Cours – Représentations équivalentes : expressions régulières, automates (à nombre) d'états fini, équations de langage

1 Des expressions régulières aux automates (à nombre) d'états fini, utilisation des ϵ-transitions

Expressions régulières : syntaxe (= notation) Une expression régulière est de la forme

	`s`	*où s est un symbole de l'alphabet Σ*
ou bien	`{}`, `$`, `_`	
ou bien	e_1 `.` e_2	
ou bien	e_1 `|` e_2	
ou bien	e_1 `&` e_2	
ou bien	(e)`*`	

où e_1, e_2, e désignent des expressions régulières. On peut définir d'autres opérateurs, par exemple

- `~`e pour « complémentaire »
- e_1 `-` e_2 pour « privé de »
- e`?` pour « éventuellement e »

Expressions régulières : sémantique (= interprétation des notations) On donne une interprétation aux notations en indiquant pour chaque notation le langage qu'elle reconnaît :

$\mathscr{L}($`s`$)$	$=$	$\{s\}$	*le langage réduit au symbole s*
$\mathscr{L}($`{}`$)$	$=$	$\{\,\}$	*le langage vide*
$\mathscr{L}($`$`$)$	$=$	$\{\epsilon\}$	*le langage contenant uniquement le symbole invisible*
$\mathscr{L}($`_`$)$	$=$	Σ	*n'importe quel symbole de l'alphabet*
$\mathscr{L}(e_1$ `.` $e_2)$	$=$	$\mathscr{L}(e_1) \cdot \mathscr{L}(e_2)$	*concaténation de langages*
$\mathscr{L}(e_1$ `|` $e_2)$	$=$	$\mathscr{L}(e_1) \cup \mathscr{L}(e_2)$	*union de langages = somme des automates*
$\mathscr{L}(e_1$ `&` $e_2)$	$=$	$\mathscr{L}(e_1) \cap \mathscr{L}(e_2)$	*intersection de langages = produit des automates*
e`*`	$=$	$\big(\mathscr{L}(e)\big)^*$	*fermeture de Kleene du langage*

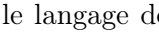
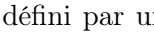
Exemple Quel est le langage reconnu par l'expression régulière $($`_*` $\,.\,(b$ `|` $a))$ `&` $(a$`*` $\,.\,b$`*`$)$

SOLUTION

$$\begin{aligned}
\mathscr{L}\big(\texttt{_*}\,.\,(b\ \texttt{|}\ a))\ \texttt{\&}\ (a\texttt{*}\,.\,b\texttt{*})\big) &= \mathscr{L}\big(\texttt{_*}\,.\,(b\ \texttt{|}\ a))\big) \cap \mathscr{L}\big(a\texttt{*}\,.\,b\texttt{*}\big) \\
&= \big(\mathscr{L}(\texttt{_*}) \cdot \mathscr{L}(b\ \texttt{|}\ a)\big) \cap \big(\mathscr{L}(a\texttt{*}) \cdot \mathscr{L}(b\texttt{*})\big) \\
&= \big(\Sigma^* \cdot (\{a\} \cup \{b\})\big) \cap \big(\{a\}^* \cdot \{b\}^*\big) \\
&= \big(\Sigma^* \cdot \{a,b\}\big) \cap \big(\{a\}^* \cdot \{b\}^*\big) \\
&= \big\{a^i,\ a^j b^i \mid i,j \in \mathbb{N},\ i \geq 1\big\}
\end{aligned}$$

L'ensemble de mots qu'on obtient peut être décrit par d'autres expressions régulières équivalentes :

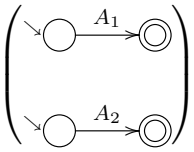
- *soit* $(a\texttt{*}\,.\,a)$ `|` $(a\texttt{*}\,.\,b\texttt{*}\,.\,b)$
- *soit* $(a\texttt{*}\,.\,b\texttt{*})$ `-` `$` *à condition d'avoir définir l'opérateur « privé de » $(-)$ sur les expressions régulières.*

Automate associé à une expression régulière On donne un procédé (récursif) de construction d'un automate qui reconnaît le langage défini par une expression régulière. Soit  ,  et  les automates qui reconnaissent les langages définis par les expressions régulières e, e_1, e_2.

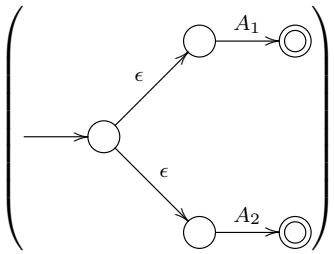
$$\begin{aligned}
\mathscr{L}(\texttt{\{\}}) &= \{\,\} = \mathscr{L}(\rightarrow\bigcirc) \\
\mathscr{L}(\texttt{\$}) &= \{\epsilon\} = \mathscr{L}(\rightarrow\textcircled{\textcircled{}}) \\
\mathscr{L}(\texttt{s}) &= \{s\} = \mathscr{L}(\rightarrow\bigcirc \xrightarrow{s} \textcircled{\textcircled{}}) \\
\mathscr{L}(\texttt{_}) &= \Sigma = \mathscr{L}(\rightarrow\bigcirc \xrightarrow{\Sigma} \textcircled{\textcircled{}}) \\
\mathscr{L}(e_1\,.\,e_2) &= \mathscr{L}(A_1) \cdot \mathscr{L}(A_2) = \mathscr{L}\left(\rightarrow\bigcirc \xrightarrow{A_1} \otimes \xrightarrow{\epsilon} \bigcirc \xrightarrow{A_2} \textcircled{\textcircled{}}\right) \\
&= \mathscr{L}\left(\rightarrow\bigcirc \xrightarrow{A_1} \bigcirc \xrightarrow{A_2} \textcircled{\textcircled{}}\right)
\end{aligned}$$

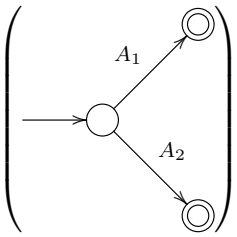
cette simplification est possible uniquement si A_1 n'a pas de cycle contenant un état accepteur et A_2 n'a pas de cycle contenant son état initial

$\mathscr{L}(e_1 \,\&\, e_2) = \mathscr{L}(A_1) \cap \mathscr{L}(A_2) = \mathscr{L}(A_1 \times A_2)$

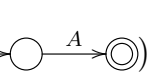
$\mathscr{L}(e_1 \mid e_2) = \mathscr{L}(A_1) \cup \mathscr{L}(A_2) = \mathscr{L}(A_1 + A_2) = \mathscr{L}$ 

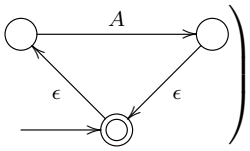
automate avec deux états initiaux

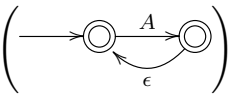
ou bien $= \mathscr{L}$ 

$= \mathscr{L}$ 

cette simplification est possible uniquement si ni A_1 ni A_2 n'ont de cycle contenant leur état initial

$\mathscr{L}(e*) = \left(\mathscr{L}($ 

$)\right)^* = \mathscr{L}$ 

$= \mathscr{L}$ 

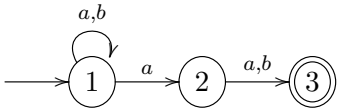
cette simplification est possible uniquement si ni A n'a pas de cycle contenant son état initial

2 Élimination des ϵ / Fermeture par ϵ^*

en TD

3 Des automates (à nombre) d'états fini aux équations d'Arden

Considérons l'automate



On note L_q le langage reconnu par l'état q de l'automate. On peut alors décrire l'automate comme un système d'équations sur les langages :

$$\begin{cases} L_1 = (a+b).L_1 + a.L_2 \\ L_2 = (a+b).L_3 \\ L_3 = \epsilon \end{cases}$$

Le langage reconnu par l'automate est le langage de son état initial (L_1 donc)

4 Des équations d'Arden aux expressions régulières

Pour obtenir l'expression régulière qui correspond à chaque langage L_q on résout le système d'équations à l'aide du lemme d'Arden.

Lemme d'Arden

Soient A et B des langages ou des automates ou des expressions régulières (les trois représentations sont équivalentes),
l'équation de langage $L = A \cdot L \cup B$ avec $\epsilon \notin A$ admet pour unique solution le langage

$L = A^* \cdot B$

Remarques

— *Si $\epsilon \in A$ alors il y a plusieurs solutions :*
$L = A^* \cdot B$ est le plus petit ensemble solution, $L = \Sigma^*$ est aussi une solution, en effet :

Montrons que $\Sigma^* = A.\Sigma^* \cup B$ *Pour cela on montre la double inclusion.*

$\supseteq$ *est triviale puisque Σ^* inclut tous les langages*

$\subseteq$ *: on doit montrer que* $\Sigma^* \subseteq A.\Sigma^* \cup B$ *sachant que* $\epsilon \in A$.
Puisque $\epsilon \in A$ *alors*

$$\begin{array}{ccccc} \{\epsilon\} \cdot \Sigma^* & \subseteq & A \cdot \Sigma^* & & \\ = & & & & \\ \Sigma^* & \subseteq & A \cdot \Sigma^* & \subseteq & A \cdot \Sigma^* \cup B \end{array}$$

□

— *Si* $\epsilon \notin A$ *alors* $L = A^* \cdot B$ *est la plus petite solution. Remarquez que si le langage* $B = \emptyset$ *alors la solution est* $L = \emptyset$

Preuve du Lemme d'Arden

1. On vérifie que $L = A^* \cdot B$ est solution de l'équation $L = A \cdot L \cup B$ en démontrant l'égalité
$$A^* \cdot B = A \cdot (\underbrace{A^* \cdot B}_{L}) \cup B$$
Pour cela on montre la double inclusion entre les langages : à rédiger.
2. Preuve de l'unicité de la solution si $\epsilon \notin A$

Soient L *et* L' *deux solutions avec* $L \neq L'$.
Soit ω *le plus petit mot tel que* $\omega \in L$ *et* $\omega \notin L'$ *(un tel mot existe si* $L \neq L'$*)*

— *puisque* L *est une solution alors* $L = A.L \cup B$ *donc* $B \subseteq L$

— *puisque* L' *est une solution alros* $L' = A.L' \cup B$ *donc* $B \subseteq L'$

donc $\omega \notin B$ *sinon on aurait* $\omega \in L'$
donc forcément $\omega \in A.L$ *et alors* ω *s'écrit* $a.u$ *avec* $u \in L$
mais alors pour avoir $\omega \stackrel{def}{=} a.u \notin L'$ *sachant que* $a.u \notin B$*, il faut que* $a.u \notin A \cdot L'$ *donc il faut avoir* $u \notin L'$
On a donc construit un mot u *plus petit que* ω *avec* $u \in L$ *et* $u \notin L'$ *: Contradiction (*ω *était le plus petit mot tel que* $\omega \in L$ *et* $\omega \notin L'$*).*
Conclusion : *On a supposé qu'il existait deux solutions distinctes* L *et* L' *et on a abouti à une contradiction donc la solution de l'équation* $L = A \cdot L \cup B$ *est unique quand* $\epsilon \notin A$.

Application En appliquant le lemme d'Arden au système d'équations précédent on obtient

$$\begin{cases} L_1 = \underbrace{(a+b)}_{A}.L_1 + \underbrace{a.L_2}_{B} = \underbrace{(a+b)}_{A}^*.\underbrace{a.L_2}_{B} = (a+b)^*.a.(a+b) \\ L_2 = (a+b).L_3 = (a+b).\epsilon = (a+b) \\ L_3 = \epsilon \end{cases}$$

5 Résultat

Les représentations suivantes permettent de décrire les mêmes langages :
ANDEF, ADEF, ADEF *minimaux, expressions régulières, équations d'Arden*
Autrement dit :
Ces représentations ont le même pouvoir d'expression
Autrement dit :
Il n'est pas possible de décrire avec une représentation un langage qu'on ne pourrait pas décrire par une autre des représentations.

*On nomme «**Langages réguliers**» cette classe de langages pour insister sur le fait que les mots de ces langages comportent un certaine régularité.*