TD N°0

Analyse lexicale

Exercice 1:

Dans ce qui suit, une chaîne de chiffres est une séquence non vide de chiffres décimaux, c'est-à-dire quelque chose dans le langage défini par l'expression régulière [0-9]+. La valeur d'une chaîne de chiffres est l'interprétation habituelle d'une chaîne de chiffres comme un nombre entier. Notez que les zéros non significatifs sont autorisés.

Créez pour chacun des langages suivants une expression régulière qui décrit ce langage.

a) Toutes les chaînes de chiffres qui ont la valeur 42.
b) Toutes les chaînes de nombres qui n'ont pas la valeur 42.
c) Toutes les chaînes de nombres dont la valeur est strictement supérieure à 42.

Exercice 2:

Étant donné l'expression régulière a $*$(a|b)aa :

a) Construire un NFA équivalent en utilisant la méthode donnée dans l'annexe.
b) Convertir ce NFA en un DFA en utilisant l'algorithme donné dans l'annexe.

Exercice 3:

Construire un DFA qui reconnaît des séquences équilibrées de parenthèses avec une profondeur maximale d'imbrication de 3. Par exemple ε, ()(), (()()) ou (()()()()) mais pas ((((())))) ou (()((())()).

Exercice 4:

Construire des DFA pour chacun des langages réguliers suivants. Dans tous les cas, l'alphabet est {a, b}.

a) L'ensemble des chaînes de caractères qui ont exactement trois **b**, et un nombre quelconque de **a**.
b) L'ensemble des chaînes de caractères dont le nombre de **b** est un multiple de trois, et il peut y avoir un nombre quelconque de **a**.
c) L'ensemble des chaînes de caractères dont la différence entre le nombre de **a** et le nombre de **b** est un multiple de trois.

Annexe

Automates finis non déterministes

Nous construirons un NFA (Nondeterministic finite automaten) par composition à partir d'une expression régulière, c'est-à-dire que nous construirons le NFA d'une expression régulière composite à partir des NFAs construits à partir de ses sous-expressions. Pour être précis, nous construirons un fragment NFA à partir de chaque sous-expression, puis nous combinerons ces fragments en fragments plus grands. Un fragment n'est pas un NFA complet, nous complétons donc la construction en ajoutant les composants nécessaires pour obtenir un NFA complet.

Un fragment de NFA est constitué d'un certain nombre d'états avec des transitions entre ceux-ci et en plus deux transitions incomplètes : Une qui pointe vers le fragment et une qui pointe vers la sortie du fragment. La demi-transition entrante n'est pas étiquetée par un symbole, mais la demi-transition sortante est étiquetée soit par ε soit par un symbole de l'alphabet. Ces demi-transitions constituent l'entrée et la sortie du fragment et sont utilisées pour le connecter à d'autres fragments ou à des états supplémentaires "collés". La construction de fragments NFA pour les expressions régulières est illustrée à la figure 1. La construction suit la structure de l'expression régulière en créant d'abord des fragments NFA pour les sous-expressions, puis en les reliant pour former un fragment NFA pour l'expression régulière entière. Les fragments NFA pour les sous-expressions sont représentés par des ovales en pointillés avec la demi-transition entrante à gauche et la demi-transition sortante à droite.

Lorsqu'un fragment NFA a été construit pour l'ensemble de l'expression régulière, la construction est complétée en connectant la demi-transition sortante à un état acceptant. La demi-transition entrante sert à identifier l'état de départ de l'AFN complétée. Notez que même si nous permettons à un NFA d'avoir plusieurs états finaux, un NFA construit avec cette méthode n'en aura qu'un seul : celui ajouté à la fin de la construction. Un NFA construit de cette manière pour l'expression régulière **(a|b)∗ac** est montré dans la figure 2.

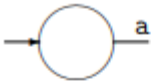
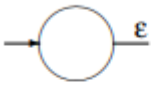
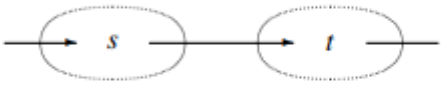
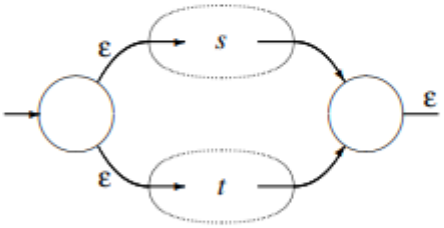
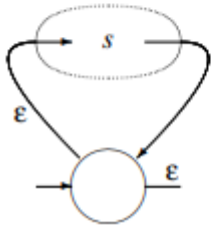
Regular expression	NFA fragment
a	
ε	
s t	
s|*t*	
*s**	

Figure 1: Construction de fragments NFA à partir d'expressions régulières

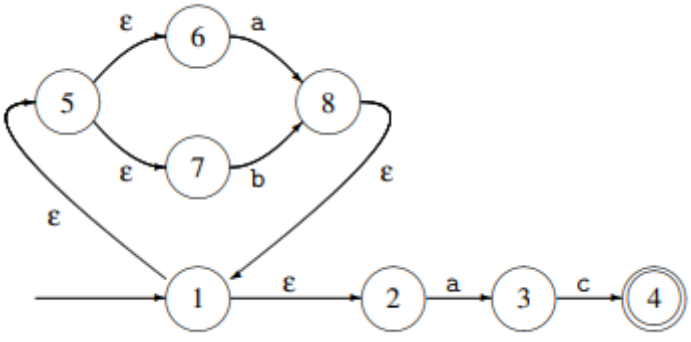


Figure 2: NFA pour l'expression régulière (a|b)∗ac

Automates finis déterministes

Étant donné un NFA N avec les états S, l'état de départ $s_0 \in S$, les états finals $F \subseteq S$, les transitions T et l'alphabet Σ, nous construisons un DFA équivalent D avec les états S', l'état de départ s'_0, les états finaux F' et une fonction de transition **move** par :

$$\begin{array}{lcl} s'_0 & = & \varepsilon\text{-closure}(\{s_0\}) \\ move(s', c) & = & \varepsilon\text{-closure}(\{t \mid s \in s' \text{ and } s^c t \in T\}) \\ S' & = & \{s'_0\} \cup \{move(s', c) \mid s' \in S', c \in \Sigma\} \\ F' & = & \{s' \in S' \mid s' \cap F \neq \emptyset\} \end{array}$$

Une petite explication :

- L'état de départ du DFA est la « epsilon-closure » de l'ensemble contenant uniquement l'état de départ du NFA, c'est-à-dire les états qui sont atteignables depuis l'état de départ par des transitions epsilon.
- Une transition dans le DFA est effectuée en trouvant l'ensemble des états du NFA qui comprennent l'état du DFA, en suivant toutes les transitions (sur le même symbole) dans le NFA à partir de tous ces états du NFA et enfin en combinant les ensembles d'états résultants et en fermant ceci sous des transitions epsilon.
- L'ensemble S' des états du DFA est l'ensemble des états du DFA qui peuvent être atteints à partir de s'_0 en utilisant la fonction move.
- Un état du DFA est un état final si au moins un des états NFA qu'il contient est un état final.