XML Schema

競合

非終端記号 n_1 と n_2 が競合するとは

n_1 を左辺に持つ生成規則と n_2 を左辺に持つ生成規則で右辺に同じタグ名を持つものがある

例:

$n_1 \to a(\ldots)$
$n_2 \to a(\ldots)$

XML Schema

XML Schema による具体的な記述例

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">

<element name="title">
  <complexType>
    <sequence>
      <element name="maintitle" type="string"/>
      <element name="subtitle" type="string"/>
    </sequence>
  </complexType>
</element>

<complexType name="secType">
```

```
  <sequence>
    <element name="title" type="string"/>
    <element name="para" type="string"
             maxOccurs="unbounded"/>
  </sequence>
</complexType>

<element name="book"/>
  <complexType>
    <sequence>
      <element ref="title">
      <element name="author" type="string"
               minOccurs="1" maxOccurs="5">
      <choice>
        <element name="chapter" type="secType">
```



```
        <element name="section" type="nameType">
      </choice>
    </sequence>
  </complexType>

</schema>
```