**Contrôle**

**le : 29/03/2021 Durée : 1h**

**Exercice 01 ( 08 pts) :**

1. Les identifiants de fichiers distants se présentent, dans la plupart des cas, comme suit: *user@hostname:filename*

Les parties des identifiants sont constituées de mots, qui sont des séquences d'une ou plusieurs lettres et chiffres. La partie *user* contient un seul mot. La partie *hostname*: se compose d'un ou plusieurs mots séparés par des points (comme dans www.uio.no),. *filename* se compose d'un ou plusieurs mots séparés par des barres obliques (/) avec une barre oblique facultative au début et / ou à la fin. La partie « *user @*» est facultative et peut être omise. La totalité de «*user @ hostname*:» peut également être omise, y compris les deux-points de fin «:». La partie «user @» peut ne pas apparaître mais la partie *hostname*: doit toujours présente.

1. Soit l’alphabet A = {a,b}. Définir les expressions régulières pour les langages ci-dessous :
2. Chaque *a* est suivi d’au moins trois b.
3. Les mots contiennent un nombre impair de a.

**Exercice 02 ( 12 pts) :**

Soit G la grammaire suivante :  
S → Aa | Bb

A → Sa | ɛ

B → Sb| ɛ

1. La grammaire G nécessite-t-elle la factorisation à gauche ? Si oui modifier toutes les règles qui nécessite la factorisation.
2. Est-ce que G contient la récursivité gauche ? Si oui éliminer tous les cas récursivité gauche.
3. Construire la table d’analyse de la grammaire.
4. La grammaire résultante est-elle de type LL(1) ? Justifier.

**Interrogation (30 minutes) :**

**Exercice1**

Eliminer la récursivité à gauche de la grammaire suivante :

S **→**  T a | S a S b

T **→** S b| T b| b

**Exercice2**

Soit la grammaire suivante, d’axiome S sur l'alphabet terminal A = {a, b, c,d } et non terminal {S, X,Y, W }

S **→**Xa | **ε**

X **→**bX | YW

Y **→** aX | **ε**

W**→** c | d

1. Construire la table d’analyse LL(1).
2. La grammaire est-elle LL(1) ? Justifier votre réponse.
3. Simuler l’analyse prédictive de : abcda$
4. Construire l’arbre de dérivation correspondant.

**Corrigé type**

**Fait le : 14/01/2020 Durée : 1h30m**

**Exercice 01 ( 08 pts) :**

1. *word* = [*a* - *zA* - *Z*0 - 9]**+**

((*word* **@**)**?***word*(***\.****word*)∗ **∶**)**?**(**/**)**?***word*(**/***word*)∗(**/**)**? (05 pts)**

1. Soit l’alphabet A = {a,b}. Définir les expressions régulières pour les langages ci-dessous :
2. Chaque *a* est suivi d’au moins trois b. b\*(abbbb\*)\* **(01.5 pts)**

Les mots contiennent un nombre impair de a. b\*ab\*(ab\*ab\*)\* **(01.5 pts)**

**Exercice 2 ( 08 pts) :**

Soit G la grammaire suivante :  
S → Aa | Bb

A → Sa | ɛ

B → Sb| ɛ

1. La grammaire G nécessite-t-elle la factorisation à gauche ? oui

La modifier des règles qui nécessite la factorisation.

S → Aa | Bb on a premier= (S)= premier (A)⋃premier (B)

mais

premier (A)∩ premier (B)!={∮}

Alors

Il faire un genre de factorisation de la partie droite la règle (1)

S → Aa | Bb🡺 S → (Sa | ɛ)a | (Sb| ɛ)b🡺 S → Saa | a | Sbb|b🡺 S → Saa | Sbb| a |b

🡺 S → S(aa | bb)| a |b

Alors G: sera

S → SS’|a |b

S’ → aa | bb

Les non terminaux A et B seront supprimés puisque ne seront plus accessible

1. Est-ce que G contient la récursivité gauche ? oui.

S → SS’|a |b🡺

S’ → aa | bb

1. Construire la table d’analyse de la grammaire.
2. La grammaire résultante est-elle de type LL(1) ? Justifier.

**Contrôle**

**le : 19/05/2021 Durée : 1h**

**Exercice 01 (08 pts)**

Ecrire les expressions régulières acceptant les langages suivants :

1. Un langage L défini sur l’alphabet {A, B, C, …, Z} est composé de mots présentant les caractéristiques suivantes :

* Les mots se commencent par une consonne et se terminent par une voyelle.
* Les consonnes sont séparées par des voyelles (pas de consonnes consécutives).
* Plusieurs voyelles (au maximum 3) peuvent se suivre.
* La longueur d’un mot est comprise entre 2 et 8 inclus.

2. L’ensemble des chaînes composées de lettres, de chiffres et de tirets. Elles ne peuvent ni commencer, ni finir par un tiret et ne contiennent pas deux tirets consécutifs.

3. L’ensemble de toutes les chaînes de 0 et de 1 telles que chaque 0 soit immédiatement suivi par au moins un 1.

**Exercice 02 (12 pts)**

Soit la grammaire G :

S ⟶ Aa | SaSb | ɛ  
A ⟶Ba | Sb  
B ⟶aBb | Bba | a

1. G-est elle LL(1) ? Pourquoi ?.
2. S’il est possible, transformez-la pour qu’elle devienne LL(1) ? Justifier

**Rattrapage**

**Le : 07/06/2021 Durée : 1h**

**Exercice 01 (08 pts) :**

1. Dans une formule mathématique d'un document LaTeX, un indice est indiqué par le caractère \_ ; par exemple x\_i désigne xi ; si l'indice n'est pas réduit à un seul caractère, il faut employer des accolades : x\_{i+1} désigne xi+1 ; la notation x\_{i} est bien sûr autorisée.

Dans un document HTML, un indice est indiqué par la balise <SUB> obligatoirement suivie de la balise fermante </SUB> ; ainsi xi s'écrit x<SUB>i</SUB> et xi+1 s'écrit x<SUB>i+1</SUB> .

Ecrire un analyseur lexical qui transforme les formules LaTeX en formules HTML.

1. Considérons A= {0*,* 1}. Pour les langages suivants, donner une ERE :

* Tous les mots non vides qui contiennent un nombre de 0 multiple de 3.
* Tous les mots tels que chaque bloc de 3 symboles consécutifs contiennent (exactement) 2 occurrences du symbole 0.
* L’ensemble de toutes les chaînes de 0 et de 1 telles que chaque 0 soit immédiatement suivi par au moins un 1.

**Exercice 02 (12 pts):**

Soit la grammaire des expressions post-fixées, G, d’axiome E :

E → E E O | n

O → + | - | \* | /

1. Vérifier les conditions LL(1).
2. G est-elle LL(1) ? (Justifiez) Si oui, effectuez l’analyse descendante de 3 5 + 4 \* en montrant les étapes de la construction de l’arbre d’analyse syntaxique. Sinon la transformer pour qu'elle le soit et analyser le mot.

**Corrigé type de Rattrapage**

**Le : 07/06/2021 Durée : 1h**

**Exercice 01 (08 pts) :**

1. Dans une formule mathématique d'un document LaTeX, un indice est indiqué par le caractère \_ ; par exemple x\_i désigne xi ; si l'indice n'est pas réduit à un seul caractère, il faut employer des accolades : x\_{i+1} désigne xi+1 ; la notation x\_{i} est bien sûr autorisée.

Dans un document HTML, un indice est indiqué par la balise <SUB> obligatoirement suivie de la balise fermante </SUB> ; ainsi xi s'écrit x<SUB>i</SUB> et xi+1 s'écrit x<SUB>i+1</SUB> .

Ecrire un analyseur lexical qui transforme les formules LaTeX en formules HTML. **(02+02 pts)**

1. Considérons A= {0*,* 1}. Pour les langages suivants, donner une ERE:

* Tous les mots non vides qui contiennent un nombre de 0 multiple de 3.

**(1\*01\*01\*01\*)(1\*01\*01\*01\*)\***  **(01.5 pts)**

* Chaque bloc de 3 symboles consécutifs contiennent 2 occurrences du symbole 0 : **(001|010|100)\*(0|1)?(0|1)? (01.5 pts)**
* Chaque 0 soit immédiatement suivi par au moins un 1 : **((01)\*1\*)\* (01 pts)**

**Exercice 02 (12 pts):**

Soit la grammaire des expressions post-fixées, G, d’axiome E :

E → E E O | n

O → + | - | \* | /

1. Vérifier les conditions LL(1). G est-elle LL(1) ?Non **(0.5 pts)**

G est recursive à gauche: E → E E O | n **(01 pts)**

G ne nécessite pas la factorisation.

1. Elimination de la récursivité à gauche. **(1.5 pts)**

E → E E O | n 🡺 E → nE ‘

E’ → EOE’|ɛ

|  |  |  |
| --- | --- | --- |
| **(2 pts)** | Premiers | Suivants |
| E | n | $ + , - ,\* , / |
| E’ | n, ɛ | $ + , - ,\* , / |
| O | + , - ,\* , / | n, $ + , - ,\* , / |

**La table d’analyse ;**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **(03 pts)** | n | + | - | \* | / | $ |
| E | nE’ |  |  |  |  |  |
| E’ | EOE’ | ɛ | ɛ | ɛ | ɛ | ɛ |
| O |  | + | - | \* | / |  |

L’analyse descendante de 3 5 + 4 \* en montrant les étapes de la construction de l’arbre d’analyse syntaxique **(03+02pts)**

|  |  |  |  |
| --- | --- | --- | --- |
| PILE | Entrée | Action | Sortie |
| $ *E* | 3 5 + 4 \*$ | dépiler, empiler E’n | E → nE ‘ |
| $E’n | 3 5 + 4 \*$ | dépiler, avancer |  |
| $E’ | 5 + 4 \*$ | dépiler, empiler EOE ‘ | E’ → EOE’ |
| $E’OE | 5 + 4 \*$ | dépiler, empiler E’n | E → nE ‘ |
| $E’OE’n | 5 + 4 \*$ | dépiler, avancer |  |
| $E'OE’ | + 4 \*$ | dépiler, empiler ɛ | E’ → ɛ |
| $E'O | + 4 \*$ | dépiler, empiler + | E’ → + |
| $E'+ | + 4 \*$ | dépiler, avancer |  |
| $E' | 4 \*$ | dépiler, empiler E’n | E’ → EOE’ |
| $E’OE | 4 \*$ | dépiler, empiler E’n | E → nE ‘ |
| $E’OE’n | 4\*$ | dépiler, avancer |  |
| $E’OE’ | \*$ | dépiler, empiler ɛ |  |
| $E'O | \*$ | dépiler, empiler \* | E’ → \* |
| $E'\* | \*$ | dépiler, avancer |  |
| $E' | $ | dépiler, empiler ɛ | E’ → ɛ |
| $' | $ | Accepter |  |

**Contrôle**

**le : 15/09/2021 Durée : 1h**

**Exercice 01(08 pts) :**Ecrire les expressions régulières acceptant les langages suivants :

1. Dans une formule mathématique d'un document LaTeX, un indice est indiqué par le caractère \_ ; par exemple x\_i désigne *xi* ; si l'indice n'est pas réduit à un seul caractère, il faut employer des accolades : x\_{i+1} désigne *xi*+1 ; la notation x\_{i} est bien sûr autorisée.

Dans un document HTML, un indice est indiqué par la balise <SUB> obligatoirement suivie de la balise fermante </SUB> ; ainsi *xi* s'écrit x<SUB>i</SUB> et *xi*+1 s'écrit x<SUB>i+1</SUB> .

Ecrire un analyseur lexical qui transforme les formules LaTeX en formules HTML.

1. L’ensemble des chaînes composées de lettres, de chiffres et de tirets. Elles ne peuvent ni commencer, ni finir par un tiret et ne contiennent pas deux tirets consécutifs.
2. L’ensemble de toutes les chaînes de 0 et de 1 telles que chaque 0 soit immédiatement suivi par au plus trois 1.
3. Tous les mots non vides ne contenant pas plus que deux *0* consécutifs

**Exercice 02 (12 pts)**

Soit la grammaire G :

S⟶Aa| Bb

A⟶CB|Bb|ɛ

B⟶b

C⟶c|ɛ

1. G-est elle LL(1) ? Pourquoi ?.
2. Transformez-la pour qu’elle devienne LL(1) ? Justifier.
3. Analyser la chaine « cbbab$»