Universität Freiburg
Institut für Informatik
Prof. Dr. G. Lausen
Dr. Fang Wei

Georges-Köhler Allee, Geb. 51
D-79110 Freiburg

fwei@informatik.uni-freiburg.de

Webbasierte Informationssysteme
Wintersemester 10
Besprechung 02.02.2011

10. Übungsblatt: XPath und XQuery

Übung 1 (XPath Enthaltenseinsbeziehungen)

Wir betrachten das XPath-Fragment $XP^{[/,*,node(),text()]}$, bei dem nur Knotentests in Form von Tags, Wildcards ($*$), *node()*- und *text()*-Ausdrücken erlaubt sind und lediglich *child*-Achsen vorkommen dürfen. Insbesondere gibt es keine Bedingungen. Weiterhin beschränken wir uns auf absolute Pfadausdrücke.

Gegeben zwei Ausdrücke $q_1, q_2 \in XP^{[/,*,node(),text()]}$ sagen wir dass q_1 in q_2 enthalten ist ($q_1 \subseteq q_2$) genau dann wenn für alle möglichen XML-Dokumente D der Ausdruck q_2 mindestensc diejenigen Knoten des Dokuments selektiert, die auch von q_1 selektiert werden. Die Ausdrücke sind äquivalent ($q_1 \equiv q_2$) g.d.w. $q_1 \subseteq q_2$ und $q_2 \subseteq q_1$.

Geben Sie alle Enthaltenseinsbeziehungen und Äquivalenzbeziehungen zwischen den folgenden Paaren von XPath-Ausdrücken an. Wenn für ein Paar (q_1, q_2) von Ausdrücken $q_1 \not\subseteq q_2$ gilt, so geben Sie ein Dokument an, auf dem die durch q_1 selektierten Knoten nicht in der Menge der von q_2 selektierten Knoten enthalten sind.

a) q_1 : `/a/b/c` und q_2 : `/a/b/*`
b) q_1 : `/a/*/c` und q_2 : `/a/node()/c`
c) q_1 : `/a/b/*` und q_2 : `/a/b/node()`
d) q_1 : `/a/node()/c` und q_2 : `/a/*/node()`
e) q_1 : `/a/text()/a` und q_2 : `/c/text()/d/e`
f) q_1 : `/a/text()/a` und q_2 : `/a/node()/a`
g) q_1 : `/*/node()/b/c/text()` und q_2 : `/a/node()/*/c/node()`
h) q_1 : `/*/node()/node()/b/node()` und q_2 : `/a/node()/*/b/text()`

Übung 2 (Auswertung von XQuery-Anfragen)

Gegeben sei das XML-Dokument "book.xml", das den Aufbau des Buchs "Data on the Web" beschreibt.

```
<book>
  <title>Data on the Web</title>
  <author>Serge Abiteboul</author>
  <author>Peter Buneman</author>
  <author>Dan Suciu</author>
  <section id="intro" difficulty="easy" >
    <title>Introduction</title>
    <p>T1</p>
    <section>
      <title>Audience</title>
      <p>T1</p>
    </section>
    <section>
      <title>Web Data and the Two Cultures</title>
      <p>T2</p>
      <figure height="400" width="400">
        <title>Traditional client/server architecture</title>
        <image source="csarch.gif"/>
      </figure>
      <p>T2</p>
```

```
    </section>
  </section>
  <section id="syntax" difficulty="medium" >
    <title>A Syntax For Data</title>
    <p>T1</p>
    <figure height="200" width="500">
      <title>Graph representations of structures</title>
      <image source="graphs.gif"/>
    </figure>
    <p>T1</p>
    <section>
      <title>Base Types</title>
      <p>T1</p>
    </section>
  </section>
</book>
```

a) Geben Sie eine dem Dokument entsprechende DTD an. Ist die DTD rekursiv?

b) Geben Sie für die folgenden XQuery-Expressions das Ergebnis der Auswertung an. Formulieren Sie in Worten, was genau die Anfrage berechnet.

(a)
```
<result> {
    for $x1 in doc("book.xml")//p
    where $x1/text()="T2"
    return
      for $x2 in doc("book.xml")//section
      where (some $x3 in $x2//p satisfies $x3/text()=$x1/text())
      return <x/>
  } </result>
```

(b)
```
<results> {
    for $x1 in doc("book.xml")//section
    where $x1//p/text()="T2"
    return
      <result> {
        for $x2 in $x1//p return $x2
      } </result>
  } </results>
```

(c)
```
<results> {
    (
      <stars>{ fn:count(doc("book.xml")//*) }</stars>,
      <nodes>{ fn:count(doc("book.xml")//node()) }</nodes>,
      <text>{ fn:count(doc("book.xml")//text()) }</text>
    )
  } </results>
```

Übung 3 (Auswertung von XQuery-Anfragen)
Gegeben sei das XML-Dokument "bib.xml" aus den W3C XML Usecases (das Dokument können Sie von der Homepage herunterladen). Formulieren Sie die folgenden Anfragen in XQuery und testen Sie ihre Anfragen mit Hilfe des Beispieldokuments. Modifizieren Sie das Beispieldokument gegebenenfalls um kompliziertere Anfragen zu testen.

a) Geben Sie alle Bücher aus, bei denen der Nachname eines Autors mit dem Nachnamen des Publishers übereinstimmt. Nehmen Sie dabei an, dass alle Publisher im XML-Dokument durch Nachnamen spezifiziert sind.

b) Geben Sie für jedes Buch von Autor Peter Buneman den Titel und die Anzahl der Autoren aus. Falls der Preis des Buches über 20 liegt, soll auch dieser ausgegeben werden.

c) Geben Sie alle Paare von unterschiedlichen Büchern des gleichen Verleger aus. Das Ergebnis darf keine Duplikate enthalten. Sie dürfen davon ausgehen, dass alle im Dokument vorkommenden Buchtitel unterschiedlich sind.

d) Geben Sie für jeden Autoren den Nachnamen, Vornamen, und die Summe der Preise aller von ihm (mit)verfassten Bücher aus.

e) Geben Sie für jeden Autor eine nach Preis geordnete Liste aller seiner Buchtitel aus. Ein Element der Liste soll Name und Vorname des Autors enthalten und auf maximal drei Titel begrenzt werden. Zusätzlich soll ein Tag "minprice" ausgegeben werden, in dem der Preis des billigsten Buchs vermerkt ist.

f) Erzeugen Sie ein HTML-Dokument, das im Titel den Text "Anzahl der Bücher: $x" stehen hat, wobei $x die Anzahl der insgesamt vorhandenen Bücher ist. Als Überschrift soll der Text "Das Dokument enthält Bücher von $y Autoren" erscheinen, mit Variable $y entsprechend ersetzt. Abschließend soll das Dokument eine nach Titel sortierte Auflistung aller Buchtitel und Preise in Tabellenform enthalten.

Übung 4 (XQuery Äquivalenz von Ausdrücken)
Welche der jeweils drei XQuery Anfragen sind paarweise äquivalent? Für nicht äquivalente Anfragen-Paare geben Sie ein Dokument an, auf dem die beiden Anfragen unterschiedliche Ergebnisse berechnen. In den Anfragen wurde die explizite Dokumentenangabe weggelassen, beispielsweise steht `/a//b` als Abkürzung für `doc("document.xml")/a//b`.

a) Anfrage 1.1:
```
<q1> { for $a in /a, $b in $a//b return <match/> } </q1>
```
Anfrage 1.2:
```
<q1> { for $a in /a return
         for $b in $a//b return <match/> } </q1>
```
Anfrage 1.3:
```
<q1> { for $b in /a//b return <match/> } </q1>
```

b) Anfrage 2.1:
```
<q2> { for $a in //a return
         for $b in $a//b return <match/> } </q2>
```
Anfrage 2.2:
```
<q2> { for $a in //a return
         for $b in //a//b return <match/> } </q2>
```
Anfrage 2.3:
```
<q2> { for $b in //a//b return <match/> } </q2>
```

c) Anfrage 3.1:
```
<q3> { let $x := (1, 2, 3)
       return <a>{ $x }</a> } </q3>
```
Anfrage 3.2:
```
<q3> { let $x := (1, 2, 3)
       for $y in $x return <a>{ $y }</a> } </q3>
```
Anfrage 3.3:
```
<q3> { for $x in (1, 2, 3) return <a>{ $x }</a> } </q3>
```

d) Anfrage 4.1:
```
<q4> { if (some $x in //x satisfies $x/a)
       then <true/> else <false/> } </q4>
```
Anfrage 4.2
```
<q4> { if (for $x in //x return $x/a)
       then <true/> else <false/> } </q4>
```
Anfrage 4.3
```
<q4> { if (fn:exists(//x/a))
       then <true/> else <false/> } </q4>
```

e) Aufgabe 5.1
```
<q5> { for $book in //book return
         for $article in //article
         where $article/author=$book/author
```

```
         return $book } </q5>
```

Aufgabe 5.2

```
<q5> { for $book in //book return
         for $article in //article return
         if ($article/author=$book/author)
         then $book else () } </q5>
```

Aufgabe 5.3

```
<q5> { for $book in //book return
         for $author in //article/author
         where $book/author=$author
         return $book } </q5>
```