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

  - Separation Logic: A logic for shared mutable data structure.
## Adopters of Separation Logic

<table>
<thead>
<tr>
<th>Domain</th>
<th>Authors</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro-controller</td>
<td>Klein et al</td>
<td>NICTA</td>
</tr>
<tr>
<td>Assembly language</td>
<td>Chlipala et al</td>
<td>MIT</td>
</tr>
<tr>
<td>Operating system</td>
<td>Shao et al</td>
<td>Yale</td>
</tr>
<tr>
<td>C (drivers)</td>
<td>Yang et al</td>
<td>Oxford</td>
</tr>
<tr>
<td>C-light</td>
<td>Appel et al</td>
<td>Princeton</td>
</tr>
<tr>
<td>C11 (concurrent)</td>
<td>Vafeiadiis, Parkinson et al</td>
<td>MPI and MSR</td>
</tr>
<tr>
<td>ML</td>
<td>Morisset et al</td>
<td>Harvard</td>
</tr>
<tr>
<td>Java</td>
<td>Parkinson et al</td>
<td>MSR and Cambridge</td>
</tr>
<tr>
<td>Javascript</td>
<td>Jacobs et al</td>
<td>Leuven</td>
</tr>
<tr>
<td>Caml</td>
<td>Gardner et al</td>
<td>Imperial College</td>
</tr>
<tr>
<td>...</td>
<td>Chaguéraud</td>
<td>Inria</td>
</tr>
</tbody>
</table>

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

[http://www0.cs.ucl.ac.uk/staff/p.ohearn/SeparationLogic/SeparationLogic/SL_Home.html](http://www0.cs.ucl.ac.uk/staff/p.ohearn/SeparationLogic/SeparationLogic/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 $\neg \ast$, 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