Méthodes Formelles pour XML : validation

Pierre-Alain Reynier
http://www.lif.univ-mrs.fr/~preynier/XML/

Plan du cours

Rappels et compléments :

1 - Automates d'arbres et complexité

Aujourd'hui :

2 - Langages de spécifications :

- DTDs
- XML Schémas

I - Automates d'arbres et complexité : rappels et compléments

Arbres d'arité fixe égale à d

Automates ascendants

Forme : (Σ, Q, ∂, F)

Transitions :

Feuilles : $\partial : \Sigma \rightarrow P(Q)$

Noeuds internes :

$\partial : \Sigma \times Q^d \rightarrow P(Q)$

Arbre *accepté* ssi il existe un calcul t.q. *la racine est dans F*

Automates descendants

Forme : (Σ, Q, ∂, I)

La racine part de I

Transitions :

Noeuds internes :

$\partial : \Sigma \times Q \rightarrow P(Q^d)$

Feuilles : $\partial : \Sigma \times Q \rightarrow P(Q)$

Arbre *accepté* ssi il existe un *calcul terminant*

Arbres d'arité variable

Automates ascendants

Forme : (Σ, Q, ∂, F)

Transitions :

Feuilles : $\partial : \Sigma \rightarrow P(Q)$

Noeuds internes :

$\partial : \Sigma \times Reg(Q) \rightarrow P(Q)$

Arbre *accepté* ssi il existe un calcul t.q. *la racine est dans F*

Automates descendants

Forme : (Σ, Q, ∂, I)

La racine part de I

Transitions :

Noeuds internes :

$\partial : \Sigma \times Q \rightarrow P(Reg(Q))$

Arbre *accepté* ssi il existe un *calcul terminant*

Complexité

- Le test d'appartenance peut être fait en temps linéaire pour un automate déterministe, et polynomial sinon.

- Le test d'inclusion peut être fait en temps polynomial pour des automates déterministes, et exponentiel sinon.

2 - Langages de spécifications

- DTDs -

DTDs

- Décrire les fils d'un noeud de label 'a' par une expression régulière
- Un exemple de DTD :

<!ELEMENT populationdata (continent*) >
<!ELEMENT continent (name, country*) >
<!ELEMENT country (name, province*)>
<!ELEMENT province (name, city*) >
<!ELEMENT city (name, pop) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT pop (#PCDATA) >

DTDs et déterminisme

- **Restriction :** Les expressions régulières utilisées dans les DTDs doivent être déterministes
- Extrait de la W3C :

  "For Compatibility, it is an error if the content model allows an element to match more than one occurrence of an element type in the content model."

- l'automate correspondant doit être déterministe :
  - (a+b)*a ne l'est pas
  - quand on lit <a>, on ne sait pas si on est dans (a+b) ou dans le a final
  - (b*a)(b*a)* est une expression déterministe équivalente

A propos des expr. rég. dét.

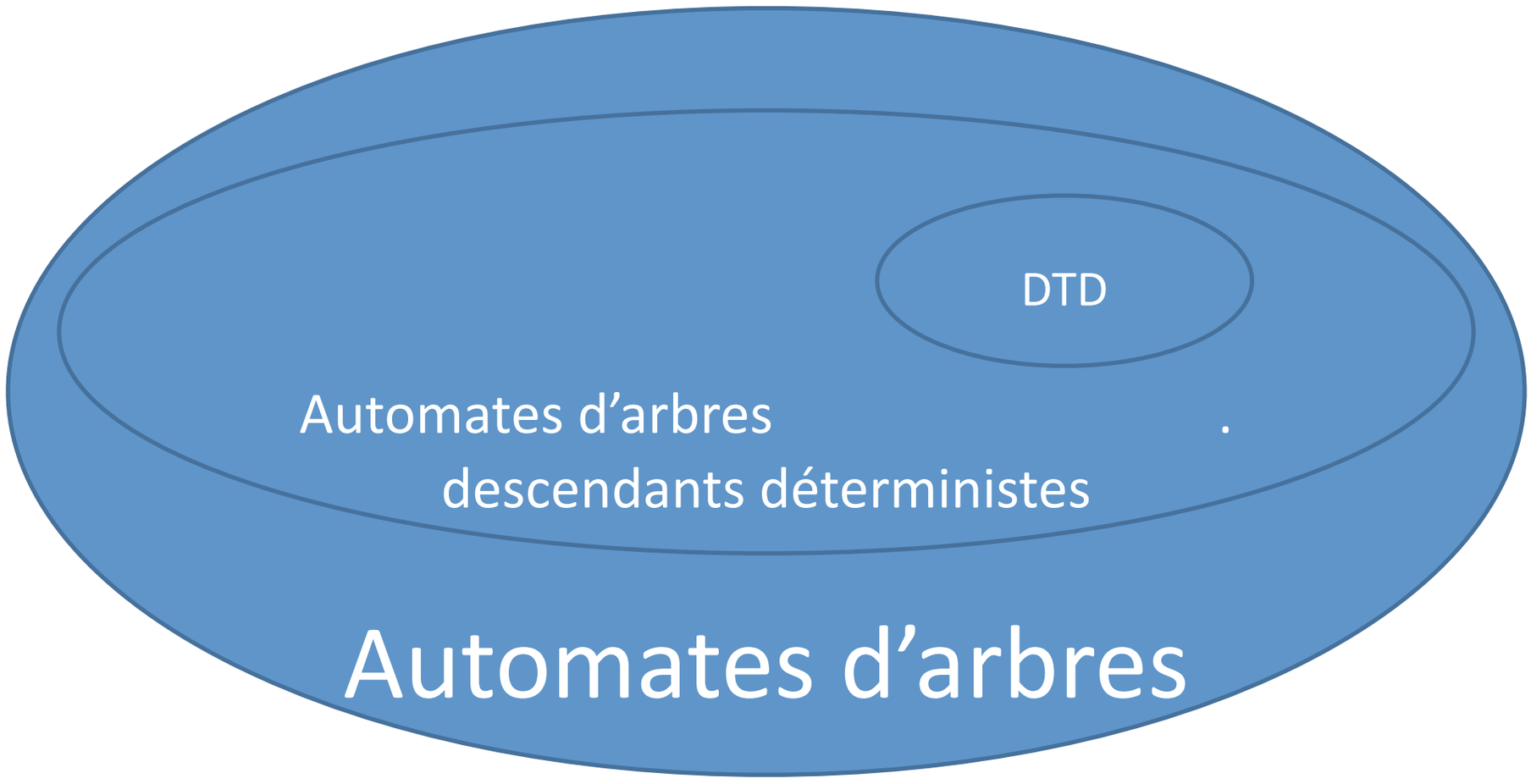
- Important : Les expressions régulières déterministes sont **strictement moins expressives** !!
- Exemple : (a+b)*a(a+b)
- Elles peuvent être traduites en temps polynomial dans des automates finis déterministes
- On sait décider si une expression régulière peut ou non être rendue déterministe (complexité : exponentielle en temps)

En termes d'automates d'arbres

- Une DTD est équivalente à un automate de haies
  - descendant déterministe
  - tel que Q = ∑
  - tel que les expressions régulières utilisées sont déterministes

- En particulier, on obtient que les DTDs sont strictement moins expressives que les langages réguliers d'arbres reconnus par des automates descendants déterministes.

Où situer les DTDs ?



Vérification très efficace

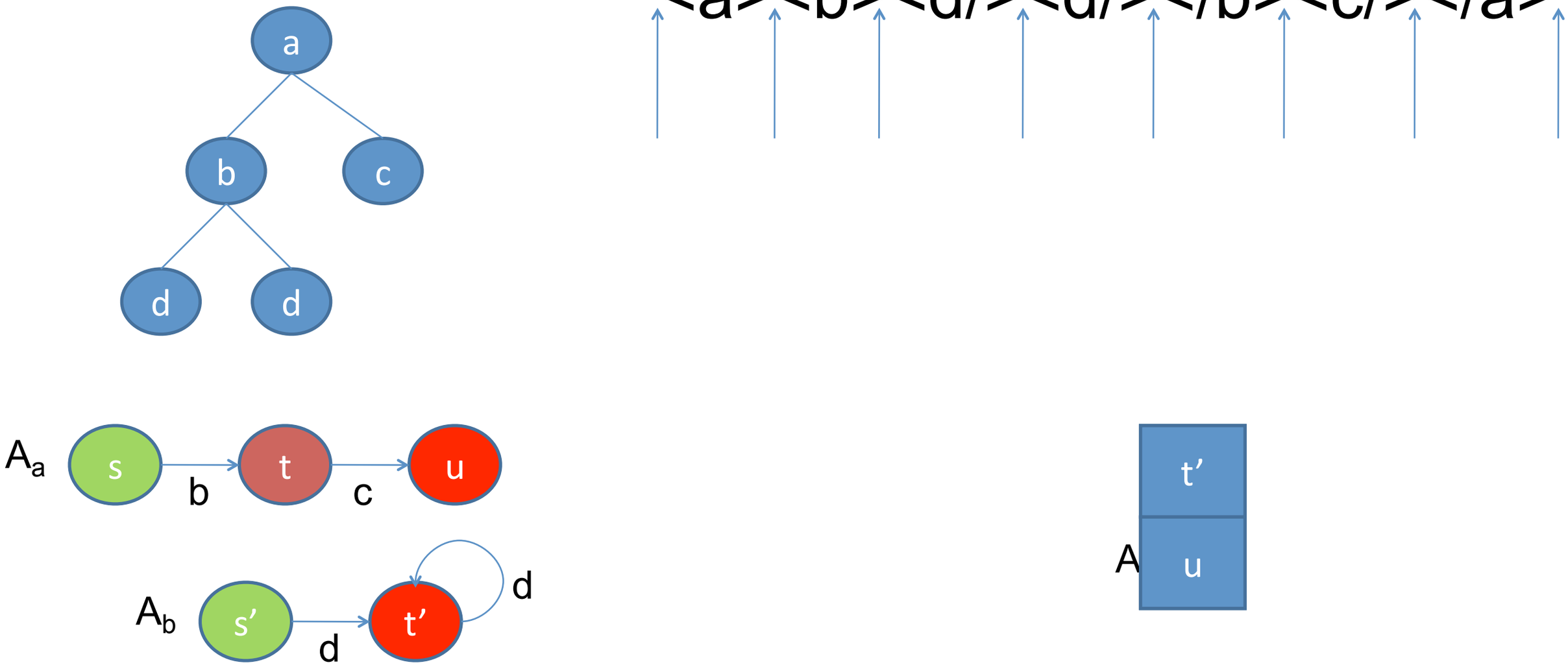
- Il suffit de vérifier pour tout noeud ‘a’ que le mot formé par les étiquettes de ses fils est accepté par un automate fini A_a déterministe

- Problème ! Lors de la lecture d’un document XML, on n’a pas accès en même temps à tous les fils d’un noeud.

- On utilise une pile pour mémoriser les états des automates correspondants à nos ancêtres.

Vérification très efficace (2)

```
<!ELEMENT a ( b c ) >
<!ELEMENT b ( d+ ) >
```



Attention !

- L'exemple précédent pourrait être testé avec un simple automate sur mots
- Mais pas l'exemple suivant :

```
<!ELEMENT part ( part* ) >
```

- Une pile est nécessaire pour vérifier :

<a>...<a>...

Mauvaises nouvelles pour les DTDs

- Pas fermées par union :

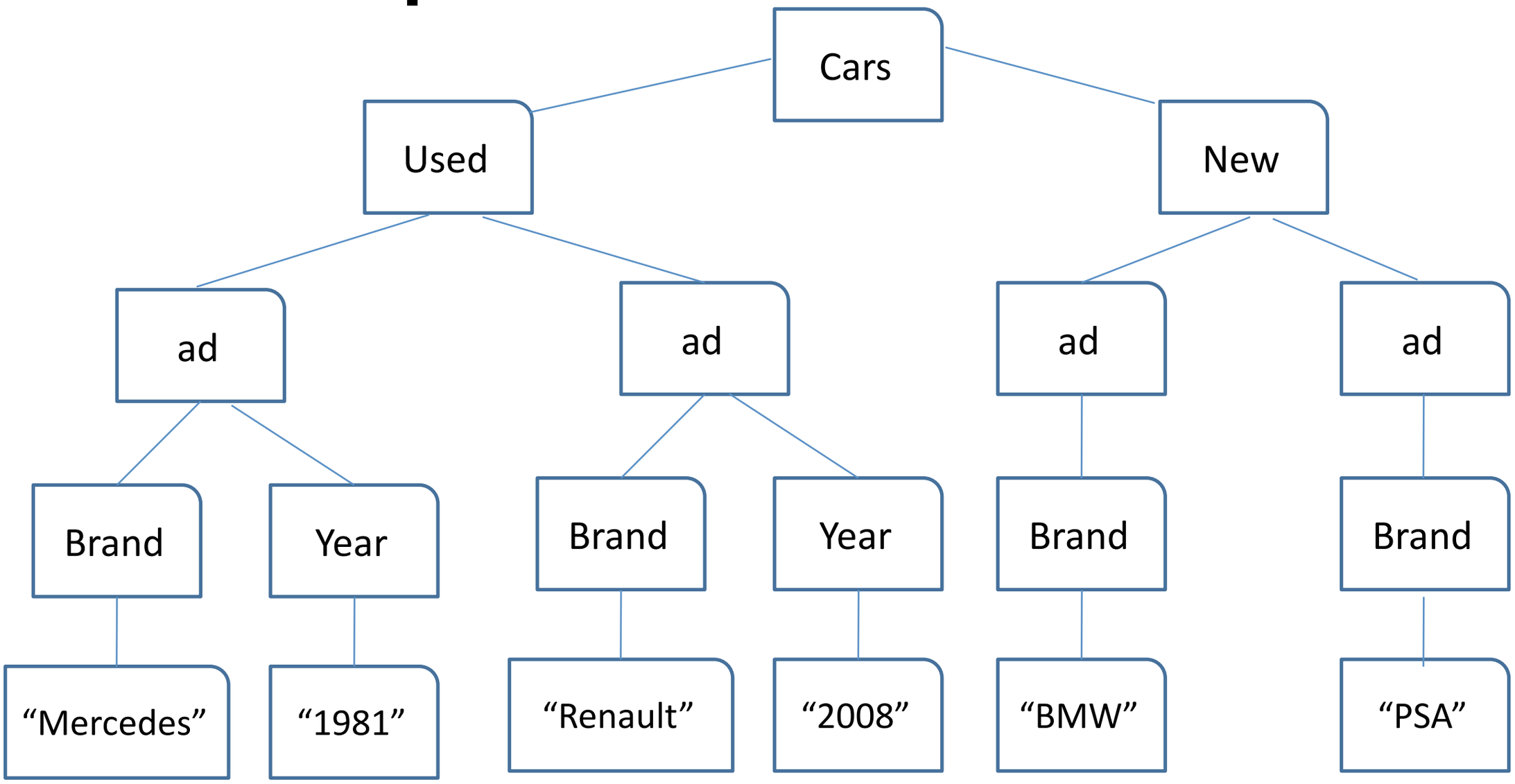
```
DTD1        …
            <!ELEMENT used( ad*) >
            <!ELEMENT ad ( year, brand )>
DTD2        …
            <!ELEMENT new( ad*) >
            <!ELEMENT ad ( brand )>
```

- L(DTD1) ∪ L(DTD2) ne peut pas être décrite par une DTD mais peut facilement être décrite par un automate d'arbres

–Problème avec le type de ad qui dépend de son père

- Pas fermées non plus par complément

Exemple de la voiture



- La meilleure DTD que l'on peut choisir ne distingue pas les éléments ad entre nouvelles et anciennes voitures.
  - `<!ELEMENT ad (year?, brand) >`

2 - Langages de spécifications

- XML Schémas -

XML Schémas

- Souvent critiqué car inutilement compliqué
- Beaucoup utilisé par les Services Web
- Plus riche que les DTDs (types découplés)
- Les Schémas XML sont extensibles
- Beaucoup d'autres fonctionnalités utiles :
  - namespaces
  - types atomiques
  - contraintes d'intégrité, etc.

```
<?xml version="1.0" encoding="utf-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
             targetnamespace="http://www.net-language.com">
    <xs:element name="book">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="title" type="xs:string"/>
          <xs:element name="author" type="xs:string"/>
          <xs:element name="character"
                      minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="name" type="xs:string"/>
                <xs:element name="friend-of" type="xs:string"
                            minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="since" type="xs:date"/>
                <xs:element name="qualification" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="isbn" type="xs:string"/>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

Exemple : une DTD

```
<!ELEMENT EMAIL (TO+, FROM, CC*, BCC*, SUBJECT?, BODY?)>
<!ATTLIST EMAIL
  LANGUAGE (Western|Greek|Latin|Universal) "Western"
  ENCRYPTED CDATA #IMPLIED
  PRIORITY (NORMAL|LOW|HIGH) "NORMAL">
<!ELEMENT TO (#PCDATA)>
<!ELEMENT FROM (#PCDATA)>
<!ELEMENT CC (#PCDATA)>
<!ELEMENT BCC (#PCDATA)>
<!ATTLIST BCC
  HIDDEN CDATA #FIXED "TRUE">
<!ELEMENT SUBJECT (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ENTITY SIGNATURE "Bill">
```

Le même en XML Schéma (verbeux...)

```
<?xml version="1.0" ?>
<Schema name="email" xmlns="urn:schemas-microsoft-com:xml-data"
                     xmlns:dt="urn:schemas-microsoft-com:datatypes">
 <AttributeType name="language"
                dt:type="enumeration" dt:values="Western Greek Latin Universal" />
 <AttributeType name="encrypted" />
 <AttributeType name="priority" dt:type="enumeration" dt:values="NORMAL LOW HIGH" />
 <AttributeType name="hidden" default="true" />
 <ElementType name="to" content="textOnly" />
 <ElementType name="from" content="textOnly" />
 <ElementType name="cc" content="textOnly" />
 <ElementType name="bcc" content="mixed">
<attribute type="hidden" required="yes" />
</ElementType>
<ElementType name="subject" content="textOnly" />
<ElementType name="body" content="textOnly" />
<ElementType name="email" content="eltOnly">
<attribute type="language" default="Western" />
<attribute type="encrypted" />
<attribute type="priority" default="NORMAL" />
<element type="to" minOccurs="1" maxOccurs="*" />
<element type="from" minOccurs="1" maxOccurs="1" />
<element type="cc" minOccurs="0" maxOccurs="*" />
<element type="bcc" minOccurs="0" maxOccurs="*" />
<element type="subject" minOccurs="0" maxOccurs="1" />
<element type="body" minOccurs="0" maxOccurs="1" />
</ElementType>
```

Types découplés en XML Schéma

- Chaque type correspond à un label, pas réciproquement

```
car:        [car]      (used + new)*
used:       [used]     (ad1*)
new:        [new]      (ad2*)
ad1:        [ad]       (year, brand)
ad2:        [ad]       (brand)
```

- Les labels sont en vert, les noms de types en bleu.
- Bonnes propriétés de clôture : résout le problème de l'union des DTDs
- D'autres « gadgets » dans XML schéma

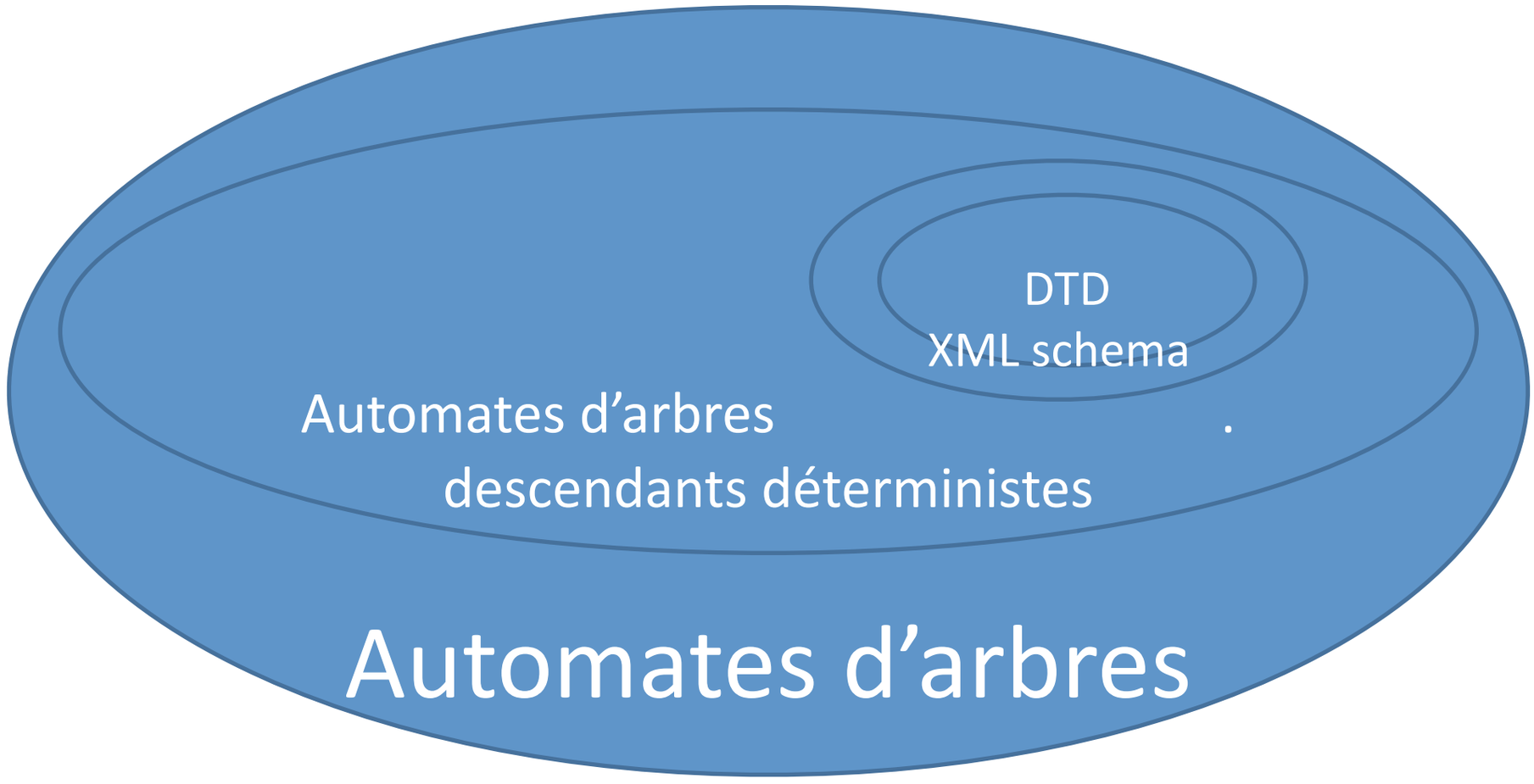
Une restriction sur les types

- “Element Declarations Consistent” :
  des éléments de même nom apparaissant dans le même modèle de contenu doivent avoir le même type.

= limitation du découplage des types...

```
<xsd:complexType name="track">
    <xsd:sequence minOccurs="1" maxOccurs="unbounded">
       <xsd:choice>
          <xsd:element name="session" type="invSession" />
          <xsd:element name="session" type="conSession" />
       </xsd:choice>
        <xsd:element name="break" type="xsd:string" />
    </xsd:sequence>
 </xsd:complexType>
```

Où situer les Schémas XML ?



- Limitation sur les expressions régulières déterministes
- Plus proche des DTDs que des automates d’arbres (types)
- Validation de type efficace (même approche)

Relax NG

- Il existe un troisième type de spécifications appelé Relax NG.
- Ce format est plus général que les schémas XML.
- Les expressions régulières n'ont plus à être déterministes.
- Il existe un opérateur supplémentaire qui induit une complexité dans NP (au lieu de P)