Lab 2

DTD validation

- Provide a DTD D such that the document t is valid with regard to D
- Write D as a regular grammar G
- Show that t is in L(G)
- Generate another non-trivial document that is in L(G)

D.dtd:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT a ((b|e)*)>
<!ELEMENT b ((c|f)*)>
<!ELEMENT e ((c|f|d)*)>
<!ELEMENT d (#PCDATA)>
<!ELEMENT f (#PCDATA)>
<!ELEMENT c (d?)>
```

Regular Grammar G

Reminder : G = (N, T , S, P)

where N = non-terminal, T = terminal, S = start, P = rules

- N = (A,B,C,D,F)
- T = (a,b,e,c,f,d)
- S = (a)
- P =
  - A -> a[B,E]
  - B -> b [(C,F)*]
  - C -> c[D?]
  - D -> d[epsilon]
  - E -> e[D, C+]
  - F -> d[epsilon]

In order to prove that t is in L(G), we tranform the grammar in automaton

- a(q_b,q_e) -> q_a
- b((q_c,q_f)*) -> q_b
- c(q_+) -> q_c
- d(epsi) -> q_D
- e(q_d,q_c+) -> q_e
- f(epsi) -> q_f

\sigma = {a,b,c,d,e,f}
\Q = {q_a,q_b,q_c,q_d,q_e,q_f}
F= {q_a}

Another non-trivial example

```
<a>
  <b>
    <c>
    </c>
  </b>
</a>
```

Properties of Regular Tree Grammars

1

- FAUX, student without address in G1 cannot be in G2

2

- TO G1 no
- TO G2 yes

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT  directory ((student|teacher)*)>
<!ELEMENT student (contact)>
<!ELEMENT teacher (contact)>
```

```
<!ELEMENT contact (Name,address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
```

3

```
<directory>
  <student>
    <contact>
      <name> coucou </name>
      <address> 46 rue barrault </address>
    </contact>
  </student>
</directory>
```

XML Schema

regular tree Grammar

R = (N,T,S,P)
N = (A,B,C,D,E,F,G,H,I,J,K,L,M)
T = (shiporder,orderperson,shipto,name,address,country,item
, title,note,quantity,price)
S = (A)
P =

- A -> shiporder[B,C,H,M]
- B -> orderperson[epsilon]
- C -> shipto[D,E,F,G]
- D -> name[epsilon]
- E -> city[epsilon]
- F -> address[epsilon]
- G -> city[epsilon]
- H -> item[I,J,K,L]
- I -> title[epsilon]
- J -> note[epsilon]
- K -> qty[epsilon]
- L -> price[epsilon]

DTD and RTG

1

The grammar allows only patient* follow doctors * therefore there is a valid DTD D file that cannot be generated by the RTG G.

2

The opposite is true because of the flexibility of dtd syntax (correct even if there are more attributes)

3

G writen in DTD

```
<!ELEMENT files ((file|patient|Doctor)*)>
<!ELEMENT Doctor (person)>
<!ELEMENT patient
<!ELEMENT person (FName,LName,Tel)>
<!ELEMENT file ((file|person|Doctor)*)>
```

THIS IS NOT POSSIBLE -> CANNOT DEFINE TWO PERSON FOR PATIENT AND DOCTOR

4

```
<Files>
  <file>
  <consultation>
    <symptom>
      data
    </symptom>
    <Prescription>
      <Medication>
        data
      </Medication>
    </Prescription>
  </consultation>
  </file>
  <patient>
```

```
      <person>
      <FName>
      </FName>
      <LName>
      </LName>
      </person>
    </patient>
    <Doctor>
      <person>
        <FName>
        </FName>
        <LName>
        </LName>
        <Tel></Tel>
      </person>
    </Doctor>
```

Properties of Regular Tree Grammars

Consider two regular tree grammars G1 = (N1, T, S1, P1) and G2 = (N2, T, S2, P2), having the same terminal symbols T.
Can you build the regular tree grammar G3 = (N3,T,S3,P3) that captures the intersection of L(G1) and L(G2)?

idea :

N3 = N1 * N2

S3 = S1 * S2

P3 : n \in N3 -> (R_1,R_2) where
R_1 \in P_1
and
R_2 in P_2