Interopérabilité vérifiée entre OCaml et C en présence d'exceptions

Proposition de stage de recherche (niveau M1)

2023

Encadrant

Armaël Guéneau, chargé de recherche, LMF et Inria Saclay (armael.gueneau@inria.fr)

Situation du sujet

Il est courant pour des bibliothèques OCaml de s'interfacer à des bibliothèques plus bas niveau écrites en C. Ceci est rendu possible via l'implémentation de code « glue » écrit en C mais soumis à diverses restrictions dictées par l'implémentation du langage OCaml, et décrites dans la « *Foreign Function Interface* » (FFI) d'OCaml.

Je mène, avec mes collaborateurs, le projet Melocoton [mel] consacré à la formalisation de la FFI OCaml afin de pouvoir vérifier rigoureusement la sûreté de bibliothèques OCaml s'interfaçant avec du code C. Les fondations de Melocoton sont déjà posées dans une publication récente [GHS+23] : nous avons construit un modèle formel *d'une partie* de la FFI OCaml, et énoncé des règles de raisonnement associées permettant déjà de vérifier la sûreté de programmes non triviaux. Ce modèle est implémenté en Coq en utilisant le framework Iris [JKJ+18].

Un certain nombre de fonctionnalités de la FFI OCaml ne sont pas encore modélisées formellement dans Melocoton. Notamment, Melocoton ne prend aujourd'hui pas en compte le mécanisme d'*exceptions* fourni par OCaml. Or, *les exceptions d'OCaml interagissent de manière non-triviale avec la FFI* : le code C peut notamment lancer des exceptions via la FFI, ou rattraper des exceptions émises par du code OCaml via des *callbacks*, le tout ayant une sémantique souvent subtile.

Je propose dans ce stage d'étendre la formalisation Coq/Iris de Melocoton pour : 1) modéliser la sémantique des exceptions d'OCaml et leurs interactions avec la FFI, et 2) définir des règles de vérification de programmes correctes vis-à-vis de cette sémantique.

Objectifs

Le programme de travail est le suivant. Il s'agit des grandes lignes à suivre, en laissant ouvertes des pistes de recherche intéressantes (notamment sur la conception de la sémantique et des règles de raisonnement : lesquelles sont non seulement correctes, mais aussi élégantes, expressives, ... ?).

1. Écrire des petits exemples de programmes OCaml + C utilisant la FFI, illustrant les différentes opérations et aspects intéressants de la FFI liés aux exceptions.

2. Formaliser en Coq/Iris la sémantique d'exceptions et leurs principes de raisonnement correspondant en logique de séparation. On se placera dans un cadre le plus simple possible, par exemple en s'inspirant du langage mini-ML "`heap_lang`" déjà formalisé dans Iris.
3. Étendre la notion de "linking" présente dans Melocoton pour autoriser des fonctions de la FFI à lancer des exceptions. Étendre ensuite Melocoton avec les opérations de la FFI liées aux exceptions, définir leur sémantique et leurs règles de raisonnement.
4. Valider ces nouvelles règles en les utilisant pour vérifier la correction des petits exemples de code écrits au début du stage.

Pré-requis

Avoir de bases solides en programmation, et une bonne familiarité avec les langages OCaml et C. Avoir suivi un cours de programmation avancée, par exemple le cours « Programmation Avancée » à l'ENS Paris-Saclay (L3). Souhaitable : de l'expérience avec la preuve de programmes en Coq.

Détails pratiques

Le stage se déroulera au LMF, bâtiment 650, sur le plateau de Saclay. (Campus Universitaire, Rue Raimond Castaing, Bâtiment 650, 91190 Gif-sur-Yvette)

Références

[GHS+23] Armaël Guéneau, Johannes Hostert, Simon Spies, Michael Sammler, Lars Birkedal, and Derek Dreyer. Melocoton : A program logic for verified interoperability between OCaml and C. In *OOPSLA*, 2023. https://doi.org/10.1145/3622823.

[JKJ+18] Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Aleš Bizjak, Lars Birkedal, and Derek Dreyer. Iris from the ground up : A modular foundation for higher-order concurrent separation logic. 2018.

[mel] The Melocoton project (web page). https://melocoton-project.github.io.