UNIVERSITÉ DE SHERBROOKE

Département d'informatique

IFT313 — Introduction aux langages formels

Plan de cours

Enseignant :

Courriel :
Site du cours :
Disponibilité :

Professeur responsable :

Horaire :

Description officielle de l'activité pédagogique[1]

Objectifs	S'initier aux fondements théoriques des langages de programmation, en particulier aux langages formels, à la théorie des automates ainsi qu'à l'analyse lexicale et syntaxique.
Contenu	Langages réguliers et expressions régulières. Automates finis et analyseurs lexicaux. Langages et grammaires hors contexte. Arbre syntaxique et grammaire ambiguë. Automates à pile de mémoire, analyseurs syntaxiques descendants et analyseurs syntaxiques ascendants. Machines caractéristiques. Classes de grammaires hors contexte : LL, SLR, LALR et LR. Applications aux langages de programmation. Générateurs d'analyseurs lexicaux et syntaxiques.
Crédits	3
Organisation	3 heures d'exposé magistral par semaine 1 heure d'exercices par semaine 5 heures de travail personnel par semaine
Préalable	MAT 115

1 http://www.usherbrooke.ca/fiches-cours/ift313

1 Présentation

Cette section présente les objectifs et le contenu détaillé du cours. Cette section représente la description officielle du cours telle qu'adoptée par les comités de programme du département d'informatique. Elle ne peut être modifiée sans l'autorisation des comités de programme.

1.1 Mise en contexte

On sait que les langages de programmation occupent une place prépondérante en informatique. Par leur structure et leurs particularités, ils exercent une influence certaine sur le style de programmation et par conséquent sur la conception des programmes. Par exemple, le langage *Prolog* est conçu pour faire de la programmation logique, le langage Eiffel de la programmation orientée objet, le langage *Scheme* de la programmation fonctionnelle, le langage *Occam* de la programmation parallèle et enfin le langage *C* de la programmation procédurale et impérative.

Les langages de programmation sont loin d'être les seuls langages utilisés en informatique. En effet, il arrive souvent qu'il faille exprimer des instructions ou des spécifications de manière formelle, par exemple pour décrire un système complexe (langages de spécification), faire exécuter une suite de t\^aches (langages de commandes), interroger une base de données (langages de quatrième génération), voire formater un texte ou utiliser un logiciel de calcul symbolique (langages spécialisés).

Tous ces langages ont plusieurs points en commun. D'abord, de la même manière que les langues naturelles peuvent être vues comme des ensembles de phrases constituées de mots, on peut définir ces langages comme des ensembles de suites de lexèmes qui obéissent à des *règles lexicales* (dans le cas des langages de programmation, les lexèmes sont les constantes numériques, les noms de variables, les mots-clés et les opérateurs comme + et =). Ensuite, on distingue les phrases valides des autres à l'aide d'un ensemble de *règles syntaxiques*, appelé *grammaire*.

Définir un langage de manière rigoureuse et formelle permet l'analyse automatique de programmes ou de textes écrits dans ce langage par un outil qui, selon le type de langage, est appelé un compilateur, un assembleur, un interpréteur, un préprocesseur ou un éditeur. Par exemple, un programme écrit en *Java* est soumis à un compilateur, qui vérifie s'il est conforme aux règles syntaxiques du langage *Java*; si c'est le cas, le compilateur pourra traduire le programme en *bytecode*, ce qui lui permettra d'être exécuté sur une machine virtuelle *Java*.

L'activité pédagogique intitulée *Introduction aux langages formels* présente les principaux outils formels de description de langages et de mise en oeuvre d'analyseurs lexicaux et syntaxiques. Ces outils sont basés sur la théorie des automates et des langages formels.

1.2 Objectifs spécifiques

À la fin de cette activité pédagogique, l'étudiante ou l'étudiant sera capable :

1. de comprendre et d'expliquer les principales notions liées à la définition formelle de langages de programmation;
2. de comprendre et d'expliquer les principales méthodes de spécification et d'analyse lexicale et syntaxique;
3. d'appliquer les notions vues en cours à des exemples concrets;
4. d'utiliser des outils d'écriture automatique d'analyseurs lexicaux et syntaxiques.

1.3 Contenu détaillé

Thème	Contenu	Heures	Objectifs	Références	Travaux
1	Introduction : présentation du domaine des automates et des langages formels; présentation du plan de cours.	2			
2	Langage régulier : alphabet, chaîne et langage; opérations sur les chaînes et les langages; façons de définir formellement un langage; ensemble régulier et expressions régulières; exemples concrets à partir de langages de programmation.	6	1,3	chap. 2 de [7]	
3	Automate fini déterministe : rappel de la notion d'automate fini déterministe, brève introduction à la notion de non déterministe, exemples d'automates finis;algorithmes de traduction directe d'une expression régulière en un automate fini déterministe.	6	1	sec. 5.1 à 5.3 de [7]	
4	Construction d'analyseurs lexicaux : introduction à un outil de construction automatique d'analyseurs lexicaux.	2	4	[2] ou [6]	
5	Grammaire hors contexte : notion de grammaire hors contexte, exemples concrets de langages hors contexte; notions de dérivation, d'arbre de dérivation et de grammaire ambiguë; transformation d'une grammaire ambigu\"e en une grammaire non ambiguë; formes de règles de production et formes de grammaire hors-contexte; grammaires régulières et grammaires linéaires.	8	1,3	chap. 3 (sauf sec. 3.4) et survol du chap. 4 de [7]	
6	Du lexical au syntaxique : frontière entre l'analyse lexicale et syntaxique, lemme de l'étoile.	1	2	sec. 6.6 de [7]	
7	Introduction à l'analyse syntaxique : graphe d'une grammaire; analyse descendante et analyse ascendante à l'aide d'une recherche en largeur.	3	2	chap 18 de [7]	
8	Automate à pile : notion d'automate à pile, types d'automate à pile et leur équivalence.	4	1	sec. 7.1 à 7.3 de [7]	
9	Analyse syntaxique descendante : notion de grammaire LL(k); calcul des ensembles *First*, *Follow* et calcul des *lookaheads*; construction d'analyseurs syntaxiques descendants, descente récursive; transformation d'une grammaire non LL(k) en une grammaire LL(k); introduction à un outil de construction automatique d'analyseurs syntaxiques descendants.	8	2,4	chap 19 de [7]	

10	Analyse syntaxique ascendante :
	notion de grammaire LR(k) et notions d'items;
	construction d'analyseurs syntaxiques ascendants;
	grammaires LR(0), SLR(1), LR(1), LALR(1);
	transformation d'une grammaire non LR(k) en une grammaire LR(k);
	introduction à un outil de construction automatique d'analyseurs syntaxiques ascendants.
		8	2,4	chap 20 de [7]	

2 Organisation

Cette section propre à l'approche pédagogique de chaque enseignante ou enseignant présente la méthode pédagogique, le calendrier, le barème et la procédure d'évaluation ainsi que l'échéancier des travaux. Cette section doit être cohérente avec le contenu de la section précédente.

2.1 Méthode pédagogique

Le cours comprend trois heures d'exposé magistral, une heure d'exercices dirigés et cinq heures de travail personnel par semaine. Si la situation l'exige, selon l'appréciation du professeur, il pourrait y avoir quatre heures d'exposé magistraux durant une semaine, suivi de deux heures d'exposé magistral et deux heures d'exercices la semaine suivante.

2.2 Calendrier du cours

Semaine	Dates	Contenu [thèmes entre crochets; réf. section 1.4]	Laboratoire / Travaux pratiques	Lecture *
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				

** Pour les lectures, les lettres correspondent aux manuels référencés à la fin du plan de cours et les chiffres aux sections.*

2.3 Évaluation

Travaux pratiques :
Examen périodique :
Examen final :

2.3.1 Qualité du français et de la présentation

Conformément aux articles 36, 37 et 38 du règlement facultaire d'évaluation des apprentissages[2] l'enseignant peut retourner à l'étudiante ou à l'étudiant tout travail non conforme aux exigences quant à la qualité de la langue et aux normes de présentation.

2.3.2 Plagiat

Un document dont le texte et la structure se rapporte à des textes intégraux tirés d'un livre, d'une publication scientifique ou même d'un site Internet, doit être référencé adéquatement. Lors de la correction de tout travail individuel ou de groupe une attention spéciale sera portée au plagiat, défini dans le Règlement des études comme « le fait, dans une activité pédagogique évaluée, de faire passer indûment pour siens des passages ou des idées tirés de

[2] http://www.usherbrooke.ca/sciences/intranet/informations-academiques/reglement-d-evaluation/

l'œuvre d'autrui. ». Le cas échéant, le plagiat est un délit qui contrevient à l'article 8.1.2 du Règlement des études[3] : « tout acte ou manœuvre visant à tromper quant au rendement scolaire ou quant à la réussite d'une exigence relative à une activité pédagogique. » À titre de sanction disciplinaire, les mesures suivantes peuvent être imposées : a) l'obligation de reprendre un travail, un examen ou une activité pédagogique et b) l'attribution de la note E ou de la note 0 pour un travail, un examen ou une activité évaluée. Tout travail suspecté de plagiat sera référé au Secrétaire de la Faculté des sciences.

2.4 Échéancier des travaux

TP	Spécification donnée le	Thème	Pondération	Date de remise

Directives particulières

- Les travaux sont effectués par équipe de 1 à 3 étudiant(e)s.
- La qualité du français et de la présentation peut être considérée dans le résultat du travail.
- Toute soumission en retard vaut zéro, sauf celles motivées par des raisons valables, conformes au règlement des études (par exemple, maladie avec attestation d'un médecin).

3 Matériel nécessaire pour le cours

Le manuel de Sudkamp [8] est le manuel du cours. Tous les programmes et les diapositives présentés dans le cours sont disponibles dans le répertoire `/home/public/cours/ift313`.

4 Matériel nécessaire pour le cours

1. J. Grosch,
   The Parser Generator Ell,
   CoCoLab – Datenverarbeitung,
   Document No. 8,
   1998,
   (Disponible dans le répertoire opt/cocktail-0210/doc.pdf
)
2. J. Grosch,
   Rex - A Scanner Generator,
   CoCoLab – Datenverarbeitung,
   Document No. 5,
   2000, (Disponible dans le répertoire opt/cocktail-0210/doc.pdf
)
3. J. Grosch,
   Lark - An LALR(2) Parser Generator with Backtracking,
   CoCoLab – Datenverarbeitung,
   Document No. 32,
   2002,
   (Disponible dans le répertoire opt/cocktail-0210/doc.pdf
)
4. JavaCC is a parser / scanner generator for Java https://javacc.dev.java.net/

[3] http://www.usherbrooke.ca/programmes/etude

5. JavaCUP : LALR Parser Genetrator for Java
   http://www.cs.princeton.edu/~appel/modern/java/CUP/
6. JFlex – The Fast Scanner Generator for Java http://jflex.de/
7. T. A. Sudkamp,
   Languages and Machines : An Introduction to the Theory of Computer Science,
   Addison-Wesley,
   2005,
   (Manuel disponible à la Coop de l'Université de Sherbrooke et en réserve à la bibliothèque)
8. Java Standard Edition Development Kit 6 (JDK) http://java.sun.com/
9. Environnements de développement intégré (IDE) : NetBeans : http://www.netbeans.org/ , Eclipse: http://www.eclipse.org/
10. Normes de rédaction et de programmation du département
http://www.dmi.usherb.ca/~fraikin/cours/Normes/normes-de-programmation.pdf
11. Connexion par SSH et la soumission par turnin : http://www.dmi.usherb.ca/~fraikin/cours/SSH-turnin

12. A. W. Appel. *Modern Compiler Implementation in Java*. Second Edition. Cambrdidge, 2002.
13. P. Wolper**.** *Introduction à la calculabilité.* 3e Édition. Dunod, 2006.
14. A.V. Aho, R. Sethi & J.D. Ullman**.** *Compilers: Principles, Techniques and Tools.* Addison-Wesley, 1988.
15. D. Grune, H.E. Bal, C.J.H. Jacobs & K.G. Langendoen. *Modern Compiler Design.* Wiley, 2000.