FH Aargau: Vorlesung XML Sommersemester 2004

Test XML Schemasprachen (DTD und XML Schema)

MUSTERLÖSUNG (Lösungen in grün)

17.5.2004, 13^{15} - 14^{15}

Bitte schreiben Sie *sorgfältig* und *gut lesbar*! Danke!
Für diesen Test dürfen Sie Ihre *schriftlichen Unterlagen* verwenden, jedoch keine elektronischen Hilfsmittel wie Laptop oder Organizer!

Die folgenden Aufgaben lösen Sie am besten, indem Sie für Schemas (*DTD* und *XML Schema*) die jeweilige Syntax verwenden. Syntaxfehler werden nicht bewertet, solange sie nicht die Interpretation der Schemas beeinträchtigen. Für XML Schema können Sie als Alternative eine graphische Notation verwenden, die sich z.B. an der graphischen Sicht des XML Spy orientieren könnte. Bedenken Sie aber dabei, dass die graphische Sicht (je nach Aufgabe) *Elemente*, *Attribute*, und *Typen* sowie deren *Beziehungen* wiedergeben muss!

1) Sie sollen Software schreiben, die aus einem XML Schema eine möglichst gute DTD macht, also möglichst wenig Modellinformation verliert. Dazu sollen Sie die XML Schema Informationen in DTD Syntax mit *Parameter Entities* umwandeln, und die Informationen, die sich nicht abbilden lassen, in *Kommentaren* in der DTD unterbringen. Beschreiben Sie in einigen Worten und Beispielen Ihre generelle Implementierungsstrategie, und nehmen Sie dabei zur Abbildung der folgenden Konstrukte explizit Stellung:
   1. Simple Types (für Elemente und Attribute, anonym und benannt)
      Abbildung in Parameter Entities, dabei notwendige Unterscheidung zwischen unterschiedlichen Konstrukten in DTDs (#PCDATA in Elementen, CDATA in Attributen). Anonyme Typen müssen nicht abgebildet werden, da sie nicht wiederverwendet werden können.
   2. Simple Type Restrictions
      Kein Mechnanismus in DTDs, Kommentar generieren.
   3. Complex Types (anonym und benannt)
      Abbildung in Content Model und Attribute Lists, Zusammenhang über den Namen und Kommentar erkenntlich machen. Anonyme Typen müssen nicht abgebildet werden, da sie nicht wiederverwendet werden können.
   4. Complex Type Restrictions
      Kein Mechnanismus in DTDs, Kommentar generieren.
   5. Complex Type Extensions
      Extension des Content Models über Parameter Entity mit der Erweiterung und Sequence mit dem erweiterten Typen abbilden, Extension der Attribute List über Parameter Entity mit der Attribute List abbilden.

6. Elementdeklarationen (lokal und global)
   Lokale Deklarationen müssen als global abgebildet werden (in DTDs gibt es keine lokalen Elementdeklarationen), Information in Kommentar erhalten.
7. Attributdeklarationen (lokal und global)
   Globale Deklarationen müssen als Parameter Entity und Referenz abgebildet werden (in DTDs gibt es keine globalen Attributdeklarationen),
8. Identity Constraints
   Keine Entsprechung in DTDs, als Kommentar erhalten.

Beschreiben Sie die Abbildungen so *kurz* und *prägnant* wie möglich, so dass die Idee Ihrer Abbildungsvorschrift deutlich hervortritt.

2) Zeichen in *Mixed Content* sind (neben Kommentaren und Processing Instructions) der einzige typfreie Inhalt in XML Schema. Beschreiben Sie kurz
   - warum das so ist (was das Problem wäre, Mixed Content im Typmodell von XML Schema einzuordnen)

   Mixed Content bezieht sich immer auf das gesamte Inhaltsmodell eines Typs. Dieser Inhalt können Zeichen sein gemischt mit Elementen. Diese Mischung zu beschreiben würde über das Modell von Simple Types (die fortlaufende Zeichenketten voraussetzen) hinausgehen.
   - sowie die Konsequenzen dieser Tatsache

   Mixed Content kann in keiner Weise eingeschränkt werden, d.h. wo Mixed Content erlaubt ist, kann das volle Unicode-Repertoire an Zeichen erscheinen. Dies kann ein Problem sein, wenn Applikationen den Zeichenumfang von Instanzen einschränken wollen.

   Hier geht es um konzeptionelle Fragen, nicht um technische Details!

3) Bei der *Erweiterung (Derivation by Extension)* von *Complex Types* kann man *Element Only* Types nur zu *Element Only* Types erweitern und *Mixed* Types nur zu *Mixed* Types. Warum? (der Grund liegt in XML selber...)
   Mixed Content bezieht sich immer auf das gesamte Element, es ist also nicht möglich, einen Teil des Inhaltsmodells als Mixed und einen anderen Teil als Element Only zu deklarieren. Aus diesem Grund kann eine Erweiterung nichts an der Mixed/Element Only Eigenschaft ändern, weil ja sonst der Teil des Base Types, der definitionsgemäss gleich bleiben muss (Typerweiterung hängt nur etwas an das Inhaltsmodell des Base Types an), auch geändert würde.

4) Gegeben sein ein XML Schema, das Elemente und Attribute enthält, die der folgenden DTD entsprechen:

```
<!ELEMENT dokument (kunden,bestellungen,lieferanten) >
<!ELEMENT kunden (kunde+) >
<!ELEMENT bestellungen (bestellung+) >
<!ELEMENT lieferanten (lieferant+) >
<!ELEMENT kunde (#PCDATA) >
<!ATTLIST kunde
      id          ID          #REQUIRED >
<!ELEMENT bestellung (#PCDATA) >
<!ATTLIST bestellung
      bestellung ID          #REQUIRED
      kunde      IDREF       #REQUIRED
      lieferant  IDREF       #REQUIRED >
<!ELEMENT lieferant (#PCDATA) >
<!ATTLIST lieferant
      id          ID          #REQUIRED >
```

Dabei enthalten verschiedene Elemente Informationen in PCDATA, die hier nicht interessieren, es geht in dieser Aufgabe nur um Referenzen. Geben Sie für die folgenden Anforderungen jeweils Identity Constraints an:

1. IDs von Kunden und Lieferanten müssen eindeutig sein (es darf also kein Kunde die ID eines Lieferanten haben und umgekehrt).
   In der Elementdeklaration von dokument:
   ```
   <unique name="kundelieferantIDunique">
    <selector xpath="kunden/kunde | lieferanten/lieferant"/>
    <field xpath="@id"/>
   </unique>
   ```
2. IDs von Bestellungen müssen eindeutig sein.
   In der Elementdeklaration von bestellungen:
   ```
   <unique name="bestellungID">
    <selector xpath="bestellung"/>
    <field xpath="@bestellung"/>
   </unique>
   ```
3. Der Kunde in einer Bestellung muss auf einen Kunden verweisen.
   In der Elementdeklaration von dokument:
   ```
   <key name="kundeID">
    <selector xpath="kunden/kunde"/>
    <field xpath="@id"/>
   </key>
   <keyref name="kunderef" refer="kundeID">
    <selector xpath="bestellungen/bestellung"/>
    <field xpath="@kunde"/>
   </unique>
   ```
4. Der Lieferant in einer Bestellung muss auf einen Lieferanten verweisen.
   In der Elementdeklaration von dokument:
   ```
   <key name="lieferantID">
    <selector xpath="lieferanten/lieferant"/>
    <field xpath="@id"/>
   </key>
   <keyref name="leferantref" refer="lieferantID">
    <selector xpath="bestellungen/bestellung"/>
    <field xpath="@lieferant"/>
   </unique>
   ```
5. Es darf nicht zwei Bestellungen von einem Kunden geben.
   In der Elementdeklaration von bestellungen:
   ```
   <unique name="bestellungskundeunique">
    <selector xpath="bestellung"/>
    <field xpath="@kunde"/>
   </unique>
   ```

Bedenken Sie, dass *drei* Aspekte wichtig sind: der *Kontext*, der *Selector*, und das oder die *Fields*. Beachten Sie die Benennung der Elemente und Attribute!