Separation Logic
Introduction

Jean-Marie Madiot

Inria Paris

January 21, 2020
Separation Logic:
a set of rules for reasoning about programs that mutate the heap.
Separation Logic

Separation Logic:
a set of rules for reasoning about programs that mutate the heap.

Separation Logic with interactive proofs:
a successful approach to the verification of imperative programs.
Where does it come from?

- 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.
# How successful?

<table>
<thead>
<tr>
<th>Micro-controller</th>
<th>Klein et al</th>
<th>NICTA</th>
</tr>
</thead>
<tbody>
<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 (concurrent)</td>
<td>Appel et al</td>
<td>Princeton</td>
</tr>
<tr>
<td>C11 (concurrent)</td>
<td>Vafeiadis et al</td>
<td>MPI and MSR</td>
</tr>
<tr>
<td>Java</td>
<td>Parkinson et al</td>
<td>MSR and Cambridge</td>
</tr>
<tr>
<td>Java</td>
<td>Jacobs et al</td>
<td>Leuven</td>
</tr>
<tr>
<td>Javascript</td>
<td>Gardner et al</td>
<td>Imperial College</td>
</tr>
<tr>
<td>ML</td>
<td>Morisset et al</td>
<td>Harvard</td>
</tr>
<tr>
<td>OCaml</td>
<td>Charguéraud</td>
<td>Inria</td>
</tr>
<tr>
<td>SML</td>
<td>Myreen et al</td>
<td>U. of Cambridge</td>
</tr>
</tbody>
</table>

Isabelle
Coq
Coq
Other
Coq
Paper
Other
Verifast
Paper
Coq
Coq
HOL

and many more...
Why successful?

- Specifications that one can read: concise and intuitive.
- Proofs that one can understand: step-by-step analysis.
- Modularity that enable scaling up: > 10k loc.
Why successful?

- Specifications that one can read: concise and intuitive.
- Proofs that one can understand: step-by-step analysis.
- Modularity that enable scaling up: $>10k$ loc.

Separation Logic $+$ interactive proofs $=$ no limits
Purpose of the course

Goal: make you an expert in Separation Logic for sequential programs.

You will learn how Separation Logic handles:

- Tree-shaped structures
- Structures with sharing
- Polymorphic functions
- Higher-order functions
- Abstraction
- Modularity
- Interactive proofs
Choice of the logic

We will *define* Separation Logic in terms of a standard higher-order logic.

Our definitions will be given in Coq, because:

1. most formalizations of SL are in Coq;
2. my own work belong to this majority.
Choice of the source language

Separation Logic can be adapted to pretty much any language.

To best present all its aspects, we need a language with:

- a clean and concise syntax,
- null pointers,
- pointer arithmetic,
- types and polymorphism,
- higher-order functions.
Choice of the source language

Separation Logic can be adapted to pretty much any language.

To best present all its aspects, we need a language with:

- a clean and concise syntax,
- null pointers,
- pointer arithmetic,
- types and polymorphism,
- higher-order functions.

We will use a virtual language: OCaml extended with null pointers and pointers arithmetic; we ignore the existence of headers used by the GC.

(see Coq definitions)