UNIVERSITÉ DE SHERBROOKE

DÉPARTEMENT D'INFORMATIQUE

IFT 313

Introduction aux langages formels

EXAMEN PÉRIODIQUE[1]

Le lundi 22 juin 2015 de 8 h 30 à 10 h 20 Professeur : Richard St-Denis

— Toute documentation est permise.

— **Tout appareil électronique, incluant le téléphone cellulaire et l'ordinateur, est interdit.**

— Ne dégrafez pas ce questionnaire.

— Répondez dans les espaces prévus à cet effet.

— **Personne ne peut quitter la salle d'examen avant 8 h 45.**

— **Personne ne peut quitter son siège entre 10 h 10 et 10 h 20.**

— La correction est, entre autres, basée sur le fait que chacune de vos réponses soit :

  — claire, c'est-à-dire lisible et compréhensible pour le correcteur ;

  — précise, c'est-à-dire exacte ou sans erreur ;

  — complète, c'est-à-dire que toutes les étapes de résolution du problème sont présentes ;

  — concise, c'est-à-dire que la méthode de résolution est la plus courte possible.

Nom : _____ Prénom : _____

Signature : _____ Matricule : _____

Question	Barème
1	/6
2	/6
3	/6
4	/6
5	/6
total :	/30

1. Ce questionnaire comporte huit (8) pages.

1. (6 points) Voici la définition d'un identificateur (*identifier* en anglais) pour le langage de programmation *Ruby*, sous la forme de règles de production d'une grammaire hors contexte écrites dans la notation étendue. [2]

```
identifier :: local-variable-identifier
            | global-variable-identifier
            | class-variable-identifier
            | instance-variable-identifier
            | constant-identifier
            | method-only-identifier
            | assignment-like-method-identifier

local-variable-identifier :: ( lowercase-character | "_" ) { identifier-character }
global-variable-identifier :: "$" identifier-start-character { identifier-character }
class-variable-identifier :: "@" "@" identifier-start-character { identifier-character }
instance-variable-identifier :: "@" identifier-start-character { identifier-character }
constant-identifier :: uppercase-character { identifier-character }
method-only-identifier :: ( constant-identifier | local-variable-identifier ) ( "!" | "?" )
assignment-like-method-identifier :: ( constant-identifier | local-variable-identifier ) "="

identifier-character :: lowercase-character
                      | uppercase-character
                      | decimal-digit
                      | "_"
identifier-start-character :: lowercase-character | uppercase-character | "_"

uppercase-character :: "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J"
                     | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T"
                     | "U" | "V" | "W" | "X" | "Y" | "Z"
lowercase-character :: "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j"
                     | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t"
                     | "u" | "v" | "w" | "x" | "y" | "z"
decimal-digit :: "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

2. Les accolades pour 0 ou plusieurs fois, les crochets pour 0 ou 1 fois et la barre verticale pour le choix.

a) (5 points) Donnez une expression régulière pour les identificateurs du langage de programmation *Ruby*, la plus simplifiée possible. Vous pouvez utiliser l pour désigner une lettre minuscule, u pour une lettre majuscule et d pour un chiffre. Donnez quelques explications qui montrent que l'expression régulière définit les identificateurs qui sont conformes à leur description originale donnée sous la forme de règles de production.

b) (1 point) Serait-il pertinent de considérer sept expressions régulières au niveau de l'analyse lexicale, une pour chaque type d'identificateur (`local-variable-identifier` à `assignment-like-method-identifier`), à la place d'une seule pour tous les types d'identificateur (`identifier`) ? Expliquez votre réponse.

2. (6 points) Voici une expression régulière sur l'alphabet $\Sigma = \{\mathbf{c}, \mathbf{,}, —, +, \mathbf{-}\}$ écrite dans la notation étendue pour les constantes numériques d'un certain langage de programmation :

$$r = \mathbf{c}^+[\mathbf{,}[\mathbf{c}^+]]—(+ \mid \mathbf{-})$$

a) (2 points) Décrivez en français le langage défini par l'expression régulière r, c'est-à-dire $L(r)$. Vous pouvez illustrer votre description à l'aide de quelques constantes numériques.

b) (4 points) Passez de l'expression régulière r à un automate fini déterministe qui accepte le même langage que celui défini par r en utilisant la méthode basée sur la construction d'un arbre d'analyse.

Répondez sur la page suivante.

(2 points) Arbre d'analyse (incluant les décorations *firstpos*, *nullable* et *lastpos*) :

(1 point) La fonction *followpos* :

(1 point) L'automate fini déterministe sous la forme d'un graphe de transition d'états :

3. (6 points) À partir de la définition d'un identificateur pour le langage de programmation *Ruby* introduite à la question 1, donnez le graphe de transition d'états d'un automate fini **déterministe** qui accepte tout identificateur *Ruby*. Pour cette question, il n'est **PAS** demandé d'appliquer systématiquement l'un (celui basé sur la construction d'un arbre d'analyse) ou l'autre (celui basé sur la notion d'item) des algorithmes présentés en classe, mais de construire cet automate de manière intuitive.

4. (6 points) Répondez aux questions suivantes.

   a) (2 points) Quels sont les symboles terminaux de la grammaire hors contexte introduite au numéro 1, celle dont les règles de production définissent les identificateurs *Ruby* ?

   b) (2 points) Traduisez, dans la notation non étendue, la règle de production pour `local-variable-identifier`, c'est-à-dire la règle :

   ```
   local-variable-identifier :: ( lowercase-character | "_" ) { identifier-character }
   ```

   Vous pouvez utiliser l pour désigner une lettre minuscule, u pour une lettre majuscule et d pour un chiffre.

   c) (2 points) À partir, entre autres, de votre réponse en b), donnez une dérivation à gauche pour la chaîne « nOn » à partir de l'axiome de la grammaire.

5. (6 points) Nous avons vu dans le cours différents types d'automate à pile de même expressivité, mais qui sont tous utiles selon le langage que l'on veut bien faire accepter ou reconnaître par un automate à pile. Voici un langage L :

$$L = \{a^n b^n bbbb \mid n \geq 0\}.$$

Peut-être pour ce langage qu'il serait intéressant de définir un nouveau type d'automate à pile de façon à ce qu'un tel automate, qui accepte L, soit petit en termes du nombre d'états. (4 points) Donnez le graphe de transition d'états d'un automate à pile qui accepte le langage L et indiquez le type d'automate à pile que vous avez utilisé, incluant le critère d'acception d'une chaîne.

(1 point) Type d'automate à pile : _____

_____

(1 point) Critère d'acception : _____

_____

FIN DE L'EXAMEN