

## FICHE ECTS

<b>Composante (dépt ou centre)</b>	DGEI		
<b>Code ECTS</b>	I4IRIF11		
<b>Crédits ECTS</b>	8		
<b>Enseignant responsable de l'UF</b>	Patrick Esquirol		
<b>Section CNU</b>			
<b>Libellé de l'UF (français)</b>	Informatique fondamentale		
<b>Libellé de l'UF (anglais)</b>	Fundamentals in Computer Science		
<b>Semestre :</b>	Semestre 1 <input checked="" type="checkbox"/>	Semestre 2 <input type="checkbox"/>	Annuel <input type="checkbox"/>

## DESCRIPTION GENERALE

## VERSION FRANÇAISE

## VERSION ANGLAISE

<b>Objectifs, finalités <sup>1</sup> (max 1000 caract.)</b>	<b>Objectives (max 1000 charact.)</b>
<p>Cet enseignement est hétérogène et regroupe 3 cours :            Programmation Fonctionnelle – Caml (« PF-Caml »)            Logique formelle et programmation logique en prolog (« LF-Prolog »)            Algorithmique Avancée (« AA »)</p> <p>A la fin de ce module, les étudiants doivent être capables de :</p> <p><b>[PF-Caml]</b></p> <ul style="list-style-type: none"> <li>- comprendre et écrire des programmes fonctionnels purs,</li> <li>- concevoir des fonctions récursives pour itérer sur des structures récursives,</li> <li>- définir des types variants et des types paramétrés,</li> <li>- plus généralement penser en termes de fonctions d'ordre supérieur afin d'écrire du code ré-utilisable.</li> <li>- décrire la sémantique de lambda-termes simples</li> <li>- comprendre superficiellement la théorie des systèmes de types</li> </ul> <p><b>[LF- Prolog]</b></p> <ul style="list-style-type: none"> <li>- traduire des énoncés du langage naturel en formules de logique des prédicats du 1er ordre.</li> <li>- appliquer plusieurs méthodes pour évaluer la validité et/ou la consistance d'une formule logique</li> <li>- expliquer les fondements de la programmation logique et de Prolog</li> <li>- aborder la résolution d'un problème comme une démonstration logique basée sur des axiomes et des théorèmes exprimant les propriétés spécifiques du problème.</li> <li>- programmer en Prolog, tracer l'exécution d'un programme</li> </ul> <p><b>[AA]</b></p> <p>Quelques grands paradigmes algorithmiques pour l'optimisation discrète</p> <ul style="list-style-type: none"> <li>- Enumération exhaustive</li> <li>- Diviser pour régner</li> <li>- Programmation dynamique</li> <li>- Algorithmes gloutons</li> </ul>	<p>This course is heterogeneous course and groups 3 parts :</p> <ul style="list-style-type: none"> <li>- Functionnal Programming – Caml (“FP- Caml”)</li> <li>- Formal Logic and Logic Programming in Prolog (“FL- Prolog”)</li> <li>- Advanced Algorithmics (« AA »)</li> </ul> <p>At the end of this module, students are expected to :</p> <p><b>[FP-Caml]</b></p> <ul style="list-style-type: none"> <li>- understand and write pure functional programs,</li> <li>- design recursive functions to iterate over recursive data types,</li> <li>- define variants or parameterized types,</li> <li>- more generally think in terms of higher-order functions in order to write reusable codes.</li> <li>- describe the semantics of simple lambda terms</li> <li>- have a basic theoretical understanding of the type systems theory</li> </ul> <p><b>[FL-Prolog part]</b></p> <ul style="list-style-type: none"> <li>- translate natural language statements into formulas of propositional logic and of 1<sup>st</sup> order predicate calculus</li> <li>- apply several methods in order to check the validity and the consistency of a formula</li> <li>- explain the fundamentals of logic programming and of Prolog.</li> <li>- express problem solving as a demonstration (proof) based on axioms and theorems describing the particular properties of the problem.</li> <li>- design a Prolog program and trace its execution</li> </ul> <p><b>[AA Part]</b></p> <p>Some paradigms in algorithmics for discrete optimization :</p> <ul style="list-style-type: none"> <li>- Exhaustive enumeration</li> <li>- Divide and Conquer</li> <li>- Dynamic Programming</li> <li>- Greedy Algorithms</li> </ul>

<b>Contenu (max 1000 caract.)</b>	<b>Description (max 1000 caract.)</b>
<p><b>Partie PF-Caml :</b> Les étudiants travaillent immédiatement sur des sujets de TP, et abordent les concepts fondamentaux (currification, fermetures, filtrage, récursion, variants, polymorphisme paramétrique, ...). Par la suite, un petit projet de programmation est proposé.</p> <p><b>Partie LF- Prolog :</b> Logique propositionnelle : - Syntaxe et sémantique, - Méthode des tables de vérité, - Méthode des tableaux sémantiques, - Système de preuve de Hilbert - Démonstrations et preuves. - Problem SAT- Algorithmes DPLL - Diagrammes de décisions binaires (BDD) - Diagrammes de décisions binaires et ordonnés Logique des prédicats du 1er ordre : - Principe de résolution and démonstration par réfutation - Arbres de dérivation - Forme prénexe, skolémisation, plus grand unifieurs - Univers et base de Herbrand. Programmation logique en prolog : - Résolution linéaire pour les clauses définies, - Négation par échec, - Récursivité - Prédicats prédéfinis, - Applications et extensions</p> <p><b>Partie AA</b> Les caractéristiques essentielles des algorithmes génériques pour l'optimisation sont présentées. Des éléments de comparaison sont donnés en termes de classes de problèmes, de stratégie d'optimisation et de complexité.</p>	<p><b>[FP-Caml]</b> The students undergo practical work and assessments dedicated to basic concepts used in functional programming (curried functions, closures, pattern matching, recursion, variant data types, parametric polymorphism, ...) before working on a small programming project.</p> <p><b>[FL-Prolog part]</b> -Propositional logic - Syntax and semantics - Truth table method - Method of analytic tableaux - Hilbert deduction system – derivations and proofs - SAT problem- DPLL algorithms - Binary Decisions Diagrams - Reduced and Ordered Binary Decision Diagrams First order predicate calculus - Resolution principle and refutation based demonstration - Derivation trees - Prenex normal form, skolemization, most general unifiers - Herbrand Base, Herbrand universe Logic programming in Prolog - Linear resolution for definite clauses - Negation by failure - Recursive clauses - Built-in predicates - Applications and extensions</p> <p><b>[AA part]</b> Main principles of generic algorithms are presented. Their features are compared, taking into account classes of problems, optimization strategies and complexity.</p>

<b>Recommandation (max 1000 caract.)</b>	<b>Recommendation (max 1000 caract.)</b>
<i>Principales difficultés habituellement rencontrées par les étudiants</i>	

<b>Pré-requis (Code UF + intitulé, sinon notions nécessaires) (max 200 caract.)</b>	<b>Necessary knowledge (UF Code + title, or required knowledge) (max 200 caract.)</b>

<b>Organisation, méthodes pédagogiques</b>			<b>Organisation, teaching methods</b>		
<b>Horaires</b>	<b>présentiel (tel que l'enseignement est comptabilisé)</b>	<b>Travail personnel<sup>2</sup></b>	<b>Contact hours</b>		<b>Personal work</b>
CM	28,75		Lectures	28,75	
TD	21,25		Tutorials	21,25	
TP	22		Lab work	22	
Projet			Project		
Examen formatif			Coursework		
Examen certificatif	4,5		Exam	4,5	

<b>Format d'enseignement :</b>	Présentiel <input checked="" type="checkbox"/>	Distanciel <input type="checkbox"/>	Hybride <input type="checkbox"/>
--------------------------------	--	-------------------------------------	----------------------------------

## DESCRIPTION COMPLEMENTAIRE

<b>Modalités d'évaluation (max 1000 caract.)</b>	<b>Assessment (max 1000 charact.)</b>
<b>Comment évaluez-vous que ces objectifs sont atteints ?</b>	
1 examen écrit dans chaque partie 1 évaluation des travaux pratiques  Exercices d'auto-évaluation et annales corrigées	1 written exam on each part (FP, FL-Prolog, AA) 1 practical work assessment  Self-assessment exercices, Annals and solutions
Examen écrit <input checked="" type="checkbox"/> oral <input type="checkbox"/> Rapport <input type="checkbox"/> Exposé <input type="checkbox"/> TP <input checked="" type="checkbox"/>	Written ex. <input checked="" type="checkbox"/> Oral ex. <input type="checkbox"/> Report <input type="checkbox"/> Presentation <input type="checkbox"/> Labwork <input checked="" type="checkbox"/>
<b>Autre (préciser)</b>	<b>Other (please describe)</b>

<b>Aides aux étudiants</b>	<b>Student aid</b>

<b>Public ciblé</b>	<b>Student aid</b>
<b>Type de formation</b>	Formation initiale <input checked="" type="checkbox"/> Formation continue <input type="checkbox"/> Apprentissage <input type="checkbox"/> VAE <input checked="" type="checkbox"/>

<b>Admission</b>	<b>Admission</b>

<b>Besoins particuliers</b>	<b>Particular needs</b>

<b>Langue(s) utilisée(s) pour l'enseignement : français</b>	French
<b>Langue(s) utilisée(s) pour le support de cours : français</b>	French
<b>Langue(s) utilisée(s) pour l'évaluation : français</b>	French

<b>Mots clés :</b>	<b>Keywords :</b>
Programmation Fonctionnelle, Programmation Logique, Algorithmes génériques pour l'optimisation	Functionnal Programming, Logic Programming, Generic algorithms for optimization

<b>Bibliographie (auteur, titre, éditeur, année, ISBN)</b>	<b>Bibliography (author, title, publisher, year, ISBN)</b>
<p><b>[Partie PF Caml]</b></p> <ul style="list-style-type: none"> <li>X. Leroy - Documentation Ocaml en ligne</li> <li>O. Chailloux - Développement d'application Ocaml, O'Reilly, 2000</li> <li>JM Alliot – Informatique théorique, Cépadues</li> </ul> <p><b>[Partie LF- Prolog]</b></p> <ul style="list-style-type: none"> <li>Outils logiques pour l'intelligence artificielle, Jean-Paul Delahaye, Eyrolles, 1988.</li> <li>Logique et fondements de l'informatique, R. Lassaigne, M. de Rougemont, Hermes, 1993.</li> <li>Logique, Méthodes pour l'informatique fondamentale, Volume 1, P. Gochet, P. Gribomont, 1998</li> <li>Logic, Programming and Prolog, U. Nilsson, J. Maluszynski, John Wiley &amp; Sons, 1995 (<a href="http://www.ida.liu.se/~ulfni/lpp">http://www.ida.liu.se/~ulfni/lpp</a>)</li> <li>Prolog Programming in Depth, Michael A. Covington, Donald Nute, André Vellino, Artificial Intelligence Programs, University of Georgia, USA, 1995.</li> </ul> <p><b>[Partie AA]</b></p> <p>Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest et Clifford Stein, Introduction to Algorithms, Cambridge (Mass.), MIT Press, 2009, 3e éd.</p>	

<sup>1</sup>[http://enseignants.insa-toulouse.fr/fr/ameliorer\\_mon\\_cours/comment\\_rediger\\_les\\_objectifs\\_de\\_son\\_enseignement.html](http://enseignants.insa-toulouse.fr/fr/ameliorer_mon_cours/comment_rediger_les_objectifs_de_son_enseignement.html)  
<sup>2</sup> à titre d'exemple, on peut multiplier le présentiel par un facteur fonction du type de pédagogie : 0,9 pour les CM, 0,7 pour les TD, 0,3 pour les TP, 1,5 pour les APP et autres pédagogies actives. Dans tous les cas, cette valeur doit être la plus authentique possible et s'appuyer sur des moyens appropriés pour guider le travail personnel de l'étudiant (exercices non corrigés lors des TD, préparation de TP, exercices « pour aller plus loin », grilles d'auto-évaluation, travail personnel à faire utilisant la bibliographie recommandée, ...)