Expressions régulières

Théorie des langages et de la compilation
Travaux pratiques

S. COLLETTE G. GEERAERTS

Expressions Régulières

expression régulière	langage défini
ϕ	$\emptyset$
ε	$\{\varepsilon\}$
$a \qquad (\forall a \in \Sigma)$	$\{a\}$

Expression Régulière, suite

Si p et q sont des expressions régulières et représentent les langages P et Q respectivement alors

expression régulière	langage défini
$p + q$	$P \bigcup Q$
pq (ou $p.q$)	$P.Q$
p^*	P^*

Remarque : $p^+ = pp^*$

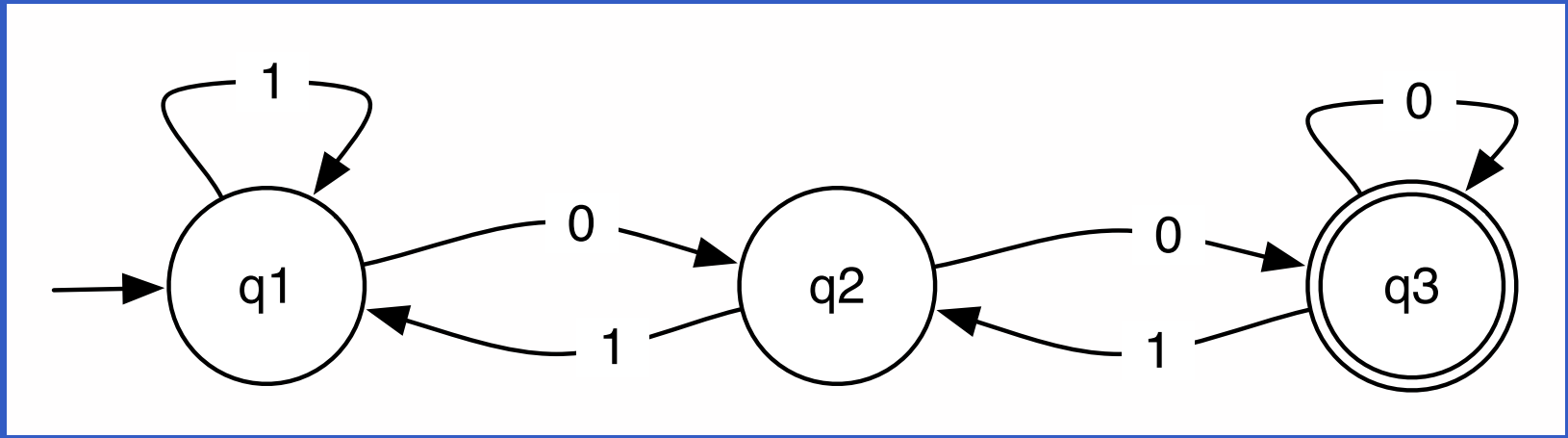
Exercice 1

Donnez une expression régulière qui accepte chacun des langages suivants (définis sur l'alphabet $\Sigma = \{0, 1\}$) :

1. Toutes les chaînes qui se terminent par 00.
2. Toutes les chaînes dont le 10ème symbole, compté à partir de la fin de la chaîne, est un 1.
3. Ensemble de toutes les chaînes dans lesquelles chaque paire de 0 apparaît devant une paire de 1.
4. Ensemble de toutes les chaînes ne contenant pas 101.
5. Tous les nombres binaires divisibles par 4.

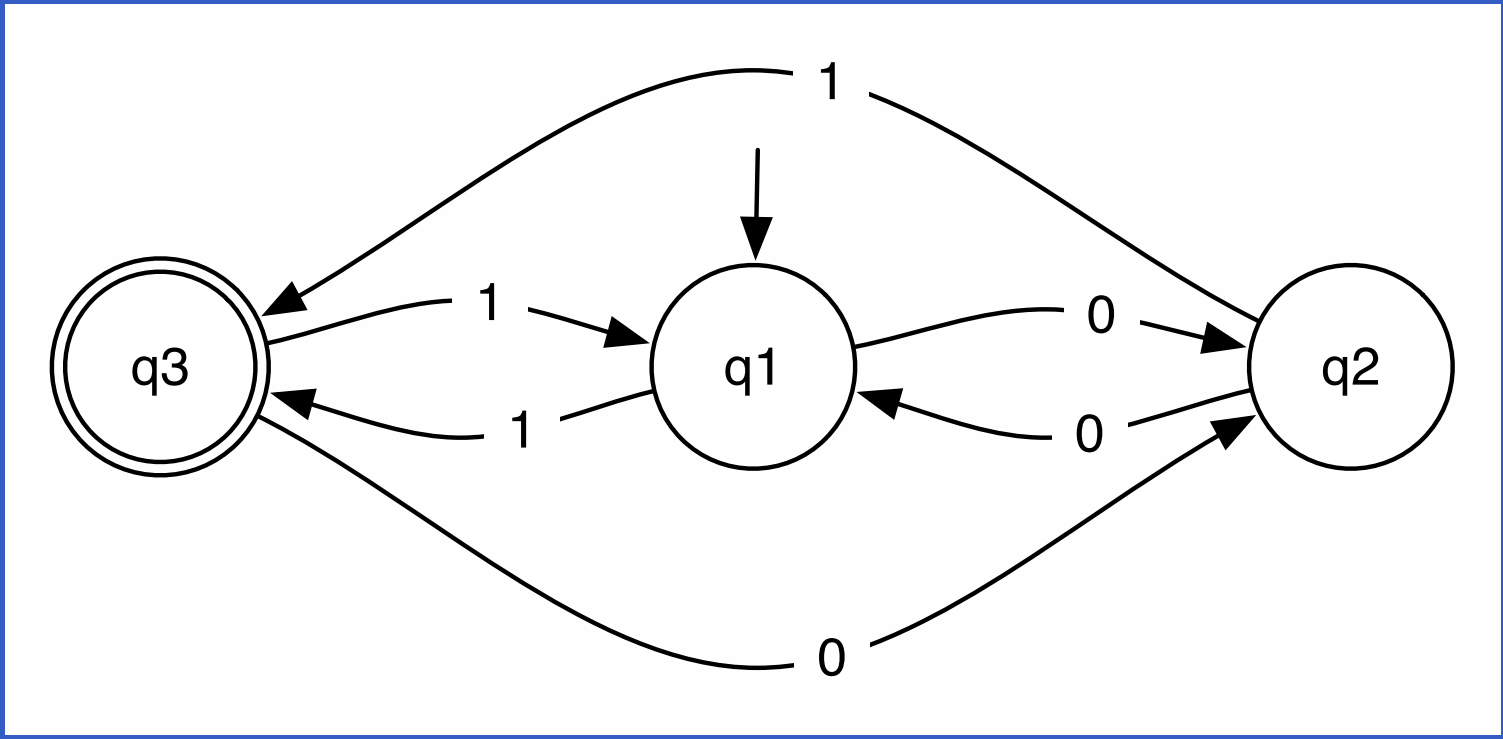
Exercice 2.1

Construisez l'ER acceptant le même langage que :



Exercice 2.2

Construisez l'ER acceptant le même langage que :



Exercice 3

Convertissez les expressions régulières suivantes en NFA_ε :

1. 01^*
2. $(0+1)01$
3. $00(0+1)^*$

Exercice 1 – correction

1. $(0+1)^*00$
2. $(0+1)^*1(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)$
3. $(1+01+0011)^*(0+\varepsilon)$
4. $0^*(1+00^+)^*0^*$
5. $(0+1)^*00$

Exercice 2.1 – correction

- On calcule les $R^k_{i,j}$ pour $k = 0, 1, 2$ et on en déduit $R^3_{1,3}$.
- Rappel : $R^k_{i,j}$= tout ce qu'on peut accepter sur des chemins allant de l'état i à l'état j, en traversant des états de numéro $\leq k$.

Exercice 2.1 – correction

- Donc, pour tout $k \geq 1$:

$$R^k_{i,j} = \left(\underbrace{R^{k-1}_{i,k}}_{(1)} \cdot \underbrace{\left(R^{k-1}_{k,k}\right)^*}_{(2)} \cdot \underbrace{R^{k-1}_{k,j}}_{(3)} \right) + \underbrace{R^{k-1}_{i,j}}_{(4)}$$

  - (1) : accepté entre i et k
  - (2) : boucle sur k
  - (3) : accepté entre k et j
  - (4) : si on choisit de ne pas passer par k

Exercice 2.1 – correction

$k = 0$:

i, j	1	2	3
1	$1 + \varepsilon$	0	$\emptyset$
2	1	ε	0
3	$\emptyset$	1	$0 + \varepsilon$

$k = 1$:

i, j	1	2	3
1	$(1+\varepsilon)(1+\varepsilon)^*(1+\varepsilon) + (1+\varepsilon)$	$1(1+\varepsilon)^*0 + 0$	$\emptyset$
2	$1(1+\varepsilon)^*0 + 1$	$1(1+\varepsilon)^*0 + \varepsilon$	0
3	$\emptyset$	1	$0 + \varepsilon$

Exercice 2.1 – correction

Après simplification :

$k = 1$:

i, j	1	2	3
1	1^*	1^*0	$\emptyset$
2	$11^*0 + 1$	$11^*0 + \varepsilon$	0
3	$\emptyset$	1	$0 + \varepsilon$

La réponse est $R^3_{1,3} = R^2_{1,3}(R^2_{3,3})^* R^2_{3,3} + R^2_{1,3}$, avec :

- $R^2_{1,3} = 1^*0(11^*0 + \varepsilon)^*0$
- $R^2_{3,3} = 1(11^*0 + \varepsilon)^*0 + (0 + \varepsilon)$

Exercice 2.2 – correction

$k = 0$:

i, j	1	2	3
1	ε	0	1
2	0	ε	1
3	1	0	ε

$k = 1$:

i, j	1	2	3
1	ε	0	1
2	0	$00 + \varepsilon$	$01 + 1$
3	1	$10 + 0$	$11 + \varepsilon$

Exercice 2.2 – correction

La solution est $R^3_{1,3} = R^2_{1,3}(R^2_{3,3})^* R^2_{3,3} + R^2_{1,3}$, avec :

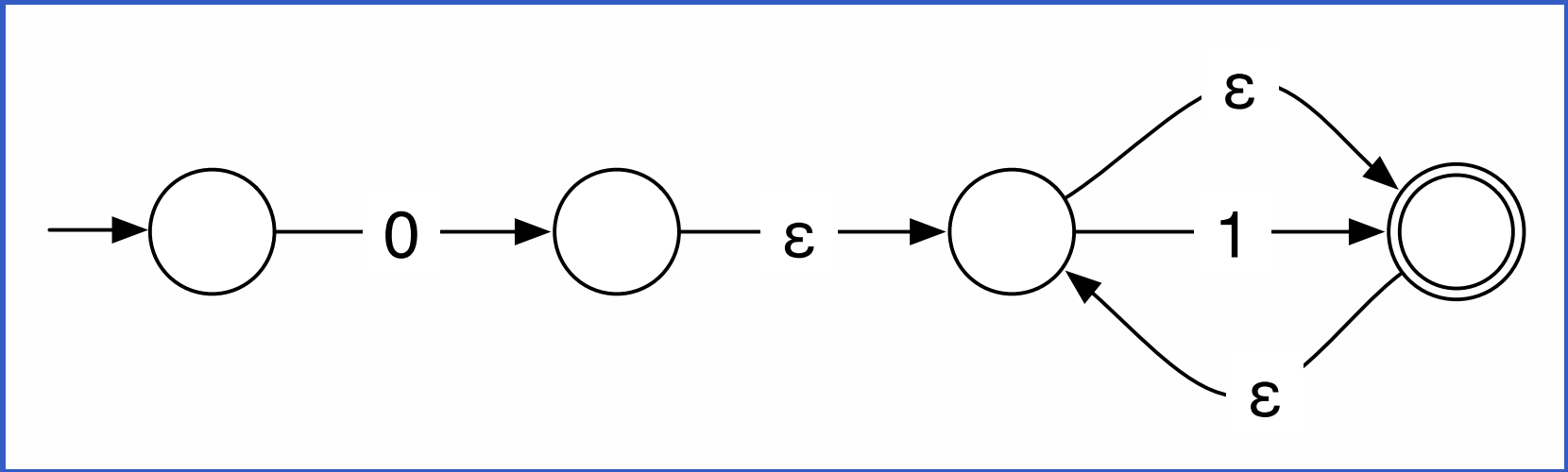
$$\begin{aligned} R^2_{1,3} &= 0(00+\varepsilon)^*(01+1)+1 \\ &= 0^*1 \end{aligned}$$

$$\begin{aligned} R^2_{3,3} &= (10+0)(00+\varepsilon)^*(01+1)+11+\varepsilon \\ &= (1+\varepsilon)0(00+\varepsilon)^*(0+\varepsilon)1+11+\varepsilon \\ &= (1+\varepsilon)00^*1+\varepsilon \end{aligned}$$

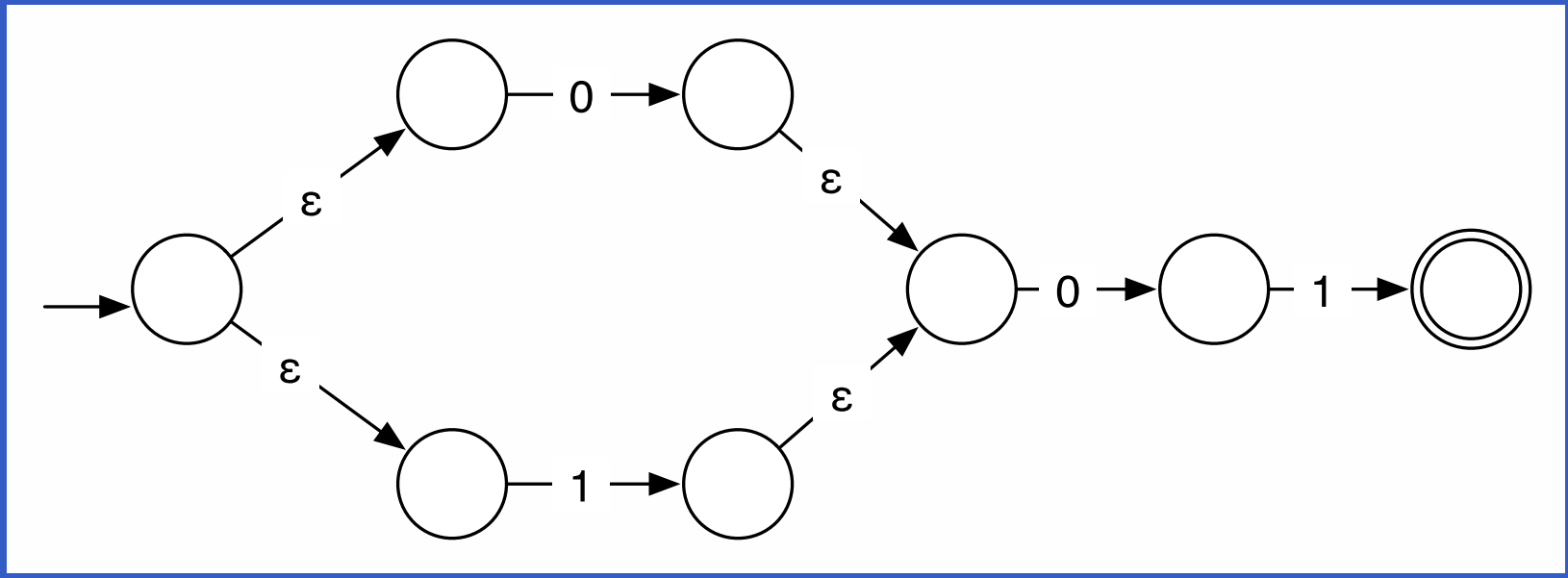
Au final :

$$\begin{aligned} R^3_{1,3} &= 0^*1((1+\varepsilon)00^*1+\varepsilon)^*((1+\varepsilon)00^*1+\varepsilon)+0^*1 \\ &= 0^*1(100^*1+00^*1)^* \end{aligned}$$

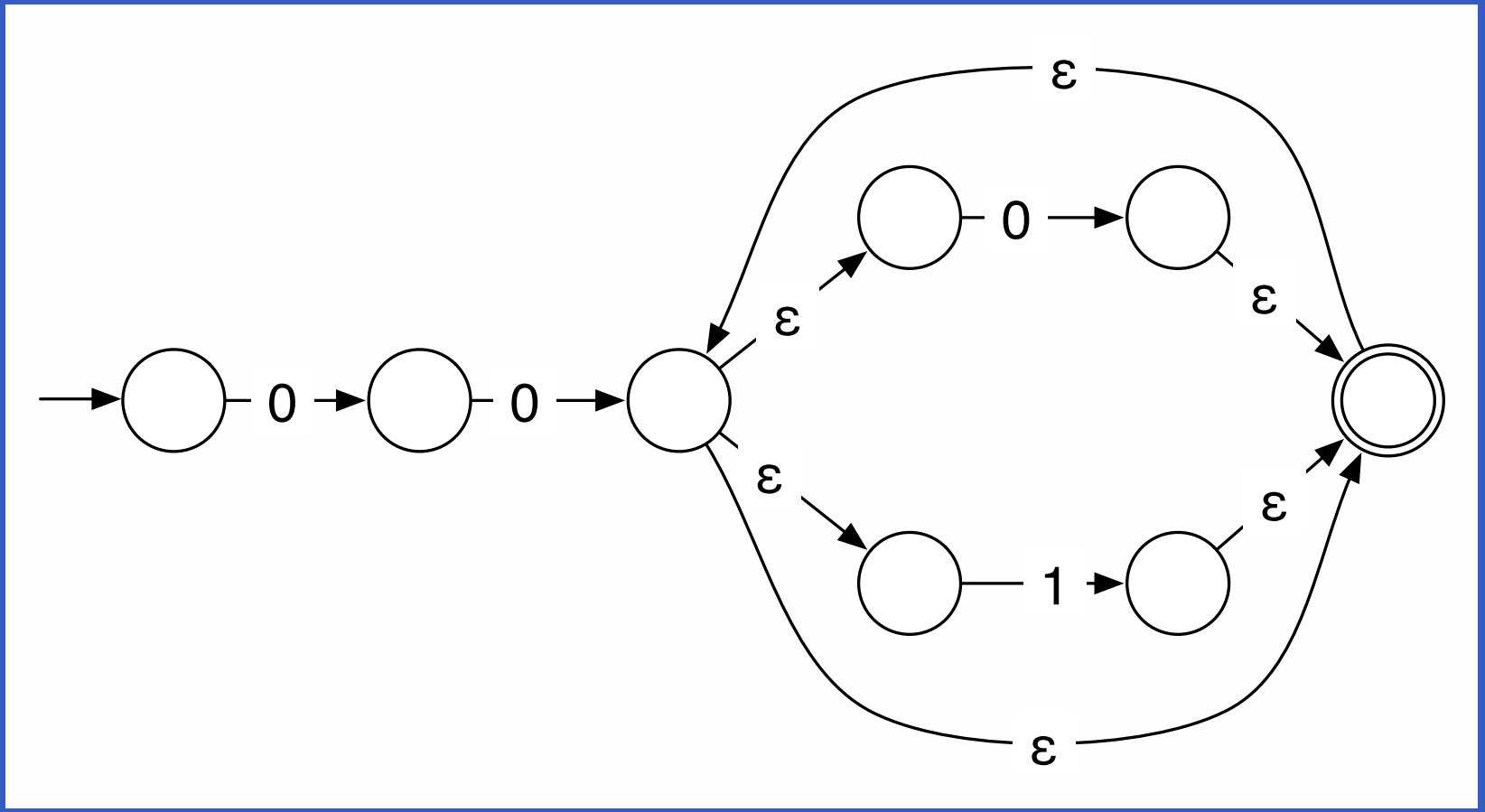
Exercice 3.1 – correction



Exercice 3.2 – correction



Exercice 3.3 – correction



Les expressions régulières étendues

- Très utilisées dans UNIX (`grep`, `find`, *etc*.)
- Plus flexibles que les expressions régulières traditionnelles.
- Elles sont aussi utilisées dans `lex`.

Syntaxe 1

Expression	Langage accepté
`r*`	0 ou plusieurs `r`
`r+`	1 ou plusieurs `r`
`r?`	0 ou 1 `r`
`[abc]`	a ou b ou c
`[a-z]`	N'importe quel caractère dans l'intervalle a..z
`.`	N'importe quel caractère sauf \n
`[^s]`	N'importe quel caractère sauf ceux de `s`
`r{m,n}`	Entre `m` et n occurences de `r`
`r1 r2`	La concaténation de `r1` et `r2`

Syntaxe 2

Expression	Langage accepté
`r1 | r2`	`r1` ou `r2`
`(r)`	`r`
`^r`	`r` en début de ligne
`r$`	`r` en fin de ligne
`"s"`	Le string s
`\c`	Le caractère c
`r1 / r2`	`r1` quand il est suivi de `r2`

Exemples

- `[a-zA-z]` Une lettre.
- `[0-9]` Un chiffre.
- `a[^A-Za-z]b` Un a, suivi d'un caractère non alphabétique, suivi d'un b.
- `^Monsieur` `Monsieur` en début de ligne.
- `[a-zA-Z]([a-zA-Z]|[0-9])*` Un identifiant Pascal. . . .

Exercice 4

1. Donnez l'expression régulière étendue (ERE) qui désigne n'importe quelle suite de 5 caractères, y compris `\n`.
2. Donnez l'ERE qui désigne une chaîne formée de n'importe quel nombre de `\`, suivi de n'importe quel nombre de `*`.
3. Les shells UNIX (du type `bash`) permettent d'écrire des fichiers *batch* dans lesquels on peut insérer des commentaires. Une *ligne* est considérée comme commentaire si elle commence par #. Quelle est l'ERE qui accepte de tels commentaires ?
4. Donnez l'ERE qui désigne un nombre en notation scientifique. Ce nombre sera composé d'au moins un chiffre. Il comportera deux parties optionnelles : une partie «décimale» (un `.` suivi d'une série de chiffres) et une partie «exposant» (un `E` suivi d'un nombre entier, éventuellement préfixé d'un + ou un `-`).

Exercice 4

5 Donnez l'ERE acceptant l'ensemble des phrases «correctes» selon les critères suivants :

- Le premier mot de la phrase a une majuscule ;
- la phrase se termine par un point ;
- la phrase est composée d'un ou plusieurs mots (caractères `a...z` et `A...Z`), séparés par un espace ;
- on trouve une phrase par ligne.

Remarquons que les caractères de ponctuation autres que le point ne sont pas admis.

6 Écrivez l'ERE qui accepte tous les noms de fichiers DOS (composés de 8 caractères : `A...Z`, `a...z` et _), dont l'extension est `ext` et commençant par la chaîne `abcde`. Attention, l'ERE ne doit accepter que *le nom du fichier sans l'extension !*

Exercice 4 – correction

1. `(.|\n){5}`

Exercice 4 – correction

1. `(.|\n){5}`
2. `\\*\**`

Exercice 4 – correction

1. `(.|\n){5}`
2. `\\*\**`
3. `^#.*$`

Exercice 4 – correction

1. `(.|\n){5}`
2. `\\*\**`
3. `^#.*$`
4. `[0-9]+(\.[0-9]+)?(E[+-]?[0-9]+)?`

Exercice 4 – correction

1. `(.|\n){5}`
2. `\\*\**`
3. `^#.*$`
4. `[0-9]+(\.[0-9]+)?(E[+-]?[0-9]+)?`
5. `^ [A-Z][A-Za-z]*(\ [A-Za-z]+)*\.$`

Exercice 4 – correction

1. `(.|\n){5}`
2. `\\*\**`
3. `^#.*$`
4. `[0-9]+(\.[0-9]+)?(E[+-]?[0-9]+)?`
5. `^ [A-Z][A-Za-z]*(\ [A-Za-z]+)*\.$`
6. `abcde[A-Za-z_]{3}/\.ext`