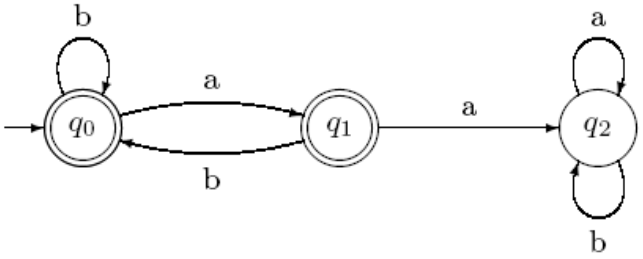
2-1 Soit A un alphabet de n symboles. On rappelle que le cardinal représente le nombre d'éléments d'un ensemble. Ici card(A)=n
Que représentent A^0 , A^1 , card(A^0) , card(A^1) , card(∅), card(A^2) , card(A^3), card(A^P) , card(A*), card(A^k*) (tous les mots de longueur ≤ k)
Application pour A={a,b,c} donnez les valeurs ci-dessus donnez aussi A^2 et A^3 , A^3* et A^7*

2-2: Un automate qui fonctionne comme un an.lex.
Q 1.1 : Donner un AFD qui reconnaît dans une chaîne de caractères les symboles :
– les opérateurs + et - ;
– les identificateurs `a la Java composes uniquement de chiffres et de lettres (exemple : ps2pdf) ;
– les entiers non signes (exemple : 209) ;
– un sous-ensemble des réels comportant syntaxiquement une suite de chiffres suivis de la lettre e suivis d'une suite de chiffres éventuellement signes (exemple : 12e-3, 4e26).
Noter l'absence de tout séparateur.
Q 1.2 : Donner pour chaque symbole une description régulière qui le définit.
Q 1.3 : Comment l'analyseur lexical décompose-t-il les chaînes suivantes ?
– + −;
– aa + 9 ;
– a9e + 9 ;
– 9e + a.
Q 1. 4 : Reprendre la question précédente en cherchant combien de caractères l'analyseur lexical doit lire après la fin d'un symbole avant de déclarer qu'il a reconnu ce symbole. Peut-on borner ce nombre ?
2-3 : Vive l'automatisation
Le but de cet exercice est de vous sensibiliser aux avantages de la génération de code comme solution aux taches répétitives et fastidieuses. On considère un langage de programmation contenant les mots-clé do, od et les identificateurs à la Java.
Q1 : Donner un AFD qui effectue l'analyse lexicale de ces trois classes de symboles.
Q 2 : Vous avez implante à la main l'analyseur lexical `a partir de l'AFD obtenu. Votre chef est content, vous aussi. Mais votre chef décide que tout compte fait, done serait un mot-clé plus agréable que od ! Recommencez. . .
2-4 : Un langage de commande
On souhaite écrire un analyseur lexical pour le langage de commandes suivant :
– une commande est composée d'un nom de commande, suivi d'une liste optionnelle d'arguments, suivie d'une liste facultative d'options ;
– une liste d'arguments est une suite d'arguments ;
– une liste d'options est une suite non vide d'options encadrée par [et], `a l'intérieur de laquelle les options sont séparées par , ;
– une option est un caractère précède d'un tiret ;
– un argument est un identificateur, de même qu'un nom de commande.
Par exemple, macom arg1 arg2 [-a,-b] est une commande, ainsi que macom [-f] et macom.
Q1 : Quelles sont les unités lexicales nécessairesà la description d'une commande ? Donner pour chacune d'entre elles une description régulière qui la définit.

2-5 (Expressions régulières)
Définir les expressions régulières suivantes :
1. Nombres binaires multiples de 2 ;
2. Nombres binaires multiples de 4 ;

3. Chaînes de caractères sur {a, b} contenant 2 “a” consécutifs ;
4. Chaînes de caractères sur {a, b} ne contenant pas 2 “a” consécutifs ;
5. Chaînes sur {a, b, c} où le premier “a” précède le premier “b”.

2-6 Q1 : Décrire formellement l’automate suivant :

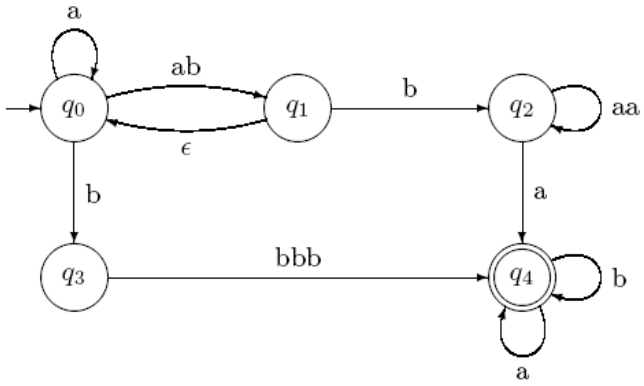


Q2 Cet automate est-il déterministe ?

Q3 Quel langage accepte-t-il ?

Q4 Écrire une fonction en C ou java pour simuler cet automate.

2-7 Quel est le langage accepté par :



2-8 (Construction d’automates)
Construisez des automates qui reconnaissent les langages suivants :
– (aa*)|bb*
– (a|b)*abb

2-9 Transformation d’AFN en AFD et minimisation
Q1 Construire un automate qui reconnaît le langage défini à partir de l’expression régulière suivante : (a|abb|a*b$^+$)
Q2 Transformez l’automate fini non déterministe (AFN) en automate fini déterministe (AFD).
Q3 Minimisez l’automate fini déterministe obtenu.

2-10 Transformation d’AFN en AFD et minimisation)
Q1 Construire l’automate fini non déterministe pour (a|b)*abb
Q2 Écrire une fonction C ou java le simulant.
Q3 Exécuter cette fonction pour le mot “abbab”.
Q4 Transformer l’AFN en AFD.

2-11 (De l’expression régulière à l’AFD)
Trouver une expression régulière dont le langage est l’ensemble des mots sur {a, . . . , z} et qui contient un “a”,
un “e”, un “i ” dans cet ordre et qui se termine par “ou”.
Par exemple : ebdaiecaidou
En déduire les automates associés : AFN et AFD.