

## 1 Incontournables

Chacune des questions suivantes est notée 0 pour la réponse correcte, -5 dans tous les autres cas.

Q.2 Si  $\{a^n b^n \mid n \in \mathbb{N}\} \subseteq L$ , alors  $L$  n'est pas rationnel.

☐ vrai

☒ faux

Q.3 Le langage  $\{a^m b^m \mid m \in \mathbb{N}\}$  est :

☐ ambigu

☒ hors-contexte

☐ rationnel

☐ fini

Q.4 Combien existe-t-il de sous-ensembles de  $\{1, 2, \dots, n\}$  ?

☐  $\frac{n(n+1)}{2}$

☒  $2^n$

☐  $n!$

☐  $\frac{n(n-1)}{2}$

☐  $n^2$

Q.4 Si une grammaire n'est pas LR(1), alors elle est ambiguë.

☒ faux

☐ vrai

## 2 Théorie des langages rationnels

Q.5 Que vaut  $\text{Fact}(\{ab, c\})$  (l'ensemble des facteurs) :

☐  $\{\varepsilon\}$

☒  $\{ab, a, b, c, \varepsilon\}$

☐  $\{a, b, c, \varepsilon\}$

☐  $\emptyset$

☒  $\{a, b, c\}$

Q.5 Que vaut  $\text{Fact}(L)$  (l'ensemble des facteurs) :

☐  $\text{Pref}(\text{Pref}(L))$

☒  $\text{Suff}(\text{Pref}(L))$

☐  $\text{Suff}(\overline{\text{Pref}(L)})$

☐  $\text{Suff}(\text{Suff}(L))$

☐  $\text{Pref}(\overline{\text{Pref}(L)})$

Q.6 Que vaut  $\text{Fact}(\{a\}\{b\}^*)$  (l'ensemble des facteurs)

☐  $\{b\}\{a\}^* \cup \{b\}^*$

☒  $\{a\}\{b\}^* \cup \{b\}^*$

☐  $\{a, b\}^* \{b\}\{a, b\}^*$

☐  $\{a\}\{b\}^* \{a\}$

☐  $\{\varepsilon\} \cup \{a\}\{a\}\{a\}^*$

Q.6 Que vaut  $\overline{\{a\}^*}$ , avec  $\Sigma = \{a, b\}$ .

☐  $\{\varepsilon\} \cup \{a\}\{a\}\{a\}^*$

☒  $\{a, b\}^* \{b\}\{a, b\}^*$

☐  $\{a\}\{b\}^* \{a\}$

☐  $\{a\}\{b\}^* \cup \{b\}^*$

☐  $\{b\}\{a\}^* \cup \{b\}^*$

Q.6 Que vaut  $(\{a\}\{b\}^*\{a\}^*) \cap (\{a\}^*\{b\}^*\{a\})$

☐  $\{b\}\{a\}^* \cup \{b\}^*$

☐  $\{\varepsilon\} \cup \{a\}\{a\}\{a\}^*$

☒  $\{a\} \cup \{a\}\{b\}^*\{a\}$

☐  $\{a, b\}^* \{b\}\{a, b\}^*$

☐  $\{a\}\{b\}^* \cup \{b\}^*$

Q.6 Que vaut  $\overline{\{a\}\{b\}^*} \cap \{a\}^*$

- ☐  $\{a\}\{b\}^* \cup \{b\}^*$ 
☐  $\{b\}\{a\}^* \cup \{b\}^*$ 
☐  $\{a, b\}^* \{b\}\{a, b\}^*$ 
☐  $\{a\}\{b\}^* \{a\}$   
☒  $\{\varepsilon\} \cup \{a\}\{a\}^*$

Q.7 Soit  $\Sigma$  un alphabet. Pour tout  $a \in \Sigma, L_1, L_2 \subseteq \Sigma^*$ , on a  $L_1^* = L_2^* \implies L_1 = L_2$ .

- ☒ faux
 ☒ vrai

Q.7 Soit  $\Sigma$  un alphabet. Pour tout  $a \in \Sigma, L \subseteq \Sigma^*$ , on a  $\{a\}.L = \{a\}.M \implies L = M$ .

- ☐ faux
 ☒ vrai

Q.7 Soit  $\Sigma$  un alphabet. Pour tout  $a \in \Sigma, L \subseteq \Sigma^*$ , on a  $\forall n > 1, L^n = \{u^n | u \in L\}$ .

- ☒ faux
 ☐ vrai

Q.7 Si  $e$  et  $f$  sont deux expressions rationnelles, quelle identité n'est pas nécessairement vérifiée?

- ☐  $(ef)^*e \equiv e(fe)^*$ 
☐  $(e + f)^* \equiv (e^*f^*)^*$ 
☐  $(e + f)^* \equiv (f^*(ef)^*e^*)^*$   
☒  $(ef)^* \equiv e(fe)^*f$ 
☐  $\emptyset^* \equiv \varepsilon$

Q.6 Pour toutes expressions rationnelles  $e, f$ , simplifier  $e^*(e + f)^*f^*$ .

- ☒  $(e + f)^*$ 
☐  $e^* + f$ 
☐  $e + f^*$ 
☒  $e^*f^*$

Q.6 Pour  $e = (a + b)^* + \varepsilon, f = (a^*b^*)^*$  :

- ☒  $L(e) = L(f)$ 
☐  $L(e) \not\subseteq L(f)$ 
☐  $L(e) \supseteq L(f)$ 
☐  $L(e) \subseteq L(f)$

Q.6 Pour  $e = (ab)^*, f = a^*b^*$  :

- ☐  $L(e) = L(f)$ 
☒  $L(e) \not\subseteq L(f)$ 
☐  $L(e) \subseteq L(f)$ 
☐  $L(e) \supseteq L(f)$

Q.6 Pour  $e = (a + b)^*, f = a^*b^*$  :

- ☐  $L(e) = L(f)$ 
☐  $L(e) \subseteq L(f)$ 
☐  $L(e) \not\subseteq L(f)$ 
☒  $L(e) \supseteq L(f)$

Q.6 Pour  $e = (ab)^*, f = (a + b)^*$  :

- ☒  $L(e) \subseteq L(f)$ 
☐  $L(e) \not\subseteq L(f)$ 
☐  $L(e) = L(f)$ 
☐  $L(e) \supseteq L(f)$

**Q.6** Que vaut  $\text{Suff}(\{a\}\{b\}^*)$

- ☒  $\{a\}\{b\}^* \cup \{b\}^*$ 
☐  $\{a\}\{b\}^*\{a\}$ 
☐  $\{b\}\{a\}^* \cup \{b\}^*$ 
☐  $\{a, b\}^*\{b\}\{a, b\}^*$   
☐  $\{\varepsilon\} \cup \{a\}\{a\}\{a\}^*$

Q.5 Que vaut  $\text{Suff}(\{ab, c\})$  :

- ☐  $\{b, c, \varepsilon\}$       ☐  $\{b, \varepsilon\}$       ☐  $\emptyset$       ☒  $\{ab, b, c, \varepsilon\}$       ☐  $\{a, b, c\}$

**Q.5** Que vaut  $\text{Pref}(\{ab, c\})$  :

- ☐  $\emptyset$     ☒  $\{ab, a, c, \varepsilon\}$     ☐  $\{b, c, \varepsilon\}$     ☐  $\{a, b, c\}$     ☐  $\{b, \varepsilon\}$

**Q.8** Ces deux expressions rationnelles :

$$(a^{\star} + b)^{\star} + c((ab)^{\star}(bc))^{\star}(ab)^{\star} \quad c(ab + bc)^{\star} + (a + b)^{\star}$$

- ☐ ne sont pas équivalentes    
 ☐ sont identiques    
 ☐ dénotent des langages différents  
☒ sont équivalentes

**Q.6** Un langage quelconque

- ☒ est toujours inclus ( $\subseteq$ ) dans un langage rationnel
- ☐ peut n'être inclus dans aucun langage dénoté par une expression rationnelle
- ☒ n'est pas nécessairement dénombrable
- ☐ peut avoir une intersection non vide avec son complémentaire

Q.8 L'expression Perl '([+]\*[0-9A-F]+[+/\*])\*[+]\*[0-9A-F]+' n'engendre pas :

- ☐
- '-+-1+--2'
- ☐
- 'DEADBEEF'
- ☐
- '0+1+2+3+4+5+7+8+9'
- ☒
- '(20+3)\*3'

**Q.7** L'expression Perl "`([a-zA-Z]|\\|)+`" engendre :

- ☐ "eol" (eol est le caractère « retour à la ligne »)    ☒ "\\\\"    ☐ ""    ☐ "\"

Q.8 L'expression Perl '[+]?[0-9]+(,[0-9]+)?(e[+]?[0-9]+)' n'engendre pas :

- ☒
- '42,e42'
- ☐
- '42,42e42'
- ☐
- '42,4e42'
- ☐
- '42e42'

Q.8 L'expression Perl '[+]?[0-9A-F]+([+/\*][+]?[0-9A-F]+)\*' n'engendre pas :

- ☐ '-42-42'    ☒ '42+(42\*42)'    ☐ '42+42'    ☐ '-42'

2.6 L'expression Perl '[a-zA-Z][a-zA-Z0-9\_]\*' n'engendre pas :

- ☒
- `'__STDC__'`
- ☐
- `'main'`
- ☐
- `'exit_42'`
- ☐
- `'eval_expr'`

**Q.6** L'expression Perl `'[-+]?[0-9]+,[0-9]*'` n'engendre pas :

- ☒ '42'    ☐ '42,'    ☐ '42,42'    ☐ '42,4'

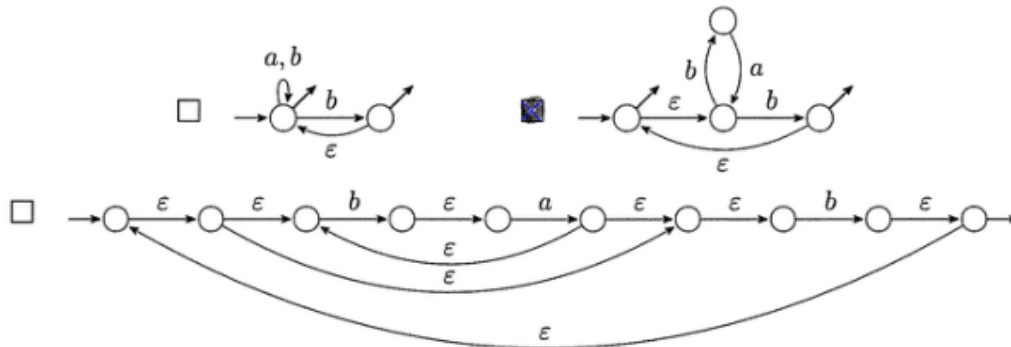
Q.8 Combien d'états a l'automate de Thompson auquel je pense ?

- ☐ 1    ☐ 9    ☒ 4    ☐ 7

Q.8 Quel automate ne reconnaît pas le langage décrit par l'expression  $(a^*b^*)^*$ .

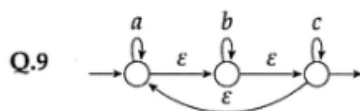


Q.8 Quel automate reconnaît le langage décrit par l'expression  $((ba)^*b)^*$

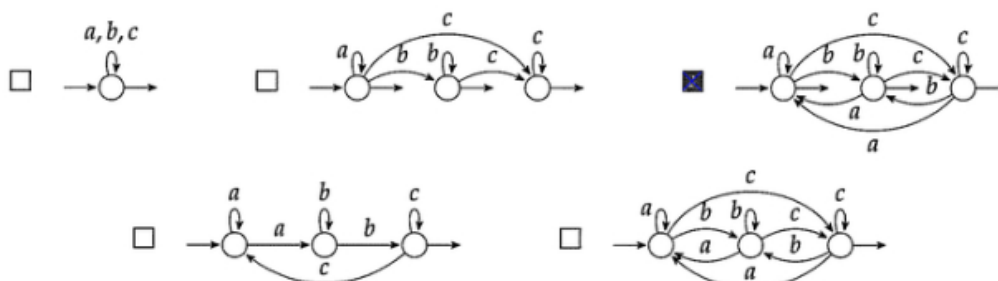


Q.10 Si  $L_1 \subseteq L \subseteq L_2$ , alors  $L$  est rationnel si :

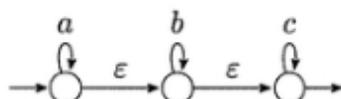
- ☒  $L_1, L_2$  sont rationnels et  $L_2 \subseteq L_1$     ☐  $L_2$  est rationnel    ☐  $L_1, L_2$  sont rationnels  
☐  $L_1$  est rationnel



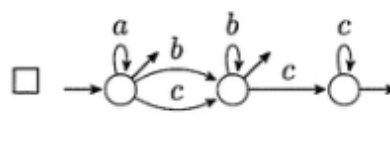
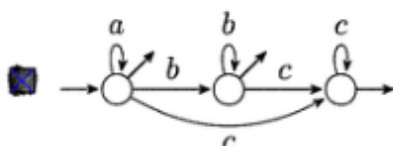
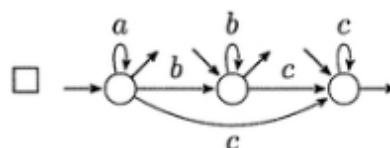
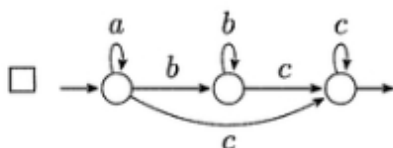
Quel est le résultat d'une élimination arrière des transitions spontanées ?



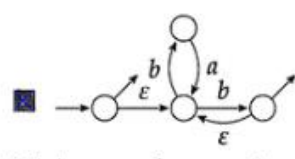
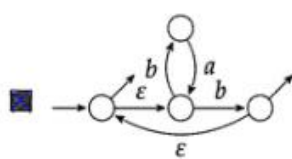
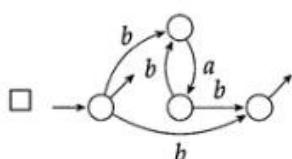
Q.8



Quel est le résultat d'une élimination *arrière* des transitions spontanées ?



Q.10 🎵 Parmi les 3 automates suivants, lesquels sont équivalents ?



☐ Aucune de ces réponses n'est correcte.

Q.10 Si un automate de  $n$  états accepte  $a^n$ , alors il accepte...

☐  $a^{n+1}$

☒  $a^p(a^q)^*$  avec  $p \in \mathbb{N}, q \in \mathbb{N}^* : p + q \leq n$

☐  $a^n a^m$  avec  $m \in \mathbb{N}^*$

☐  $(a^n)^m$  avec  $m \in \mathbb{N}^*$

Q.11 Quelle séquence d'algorithmes teste l'appartenance d'un mot au langage d'une expression rationnelle ?

☐ Thompson, déterminisation, élimination des transitions spontanées, évaluation.

☒ Thompson, élimination des transitions spontanées, déterminisation, minimisation, évaluation.

☐ Thompson, déterminisation, évaluation.

☐ Thompson, déterminisation, Brzozowski-McCluskey.

Q.12 Combien d'états au moins a un automate déterministe émondé qui accepte les mots sur  $\Sigma = \{a, b, c, d\}$  dont la  $n$ -ième lettre avant la fin est un  $a$  (i.e.,  $(a + b + c + d)^* a (a + b + c + d)^{n-1}$ ) :

☐  $\frac{n(n+1)(n+2)(n+3)}{4}$

☐  $4^n$

☒  $2^n$

☐ Il n'existe pas.

Q.10 Combien d'états au moins a un automate déterministe émondé qui accepte les mots sur  $\Sigma = \{a, b\}$  dont la  $n$ -ième lettre avant la fin est un  $a$  (i.e.,  $(a + b)^* a (a + b)^{n-1}$ ) :

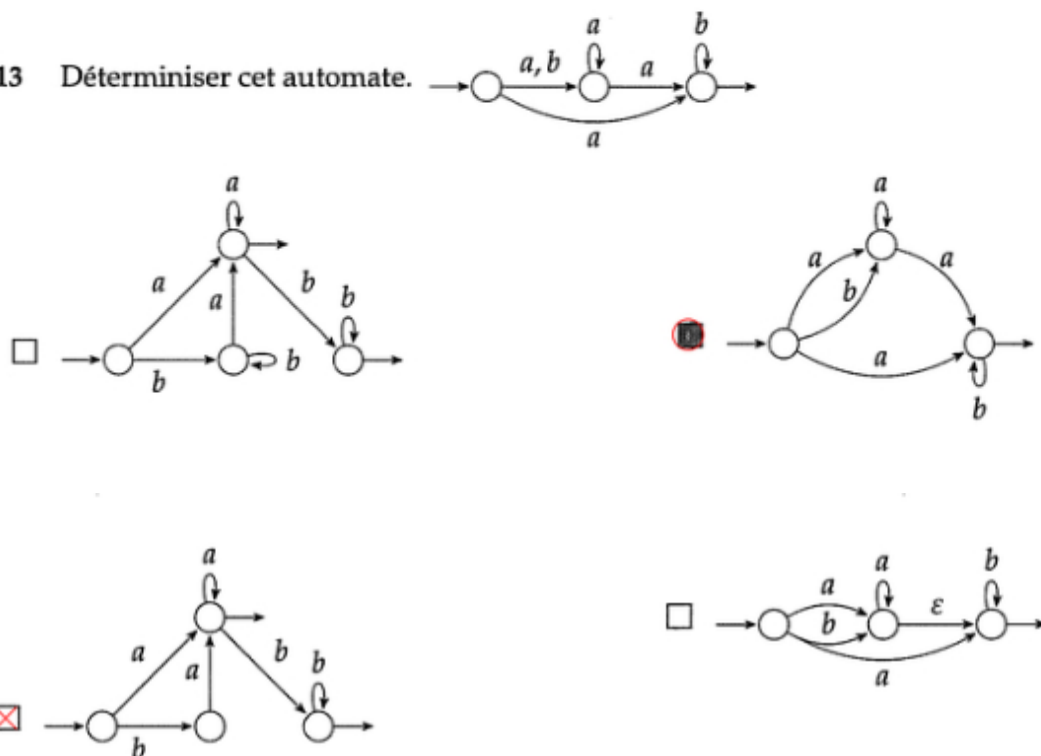
☐  $n + 1$

☒  $2^n$

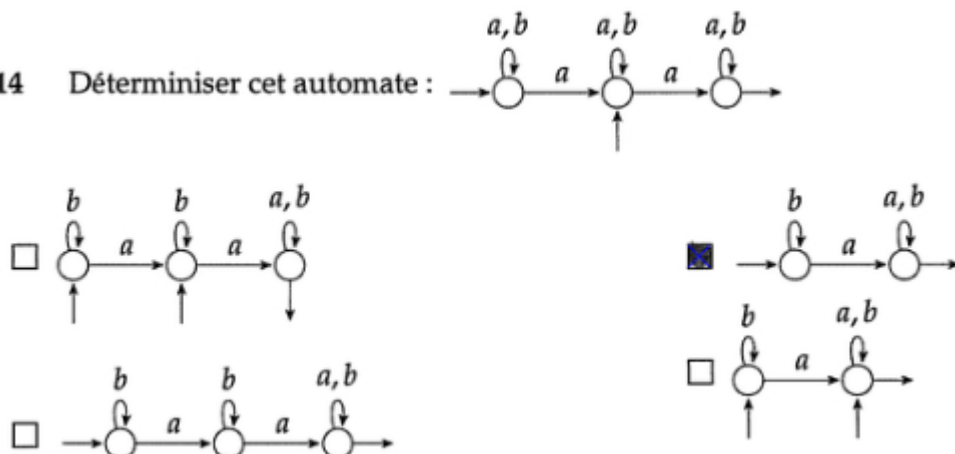
☐ Il n'existe pas.

☐  $\frac{n(n+1)}{2}$

Q.13 Déterminer cet automate.



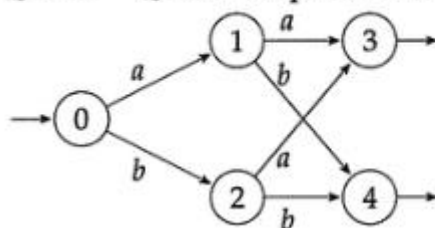
Q.14 Déterminer cet automate :



Q.15 Considérons  $\mathcal{P}$  l'ensemble des *palindromes* (mot  $u$  égal à son transposé/image miroir  $u^R$ ) de longueur paire sur  $\Sigma$ , i.e.,  $\mathcal{P} = \{v \cdot v^R \mid v \in \Sigma^*\}$ .

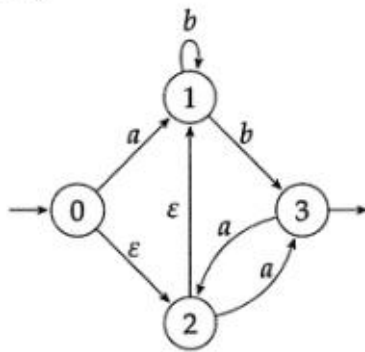
- ☐ Il existe un DFA qui reconnait  $\mathcal{P}$       ☐ Il existe un NFA qui reconnait  $\mathcal{P}$   
☒  $\mathcal{P}$  ne vérifie pas le lemme de pompage      ☐ Il existe un  $\varepsilon$ -NFA qui reconnait  $\mathcal{P}$

Q.16 🎵 Quels états peuvent être fusionnés sans changer le langage reconnu.



- ☐ 2 avec 4  
☒ 1 avec 2  
☐ 0 avec 1 et avec 2  
☐ 1 avec 3  
☒ 3 avec 4  
☐ Aucune de ces réponses n'est correcte.

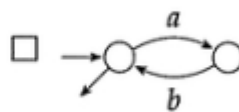
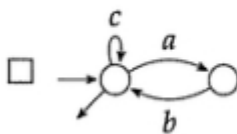
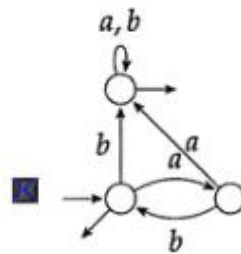
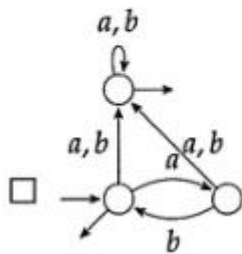
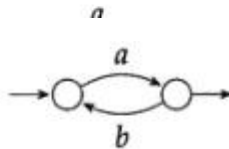
Q.17



Quel est le résultat de l'application de BMC en éliminant 1, puis 2, puis 3 et enfin 0?

- ☐  $(ab^* + (a + b)^*)(a + b)^+$
- ☐  $(ab^* + a + b^*)a(a + b^*)$
- ☐  $(ab^* + (a + b)^*)a(a + b)^*$
- ☐  $(ab^* + a + b^*)a(a + b)^*$
- ☒  $(ab^+ + a + b^+)(a(a + b^+))^*$

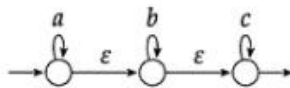
Q.18 Sur  $\{a, b\}$ , quel est le complémentaire de  $\rightarrow \text{---} \text{---} \text{---} ?$



Q.15 🎵 Qu'un langage vérifie le lemme de pompage

- ☒ est nécessaire s'il est rationnel
- ☐ est suffisant pour qu'il soit rationnel
- ☐ Aucune de ces réponses n'est correcte.

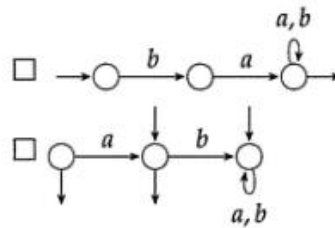
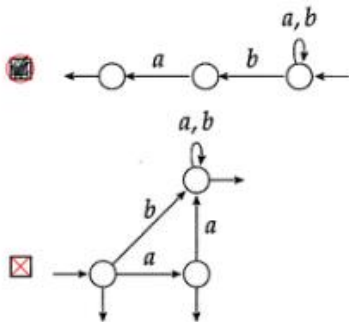
Q.16



Si on élimine les transitions spontanées de cet automate, puis qu'on applique la détermination, alors l'application de BMC conduira à une expression rationnelle équivalente à :

- ☒  $a^*b^*c^*$ 
☐  $(a + b + c)^*$ 
☐  $(abc)^*$ 
☒  $a^* + b^* + c^*$

Q.17 Sur  $\{a, b\}$ , quel automate reconnaît le complémentaire du langage de ?



### 3 Grammaires et Machines abstraites

Q.19 Une machine de Turing nondéterministe

- ☒ est sûrement plus efficace qu'une machine de Turing déterministe  
☐ permet d'aboutir à une réponse là où les machines déterministes échouent  
☐ ne sait pas ce qu'elle fait ☐ gère les ensembles flous

Q.19 Quel type de machines abstraites reconnaît les langages rationnels ?

- ☒ les automates ☐ les machines de Turing bornées linéairement  
☐ les automates à pile déterministes ☐ les automates à pile  
☐ les machines de Turing

Q.19 Quel type de machines abstraites reconnaît les langages sensibles au contexte ?

- ☐ les automates à pile ☐ les automates  
☒ les machines de Turing bornées linéairement ☐ les machines de Turing  
☐ les automates à pile déterministes

Q.19 L'équation  $P \subseteq NP$  signifie

- ☐ un problème de résolution d'équations polynomiales est plus facile qu'un problème de résolution d'équations exponentielles  
☐ un problème soluble par une machine de Turing à une bande  $P$  est soluble par une machine de Turing ayant en plus une bande  $N$ .  
☒ on ne perd pas de performances en ayant plus de crus  
☒ les problèmes solubles dans un polynôme précipitent dans une solution non polynomiale

Q.13 Quel type de machines abstraites reconnaît les langages hors-contexte ?

- ☐ les automates à pile déterministes
- ☐ les machines de Turing
- ☒ les automates à pile
- ☒ les machines de Turing bornées linéairement
- ☐ les automates

Q.20 Quelle est la classe de la grammaire suivante?  $S \rightarrow Sac \mid c$

- ☐ Sensible au contexte
- ☐ Choix Finis
- ☒ Rationnelle
- ☐ Monotone
- ☐ Hors contexte

Q.21 Quelle est la classe de la grammaire suivante?  $S \rightarrow SaS \mid c$

- ☐ Sensible au contexte
- ☐ Rationnelle
- ☐ Monotone
- ☒ Hors contexte
- ☐ Choix Finis

Q.20 Quelle est la classe de la grammaire suivante?  $S \rightarrow aS \mid Sb \mid c$

- ☒ Hors contexte
- ☐ Rationnelle
- ☐ Sensible au contexte
- ☐ Choix Finis
- ☐ Monotone

Q.14 Quelle est la classe de la grammaire suivante?  $S \rightarrow aSb \mid c$

- ☐ Monotone
- ☒ Hors contexte
- ☐ Rationnelle
- ☐ Sensible au contexte
- ☐ Choix Finis

Q.20 Quelle est la classe de la grammaire suivante?

$$S \rightarrow abc \mid aSQ \quad bQc \rightarrow bbcc \quad cQ \rightarrow Qc$$

- ☐ Choix Finie
- ☐ Rationnelle
- ☐ Sensible au contexte
- ☒ Hors contexte
- ☒ Monotone

Q.21 Quelle est la classe de la grammaire suivante?

$$\begin{array}{lll} S \rightarrow abc \mid aSQ & CQ \rightarrow CX & QX \rightarrow QC \\ bQC \rightarrow bbCC & CX \rightarrow QX & C \rightarrow c \end{array}$$

- ☐ Hors contexte
- ☒ Sensible au contexte
- ☐ Choix Finis
- ☒ Monotone
- ☐ Rationnelle

Q.23 Quelle propriété cette grammaire vérifie?  $S \rightarrow Sac \mid c$

- ☐ Hors contexte
- ☒ Linéaire à gauche
- ☒ Ambigüe
- ☐ Linéaire à droite

Q.23 Quelle propriété cette grammaire vérifie?  $S \rightarrow aSc \mid c$

- ☐ Linéaire à droite
- ☐ Ambigüe
- ☒ Hors contexte
- ☒ Linéaire à gauche

Q.22 Quelle propriété cette grammaire vérifie?  $S \rightarrow SpS \mid n$

- ☐ Rationnelle
- ☐ Linéaire à gauche
- ☒ Ambigüe
- ☐ Linéaire à droite

Q.21 Quelle est la classe du langage  $\{a^n b^n c^n \mid n \in \mathbb{N}\}$  ?

- |   |   |
|---|---|
| <input type="checkbox"/> Général (Type 0)       | <input checked="" type="checkbox"/> Sensible au contexte (Type 1) |
| <input type="checkbox"/> Hors contexte (Type 2) | <input type="checkbox"/> Fini (Type 4)                            |
| <input type="checkbox"/> Rationnel (Type 3)     |   |

Q.20 Quelle est la classe du langage  $\{a^n \mid n \in \mathbb{N}\}$  ?

- |  |   |
|--|---|
| <input type="checkbox"/> Fini (Type 4)                 | <input type="checkbox"/> Hors contexte (Type 2) |
| <input checked="" type="checkbox"/> Rationnel (Type 3) | <input type="checkbox"/> Général (Type 0)       |
| <input type="checkbox"/> Sensible au contexte (Type 1) |   |

Q.21 Quelle est la classe du langage  $\{a^n b^n \mid n \in \mathbb{N}\}$  ?

- |  |   |
|--|---|
| <input type="checkbox"/> Fini (Type 4)                     | <input type="checkbox"/> Rationnel (Type 3) |
| <input type="checkbox"/> Sensible au contexte (Type 1)     | <input type="checkbox"/> Général (Type 0)   |
| <input checked="" type="checkbox"/> Hors contexte (Type 2) |   |

Q.21 Quelle est la classe de la grammaire suivante ?

$$\begin{array}{lll} A \rightarrow aABC & CB \rightarrow BC & bC \rightarrow bc \\ A \rightarrow abC & bB \rightarrow bb & cC \rightarrow cc \end{array}$$

- |   |   |
|---|---|
| <input type="checkbox"/> Rationnelle (Type 3)         | <input checked="" type="checkbox"/> Sensible au contexte (Type 1) |
| <input checked="" type="checkbox"/> Monotone (Type 1) | <input type="checkbox"/> Choix Finis (Type 4)                     |
| <input type="checkbox"/> Hors contexte (Type 2)       |   |

Q.15 Quelle est la classe de la grammaire suivante ?  $P \rightarrow P \text{ "stm" ";" } \mid \text{ "stm" ";" }$

- |  |  |
|--|--|
| <input type="checkbox"/> Hors contexte (Type 2)          | <input type="checkbox"/> Sensible au contexte (Type 1) |
| <input type="checkbox"/> Monotone (Type 1)               | <input type="checkbox"/> Finie (Type 4)                |
| <input checked="" type="checkbox"/> Rationnelle (Type 3) |  |

Q.19 Quel type de machines abstraites reconnaît les langages de type général (type 0) ?

- |  |  |
|--|--|
| <input type="checkbox"/> les machines de Turing bornées linéairement | <input checked="" type="checkbox"/> les automates          |
| <input type="checkbox"/> les automates à pile déterministes          | <input checked="" type="checkbox"/> les machines de Turing |
| <input type="checkbox"/> les automates à pile                        |  |

Q.22 Toute grammaire hors contexte ambiguë produit un langage...

- |  |  |                                 |                                    |
|--|--|---------------------------------|------------------------------------|
| <input type="checkbox"/> non rationnel | <input checked="" type="checkbox"/> non vide | <input type="checkbox"/> infini | <input type="checkbox"/> rationnel |
|--|--|---------------------------------|------------------------------------|

Q.23 Il existe un formalisme qui permette une description finie de tout langage.

- |  |                               |   |   |
|--|-------------------------------|---|---|
| <input checked="" type="checkbox"/> Non. | <input type="checkbox"/> Oui. | <input type="checkbox"/> Ça dépend de l'alphabet. | <input type="checkbox"/> Ça dépend du formalisme. |
|--|-------------------------------|---|---|

Q.19 Un transducteur est

- ☐ un automate infini  
☐ un automate fini avec des transductions spontanées  
☐ un élément de transistor  
☒ une machine ayant une entrée et une sortie

Q.23 Une grammaire hors contexte est ambiguë ssi il existe

- ☐ un automate nondéterministe qui reconnaisse ses arbres de dérivation.  
☒ un mot ayant deux arbres de dérivation.  
☐ un mot ayant une dérivation droite et une dérivation gauche.  
☐ une dérivation gauche (ou droite) ayant deux arbres de dérivation.

## 4 Analyseurs

Q.24 Les "start conditions" de Lex/Flex (%s et %x) permettent

- ☐ la conversion des chaînes de chiffres en la valeur qu'elles représentent  
☐ le choix du parseur à utiliser  
☐ de déterminer quand l'analyse lexicale doit commencer  
☒ de supporter différents contextes lexicaux

Q.25 Un parser sert à

- ☐ segmenter un flux de caractères en un flux de tokens  
☐ construire un analyseur syntaxique  
☐ éliminer les récursions terminales  
☐ s'assurer de la correction du typage  
☒ faire de l'analyse syntaxique

Q.26 Comment désambigüiser pour Yacc/Bison le morceau d'arithmétique suivant :

exp: exp '+' exp | exp '-' exp | NUM;

- ☐ %left '+' %left '-'  
☐ %left '+' %left '-' %nonassoc NUM  
☒ %left '+' '-'  
☐ %left '-' %left '+'

Q.27 Avec la grammaire suivante, quel état atteint l'automate LR(1) après une transition sur E puis sur '?'?

$S \rightarrow E \$$   
 $E \rightarrow E ? E : E \mid E + E \mid 0$


- ☐  $S \rightarrow E \bullet \$$  [\$]  
 $E \rightarrow E \bullet ? E : E$  [\$?+]  
 $E \rightarrow E \bullet + E$  [\$?+]  
☒  $E \rightarrow E ? \bullet E : E$  [\$?+]  
 $E \rightarrow \bullet E ? E : E$  [?+ :]  
 $E \rightarrow \bullet E + E$  [?+ :]  
 $E \rightarrow \bullet 0$  [?+ :]

- ☐  $E \rightarrow E ? \bullet E : E$  [\$?+]  
 $E \rightarrow \bullet E ? E : E$  [\$?+ :]  
 $E \rightarrow \bullet E + E$  [\$?+ :]  
 $E \rightarrow \bullet 0$  [\$?+ :]  
☐  $S \rightarrow E \bullet \$$  [\$]  
 $E \rightarrow E \bullet ? E : E$  [\$?+ :]  
 $E \rightarrow E \bullet + E$  [\$?+ :]

- ☐  $E \rightarrow E ? \bullet E : E$  [\$?+]  
 $S \rightarrow \bullet E \$$  [\$]  
 $E \rightarrow \bullet E ? E : E$  [\$?+ :]  
 $E \rightarrow \bullet E + E$  [\$?+ :]  
 $E \rightarrow \bullet 0$  [\$?+ :]

Q.19 Lex/Flex sont des

- ☐ générateurs de parsers      ☐ parseurs      ☒ scanners      ☒ générateurs de scanners  
☐ générateurs de préprocesseurs

Q.20  Quels sont les adjectifs usuels pour désigner un parseur LL :

- ☐ ascendant      ☒ descendant      ☒ prédictif      ☐ additif      ☐ multiplicatif  
☒ récursif      ☐ itératif

Q.21 Si une grammaire hors contexte est non ambiguë, alors...

- ☐ elle est LL(k)      ☒ elle n'est pas nécessairement LL  
☐ elle produit nécessairement des conflits dans un parseur LL      ☐ elle est LL(1)

Q.25 LL(k) signifie

- ☒ lecture en une passe de gauche à droite, avec  $k$  symboles de regard avant  
☐ lecture en une passe de gauche à droite, avec une pile limitée à  $k$  symboles  
☐ lecture en deux passes de gauche à droite, avec  $k$  symboles de regard avant  
☐ lecture en deux passes de gauche à droite, avec une pile limitée à  $k$  symboles

## 5 Logique Propositionnelle

Soit le langage de la logique propositionnelle, composé de deux symboles  $\top$  (vrai) et  $\perp$  (faux), de l'opération unaire  $\neg$  (non), des opérations binaires  $\vee$  (ou) et  $\wedge$  (et), et des parenthèses notées  $[, ]$ . Ce langage inclut des mots tels que  $\perp \wedge \perp$ ,  $\top \vee \perp$  et  $\neg[\top \wedge \top] \vee [\perp \wedge \perp]$ .

Q.28 Que dire de la grammaire suivante?

$$S \rightarrow S \wedge S \mid S \vee S \mid \neg S \mid [S] \mid \top \mid \perp \quad (G_1)$$

- ☐ rationnelle      ☒ ambiguë      ☐ non ambiguë      ☐ infiniment ambiguë

Q.29 Dans la grammaire suivante, quelles sont les priorités/associativités des opérateurs

$$S \rightarrow S \vee T \mid T \quad T \rightarrow T \wedge F \mid F \quad F \rightarrow \neg F \mid [S] \mid \top \mid \perp \quad (G_2)$$

- ☐  $\wedge$  et  $\vee$  associatives à droite, priorités croissantes :  $\neg < \wedge < \vee$   
☐  $\wedge$  et  $\vee$  associatives à droite, priorités croissantes :  $\vee < \wedge < \neg$   
☐  $\wedge$  et  $\vee$  associatives à gauche, priorités croissantes :  $\neg < \wedge < \vee$   
☒  $\wedge$  et  $\vee$  associatives à gauche, priorités croissantes :  $\vee < \wedge < \neg$

Q.30 Que dire de la grammaire  $(\hat{G}_2)$ ?

- ☒ non ambiguë et non LL(1)      ☐ ambiguë et LL(1)  
☐ non ambiguë et LL(1)      ☐ ambiguë et non LL(1)

Q.31 Que dire de la grammaire suivante par rapport à  $(G_2)$ ?

$$\begin{aligned} S &\rightarrow TS' & T &\rightarrow FT' & F &\rightarrow \neg F \mid [S] \mid \top \mid \perp & (G_3) \\ S' &\rightarrow \vee TS' \mid \varepsilon & T' &\rightarrow \wedge FT' \mid \varepsilon \end{aligned}$$

- ☒ même langage, priorités et/ou associativités différentes, mais LL(1)  
☐ même langage, mêmes priorités et associativités, pas LL(1)  
☐ même langage, priorités et/ou associativités différentes, pas LL(1)  
☐ langage différent  
☐ même langage, mêmes priorités et associativités, mais LL(1)

Q.32 Quels sont les symboles annulables dans la grammaire  $(G_3)$ ?

- ☐  $S, S', T, T', F$     ☐  $S, T, F$     ☐  $S', T', F$     ☒  $S', T'$     ☐  $F$

Q.33 Quels sont les FIRST dans la grammaire  $(G_3)$ ?

<input type="checkbox"/>	FIRST	<input type="checkbox"/>	FIRST	<input type="checkbox"/>	FIRST	<input checked="" type="checkbox"/>	FIRST
$S$	$\neg[\top \perp$	$S$	$\neg[\top \perp$	$S$	$T$	$S$	$\neg[\top \perp$
$S'$	$\varepsilon \vee$	$S'$	$\vee \wedge$	$S'$	$\vee$	$S'$	$\vee$
$T$	$\neg[\top \perp$	$T$	$\neg[\top \perp$	$T$	$F$	$T$	$\neg[\top \perp$
$T'$	$\varepsilon \wedge$	$T'$	$\vee \wedge$	$T'$	$\wedge$	$T'$	$\wedge$
$F$	$\neg[\top \perp$	$F$	$\neg[\top \perp$	$F$	$\neg[\top \perp$	$F$	$\neg[\top \perp$

Q.34 Quels sont les FOLLOW dans la grammaire  $(G_3)$ ?

<input type="checkbox"/>	FOLL	<input type="checkbox"/>	FOLL	<input type="checkbox"/>	FOLL	<input checked="" type="checkbox"/>	FOLL	<input type="checkbox"/>	FOLL
$S$	$\varepsilon]$	$S$	$]$	$S$	$]$	$S$	$]$	$S$	$]$
$S'$	$\vee \wedge]$	$S'$	$]$	$S'$	$\vee \wedge]$	$S'$	$]$	$S'$	$]$
$T$	$\vee]$	$T$	$\vee]$	$T$	$\vee]$	$T$	$\vee]$	$T$	$]$
$T'$	$\vee]$	$T'$	$\vee]$	$T'$	$\vee]$	$T'$	$\vee]$	$T'$	$]$
$F$	$\wedge \vee]$	$F$	$\vee]$	$F$	$\wedge \vee]$	$F$	$\wedge \vee]$	$F$	$]$

Q.35 Que dire de la grammaire étendue suivante par rapport à  $(G_2)$ ?

$$S \rightarrow T(\vee T)^* \quad T \rightarrow F(\wedge F)^* \quad F \rightarrow \neg F \mid [S] \mid \top \mid \perp \quad (G_4)$$

- ☐ même langage, priorités et/ou associativités différentes, mais LL(1)  
☐ même langage, priorités et/ou associativités différentes, pas LL(1)  
☒ même langage, mêmes priorités et associativités, mais LL(1)  
☐ même langage, mêmes priorités et associativités, pas LL(1)  
☐ langage différent

**Q.36** Quelle routine parse et calcule correctement S pour la grammaire ( $G_4$ ) de la logique booléenne ? La variable la désigne le lookahead courant, et la routine eat (*expect*) vérifie que le lookahead actuel est *expect* puis stocke le suivant dans la.



```
bool S()
{
    bool res = false;
    do
    {
        eat('v');
        res |= T();
    }
    while (la == 'v');
    return res;
}
```



```
bool S()
{
    bool res = T();
    while (la == 'v')
    {
        eat('v');
        res |= F();
        while (la == '^')
        {
            eat('^');
            res &= F();
        }
    }
    return res;
}
```



```
bool S()
{
    bool res = true;
    do
    {
        eat('v');
        res |= T();
    }
    while (la == 'v');
    return res;
}
```



```
bool S()
{
    bool res = T();
    while (la == 'v')
    {
        res |= T();
        eat('v');
    }
    return res;
}
```



```
bool S()
{
    bool res = T();
    while (la == 'v')
    {
        eat('v');
        res |= T();
    }
    return res;
}
```

Q.37 Quelle est la séquence de décalages/réductions pour un parser Yacc/Bison implémentant la grammaire ( $G_1$ ) avec des directives précisant correctement priorités et associativités ?



```

┌      T ^ T V T →
s ┌ "T"      ^ T V T →
r ┌ S      ^ T V T →
s ┌ S "Λ"      T V T →
s ┌ S "Λ" "T"      V T →
r ┌ S "Λ" S      V T →
s ┌ S "Λ" S "V"      T →
s ┌ S "Λ" S "V" "T" →
r ┌ S "Λ" S "V" S →
r ┌ S      →
s ┌ S →
accept

```



```

┌      T ^ T V T →
s ┌ "T"      ^ T V T →
r ┌ S      ^ T V T →
s ┌ S "Λ"      T V T →
s ┌ S "Λ" "T"      V T →
r ┌ S "Λ" S      V T →
r ┌ S      V T →
s ┌ S "V"      T →
s ┌ S "V" "T"      →
r ┌ S "V" S      →
r ┌ S      →
s ┌ S →
accept

```



```

┌      T ^ T V T →
s ┌ "T"      ^ T V T →
r ┌ S      ^ T V T →
s ┌ S "Λ"      T V T →
s ┌ S "Λ" "T"      V T →
r ┌ S      V T →
s ┌ S "V"      T →
s ┌ S "V" "T"      →
r ┌ S      →
s ┌ S →
accept

```



```

┌      T ^ T V T →
s ┌ "T"      ^ T V T →
r ┌ S      ^ T V T →
s ┌ S "Λ"      T V T →
s ┌ S "Λ" "T"      V T →
s ┌ S "Λ" "T" "V"      T →
s ┌ S "Λ" "T" "V" "T" →
r ┌ S "Λ" "T" "V" S →
r ┌ S "Λ" S      →
r ┌ S      →
s ┌ S →
accept

```