UNIVERSITE LIBRE DE BRUXELLES Le 4 septembre 2006

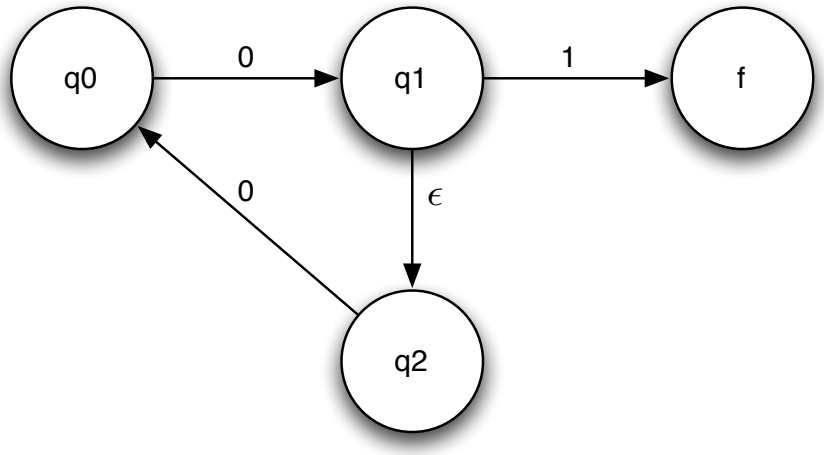
**Théorie des langages et de la compilation**
**Année académique 2005-2006**
**1ère licence en Informatique et**
**2ème année du grade de l'ingénieur civil informaticien**
**Examen de seconde session**

## Remarques préliminaires

- On vous demande de répondre à chaque question sur des feuilles séparées (les correcteurs pouvant être différents).
- N'oubliez pas d'inscrire vos nom et prénom sur chacune des feuilles.
- Vous ne pouvez pas utiliser de notes.
- L'examen comporte cinq questions pour un total de 16 points, les projets (4 points) complétant la note.
- Vous disposez de quatre heures. Pensez à gérer correctement votre temps.
- Si rien n'est spécifié pour les formalismes, les conventions utilisées habituellement dans le cours sont prises.

## Question 1 (5pts) :

Soit $M$, l'automate fini non déterministe avec transitions sur le vide suivant :




**Questions :** construisez
- un automate fini déterministe $M'$,
- une grammaire régulière $G$
- et une expression régulière **r**

qui acceptent tous les trois le même langage que $M$. Utilisez pour ce faire des méthodes générales. Démontrez que ces trois méthodes sont correctes, c'est-à-dire que, pour tout NFA$_\epsilon$ $M$, si $M'$ (resp. $G$, **r**) est le DFA (resp. la grammaire régulière, l'expression régulière) obtenue à partir de $M$ à l'aide de la méthode, alors $M'$ (resp. $G$, **r**) accepte le même langage que $M$.

## Question 2 (4pts) :

On considère la grammaire suivante qui permet de générer des *expressions booléennes* :

$$
\begin{aligned}
E &\rightarrow E \vee E \\
&\rightarrow E \wedge E \\
&\rightarrow E \oplus E \\
&\rightarrow (E) \\
&\rightarrow \neg E \\
&\rightarrow v
\end{aligned}
$$

où les opérateurs $\vee$, $\wedge$, $\oplus$ et $\neg$ représentent respectivement le *ou*, le *et*, le *ou exclusif* et la *négation*. $v$ est un *token* qui représente une variable booléenne. La priorité des opérateurs est fixée comme suit (du plus au moins prioritaire) :

1. () et $\neg$
2. $\wedge$
3. $\vee$ et $\oplus$

**Questions :** Adaptez cette grammaire de manière à ce qu'elle tienne compte de la priorité des opérateurs et de leur associativité, qui est fixée à gauche. Transformez la grammaire obtenue en grammaire *strong* LL(1). Écrivez un analyseur gauche avec un symbole de prévision pour cette grammaire (donnez la table des actions). Exécutez cet analyseur sur la chaîne $\neg a \oplus b \wedge c$.

## Question 3 (4pts) :

Soit la grammaire suivante :

$$
\begin{aligned}
S &\rightarrow T\$ \\
T &\rightarrow TbA \\
&\rightarrow \epsilon \\
A &\rightarrow TC \\
C &\rightarrow c
\end{aligned}
$$

**Questions :** Construisez un analyseur $SLR(1)$ pour cette grammaire, en détaillant les étapes nécessaires, y compris les calculs de Follows et la Table des Actions. Expliquez l'intérêt de cette méthode par rapport à un analyseur $LR(1)$. Enfin, simulez l'analyseur sur la chaîne $bbcc\$$

## Question 4 (3pts) :

En supposant que le programme à compiler contienne les variables $x$ et $y$ réelles, expliquez comment le compilateur, lors de ses différentes phases, traite l'instruction d'assignation $x = y$, pour finalement arriver à générer le code machine correspondant. Dans vos explications, concentrez vous uniquement sur les traitements liés à cette assignation en expliquant les techniques utilisées lors des différentes phases.