HMIN103
Données du Web

Rendu TD/TP 2

Auteur :

Gracia-Moulis Kévin
Canta Thomas

Master 1 - AIGLE/DECOL
Faculté des sciences de Montpellier
Année universitaire 2020/2021

Table des matières

1. Instance de données

Retrouvé la représentation d'un tweets de part nos précédentes DTD dans les fichiers suivant :

- CANTA_Thomas.xml
- GRACIA_MOULIS_Kevin.xml

2. Synthèse des modèles

Retrouver la synthèse XML et le nouveau DTD associé dans les fichiers suivant :

- merge.xml
- merge.dtd

Voici 3 différences que nous avons pus constater :

- Pour le corps du texte l'un de nous compose le corps d'un message comme étant une répétition de texte ou hashtag ou références utilisateurs.

```
0     <corps>
1         <text> toto </text>
2         <hashtag> #toto </hashtag>
3         <ref-user> @toto </ref-user>
4     </corps>
```

  Tandis que l'autre composé un message comme étant un texte composé de hashtag et/ou références utilisateurs.

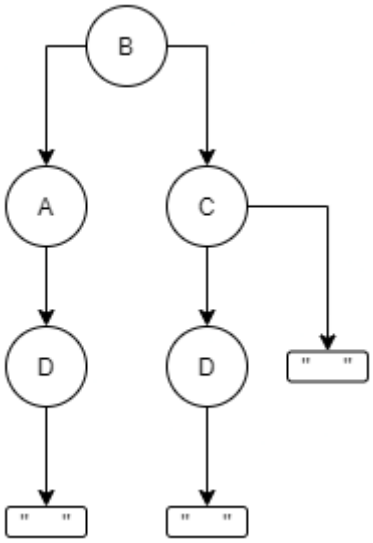
```
0     <text>
1         toto
2         <hashtag> #toto </hashtag>
3         <ref-user> @toto </ref-user>
4     </text>
```

  Nous avons conservé la première solution car c'est celle qui nous semble la plus simple et

- Pour l'attribut du langage, l'un de nous l'associé au tweet et l'autre sur le corps du message, il nous as semblé plus judicieux de l'attribuer uniquement au corps du message.

3. DTD : cas particuliers

DTD 1



DTD 2

DTD 3

Un tel arbre n'existe pas car il est sans fin.

DTD 4

Un tel arbre n'existe pas car la syntaxe est incorrecte. Utilisation de parenthèses à la place des chevrons.

DTD 5

Un tel arbre n'existe pas car la syntaxe est incorrecte. On constate, ligne 2, un crochet fermant la balise du DOCTYPE pendant la définition d'un ELEMENT.

DTD 6

Un tel arbre n'existe pas car la syntaxe est incorrecte. On constate, ligne 2, un crochet fermant la balise du DOCTYPE pendant la définition d'un ELEMENT.

4. Expressions régulières et déterminisme

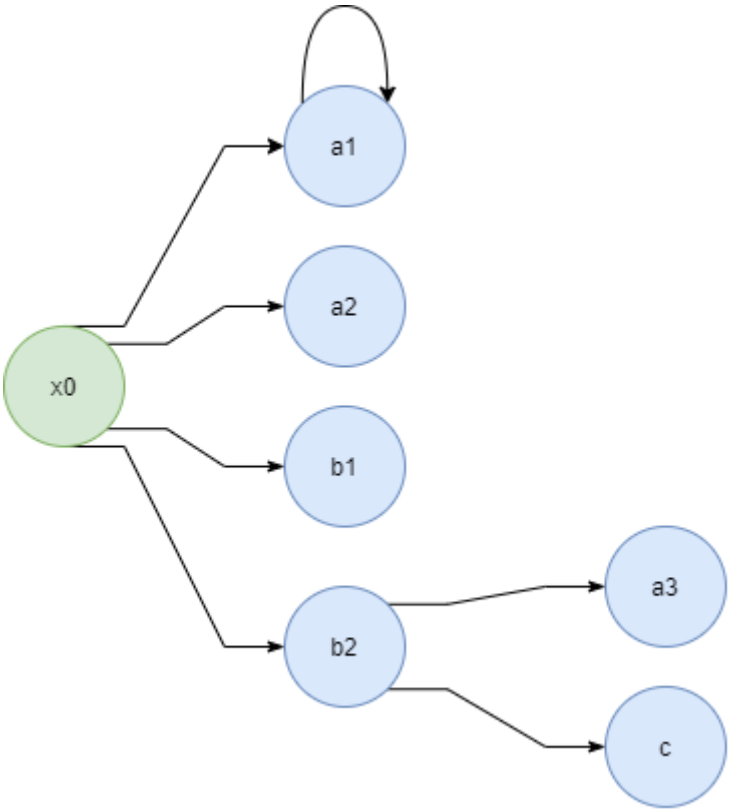
$r_1 = (a^*|(b|a)) \mid (b, (a|c))$

Reformulons notre expression pour supprimer toutes ambiguïtés :

$$r_1^{\#} = (a_1^*|(b_1|a_2)) \mid (b_2, (a_3|c))$$

$$FirstTag(r_1^{\#}) = a_1, a_2, b_1, b_2$$
$$FollowTag(r_1^{\#}, b_2) = a_3, c$$
$$LastTag(r_1^{\#}) = a_1, a_2, a_3, b_1, c$$

Nous pouvons maintenant construire l'arbre :



Ce graphe est déterministe car ...

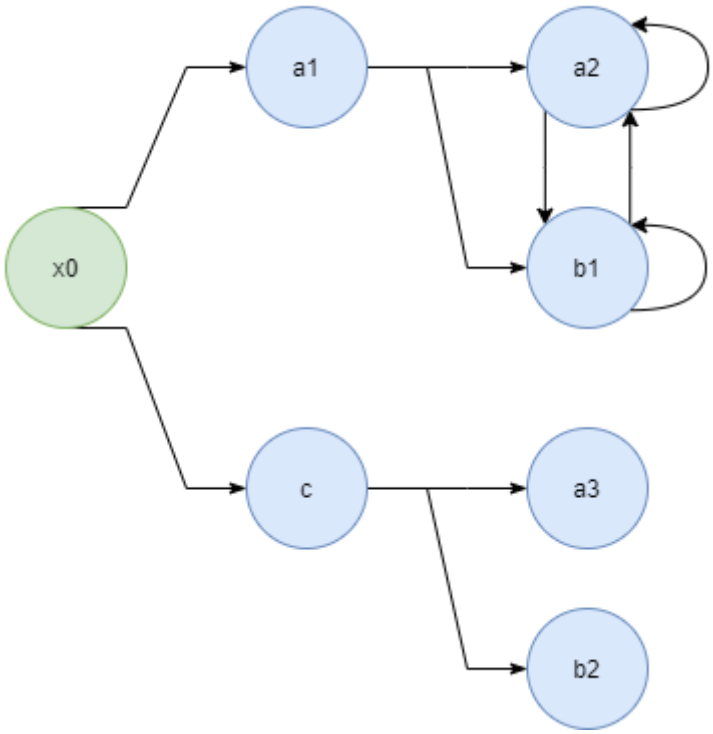
$$r_2 = (a, (a|b)^+) \mid (c, (a|b))$$

Reformulons notre expression pour supprimer toutes ambiguïtés :

$$r_2^{\#} = (a_1, (a_2|b_1)^+) \mid (c, (a_3|b_2))$$

$$FirstTag(r_2^{\#}) = a_1, c$$
$$FollowTag(r_2^{\#}, a_1) = a_2, b_1$$
$$FollowTag(r_2^{\#}, c) = a_3, b_2$$
$$LastTag(r_2^{\#}) = a_2, b_1, a_3, b_2$$

Nous pouvons maintenant construire l'arbre :



Ce graphe est déterministe car ...

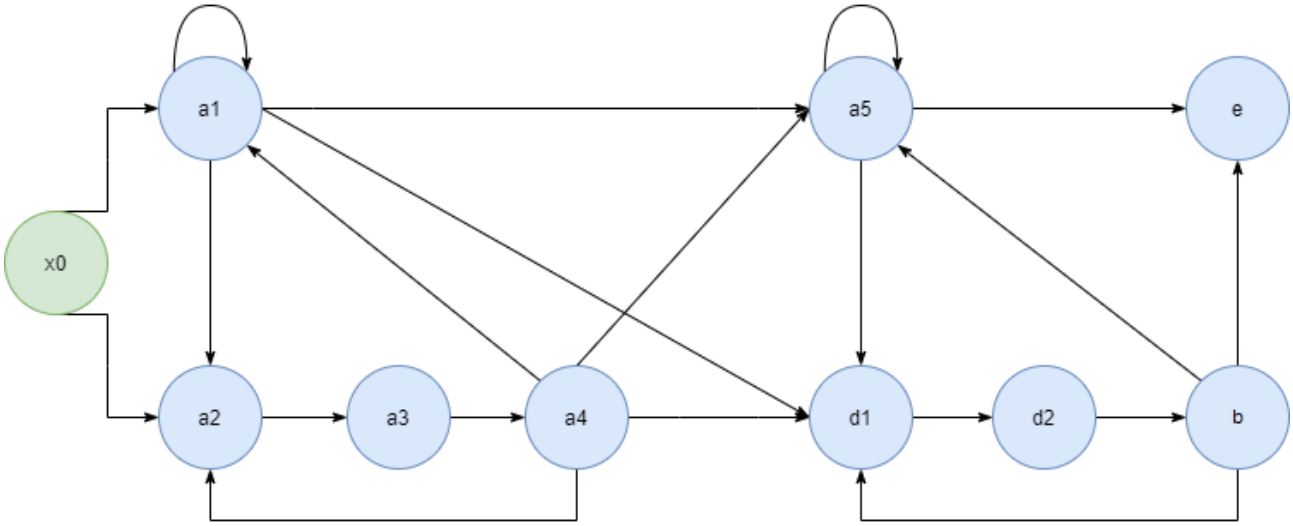
$$r_3 = (a|(a,a,a))^*, (a|(d,d,b))^*, e$$

Reformulons notre expression pour supprimer toutes ambiguïtés :

$$r_3^{\#} = (a_1|(a_2,a_3,a_4))^*, (a_5|(d_1,d_2,b))^*, e$$

$$FirstTag(r_3^{\#}) = a_1, a_5, e$$
$$FollowTag(r_3^{\#}, a_1) = a_2$$
$$FollowTag(r_3^{\#}, a_2) = a_3$$
$$FollowTag(r_3^{\#}, a_3) = a_4$$
$$FollowTag(r_3^{\#}, a_5) = d_1$$
$$FollowTag(r_3^{\#}, d_1) = d_2$$
$$FollowTag(r_3^{\#}, d_2) = b$$
$$LastTag(r_3^{\#}) = e$$

Nous pouvons maintenant construire l'arbre :



Ce graphe n'est pas déterministe car il existe des arêtes....

5. La simplification des expressions régulières

```
0 // On vérifie que r commence par un des symboles la composant
1 If FirstTag(r) = {a1,..,an,ϵ}
2     For i = 1 to n
3         // On vérifie que chaque symbole peut etre suivis par n'importe quel autre
              symbole, lui même compris
4         If FollowsTag(ai) != {a1,..,an,ϵ}
5             return false;
6 Else
7     return false;
8 return true;
```