Ecole des Jeunes Chercheurs en Programmation

À côté des groupes de travail, qui constituent le principal instrument d'animation des communautés scientifiques, l'École des Jeunes Chercheurs participe au renouvellement de ces communautés en formant et sensibilisant les jeunes chercheurs aux thématiques scientifiques présentes dans le GDR Génie de la Programmation et Logiciel (GPL).

L'École Jeunes Chercheurs en Programmation accueille chaque année pendant une semaine une quarantaine de jeunes chercheurs en première année de thèse dans la communauté informatique. L'école propose ses cours dans un ensemble de thèmes de recherche liés à la programmation, à la vérification et au génie logiciel.

L'édition 2024 de l'école aura lieu à Argelès-sur-Mer (Pyrénées Orientales), au Village Club Azureva. Elle est organisée par le laboratoire LAMPS de l'Université de Perpignan Via Domitia.

Inscriptions

Les frais d'inscription incluent l'hébergement en chambre individuelle ainsi que l'ensemble des repas en pension complète pour la durée du séjour (repas du lundi matin au vendredi midi inclus, le repas du dimanche 16 au soir n'est pas inclu). Deux options sont proposées :

- Arrivée le lundi 17 juin au matin : 450 euros
- Arrivée le dimanche 16 juin au soir : 550 euros

Programme

Lundi 17 juin 2024

- **10h00-12h00 - Mesurer la consommation énergétique : du matériel au code source logiciel**, Adel Noureddine (Université de Pau et des Pays de l'Adour) La mesure de l'énergie logicielle est une étape importante pour comprendre et analyser la consommation énergétique, optimiser le code et écrire un code éco-conçu. Dans ce cours, nous détaillons la mesure énergétique logicielle en partant de la mesure des appareils (ordinateurs et serveurs, smartphones, SBC et objets IoT) et des composants matériels (CPU notamment), et en remontant les couches applicatifs jusqu'au code source. En particulier, nous analyserons les approches permettant à diagnostiquer l'énergie d'une application en mesurant ses méthodes et ses branches d'exécution au runtime.
- **13h30-16h30 - TP :** Dans ce TP, nous analyserons et comparons la consommation énergétique de différents appareils et composants matériels, ainsi que des applications, et du code source des applications Java (méthodes, branches d'exécution).

Mardi 18 juin

- **9h00-12h00 - Automated Program Repair**, Martin Monperrus (KTH Royal Institute of Technology, Stockholm) In this lecture, we'll discuss about the foundations, challenges and current state-of-the-art of automated program repair. We'll discuss the different families of

techniques (generate-and-validate, template, learning, symbolic) and how they can be used in the software development process.

- **13h30-16h30 - TP :** The students will run a program repair tool and analyze the results. At the end, you'll present to the rest of the groups the inner working and the expected input and output format.

Mercredi 19 juin

- **9h00-12h00 Formal Methods for Machine Learning Pipelines**, Caterina Urban (INRIA-ENS PSL) Formal methods offer rigorous assurances of correctness for both hardware and software systems. Their use is well established in industry, notably to certify safety of critical applications subjected to stringent certification processes. With the rising prominence of machine learning, the integration of machine-learned components into critical systems presents novel challenges for the soundness, precision, and scalability of formal methods. This lecture serves as an introduction to formal methods tailored for machine learning pipelines, highlighting their strengths and limitations. We will present several approaches through the lens of different software properties and targeting software across all phases of a machine learning pipeline, with a focus on abstract interpretation-based techniques. We will then conclude by offering perspectives for future research directions in this evolving context.
- **13h30-16h30 - TP :** TBA

Jeudi 20 juin

- **9h00-12h00 - Analyse statique/dynamique des applications HPC**, Emmanuelle Saillard (INRIA Bordeaux) Afin de résoudre les plus grands problèmes scientifiques en un temps raisonnable, les applications sont parallélisées et lancées sur des supercalculateurs. Cependant, ces supercalculateurs sont de plus en plus complexes et puissants, ce qui entraine une évolution des applications (ex., nouveaux algorithmes pour le passage à l'échelle, combinaison de modèles de programmation parallèle) qui lève de nombreux défis de programmation et un réel besoin d'outils et techniques pour aider les développeurs et développeuses à utiliser au mieux les différentes machines et architectures à leur disposition. En effet, à grande échelle, les développeurs et développeuses d'applications font face à de nouvelles erreurs, liées au parallélisme, souvent difficiles à analyser et corriger. Aujourd'hui, s'assurer que les applications parallèles s'exécutent correctement devient aussi important que d'obtenir de bonnes performances. Dans ce cours, je présenterai les différentes méthodes qui existent pour vérifier qu'un code parallèle est correcte, avec un focus sur le modèle de programmation parallèle MPI. Ensuite, nous verrons comment une analyse statique/dynamique est un bon compromis pour de l'analyse de code.
- **13h30-16h30 - TP :** Mise en pratique du cours avec l'utilisation de l'outil PARCOACH et la création d'une nouvelle analyse statique basée sur le compilateur LLVM.

Vendredi 21 juin

- **9h00-12h00 - Compilation d'un langage synchrone avec contraintes de ressources**, Tim Bourke (INRIA-ENS PSL) Les langages synchrones, tels que Lustre et Scade, sont utilisés pour programmer les logiciels de contrôle les plus critiques. Ces logiciels sont souvent conçus

comme un ensemble de plusieurs centaines de composants qui s'exécutent à diverses périodes et qui doivent respecter des contraintes temps réel imposées entre les entrées et les sorties du système. Certains de ces logiciels sont implémentés en générant un code séquentiel en vue d'une exécution périodique sur un processeur embarqué. Dans ce cas, c'est un compilateur qui doit trouver un équilibre entre la quantité de calcul à effectuer à chaque cycle d'exécution et les contraintes temps réel. Récemment, nous avons étudié une approche utilisée chez Airbus pour laquelle nous avons développé une extension de Lustre capable d'exprimer directement les caractéristiques temps réel. Cette extension est compilée en trois temps. D'abord, le compilateur analyse le programme et produit un programme linéaire en nombres entiers. Ensuite, un solveur externe est utilisé pour chercher une solution qui satisfait toutes les contraintes. Enfin, le compilateur se sert de cette solution pour générer un code séquentiel en utilisant une généralisation du schéma de compilation modulaire habituel.

- **13h30-15h30 - TP :** Dans ce cours, vous apprendrez les bases des langages synchrones et les détails d'une approche industrielle pour leur application dans un logiciel temps réel. Dans la partie pratique, vous utiliserez OCaml et un programme linéaire en nombres entiers pour transformer un programme source en un code séquentiel.