Internship proposal at IRISA/INRIA-Rennes

TITLE
Refinement types for modular abstract interpretation

ABSTRACT
A refinement type is a type endowed with a predicate that is assumed to hold for any element of the refined type. Refinement types can express preconditions when used as function arguments or postconditions when used as return types: for instance, the type of a function which accepts natural numbers and returns natural numbers greater than 5 may be written as $f : \mathbb{N} \to \{ n : \mathbb{N} \mid n > 5 \}$. Implementations of refinement type systems can be found in functional languages like Liquid Haskell (type inference, conjunctive logic), F* (verification and proof, higher-order logic), and others.

PROJECT
The goal of our project is to combine abstract interpretation of memory transformations in C programs with refinement types in order to provide
1. type inference: use abstract interpretation to determine type and properties of individual programs
2. type checking: verify declared abstractions of programs against analysed properties
3. modular analysis: use refinement types as abstractions of C modules when analysing a program
4. verification and analysis: use SAT-SMT to verify program correctness from types and properties

Our long-term goal is the implementation of a prototype to case study the modular verification of micro-kernel libraries and unikernel applications.

DURATION 4 to 6 months
LOCATION Irisa/Inria, campus de Beaulieu, 35000 Rennes
SUPERVISORS Jean-Pierre Talpin (@inria.fr) David Pichardie (@ens-rennes.fr)

REFERENCES
- Patrick Rondon, Ming Kawaguci, Ranjit Jhala. Liquid types. 29\textsuperscript{th} ACM SIGPLAN Conference on Programming Language Design and Implementation. ACM, 2008
- Swamy et al. Dependent Types and Multi-Monadic Effects in F*. 43\textsuperscript{rd} ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM, 2016