SEG 2506 – Laboratoire 3

Par

Philippe Dore

4468472

Présenté à Robin Tropper

Dans le cadre du cours

SEG 2506

Enseigné par

M. Gregor V. Bochmann

Merci, mais toujours présenté au chef du projet : ici, le prof. - aucune pénalité.

Le 13 Février 2008

Laboratoire 3 : barême de correction

- **Partie 1:**
  total 18/25 (2points bonus)
  - Expressions régulières
    - 5/7 (1 point chacun)
  - Automates
    - 8/10 (2 points chacun)
  - Analyseur en Java
    - 10/10 => 5/10
- **Partie 2:**
  total 22/25 => 16/25
  - Introduction (analyseur des adresses de courriel)
    3/3
  - Description de l'expression régulière retenue
    5/10 => 2.5
  - Programme Lex fonctionne tel que spécifié dans l'énoncé
    7/7 => 3.5
  - Problèmes rencontrés et leçons apprises
    5/5
  - Conclusion
    2/2

Bonus : remis à la première échéance : 5% => 2.5 points.

Grand total

34/50 = 68%

Code-source Java et Lex
identiques à D.Grisé
(à quelques exceptions
néglieagles près)
-50% de ces parites
politiques sur le plagiat
de l'université

SEG 2506 – Laboratoire 3

Partie 1- Expressions Régulières et automates :

Expressions Régulières :

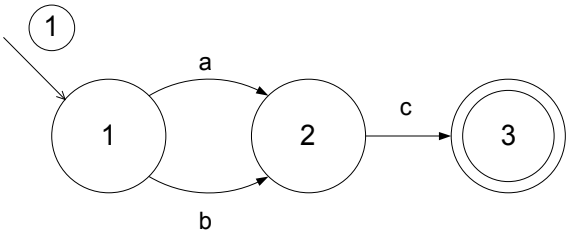
1. aa(a|b)*
2. (a|b)*aa(a|b)*
3. ((a|b)(a|b))*
4. ((ab*a)|b)*

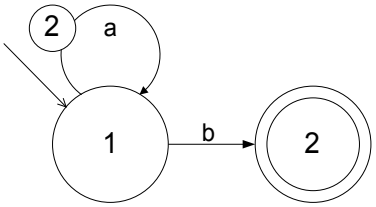
A peut ne pas se montrer du tout avec ceci.

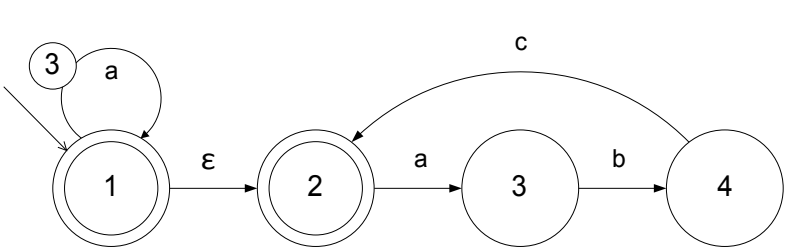
5. ((a|b)(a|b)(a|b)(a|b)(a|b))*
6. (a|b)*aaa(a|b)*
7. ((b*aa(b*))|(b*a(b*)))

Trop restrictif : voir solutions.

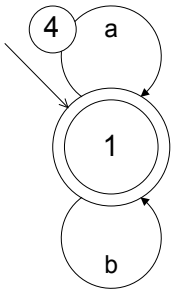
Automates:

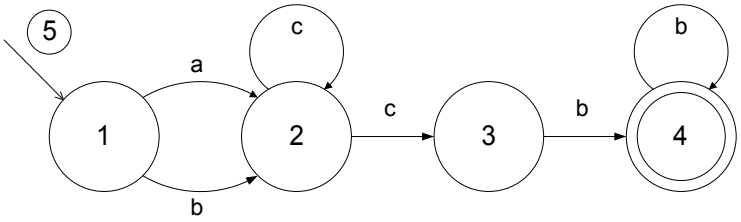






Pas d'epsilon dans les vraies MÉF.





Programme Java: (Voir RegExp.java)

Partie 2- Introduction à l'outil LEX :

Introduction (analyseur des adresses de courriel) :

L'outil « lex » est très utile dans le domaine des expressions régulières. Dans cette section du laboratoire, nous tentons d'éliminer des adresses e-mail invalide d'une liste d'adresses en implémentant une expression régulière pour modéliser la situation. Dans notre cas nous analysons les adresses d'une façon simpliste.

Description de l'expression régulière retenue:

[a-zA-Z]+[a-zA-Z0-9\-_]*\.{0,1}[a-zA-Z0-9\-_]+\@([a-zA-Z]+([a-zA-Z0-9\-_]+\.){1,2}[a-z]{2,3}|

[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3})

D'après notre expression régulière, une adresse valide se défini comme :

- Toutes adresses doivent commencer par au moins un caractère alphabétique.
- Si un point est a la gauche du '@', il doit être précéder et suivi par au moins un caractère alphabétique.

Quelles hypothèses sur les '.com' etc.?

- Toutes adresses doivent contenir un et seulement un '@'.
- Le caractère immédiatement suivant le '@' doit soit être un caractère alphabétique ou numérique.
- Dans le cas ou le caractère est un chiffre, il doit avoir un maximum de 3 chiffres ensuite un point et au plus trois autre suivi d'un autre point et finissant par un total de 3 chiffres au plus.

gauche/droite du @ :
nom d'usager & serveur courriel
=> comment les définis-tu?

Description de la spécification pour Lex :

Notre programme en « lex » compte une variable « VALID » qui identifie si l'adresse est valide. Donc si l'adresse satisfait l'expression régulière utilisée, elle est dite « VALID » donc elle est affichée.

En implémentant notre expression en « lex » nous avons utilisé les '[ ]' pour définir une classe de caractères. Le '+' nous dit qu'il faut y avoir au moins un caractère de cette classe et le '*' veut dire que la classe peut apparaitre aucune ou plusieurs fois. Ensuite le '\' signifie que le caractère suivant doit apparaitre. Et finalement nous avons fait usage des '{ }' pour définir le nombre possible d'occurrences de la classe concernée.

Problèmes rencontrés et leçons apprises:

Durant le cours de ce laboratoire, beaucoup d'obstacles ont su ralentir mon cheminement. Pour débuter, ce laboratoire était ma première expérience avec l'environnement « linux » alors il fallait que je m'adapte et que j'expérimente avec ce nouveau système. Ensuite, pour les exercices pratiques tels « Parse1 », « Parse2 », il fallait utiliser la console, mais je tentais de compiler les fichiers tout en étant dans une console du « desktop » donc j'avais tout le temps le message que le fichier était introuvable. Cela a été très facile à régler lorsque j'ai réalisé qu'il fallait ouvrir la console du fichier courant.

De plus, lorsqu'il était le temps de concevoir un algorithme qui détecte les adresses courriels non-valides, je croyais qu'il fallait le faire en C. Donc j'ai tenté de modifier le fichier « Parse5 » qui était en C. J'ai vite réalisé qu'il y avait une façon beaucoup plus simple pour faire cette tache en utilisant les expressions régulières. En faisant des modifications au fichier « Parse3 », j'ai réussi à implémenter le logiciel de manière beaucoup plus facile et élégante en format « lex ».

Conclusion:

Au cours de ce laboratoire, j'ai été initié au système d'exploitation « linux » et j'ai su comment appliquer mes connaissances en matière des expressions régulières en classe pour implémenter un programme très simple de vérification d'adresses courriels. Ce laboratoire a été très utile pour illustrer les différentes fonctionnalités utiles de « lex » dans « linux » et aussi pour démontrer les applications pratiques de la matière apprise en salle de classe.

Normalement, dans une conclusion de rapport scientifique, on résume les résultats (expres. rég. obtenue) et on commente sur le succès selon les buts de l'exercice.