EISTI **Année scolaire 2009-2010**

DÉPARTEMENT INFORMATIQUE

PÔLE INFORMATIQUE THÉORIQUE

Livrable N^{o}3. **TP de préparation**

I. Structure de fichiers JFLAP

Vous aurez remarqué que sous JFLAP vous ne pouvez testez votre automate que sur des mots séparés qu'il acceptera ou non. Le problème de décodage dans ce projet consiste à parcourir une longue suite **ininterrompue** de bits obtenue par encodage de tout un texte et constituée donc de plusieurs mots du code. Nous avons donc besoin de reconnaître les mots un par un dans l'ordre pour les "traduire" en caractères initiaux et retrouver le texte de départ.

L'objectif de cette étape est d'écrire un programme qui permettrait de décoder une séquence de plusieurs mots à l'aide de l'automate conçu à l'étape précédente. Pour cela vous allez utiliser la structure XML du fichier JFLAP qui représente l'ensemble d'états et de transitions de votre automate et écrire le programme de décodage en XSL.

Un fichier JFLAP décrit un automate d'états fini à l'aide de balises XML. Voici un exemple d'un tel automate tel que généré par JFLAP.

```
     <?xml version="1.0" encoding="UTF-8" standalone="no"?>
     <!--Created with JFLAP 4.0b14.-->
     <structure>&#13;
<type>fa</type>&#13;
<!--The list of states.-->&#13;
<state id="0">&#13;
<x>100.0</x>&#13;
<y>144.0</y>&#13;
<final/>&#13;
</state>&#13;
<state id="1">&#13;
<x>171.0</x>&#13;
<y>50.0</y>&#13;
</state>&#13;
<state id="2">&#13;
<x>238.0</x>&#13;
<y>138.0</y>&#13;
```

```
</state>&#13;
<state id="3">&#13;
<x>254.0</x>&#13;
<y>214.0</y>&#13;
</state>&#13;
<state id="4">&#13;
<x>165.0</x>&#13;
<y>264.0</y>&#13;
<initial/>&#13;
</state>&#13;
<!--The list of transitions.-->&#13;
<transition>&#13;
<from>1</from>&#13;
<to>0</to>&#13;
<read>0</read>&#13;
</transition>&#13;
<transition>&#13;
<from>1</from>&#13;
<to>0</to>&#13;
<read>1</read>&#13;
</transition>&#13;
<transition>&#13;
<from>3</from>&#13;
<to>0</to>&#13;
<read>1</read>&#13;
</transition>&#13;
<transition>&#13;
<from>4</from>&#13;
<to>3</to>&#13;
<read>1</read>&#13;
</transition>&#13;
<transition>&#13;
<from>4</from>&#13;
<to>0</to>&#13;
<read>0</read>&#13;
</transition>&#13;
<transition>&#13;
<from>2</from>&#13;
<to>0</to>&#13;
<read>1</read>&#13;
</transition>&#13;
<transition>&#13;
<from>2</from>&#13;
<to>1</to>&#13;
```

```
<read>0</read>&#13;
</transition>&#13;
<transition>&#13;
<from>3</from>&#13;
<to>2</to>&#13;
<read>0</read>&#13;
</transition>&#13;

</structure>
```

Exercice 1. Représenter sous forme d'arbre la structure de ces données XML

II. Navigation XPATH

L'objectif du livrable est d'écrire un programme XSL qui permet de reproduire le processus de décodage d'une chaîne binaire par cet automate. Pour cela il vous sera nécessaire de pouvoir accéder efficacement aux informations des différents éléments de la structure XML depuis n'importe quel noeud contexte. Les quelques exercices ci-dessous vont vous permettre de tester des solutions XPATH.

Exercice 2. Ecrire une feuille de style XSL qui affiche les indices (l'attribut "id" des éléments "state") de l'état initial et de tous les états finaux.

Exercice 3. Ajouter dans la feuille de style précédente un template nommé qui prend en entrée deux paramètres :

1. `idEtat` , le id d'un état de l'automate
2. `caractere`, un caractère de l'alphabet de l'automate

et qui analyse la possibilité d'effectuer une transition pour le couple (etat, caractère) :

- Si l'état passé en paramètre est un état final, le template affiche "Pas de transition possible : état final"
- Si une transition est possible alors le tempalte affiche le id du nouvel état
- S'il n'y a pas de transition le tempalte affiche "Pas de transition possible, état poubelle"

Appeler le template pour afficher les transitions possibles à partir de l'état initial de l'automate.

III. Décodage : un parcours récursif

Le décodage consiste à lire bit par bit une suite binaire et à effectuer les transitions correspondantes de l'automate. Voici quelques indications importantes pour concevoir l'algorithme :

– Le traitement commence avec une chaîne de caractères que l'on peut passer en paramètre global à la feuille de style ou stocker dans le fichier source XML en y ajoutant une balise
– Au début du traitement l'automate est à son état initial
– Chaque bit lu provoque une transition qui correspond à la valeur du bit et à l'état courent de l'automate. Une fois la transition effectuée, le bit lu est supprimé de la chaîne à traiter
– Si une transition débouche sur l'état final (on se souvient qu'il est unique dans l'automate minimal pour un code sans préfixe) l'automate doit être remis à l'état initial pour poursuivre la lecture de la chaîne
– Chaque séquence de bits lue entre l'état initial et l'état final correspond à un mot du code qu'il faudra retrouver dans la table du code et remplacer par le caractère correspondant (la table de code devra être ajoutée aux donnée XML)

Un parcours correspondant peut être réalisé de manière récursive.

Exercice 4. Ecrire le pseudocode de l'algorithme récursif. Identifier les paramètres en entrée de l'algorithme et les modalités des appels récursifs

---

IV. Travail à réaliser pour le livrable 3

---

1. Modifiez vos données XML, en y ajoutant l'encodage de la table du code. Cela vous permettra de traduire chaque mot reconnu en caractère de l'alphabet initial. Vous devez proposer une XSD pour le fragment XML que vous allez ajouter.
2. Ecrire une transformation XSL qui affiche dans une page HTML :
   (a) La table de code de Huffman
   (b) La séquence codée binaire
   (c) Le résultat de décodage

IV. Á rendre sur AREL pour le livrable 3

Vous devez livrer :
– la XSD des éléments ajouté dans vos donnés XML
– Le fichier Jflap modifié (avec la table de code)
– la feuille de style XSL
– le fichier de sortie HTML obtenu
– un rapport

Le rapport de ce livrable doit présenter en détails :
– l'algorithme de transformation
– les résultats obtenus
– une conclusion globale du projet et analyse critique des résultats obtenus

---