# Tutorial session 7

## DTD Validation and XPath

## 1 Movie database

For the next questions, we consider documents valid with respect to the DTD given in Figure 1. A fragment of valid document is given in Figure 2.

```xml
<!ELEMENT movies (movie *, people*) >
<!ELEMENT movie (title, genre, year, cast, director, producer, studio) >
<!ELEMENT title (#PCDATA) >
<!ELEMENT genre (#PCDATA) >
<!ELEMENT year (#PCDATA) >
<!ELEMENT cast (character+) >
<!ELEMENT director (EMPTY) >
<!ATTLIST id type IDREF #REQUIRED >
<!ELEMENT producer (EMPTY) >
<!ATTLIST id type IDREF #REQUIRED >
<!ELEMENT studio (#PCDATA) >
<!ELEMENT character (role, name) >
<!ATTLIST id type IDREF #REQUIRED >
<!ELEMENT role (#PCDATA) >
<!ELEMENT name (first, last) >
<!ELEMENT first (#PCDATA) >
<!ELEMENT last (#PCDATA) >
<!ELEMENT people (name) >
<!ATTLIST id type ID #REQUIRED >
```

**Fig. 1 – The DTD of an XML movie database**

### Questions:

1. Considering all the constraints that are represented by this DTD, can you validate a document against this DTD using only a tree automata? If not, what prevents you to do so.
2. Rewrite the elements `name` and `character` so that they can have exactly the same content but where the order doesn’t matter. (For instance, `name` can be `first, last` or `last, first` but not `last, last` for instance).
3. How many possibilities do you need to consider if you want to do the same for the movie element.
4. *(from the lecture)* describe the algorithm used to efficiently validate a DTD doing a top-down traversal of the tree.
5. Give an algorithm to validate a DTD in streaming.
<movies>
    ...
</movies>

<movie>
    <title>The Good, the Bad and the Ugly</title>
    <genre>Western</genre>
    <year>1966</year>
    <cast>
        <character id="123">
            <role>"The man with no name" is a mysterious man who travels the country on his mule.</role>
            <name><first>Blondie</first><last></last></name>
        </character>
        ...
    </cast>
</movie>

<movie>
    <title>Million Dollar Baby</title>
    <genre>Drama</genre>
    <year>2004</year>
    <cast>...</cast>
    <director id="123"/>
    <producer id="400"/>
    <studio>Warner Bros. Pictures</studio>
</movie>

<people id="123"><name><first>Clint</first><last>Eastwood</last></name>
    <last>Eastwood</last></name>
</people>

Fig. 2 – Fragment of valid XML movie database

2 XPath to English

Describe in English what the following XPath expression computes:
1. //movie/title
2. //people/name[ first = "Bruce" ]
3. //movie[ year > "1990" ]
4. //role[ contains( ., "Batman")]/../..//title
5. /descendant::movie[1]
6. /descendant::role[1]
7. //role[1]
8. /character[@id = //people[name ="Al Pacino"]/@id]/name
9. /movies/movie[ count(./cast/character) > 10 ]/title
10. /descendant::people[ position() = 1]/preceding[position() = 1]

3 English to XPath

Give an XPath expression which answers the query (There might be many possible answers):

2
– Give the title of the movies produced between 1955 and 1960
– Give the years of the 6 “Star Wars” movies were produced
– Give the id of the actors performing in “The Godfather”
– Give the id of all the producers who produced a film between 1990 and 2000.
– Give the title of all the movies where the director is also an actor
– Return the movie element occurring in the tenth position before the third movie (in document order) featuring “Bruce Willis”.
– In Question 3 and 4, can you give the name instead of the id? What is this operation called.

4 Checking constrains with XPath queries

Given a document which looks like a film database, we want to check using XPath queries that some of the constraint of the DTD of Figure 1 are verified. Check the following constraint using XPath queries. The result of the query must be non-empty if the constraint is verified:
– The root of the document is a movies element.
– The children of movies are a sequence of movie elements followed by a sequence of people elements
– the content of every movie element is exactly as in the DTD: title, genre, year, cast, director, producer, studio (only check for the labels).
– every cast element is non-empty
– every director, producer, people and character has an id attribute.