Facoltà di Ingegneria dell'informazione

Ingegneria della conoscenza 2010–11

Appello del 25 luglio 2011

Parte II			
Cognome		Laureando	si ☐ no ☐
Nome		Matricola	

4

7 punti

Sia dato il seguente schema XSD:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" ...>
<xsd:element name="a">
        <xsd:complexType><xsd:sequence>
                <xsd:element name="a">
                        <xsd:complexType mixed="true">
                                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                        <xsd:element name="b" type="xsd:int"/>
                                        <xsd:element name="c" type="xsd:int"/>
                                </xsd:choice>
                        </xsd:complexType>
                </xsd:element>
                <xsd:element ref="a" minOccurs="0"/>
        </xsd:sequence></xsd:complexType>
</xsd:element></xsd:schema>
```

1. scrivere il documento XML più semplice valido rispetto allo schema
2. descrivere per quanto possibile il linguaggio con un DTD, indicando tutte le approssimazioni necessarie
3. implementare in java la verifica di una delle caratteristiche perse nelle approssimazioni (assumendo la validità rispetto al DTD)
4. che problema sorgerebbe se si modificasse l'XSD eliminando l'ultimo attributo minOccurs="0"?

1) `<a><a>txt</a></a>`

2)

```
<!ELEMENT a (a|b|c|#PCDATA)*>
<!ELEMENT b #PCDATA>
<!ELEMENT c #PCDATA>
```

E' impossibile distinguere tra i due tipi di A, il mixed content ne risente dovendo prevedere anche una "a".
E' impossibile indicare che il testo contenuto nei b e nei c sia un intero

3)

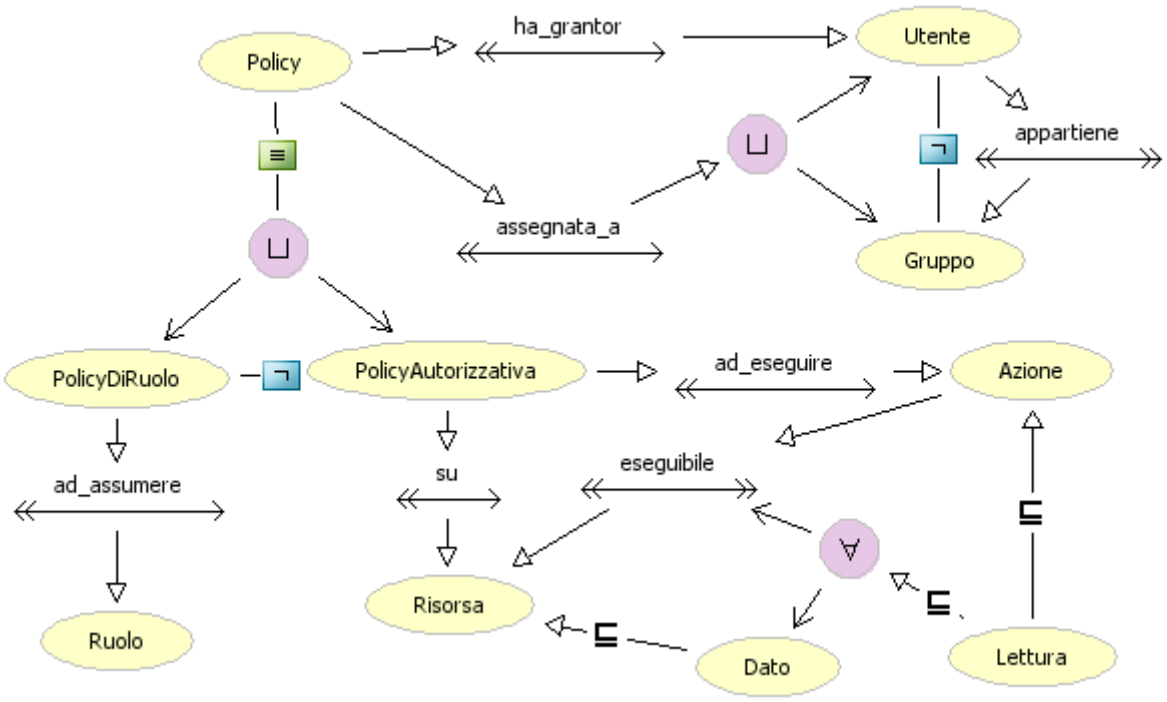
```
public class Handler extends DefaultHandler{
        private  boolean_err, _check;
        void startDocument(){_err = _check = false;}
        void startElement(String namespaceURI,String localName, String qName, Attributes atts){
                _check = localName.equals("b") || localName.equals("c");}
        void characters(char[] ch, int start, int length){
                try{
                if (_check) Integer.parseInt(new String(ch, start, length));
                } catch (Exception ex){_err = true;}
        }
        void endDocument(){System.out.println(_err ? "errore" : "ok");}
        void endElement(String namespaceURI,String localName, String qName){_check = false;}
}
```

4) eliminando il minOccurs si ottinee il valore di default (1), che impone l'esistenza di un "a" del primo dito all'interno di ogni "a" del primo tipo. Dato che "a" è l'unico elemento globale deve essere radice di ogni documento. Quindi il linguaggio contiene solo documenti infiniti.

continuare sulla pagina 4

5 7 punti	Si vuole proporre una ontologia per rappresentare le policy di sicurezza in un sistema informativo. Ogni policy può essere assegnata da un utente (grantor) a singoli utenti o a gruppi di utenti. Le policy si dividono in: • autorizzative: assegnano il privilegio di eseguire una certa azione su una certa risorsa (IT, da non confondersi con Resource di RDF) • di ruolo: assegnano il privilegio di assumere (attivare) un ruolo Ogni azione, a sua volta, può essere eseguita solo su alcune risorse. Le azioni di lettura possono essere eseguite solo su risorse di tipo "Dati". 1. descrivere l'ontologia in GrOWL, indicando le approssimazioni necessarie. 2. dire quali ulteriori approssimazioni sarebbero necessarie per ridurre l'espressività ad RDF[S] 3. scrivere una query SPARQL che restituisca tutte le policy autorizzative assegnate ad utenti specifici

1)



nessuna approssimazione necessaria

2)
non sarebbe possibile esprimere il fatto che le letture sono eseguibili solo su dati.
Le due unioni dovrebbero essere approssimate con sottoclassi ...
I ruoli non potrebbero essere indicati come funzionali, quindi …
non sarebbe possibile imporre che le policy di ruolo/autorizzativa sia una partizione (cioè che siano distinte), così come per i gruppi e gli utenti

3)

```
SELECT ?p
WHERE  {
    ?p      assegnata_a     ?u .
    ?p      rdf:type        PolicyAutorizzativa.
    ?u      rdf:type        Utente
}
```

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);
}
```