Separation Logic

Introduction

Arthur Charguéraud

February 2015

Motivation for Separation Logic

Separation Logic: a technique for modular specification and verification of programs with mutable state.

Separation Logic + Interactive proofs = No limits

Origins of Separation Logic

- John Reynolds (2000)
  - Intuitionistic Reasoning about Shared Mutable Data Structure
  - —building on ideas from Burstall (1972).
- John Reynolds, Peter O'Hearn, Hongseok Yang (2001)
  - Local reasoning about programs that alter data structures
- John Reynolds (2002)
  - Separation Logic: A logic for shared mutable data structure.

Adopters of Separation Logic

Micro-controller	Klein et al	NICTA
Assembly language	Chlipala et al	MIT
Operating system	Shao et al	Yale
C (drivers)	Yang et al	Oxford
C-light	Appel et al	Princeton
C11 (concurrent)	Vafeiadis, Parkinson et al	MPI and MSR
ML	Morisset et al	Harvard
Java	Parkinson et al	MSR and Cambridge
Java	Jacobs et al	Leuven
Javascript	Gardner et al	Imperial College
Caml	Charguéraud	Inria
...		

→ see Peter O'Hearn's webpage on Separation Logic:

`http://www0.cs.ucl.ac.uk/staff/p.ohearn/SeparationLogic/Separation_Logic/SL_Home.html`

Specificity of this course

The Separation Logic presented in this course:

- targets a clean ML language
- presents only definitions used in practice (not $-\!\!*$, not $\wedge$)
- targets higher-order logic
- supports higher-order and first-class functions
- uses polymorphic representation predicates
- integrates smoothly into Coq.

Comparison with the Why approach

Unlike in the first part of the course based on Why, the logic:

- is not optimized for proof automation
- applies to un-annotated programs
- has many similar rules, but with different interpretations
- supports aliasing and local reasoning.

Calendar

- January 29th: Course
- February 5th: Course
- February 12th: Help for the project
- February 19th: Course
- February 23th: Project deadline
- February 26th: Course
- March 12th: Exam