

Markup-Sprachen und semi-strukturierte Daten, SS 2008
Übungsblatt 11

Besprechung am Do/Fr 10./11.07.2008

Aufgabe 11-1 Rekursionsschema a la XSLT in Xcerpt

Es soll für ein XML Eingabedokument `eingabe.xml` eine rekursive Transformation geschrieben werden, die jedes Element in ein Element mit Label `element` “verpackt”.

Beispiel:

Aus folgendem Dokument

```
<a>
  <b>
    <c />
    <d />
  </b>
  <e>
    <f />
  </e>
  <g />
</a>
```

wird dann in etwa

```
<element>
  <a>
    <element>
      <b>
        <element><c /></element>
        <element><d /></element>
      </b>
    </element>
    <element>
      <e>
        <element><f /></element>
      </e>
    </element>
    <element><g /></element>
  </a>
</element>
```

Es kann angenommen werden, dass in der Eingabe nur Elemente, keine Attribute oder Textknoten, vorkommen.

Aufgabe 11-2 Xcerpt, XQuery

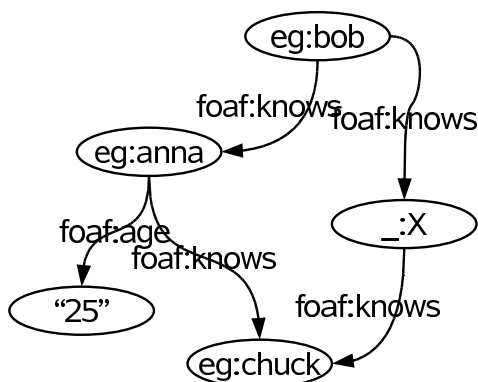
```
CONSTRUCT
  Substitutions [
    all Substitution [
      x [ var X ], y [ var Y ], z [ var Z ] ] ]
FROM
  in { resource { 'example.xml' },
    a {{ var X -> b {{ }},
      var Y -> /.*/ [ var X, var Z ]
    }} }
END
```

Gegeben sei obige Xcerpt Regel. Geben Sie eine Anfrage in XQuery an, welche für beliebige Eingabedokumente das selbe Ergebnis (bis auf die Ordnung der **Substitution**-Elemente) wie obige Regel liefert. Beachten Sie dazu folgende Punkte:

- Es ist sinnvoll jedem Knoten in der Anfrage eine eigene XQuery Variable zuzuordnen.
- Injektivität zwischen Geschwisterknoten kann mit der Funktion **is** sichergestellt werden.
- Mehrfaches Vorkommen derselben Xcerpt Variable bedeutet, dass Teile der Daten strukturell gleich sein müssen. Dies kann mit der XQuery Funktion **fn:deep-equal** getestet werden.
- Die Ordnung zwischen Elementen kann mit der Funktion **<<** verglichen werden.

Aufgabe 11-3 Semantisches Web, RDF, SPARQL

Gegeben sei der folgende RDF Graph über Freundschaftsbeziehungen und seine Turtle Serialisierung. In dem Graphen kommen drei Arten von Knoten vor: URIs (wie zum Beispiel <http://www.example.com/anna>), Literale (wie der String '25') und anonyme Knoten (**_:X**). URIs stehen für eine Resource, die man beschreiben möchte; so steht die URI <http://www.example.com/anna> wahrscheinlich für eine Person mit dem Namen Anna. Literale stehen dagegen für sich selbst, und anonyme Knoten sind bloße Platzhalter für Ressourcen, welche man nicht genauer spezifizieren kann oder will. Tripel mit anonymen Ressourcen können als existentielle Aussagen aufgefasst werden.



Listing 1: Serialisierung des RDF Graphen

```
@prefix foaf:
  <http://xmlns.com/foaf/0.1/> .
@prefix eg:
  <http://www.example.com/persons> .

eg:bob foaf:knows eg:anna .
eg:bob foaf:knows _:X .
eg:anna foaf:age "25" .
eg:anna foaf:knows eg:chuck .
_:X foaf:knows eg:chuck .
```

- Schreiben Sie eine Sparql-Anfrage, welche die Bekannten von Bob selektiert (unter der Annahme, dass die URI <http://www.example.com/persons/bob> für die Person Bob steht, und die URI <http://xmlns.com/foaf/0.1/knows> für die Bekanntschaftsbeziehung zwischen Personen)!
- Ist die Antwort das, was man erwartet?