Université de Caen **Année 2012/2013**
UFR de Sciences **1ère session**
Jeudi 28 mars

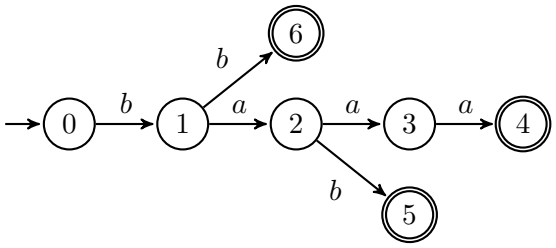
L3
ULI6BT Théorie des langages et compilation
durée 2h
Les notes de cours et TD sont autorisées.

Chaque candidat doit, en début d'épreuve, porter son nom dans le coin de la copie réservé à cet usage; il le cachettera par collage après la signature de la feuille d'émargement. Sur chacune des copies intercalaires, il portera son numéro de place.

Exercice I. Automate Fini

On considère l'automate fini déterministe $\mathcal{A} = (\{a, b\}, \{0, 1, 2, 3, 4, 5, 6\}, \delta, 0, \{4, 5, 6\})$ où δ est décrit par le graphe de transition suivant :

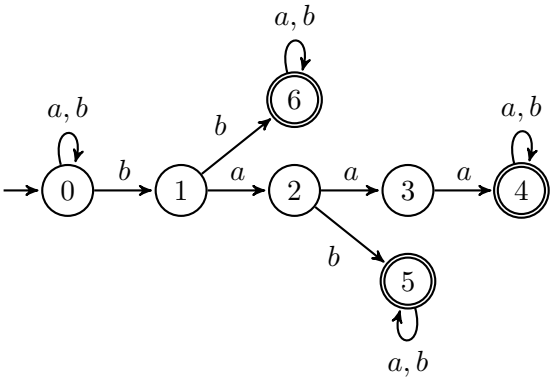
L'automate $\mathcal{A}$



Question 1. Donner une expression régulière du langage $L(\mathcal{A})$ reconnu par l'automate $\mathcal{A}$.

On modifie l'automate $\mathcal{A}$ précédent en un automate $\mathcal{B}$ par l'ajout de boucles étiquetées par les lettres de l'alphabet à la fois sur l'état initial et sur les états d'acceptation.

L'automate $\mathcal{B}$



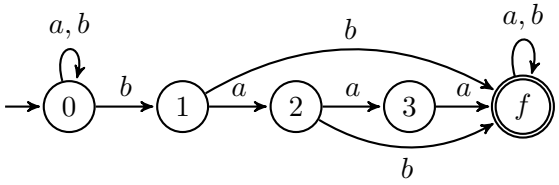
Question 2.

2.a. Pourquoi l'automate $\mathcal{B}$ n'est pas déterministe ?

2.b. Décrire en français le langage reconnu par $\mathcal{B}$. En donner une expression régulière.

Une particularité de l'automate $\mathcal{B}$ est que tous ses états d'acceptation sont absorbants (une fois arrivé dans un de ces états d'acceptation, on y reste). En identifiant tous les états d'acceptation de l'automate $\mathcal{B}$, on obtient un automate équivalent $\mathcal{C}$ avec un unique état d'acceptation absorbant.

L'automate $\mathcal{C}$



Question 3. Construire un automate fini déterministe équivalent à $\mathcal{C}$. (On pourra incidemment identifier tous les états d'acceptation à un même état absorbant.)

Question 4. Construire un automate fini déterministe dont aucun facteur n'est un mot de $L(\mathcal{A})$.

Exercice II. Compilation

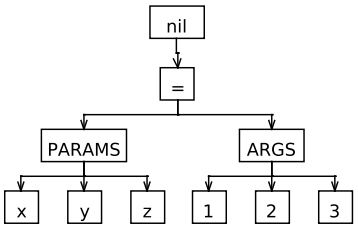
On souhaite enrichir le langage de calculette vu en TD/TP avec les assignations parallèles. On veut ainsi pouvoir affecter simultanément une suite d'expressions à une suite de variables. Par exemple, avec l'assignation `x,y,z = 1,2,3` , la variable `x` (respectivement `y`, `z`) prend la valeur `1` (respectivement `2`, `3`).

Construction de l'AST

On suppose données les règles de la grammaire et les actions de construction de l'AST pour les autres instructions de la calculette. On souhaite que l'analyse des assignations parallèles produisent des AST construits à l'aide de deux nouveaux jetons `PARAMS` et `ARGS` et de la forme suivante.

L'AST obtenu sur l'entrée

```
x,y,z = 1,2,3
```



Question 5.

5.a. Définir les règles de la grammaire pour la prise en compte des assignations parallèles.

5.b. Ajouter les actions nécessaires pour la construction de l'AST.

Production de code MVàP

Question 6. On suppose données `adr(x)`, `adr(y)`, `adr(z)`, les adresses des variables `x,y,z`. Quel code MVàP doit être produit par le compilateur pour effectuer le calcul `x,y,z = 1,2,3` ?

Voici la grammaire d'arbre qui reconnaît la structure d'arbre pour les affectations parallèles :

```
assignation : ^('='  ^(PARAMS params ) ^(ARGS args) ) ;
params : (IDENTIFIANT)*  ;
args : (expression)* ;
```

Question 7. Écrire les règles ANTLR pour la génération de code. Attention à l'ordre : c'est la première expression qui doit être affectée à la première variable et ainsi de suite.
(NB. `tablesSymboles.adresseVar($IDENTIFIANT.text)` récupère l'adresse de l'identifiant.)

Soit le code suivant produit par le compilateur.

```
PUSHI 0
PUSHI 0
PUSHI 0
PUSHI 1
PUSHI 1
PUSHI 0
STOREG 2
STOREG 1
STOREG 0
```

```
LABEL 0
    PUSHG 2
    PUSHI 8
    INF
    JUMPF 1
    PUSHG 1
    PUSHG 0
    PUSHG 1
    ADD
    PUSHG 2
    PUSHI 1
    ADD
    STOREG 2
    STOREG 1
    STOREG 0
    PUSHG 1
    WRITE
    POP
    JUMP 0
LABEL 1
    HALT
```

Question 8. Que réalise ce code ? Identifier les affectations parallèles.

Exercice III. Analyse LL

Soit la grammaire G d'axiome `E` et de terminaux $\{+, *, \texttt{a}\}$ définie par :

$$\begin{cases} \texttt{E} & \to & \texttt{E E OP} \mid \texttt{a} \\ \texttt{OP} & \to & + \mid * \end{cases}$$

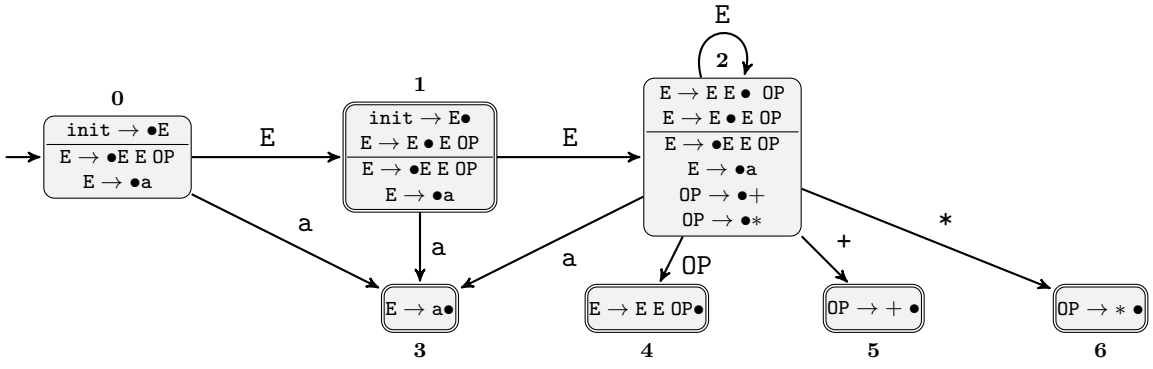
Question 9. Donner un arbre d'analyse pour le mot `a a * a +`.

Question 10. Pourquoi la grammaire G n'est pas $LL(1)$? Quel est le conflit rencontré dans la table d'analyse $LL(1)$ de G ?

Question 11. Proposer une grammaire $LL(1)$ équivalente à G et construire sa table d'analyse pour le prouver.

Exercice IV. Analyse LR

On considère la grammaire G non $LL(1)$ précédente augmentée de la règle `init` $\to$ `E`
L'automate fini caractéristique des items $LR(0)$ de la grammaire est le suivant



Question 12. Quel conflit $LR(0)$ engendre cette grammaire ? Donner un mot où l'analyse $LR(0)$ est non déterministe.

Question 13. Est-ce que une analyse SLR est suffisante pour lever le conflit ? Justifier rigoureusement.