Correction du Partiel THL
Théorie des Langages

EPITA – Promo 2011 – **Sans documents ni machine**

Novembre 2008(1h30)

Correction: Le sujet et sa correction ont été écrits par Akim Demaille. Le best of est tiré des copies des étudiants. Les erreurs, en particulier d'orthographe, sont les leurs !

Barème: Toutes les questions 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.

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

Les questions à choix multiples, numérotées Q.1, Q.2 etc., sont à répondre sur les formulaires de QCM ; aucune réponse manuscrite ne sera corrigée. 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*, sélectionner *nul* qui est plus restrictif que *positif* et *négatif*, tous deux vrais. Ne pas oublier de renseigner les champs d'identité.

1 Incontournables

Une pénalité sur la note finale sera appliquée pour les erreurs sur ces questions.

Correction: Chaque erreur aux trois questions suivantes retire 1/6 de la note finale. Avoir tout faux divise donc la note par 2. D'assez nombreux étudiants n'ont pas été capables de mettre les réponses sur le formulaire, mais l'ont fait dans leur copie. Dans ce cas, ils ont eu faux aux trois questions.

Q.1 Si un automate est non-déterministe, alors il n'est pas déterministe. a. vrai/b. faux ?

Correction: Non, il peut être déterministe, d'où l'écriture en un seul mot « non-déterministe » par opposition à « non déterministe ».
65% ont juste en promo 2009, 74% en 2010.

Q.2 Si L_1, L_2 sont des langages rationnels, alors $\{u^n v^n | u \in L_1, v \in L_2, n \in \mathbb{N}\}$ est rationnel.
a. vrai/b. faux ?

Correction: Bien sûr que non, comme le montre le cas bien connu de $L_1 = \{a\}, L_2 = \{b\}$.

Q.3 L'intersection entre un langage rationnel et un langage quelconque est toujours rationnelle.
a. vrai/b. faux ?

Correction: Le langage Σ^* est rationnel, et son intersection avec tout langage L est L. Donc bien sûr que non, il suffit de prendre L pas rationnel.

2 Contrôle

Q.4 L'expression rationnelle étendue `[-+]?[0-9A-F]+([-+]?[-+/*][0-9A-F]+)*` n'engendre pas :

a. -42
b. $42 + 42$
c. $42 + (42 * 42)$
d. $-42 - -42$

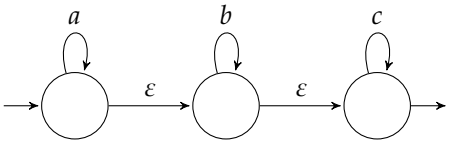
Q.5 On doit l'invention de LL et LR à. . .

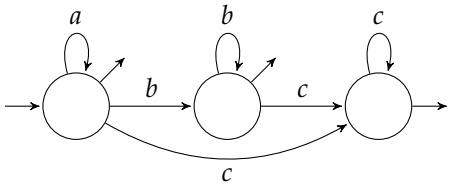
a. Frank de Remer
b. Noam Chomsky
c. Donald Knuth
d. Stephen Kleene

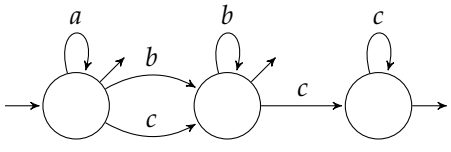
Q.6 Combien y a-t-il de kilo-octets dans un méga-octet ?

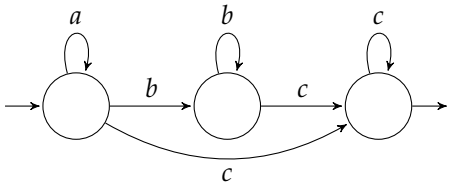
a. 256
b. 1000
c. 1024
d. 2^{16}

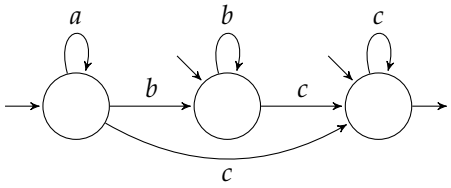
Q.7 Quelle est le résultat d'une ε-clôture *arrière* sur l'automate suivant ?



a.


b.


c.


d.


3 Parsage LL

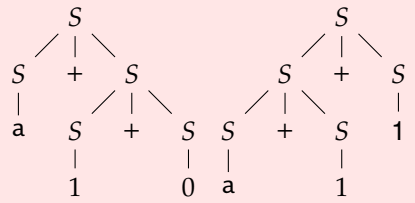
Considérons une grammaire pour les expressions rationnelles.

$$S \to 0 \mid 1 \mid \mathsf{a} \mid S + S \mid S \cdot S \mid S^* \mid (S)$$

Les parenthèses permettent de grouper, a représente une lettre de l'alphabet (c'est un terminal), et 1 désigne le mot vide des expressions rationnelles (qu'on prendra soin de ne pas confondre avec le mot vide dans nos grammaires). Lorsque l'on parle des opérateurs (infixes) binaires, on se réfère à $e + f$ et $e \cdot f$. L'étoile de Kleene, e^*, est un opérateur postfixe.

1. Montrer deux arbres de dérivation de a + 1 + 0.

Correction:

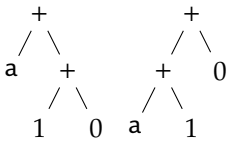


Barème:

Je n'ai pas pénalisé ceux, relativement nombreux, qui ont nommé ces arbres « dérivation à gauche » et « dérivation à droite ». Je le ferai à l'avenir, car c'est vraiment ne pas avoir compris la bijection entre dérivation gauche (ou droite d'ailleurs) et arbre de dérivation. Ici, ce qui compte c'est la forme de l'arbre, pas la chronologie de sa découverte (qui est ce qu'enregistre une dérivation).

0 si on me donne des arbres qui ne sont rien du tout, des batards entre AST et arbres de dérivation, ou encore des arbres de dérivations « simplifiés » (par exemple $S \to \mathsf{a} + 1$ etc.).

1 si on me donne des AST corrects :



2. Que peut-on dire de cette grammaire ?

Correction: La question précédente montre qu'elle est ambiguë : l'associativité de + n'est pas définie.

Barème:

0 si on me dit des horreurs (genre « cette grammaire est infinie »), même si vous avez aussi dit qu'elle était ambiguë.

1 si on me dit qu'elle est rationnelle (c'est pourtant une horreur : il y a les parenthèses bon sang !) ambiguë.

4 si on me dit qu'elle est ambiguë, ou hors-contexte et ambiguë.

Best-of:

- On voit que cette grammaire est de type 3. Elle est donc rationnelle. On remarque egalement qu'elle n'est pas ambiguë.
- De plus nous voyons clairement (notamment par le cas des parenthèses) que cette grammaire est hors contexte mais qu'elle reconnaît le langage rationnel.
- Cette grammaire est linéaire a gauche et à droite, donc ambigüe.
- Cette grammaire est régulière, infinie, linéaire à gauche et à droite. la grammaire est ambiguë. elle est de type 3.
- On peut dire beaucoup de choses de cette grammaire, comme par exemple, qu'elle est composée de divers symboles.

Q.8 Les opérateurs binaires sont pris associatifs à gauche. Les priorités des opérateurs sont, dans l'ordre croissant, $e + f$, puis $e \cdot f$, puis e^*.
Quelle forme parenthésée correspond à $\mathsf{a}^* + \mathsf{a}^* + \mathsf{a} \cdot \mathsf{a}^*$?

a. $((\mathsf{a}^*) + (\mathsf{a}^* + (\mathsf{a} \cdot (\mathsf{a}^*))))$

b. $((((\mathsf{a}^*) + \mathsf{a}^*) + \mathsf{a} \cdot (\mathsf{a}^*)))$

c. $(((\mathsf{a}^*) + \mathsf{a}^*) + (\mathsf{a} \cdot (\mathsf{a}^*)))$

d. $(((\mathsf{a}^*) + \mathsf{a}^*) + (\mathsf{a} \cdot (\mathsf{a}))^*)$

3. Étant données ces priorités et associativités, donner une grammaire non ambiguë des expressions rationnelles.

Correction:

$$\begin{array}{rcl} S & \to & S + T \mid T \\ T & \to & S \cdot F \mid F \\ F & \to & 0 \mid 1 \mid \mathsf{a} \mid F^* \mid (S) \end{array}$$

Barème: Bien sûr j'ai accepté l'introduction d'un symbole (inutile) pour l'étoile :

$$\begin{array}{rcl} S & \to & S + T \mid T \\ T & \to & S \cdot F \mid F \\ F & \to & A \mid F^* \\ A & \to & 0 \mid 1 \mid \mathsf{a} \mid (S) \end{array}$$

0/1 si l'on pose l'étoile sur $S : F \to S^* \mid \ldots$. Un problème comparable a été traité en cours — les opérateurs arithmétiques unaires — je ne vois pas de raison d'être spécialement clément.

2 si c'est correct à l'exception de l'étoile qu'on ne peut pas répéter ($F \to A \mid A^*$). Erreur trop courante.

2 si les priorités sont correctes, mais pas les associativités.

Best-of: [*En marge d'une proposition fausse :*] Les côtes déterminent les terminaux.

4. Pourquoi cette grammaire n'est pas LL(1) ?

Correction: Il est impossible en regardant le *lookahead* de distinguer les règles d'un nonterminal donné. Par exemple, les deux règles sur S ont le même préfixe T et par conséquent les mêmes First, ce qui conduit immanquablement à un conflit. La récursion gauche de la première règle interdit aussi le fait d'être LL(k).
On a également un problème avec * qui est un opérateur postfixe : le parsage de F n'est pas aussi simple que pour l'arithmétique.

Best-of:
– [*L'étudiant n'a pas répondu à la question précédente.*] Cette grammaire n'est pas LL(1) car elle est ambiguë. [*Mais à la question suivante il répond :*] Cette grammaire est LL(2).
– car d'une part, elle n'est pas ambiguë, et, d'autre part, on n'a pas besoin de regarder le lookahead pour savoir dans quel cas aller.
– car elle possède plusieurs états terminaux.

Barème:

1 si on me paraphrase la définition de « (1) » en me disant qu'un *lookahead* de 1 ne suffit pas. Clémence qui me surprend moi-même.

4 si on me parle de récursion gauche ou de FIRST non disjoints (j'accepte « FIRST identiques »).

5. Est-elle LL(2) ? Justifier.

Correction: Non, pour les mêmes raisons : T est un non terminal, et k *look-ahead* ne suffiront jamais pour trouver le terminal qui le suit, ce qui permettrait de distinguer les deux règles de S.

Best-of:
– S'agissant de toute évidence d'une question piège (ou pas), je préfère la prudence et passerait donc simplement aux questions (fort intéressantes) sur la factorielle.
– Une grammaire LL(k) implique que la grammaire est LL(1) ($k > 1$), donc cette grammaire n'est pas LL(2).
– Cette grammaire est LL(2) car chaque règle a un nombre d'éléments dans la partie droite qui ne dépasse pas 2[*Ce qui n'est d'ailleurs même pas vrai*].

6. Considérons la grammaire *factorielle* : $S \to \mathtt{n} \mid S!$ Expliquer pourquoi elle n'est pas LL(1).

Correction: Elle a une récursion gauche. Ce qui implique aussi que les deux règles ont le même First.

Best-of:
– n'est pas LL(1) car elle engendre une récursion à droite.
– Cette grammaire n'est pas LL(1) car elle n'est pas linéaire à gauche. *D'une part ça n'a pas de rapport, d'autre part elle est linéaire à gauche !)*
– Cette grammaire possède une dérivation gauche, elle n'est donc pas LL(1).
– [*Attention, celle-ci est vraiment balaise. . .*] car on ne connaît pas la priorité des opérateurs, en effet imaginons $3! = 3 \times 2 \times 1$ on ne sait pas s'il faut considérer $(3 \times 2) \times 1$ ou $3 \times (2 \times 1)$.

7. Convertir la grammaire factorielle en une grammaire **étendue**[1] à partir de laquelle il est simple de créer un analyseur LL.

Correction: Les règles 1 et 2 ont les mêmes ensembles First, {n}.

$$S \to n \ \{!\}$$

Ce qui s'analyse facilement par

```
S() { eat ('n'); while (la == '!') eat ('!'); }
```

Barème: Beaucoup trop d'entre vous n'ont pas compris la note de bas de page, et ont changé le langage étudié pour qu'il utilise les accolades au lieu de l'étoile ; il fallait faire le contraire. J'ai souvent été généreux (1 ou 2 points selon la clarté du résultat), mais en général le résultat n'est pas correct non plus.

[1] Pour éviter les collisions entre notre alphabet et les notations habituelles qui utilisent toutes deux les parenthèses et l'étoile, utilisez les notations suivantes pour les grammaires étendues : {A} signifie A^*, et on utilisera les crochets, [et], pour parenthéser.

Best-of:
- $S \to \{n\}$!
- $S \to n\{[n!]\}^*$!. Avec un analyseur LL(2) il est aisé de traiter $a^n b^n$. . .[*Je vous jure que c'est vraiment ce qui est écrit !*]
- [*D'assez nombreux étudiants ont écrit des grammaires qui ressemblent à celle-ci :*]

$$\begin{aligned} S &\to FS' \\ S' &\to (F-1)!|\varepsilon \\ F &\to n \end{aligned}$$

[*C'était un mystère pour moi, en particulier l'apparition du* -1 *jusqu'à ce que je comprenne qu'ils veulent* calculer *la factorielle ! C'est une erreur très grave, typique du problème qu'ont certains qui n'ont pas réussi à séparer syntaxe de sémantique, autrement dit séparer signifiant de signifié.*]
- $S \to n|n \times 1 + n|n \times 2 \cdots n|n \times n$ [*Non seulement il veut calculer, mais en plus il ne sait pas le faire.*]

8. Convertir la grammaire des expressions rationnelles de la question 3 en une grammaire **étendue** susceptible d'être LL. On prêtera attention à l'opérateur postfixe.

Correction:

$$\begin{aligned} S &\to T\{+T\} \\ T &\to F\{\cdot F\} \\ F &\to [0 \mid 1 \mid \mathrm{a} \mid (S)]\{^*\} \end{aligned}$$

Barème: 2 si c'est la grammaire est correcte, mais pas adéquate pour un parsing LL (par exemple $S \to \{T+\}T$).

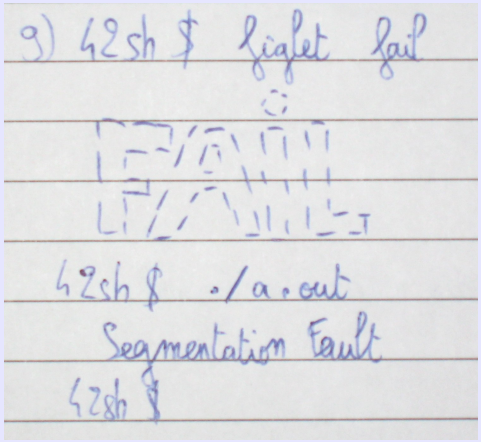
9. Parmi les non-terminaux de cette grammaire, l'un est en charge de l'opérateur postfixe. Donner en pseudo-code sa routine d'analyse prédictive récursive descendante.

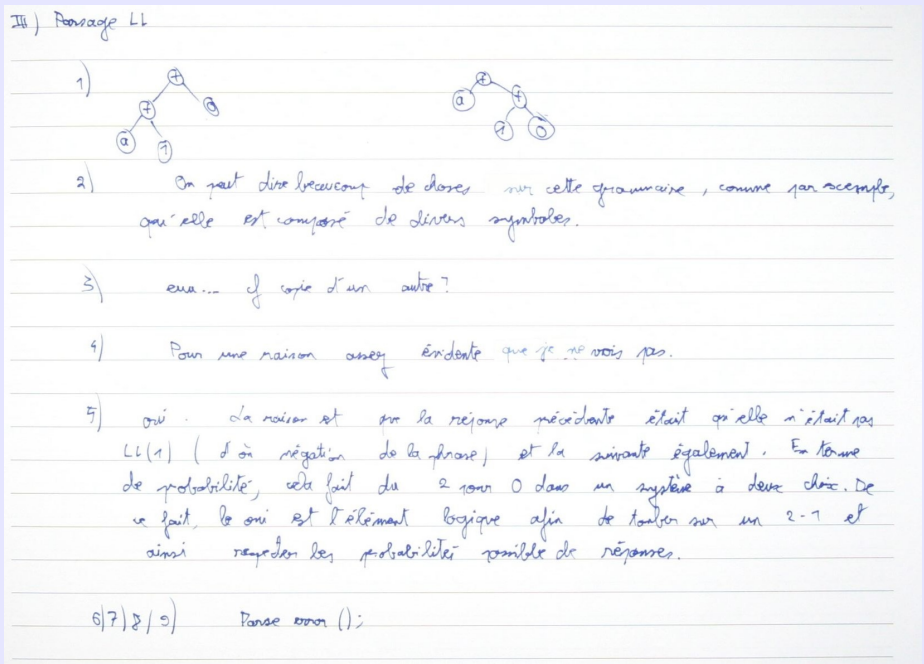
Correction: En prenant comme règle $F \rightarrow [0 \mid 1 \mid \mathtt{a} \mid (S)]\{^*\}$:

```
parse_star()
{
  switch (la)
  {
    case 0:
      res = make_zero(); eat(0);
      break;
    case z:
      res = make_one(); eat(one);
      break;
    case a:
      res = make_a(); eat(a);
      break;
    case (:
      eat('('); res = parse_S(); eat(')');
      break;
    default:
      parse_error; // unexpected la, expected 0, 1, a, or (.
      break;
  }
  while (la == *)
  {
    res = make_star(res);
    eat(*);
  }
  return res;
}
```

Barème: 3 si l'analyse est correcte, mais sans traitement des valeurs sémantiques. Bien entendu cette question a été corrigée relativement à la réponse donnée à la question précédente : du code correct pour une grammaire incorrect peut avoir une bonne note.

Best-of:

- ```
  parse_G()
    res = parse_H()
    if (la == "*") then
      return () /* appel récursif avec ajout du "*" */
    else
      return () /* appel récursif sans ajout */
  ```
- Retourner la valeur de l'accumulateur lorsque le paramètre est une opérande suivie de l'opérateur postfixe. Sinon, récursion en augmentant l'accumulateur.
- 

- 

[*Ceci était l'intégralité de la copie.*]

4 Parsage LALR(1)

On étudie la possibilité d'une implémentation en Yacc/Bison de la grammaire des expressions rationnelles de la section 3.

```
%%
```
```

```
exp:
  "0" | "e" | "a"
| exp "+" exp
| exp "." exp
| exp "*"
| "(" exp ")"
;
%%
```

Best-of: Un élève ayant écrit moins d'une page (pas feuille, vraiment une *page*) écrit ici : *Plus le temps*

1. La présence d'un opérateur postfixe est inhabituelle, il est nécessaire d'étudier les conflits avant de croire aveuglément à la puissance des directives `%left` etc.
   Montrer que l'on a deux conflits shift/reduce entre d'une part les règles des opérateurs binaires, et d'autre part la règle de l'opérateur postfixe.

   Correction: Dans la situation suivante, deux calculs sont possibles :

   ⊢exp "+" exp "*" ⊣

   La réduction donne priorité à +, le décalage à l'étoile. Il en est de même dans le cas · versus l'étoile.

   Best-of:
   – On a un confit (sic) de shift/reduce.

   Barème: Cette question et la suivante ont été mal comprises. J'admets que ma formulation n'était pas très claire ; je cherchais à vous faire travailler sur le seul cas donné ici en correction, les autres (associativité des binaires et leurs priorités relative) ont été vus en cours et ne m'intéressaient pas.
   Certains ont fait des (portions) d'automate LR(1). En général je suis clément avec ces courageux (mais ce n'était pas astucieux).
   Je me suis montré assez sévère contre les copies qui se lancent dans du verbiage pas structuré, difficile à suivre, pour plonger dans les problèmes connus de + vs. ·.

2. Dire dans chacun des cas qui du shift ou du reduce doit l'emporter.

   Correction: Dans les deux cas il faut donc choisir le décalage, ce qui se fait simplement en donnant à l'étoile une priorité supérieure aux deux autres.

   Best-of: [*Cet étudiant avait répondu « aucune idée » à la question précédente.*] Le reduce l'emporte dans tous les cas (en choisir un de chaque m'aurait obligé à nommer les cas. . .Fallait bien tenter quelque chose)

3. En déduire les directives Bison à fournir pour obtenir un analyseur correct.

   Correction:

   ```
   %left "+"
   %left "."
   %nonassoc "*"
   ```

   Barème:

   1 si c'est correct, mais à l'envers.

   3 si c'est correct, mais `%nonassoc` n'a pas été utilisé pour *. C'est une question de goût, certes, mais le bon goût compte dans la programmation.

Best-of: [*Bien noter que les textes sont aussi des étudiants eux-mêmes !*]
– Seuls les indiens d'Amérique savent communiquer avec les bisons.

```
– %right "*"
  %left "+"
  %left "*"

– %left "exp + exp"
  %left "exp . exp"
  %right "exp*"

– % no % more % conflicts %

– %left shift reduce
```

Règle générale efficace dans les deux cas.

```
– %left "+", ".", "*"
  %nonassoc "+", ".", "*"

– %left
```

4. Compléter la séquence de décalages/réductions de la phrase suivante.

```
   ⊢           0 + e . a * ⊣
s  ⊢"0"          + e . a * ⊣
r  ⊢exp          + e . a * ⊣
s  ⊢exp "+"        e . a * ⊣
s  ⊢exp "+" "e"      . a * ⊣
```

Correction:

```
   ⊢                0 + e . a * ⊣
s  ⊢"0"               + e . a * ⊣
r  ⊢exp               + e . a * ⊣
s  ⊢exp "+"             e . a * ⊣
s  ⊢exp "+" "e"           . a * ⊣
r  ⊢exp "+" exp           . a * ⊣
s  ⊢exp "+" exp "."         a * ⊣
s  ⊢exp "+" exp "." "a"       * ⊣
r  ⊢exp "+" exp "." exp       * ⊣
s  ⊢exp "+" exp "." exp "*"     ⊣
r  ⊢exp "+" exp "." exp         ⊣
r  ⊢exp "+" exp                 ⊣
r  ⊢exp                         ⊣
s  ⊢exp⊣
accept
```

Barème:

0 si des règles qui n'exitent pas sont réduites. Beaucoup d'entre vous font par exemple

```
s  ⊢exp "+" "e"           . a * ⊣
r  ⊢exp                   . a * ⊣
```

(sans parler de l'erreur de priorité ici).

2 si c'est incorrect (erreur de priorité), mais décalage et réductions correctes.

5 À propos de ce cours

Bien entendu je m'engage à ne pas tenir compte des renseignements ci-dessous pour noter votre copie. Ils ne sont pas anonymes, car je suis curieux de confronter vos réponses à votre note. En échange, quelques points seront attribués pour avoir répondu. Merci d'avance.

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

Q.9 Prises de notes

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

Q.10 Travail personnel

- **a** Rien
- **b** Bachotage récent
- **c** Relu les notes entre chaque cours
- **d** Fait les anales
- **e** Lu d'autres sources

Q.11 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.12 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.13 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