UNIVERSITÉ DE SHERBROOKE

Département d'informatique

IFT 313 – Introduction aux langages formels

Plan de cours
Hiver 2019

Enseignant

Martin Beaudry	
Courriel :	Martin.Beaudry@USherbrooke.ca
Local :	D4-2010-9
Téléphone :	(819) 821-8000 poste 62014
Site :	info.usherbrooke.ca/mbeaudry/Ens-ift313.html
Disponibilité :	flexible ; prendre rendez-vous par courriel.

Professeurs responsables : Froduald Kabanza et Richard St-denis

Horaire

Exposé magistral :	jeudi	10 h 30 à 12 h 20	salle à déterminer
	vendredi	13 h 30 à 15 h 20	salle à déterminer

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
Particularités	Cette activité pédagogique se présente sous la forme d'un cours.

1. https://www.usherbrooke.ca/admission/fiches-cours/IFT313/

1 Présentation

Cette section présente les objectifs spécifiques et le contenu détaillé de l'activité pédagogique. Cette section, non modifiable sans l'approbation d'un comité de programme du Département d'informatique, constitue la version officielle.

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 tâches (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

Le tableau suivant contient la matière présentée dans ce cours. L'étude de cette matière est accompagnée d'au moins cinq devoirs couvrant tous les sujets, dont au moins deux travaux pratiques utilisant des outils de construction automatique d'analyseurs lexicaux et syntaxiques.

Thème	Contenu	N^{bre} d'heures	Objectifs	Références
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 automatate 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 ambiguë 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], [1] ou [4]

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], [3] ou [5]

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

Une semaine comporte quatre heures de présence en classe réparties dans une proportion de trois heures de cours magistral et d'une heure d'exercices. Cependant, les proportions varieront d'une semaine à l'autre, de manière à ce que les exercices soient synchronisés avec le contenu enseigné.
Le cours suit de près le manuel de Sudkamp ; on encourage les étudiantes et les étudiants à lire à l'avance les sections du manuel abordées dans les cours à venir, afin de comprendre plus efficacement les notions qui seront présentées en classe et d'être mieux en mesure de poser des questions pertinentes.

Note : pour poser des questions en-dehors des cours, on recommande l'utilisation du courrier électronique.

2.2 Calendrier

Semaines		Thèmes	Remarques
1	10 et 11 janvier	1 - 2	
2	17 et 18	2	
3	24 et 25	3 devoir	
4	31 et 1er janvier-février	3 - 4	
5	7 et 8	4 - 5 devoir	
6	14 et 15	5	
7	21 et 22	5 - 6 - 7 devoir	
8	28 et 1er février-mars	Examen périodique	pas de cours
9	7 et 8	*Relâche*	
10	14 et 15	7 - 8	
11	21 et 22	8 - 9 devoir	
12	28 et 29	9 - 10	
13	4 et 5 avril	10 devoir	
14	11	10	
15–16		Examen final	

2.3 Évaluation

Devoirs	30%
Examen périodique	30%
Examen final	40%

Les appareils électroniques (calculatrice, portable, tablette, téléphone) sont interdits pendant les examens.

Conformément au 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. Le plagiat consiste à utiliser des résultats obtenus par d'autres personnes afin de les faire passer pour siens et dans le dessein de tromper l'enseignant. Si une preuve de plagiat est attestée, elle sera traitée en conformité, entre autres, avec

2. https://www.usherbrooke/ca/sciences/fileadmin/sites/sciences/Etudiants_actuels/Informations_academiques_et_reglements/2017-10-27_Reglement_facultaire_-_evaluation_des_apprentissages.pdf

l'article 9.4.1 du Règlement des études [3] de l'Université de Sherbrooke. L'étudiante ou l'étudiant peut s'exposer à de graves sanctions, dont automatiquement une note de zéro (0) au devoir ou à l'examen en question.
Ceci n'indique pas que vous n'ayez pas le droit de coopérer entre deux équipes tant que la rédaction finale des documents et la création du programme restent le fait de votre équipe. En cas de doute de plagiat, l'enseignant peut demander à l'équipe d'expliquer les notions ou le fonctionnement du code qu'il considère comme étant plagié. En cas de doute, ne pas hésiter à demander conseil et assistance à l'enseignant afin d'éviter toute situation délicate par la suite.

2.4 Échéancier des travaux

L'énoncé de chaque devoir est accompagné d'informations relatives aux directives et à la date de remise. La remise des devoirs s'effectue au jour et à l'heure spécifiés. Les travaux remis par courriel et les travaux remis en retard ne seront pas corrigés et recevront automatiquement la note zéro.
Toute documentation, de quelque forme que ce soit, et tous les appareils électroniques (calculatrice, ordinateur portable, téléphone cellulaire, etc.) sont interdits pendant les examens.

Directives particulières : Les devoirs peuvent être faits individuellement ou par équipe de deux personnes. Ils peuvent comporter entre autres l'écriture d'analyseurs lexicaux et d'analyseurs syntaxiques nécessitant l'utilisation d'un logiciel spécialisé.

2.5 Utilisation d'appareils électroniques et du courriel

Selon le règlement complémentaire des études, section 4.2.3 [4], l'utilisation d'ordinateurs, de cellulaires ou de tablettes pendant une prestation est interdite à condition que leur usage soit explicitement permis dans le plan de cours. *Dans le cours IFT313, le règlement 4.2.3 s'applique à moins d'avoir obtenu personnellement l'autorisation du professeur. Cette permission peut être retirée en tout temps, si l'appareil n'est pas utilisé uniquement à des fins d'apprentissage.*
Comme indiqué dans le règlement universitaire des études, section 4.2.5 [5], toute utilisation d'appareils de captation de la voix ou de l'image exige la permission du professeur.

3 Matériel nécessaire pour le cours

Le livre de Sudkamp [7] est la référence principale du cours. On consultera aussi les documents déposés sur la page web du cours.

4 Références

[1] J. GROSCH : The parser generator ell. Rapport technique Document No. 8, CoCoLab - Datenverarbeitung, 1998. Disponible dans le répertoire `opt/cocktail-0210/doc.pdf`.

[2] J. GROSCH : Rex - a scanner generator. Rapport technique Document No. 5, CoCoLab - Datenverarbeitung, 2000. Disponible dans le répertoire `opt/cocktail-0210/doc.pdf`.

[3] J. GROSCH : Lark - an lalr(2) parser generator with backtracking. Rapport technique Document No. 32, CoCoLab - Datenverarbeitung, 2002. Disponible dans le répertoire `opt/cocktail-0210/doc.pdf`.

[4] JAVACC : A parser/scanner generator for java. http ://javacc.dev.java.net/.

[5] JAVACUP : Lalr parser generator for java. http ://www.cs.princeton.edu/ appel/modern/java/CUP/.

3. https://www.usherbrooke.ca/registraire/droits-et-responsabilites/reglement-des-etudes/
4. https://www.usherbrooke.ca/sciences/fileadmin/sites/sciences/documents/Intranet/Informations_academiques/Sciences_Reglement_complementaire_2017-05-09.pdf
5. https://www.usherbrooke.ca/registraire/droits-et-responsabilites/reglement-des-etudes/

[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.