Facoltà di Ingegneria dell'informazione

Ingegneria della conoscenza 2010–11

Appello del 8 febbraio 2011

Parte II			
Cognome		Laureando	si ☐ no ☐
Nome		Matricola	

4 7 punti	Si vuole progettare un linguaggio XML per la descrizione di diagrammi di flusso. Ogni diagramma si articola in una o più pagine, sulle quali sono raffigurati i task (esecuzione, input, output) connessi tra di loro da frecce (flusso di esecuzione e flusso di messaggi). Ciascun elemento grafico ha una posizione (x ed y). Solo i task di esecuzione possono inviare e/o ricevere messaggi. - Descrivere il linguaggio tramite un DTD. - Dire se e come l'uso di XSD permetterebbe di descrivere meglio il linguaggio - Implementare un programma Java che, assumendo il documento valido rispetto al DTD, stampa a video le coordinate di tutti i task di esecuzione che non inviano messaggi ad altri task.

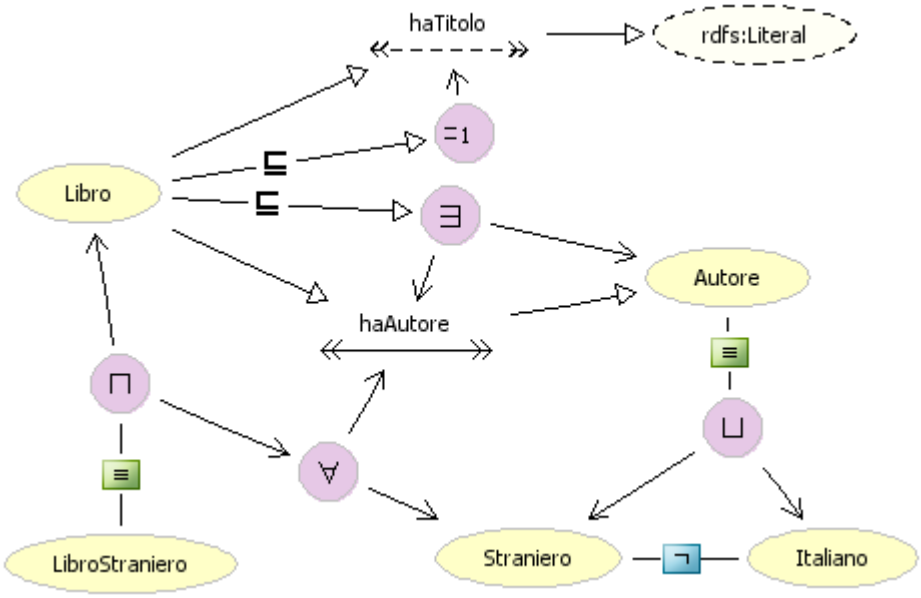
```
<!ELEMENT diagramma (pagina)+> <!ELEMENT pagina (esecuzione|input|output|freccia)*>
<!ELEMENT esecuzione (messaggio)*> <!ELEMENT input EMPTY> <!ELEMENT output EMPTY>
<!ELEMENT freccia EMPTY> <!ELEMENT messaggio EMPTY>
<!ATTLIST esecuzione cod ID #REQUIRED
   x CDATA #REQUIRED y CDATA #REQUIRED>
<!ATTLIST input cod ID #REQUIRED
   x CDATA #REQUIRED y CDATA #REQUIRED>
<!ATTLIST output cod ID #REQUIRED
   x CDATA #REQUIRED y CDATA #REQUIRED>
<!ATTLIST esecuzione cod ID #REQUIRED
   x CDATA #REQUIRED y CDATA #REQUIRED>
<!ATTLIST freccia from IDREF #REQUIRED to IDREF #REQUIRED>
<!ATTLIST messaggio to IDREF #REQUIRED>
```

neanche on l'XSD potremmo tipizzare i nodi target del messaggio, potremmo però strutturare i tipi in modo da avere un tipo task che raccolga le caratteristiche comuni (es coordinate).

```
Class myHandler extends DefaultHandler{ boolean messaggio; String id;
   void startElement(String namespaceURI,String localName, String qName, Attributes atts){
      if (localName.equals("esecuzione")){
         id = atts.getValue("cod"); messaggio = false;
      }
      if (localName.euqlas("messaggio")) messagio = true;
   }
   void endElement(String namespaceURI,String localName, String qName){
      if (localName.equals("esecuzione") && ! messaggio) System.out.println(id);
   }
}
```

5 7 punti	• Si spieghi (mostrando anche un esempio) perché gli schemi RDF-S / OWL sono detti "semantici" in contrapposizione con schemi "sintattici" quali DTD/XSD e si dica se esistono casi d'uso diversi degli uni rispetto agli altri • Si descriva in forma grafica una piccola ontologia OWL-DL per la classificazione di libri. Ogni libro ha un titolo ed uno o più autori. Gli autori possono essere (solo) italiani o stranieri. I libri stranieri sono quei libri che non hanno nemmeno un autore italiano. ◦ Che approssimazioni occorre introdurre limitando l'espressività ad OWL-Lite? ◦ Che approssimazioni occorre introdurre limitando l'espressività ad RDF-S ?

Vedi appunti delle esercitazioni... esempio
<rdf:RDF>
<Persona nome="Mario">
<amicoDi><Persona rdf:about="#p2" nome="Luigi"/></amicoDi>
</Persona></rdf:RDF>
è semanticamente equivalente a
<rdf:RDF>
<Persona amicoDi="#p2"><nome>Mario</nome></Persona>
<Persona rdf::id="p2" nome="Luigi"/>
</Persona></rdf:RDF>
per RDF[S] sono indistinguibile, per XSD no.
Gli schemi semantici ammettono altri usi oltre a quello prescrittivo: ad esempio il deduttivo sul modello ed il deduttivo sullo schema (vedi dispense)



OWL lite: non posso esprimere la cardinalità =1 (ma posso approssimarla con un esiste)

RDF-S: perdo tutte le quantificazioni, intersezioni, unioni e disgiunzioni tra classi...
il libro straniero è semplicemente sottoclasse di libro
Straniero ed italiano sono generiche sottoclassi di Autore e non posso dire che sono disgiunte
restano inalterati domini e range

Allegati

DOM

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.parse("libro.xml");
```

```
Interface Document{…
        Element getDocumentElement();
}
Interface Element extends Node {…
        String getAttribute(String name) ;
        public Node[ ] getChildNodes();
        String getTagName() ;
}
Interface CharacterData extends Node{…
        String getData();
}
```

SAX

```
Interface ContentHandler {…
        void startDocument();
        void startElement(String namespaceURI,String localName, String qName,  Attributes atts);
        void characters(char[] ch, int start, int length);
        void endDocument();
        void endElement(String namespaceURI,String localName, String qName);
}
Class DefaultHandler Implements ContentHandler();
interface Attributes{…
String getValue(String qName);
}
```