Correction du Partiel THL
Théorie des Langages

EPITA – Promo 2014 – **Sans document ni machine**

Janvier 2012 (2h)

Correction: Le sujet et sa correction ont été écrits par Akim Demaille et Jonathan Fabrizio.

Barème: Toutes les questions rédactionnelles sont notées sur 4, et c'est ainsi qu'il faut lire les barèmes qui suivent. Se reporter à la feuille de calcul pour voir les coefficients donnés aux questions.

Best-of: Le best-of est tiré des copies des étudiants. Les erreurs, en particulier d'orthographe, sont les leurs ! Les commentaires du correcteur sont notés [*ainsi*].

Bien lire les questions, chaque mot est important. Écrire court, juste, et bien. Une argumentation informelle mais convaincante est souvent suffisante.

Répondre aux questions à choix multiples (numérotées Q.1, Q.2 etc.) sur les formulaires de QCM; aucune réponse manuscrite ne sera corrigée. Renseigner les champs d'identité. Sauf mention contraire, il y a exactement une et une seule réponse juste pour ces questions. Si plusieurs réponses sont valides, sélectionner la plus restrictive. Par exemple s'il est demandé si 0 est *nul*, *non nul*, *positif*, ou *négatif*, cocher *nul* qui est plus restrictif que *positif* et *négatif*, tous deux vrais.

1 Incontournables

Chaque erreur (ou non réponse) aux trois questions suivantes retire 1/6 de la note finale. Avoir tout faux divise donc la note par 2.

Q.1 L'union de deux langages rationnels est rationnelle.

✓ vrai ✗ faux

Correction: Oui. Par définition.

Q.2 Un sous-langage (i.e., un sous-ensemble) d'un langage rationnel est rationnel.

✗ vrai ✓ faux

Correction: Bien sûr que non. Tout langage (comme $a^n b^n$ bien connu pour ne pas être rationnel) est partie de Σ^*, qui est rationnel.

Q.3 Pour toute grammaire hors-contexte non ambiguë, il existe un automate fini non-déterministe qui en reconnaît le langage engendré.

✗ vrai ✓ faux

Correction: Faux. Aucun rapport entre les deux parties de cette question. Une grammaire hors-contexte n'engendre pas nécessairement un langage rationnel. Un automate reconnaît uniquement un langage rationnel, indépendamment de son déterminisme.

2 Contrôle

Pour les questions suivantes, une réponse fausse entraîne une pénalité. Pas de réponses donne 0.

Q.4 L'expression rationnelle étendue $[\_]^?[a-z]^*[a-zA-Z0-9]^+[a-zA-Z0-9\_]^+$ n'engendre pas:

✗ *TEST* ✓ *__STDC__* ✗ *eval_expr* ✗ *exit_42*

Q.5 Soit les langages rationnels suivants : A, B et N. Le langage $ANBN$:

✗ est un langage hors-contexte ;
✓ est reconnaissable par un automate à états fini déterministe ;
✗ n'est évidement pas rationnel ;
✗ nécessiterait une mémoire infinie pour le reconnaître.

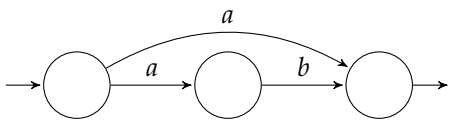
Q.6 Soit une expression rationnelle α et un automate A. Est-il possible de déterminer s'ils correspondent au même langage ?

✗ c'est parfois possible ;
✓ c'est possible en temps fini ;
✗ c'est possible en temps infini ;
✗ c'est impossible.

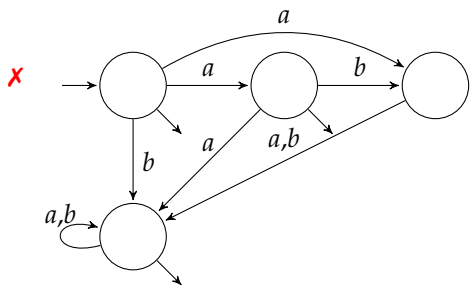
Q.7 J'ai une expression rationnelle assez longue α. J'ai construit, à l'aide de l'algorithme de Thompson un automate A reconnaissant α. Combien d'états a-t-il ?

✗ 27 ✗ 37 ✓ 72 ✗ 73

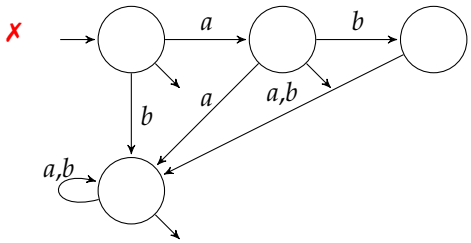
Q.8 Quel automate reconnaît le langage complémentaire du langage reconnu par l'automate suivant (sur l'alphabet $\Sigma = \{a, b\}$) :



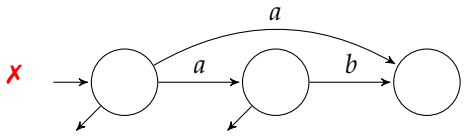
✗



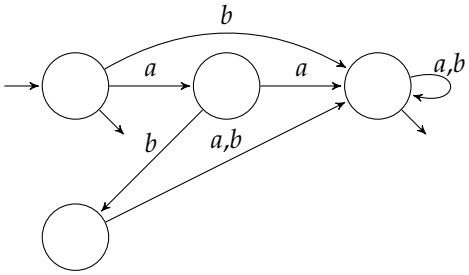
✗



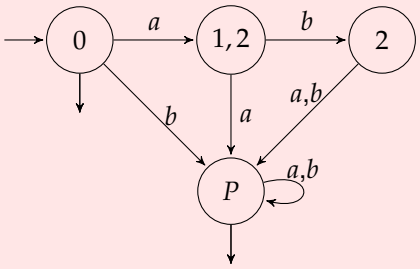
✗



✓



Correction: La bonne approche consiste à déterminiser l'automate, puis à le compléter, et enfin à le complémenter. Avec une disposition des états plus naturelle, on obtient:



Q.9 Quelle est la classe de la grammaire suivante ?

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

✗ Rationnelle (Type 3)
✗ Hors contexte (Type 2)
✗ Sensible au contexte (Type 1)
✓ Monotone (Type 1)

Q.10 Quelle est la classe de la grammaire suivante ?

$$S \rightarrow aSb \mid c$$

✗ Rationnelle
✓ Hors contexte
✗ Sensible au contexte
✗ Monotone

Q.11 Si le parseur LALR(1) associé à une grammaire présente des conflits

✗ Il n'existe pas de parseur pour cette grammaire ;
✗ Il peut exister un parseur LR(0) pour cette grammaire ;
✗ Il peut exister un parseur LLR(1) pour cette grammaire ;
✓ Il peut exister un parseur LR(1) pour cette grammaire ;
✗ Il peut exister un parseur SLR(1) pour cette grammaire.

Q.12 LL(k) signifie

✗ 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 ;
✓ 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.

Q.13 Dans une analyse classique en utilisant Lex et Yacc:

✓ on appelle la fonction `yyparse` une fois, elle appelle la fonction `yylex` plusieurs fois;
✗ on appelle la fonction `yyparse(yylex())` plusieurs fois;
✗ on appelle la fonction `yylex` plusieurs fois, puis la fonction `yyparse` une fois;
✗ on appelle la fonction `yyparse` plusieurs fois, elle appelle la fonction `yylex` chaque fois.

3 Dessine-moi un programme

Considérez la grammaire suivante, qui s'inspire d'un sous-ensemble du langage Logo.

```
⟨instr⟩  ::= "forward" ⟨exp⟩
           | "left" ⟨exp⟩
           | "right" ⟨exp⟩

           # Loop.
           | "repeat" ⟨exp⟩ "[" ⟨instrs⟩ "]"

           # Subprogram definition.
           | "to" "id" ⟨args⟩ ⟨instrs⟩ "end"

           # Subprogram call.
           | "id"

⟨exp⟩      ::= "num"
           | ":" "id"

⟨args⟩     ::= ⟨args⟩ ⟨arg⟩
           |
```

Dans cette grammaire, le terminal `"num"` désigne un entier littéral et le terminal `"id"` un identifiant (qui suit les mêmes règles lexicales que les identifiants du langage Tiger).

Le langage Logo permet (entre autres) de dessiner sur un écran à l'aide d'une "tortue" que l'on manipule avec des instructions comme `forward 10` (avance de 10 unités en traçant un trait) ou `right 30` (tourne à droite de 30 degrés).

Q.14 La grammaire donnée ne décrit qu'une seule instruction, mais on voudrait pouvoir en avoir plusieurs (mais au moins une). Quelles règles est-il préférable d'ajouter si l'on veut utiliser un parseur LR(1) :

✓
```
⟨instrs⟩ ::= ⟨instr⟩ | ⟨instrs⟩ ⟨instr⟩
```

✗
```
⟨instrs⟩ ::= ⟨instr⟩ | ⟨instr⟩ ⟨instrs⟩
```

✗
```
⟨instrs⟩ ::= | ⟨instrs⟩ ⟨instr⟩
```

✗
```
⟨instrs⟩ ::= | ⟨instr⟩ ⟨instrs⟩
```

Q.15 Si l'on voulait faire la même chose avec un parseur LL(1), on aurait écrit :

✗
```
⟨instrs⟩ ::= ⟨instr⟩ | ⟨instrs⟩ ⟨instr⟩
```

✓
```
⟨instrs⟩ ::= ⟨instr⟩ | ⟨instr⟩ ⟨instrs⟩
```

✗
```
⟨instrs⟩ ::= | ⟨instrs⟩ ⟨instr⟩
```

✗
```
⟨instrs⟩ ::= | ⟨instr⟩ ⟨instrs⟩
```

Arithmétique

On souhaite donner la possibilité à l'utilisateur de saisir des expressions arithmétiques afin de donner plus de flexibilité. Ainsi nous modifions les règles <*exp*> :

```
⟨exp⟩ ::= ⟨exp⟩ "+" ⟨exp⟩ | ⟨exp⟩ "-" ⟨exp⟩ | ⟨exp⟩ "*" ⟨exp⟩ | ⟨exp⟩ "/" ⟨exp⟩
        | ":" "id"
        | "num"
```

Malheureusement, ces règles introduisent des ambiguïtés. Elles ne codent pas l'associativité gauche comme on le souhaiterait ni la priorité de "*" et "/" sur "+" et "-".

1. Donner deux arbres de **dérivation** de l'expression :

   :step * 12 + :iteration

   Correction: Beaucoup d'étudiants ont confondu arbre de dérivation et AST.

2. Proposer une grammaire non ambiguë équivalente à la grammaire de <*exp*> pour résoudre les conflits et pour respecter l'associativité gauche ainsi que la priorité des opérateurs. Profitez-en pour ajouter les parenthèses.

   Correction: Fait en cours...

```
⟨exp⟩    ::= ⟨exp⟩ "+" ⟨term⟩
           | ⟨exp⟩ "-" ⟨term⟩
           | ⟨term⟩
           ;

⟨term⟩   ::= ⟨term⟩ "*" ⟨factor⟩
           | ⟨term⟩ "/" ⟨factor⟩
           | ⟨factor⟩
           ;

⟨factor⟩ ::= ":" "id"
           | "num"
           | "(" ⟨exp⟩ ")"
           ;
```

3. Avec Bison, quelles directives ajouteriez-vous, sans changer les règles de <exp>, pour respecter l'associativité et les priorités.

   Correction: Fait en cours aussi. . .

```
%left "+" "-"
%left "/" "*"
```

4. L'utilisation des nouvelles règles de <exp> dans la grammaire initiale introduit-elle des conflits ? Si oui, lesquels ?

   Correction: On pourrait le croire, par exemple avec `"forward" exp . * 12`: faut-il réduire <instr> ou décaler? Il est simple de voir que le décalage n'est pas admissible en LALR(1): il ne l'est même pas en SLR(1), puisque * n'est pas dans *FOLLOW*(exp).

5. Compléter, jusqu'à l'acceptation, la séquence de décalages/réductions suivante :

```
   ⊢            forward : step * 12 + : iteration ⊣
s  ⊢ "forward"          : step * 12 + : iteration ⊣
s  ⊢ "forward" ":"        step * 12 + : iteration ⊣
s  ⊢ "forward" ":" "id"         * 12 + : iteration ⊣
r  ⊢ "forward" exp              * 12 + : iteration ⊣
```

Correction:

```
   ⊢            forward : step * 12 + : iteration ⊣
s  ⊢ "forward"           : step * 12 + : iteration ⊣
s  ⊢ "forward" ":"         step * 12 + : iteration ⊣
s  ⊢ "forward" ":" "id"          * 12 + : iteration ⊣
r  ⊢ "forward" exp               * 12 + : iteration ⊣
r  ⊢ "forward" exp               * 12 + : iteration ⊣
s  ⊢ "forward" exp "*"             12 + : iteration ⊣
s  ⊢ "forward" exp "*" "num"          + : iteration ⊣
r  ⊢ "forward" exp "*" exp            + : iteration ⊣
r  ⊢ "forward" exp                    + : iteration ⊣
s  ⊢ "forward" exp "+"                  : iteration ⊣
s  ⊢ "forward" exp "+" ":"                iteration ⊣
s  ⊢ "forward" exp "+" ":" "id"                     ⊣
r  ⊢ "forward" exp "+" exp                          ⊣
r  ⊢ "forward" exp                                  ⊣
r  ⊢ instr                                          ⊣
s  ⊢ instr ⊣
a
```

Beaucoup n'ont pas mis les actions ou ne sont pas allés jusqu'au bout. De même que certains ont réduit trop tard en `<exp>` changeant ainsi l'associativité...

4 À propos de ce cours

Nous nous engageons à ne pas tenir compte des renseignements ci-dessous pour noter votre copie. Ils ne sont pas anonymes, car nous sommes curieux de confronter vos réponses à votre note. En échange, quelques points seront attribués pour avoir répondu. Merci d'avance.

Répondez sur les formulaires de QCM qui vous sont remis. Vous pouvez cocher plusieurs réponses par question.

Q.16 Prises de notes

a. Aucune
b. Sur papier
c. Sur ordinateur à clavier
d. Sur ardoise
e. Sur le journal du jour

Q.17 Travail personnel

a. Rien
b. Bachotage récent
c. Relu les notes entre chaque cours
d. Fait les annales
e. Lu d'autres sources

Q.18 Ce cours

a. Est incompréhensible et j'ai rapidement abandonné
b. Est difficile à suivre mais j'essaie
c. Est facile à suivre une fois qu'on a compris le truc
d. Est trop élémentaire

Q.19 Ce cours

a. Ne m'a donné aucune satisfaction
b. N'a aucun intérêt dans ma formation
c. Est une agréable curiosité
d. Est nécessaire mais pas intéressant
e. Je le recommande

Q.20 L'enseignant

a. N'est pas pédagogue
b. Parle à des étudiants qui sont au dessus de mon niveau
c. Me parle
d. Se répète vraiment trop
e. Se contente de trop simple et devrait pousser le niveau vers le haut