Current trends in Separation Logic

François Pottier

February 8, 2023

Some History

Early Days: 1970–2000

Floyd (1967) and Hoare (1969).

- assertions about global state

not modular!

Owicki & Gries (1976).

- is every assertion stable under the interference of every instruction?

not modular!

RG. Jones (1983).

- what interference must each thread be prepared to tolerate?
- what interference is each thread allowed to inflict?

Days of Separation: 2000–2010

SL.

O'Hearn, Reynolds & Yang (2001).

- separation / ownership
- stable knowledge / permission to update — every assertion is stable!
- malloc / free, lock / unlock — exhale/inhale: ownership transfers

CSL.

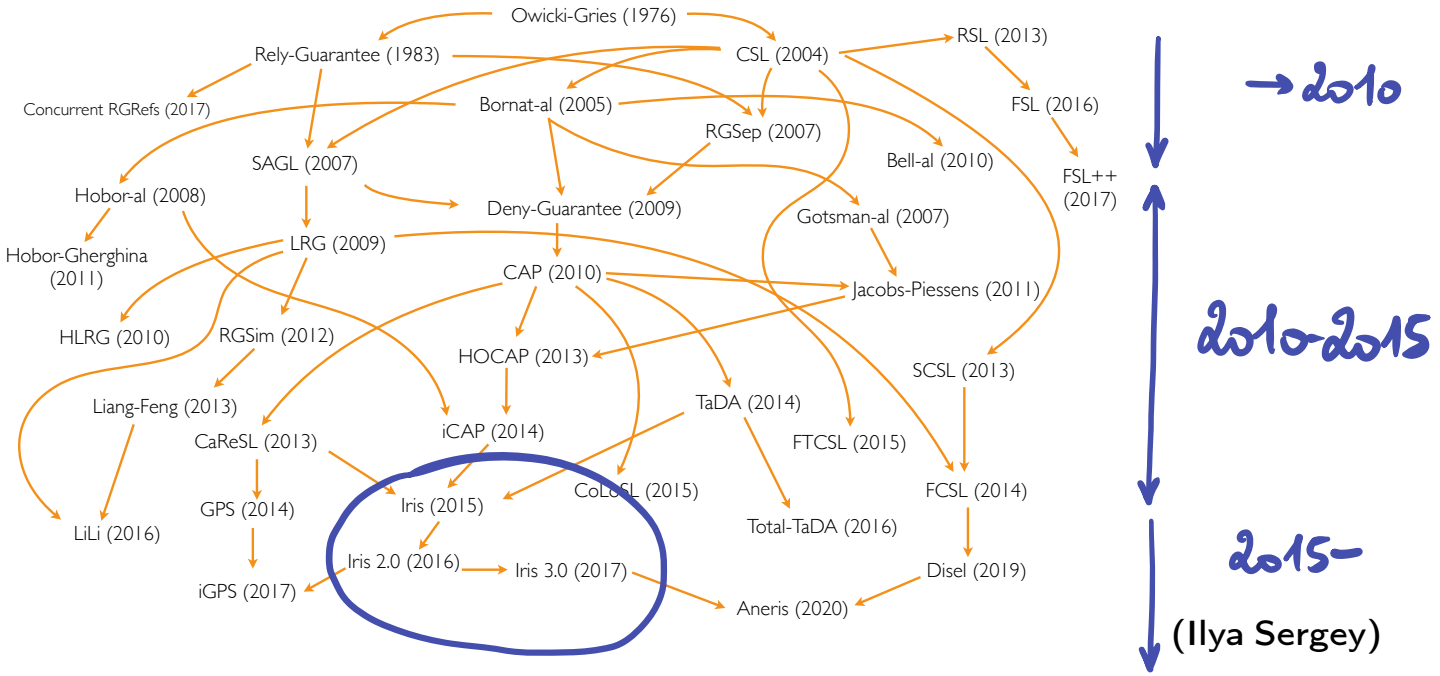
O'Hearn (2007), Gotsman+ (2007), Hobor+ (2008), Buisse+ (2011).

- primitive locks
- data-race-free programs
- dynamic allocation of new locks: challenging!

  the heap stores locks whose invariants describe the heap

  circularity!

Days of Bloom: 2010–202X



Iris.

Jung+ (2015, 2016, 2017, 2018).

want stability under controlled interference

- true separation only a special case
- fiction of separation: machine state + ghost state + invariants

user-defined

- guarded recursion

Some Recent Work and Ongoing Trends

Specification and verification of lock-free data structures.

- logically atomic triples (Jung, 2019)
- linearisation point = ghost update (Jacobs+, 2011)
- helping = letting another thread execute the ghost update
- prophecy = write-once ghost cell (Jung+, 2020) ‖ case analysis on the future!
- marriage of Sep. Logic and linearisability (Birkedal+, 2021)
- industrial use (Carbonneaux+, 2021)

Weak memory. Relaxed data structures.

Cosmo. @Inria

- OCaml 5 (Mével+, 2020, 2021). View-dependent assertions.

+ lots of work on other weak mem. models

Relational Reasoning

A relational Separation Logic relates two programs: $\{P\}\ e_1 \preceq e_2\ \{Q\}$.

Concurrency.

ReLoC.

- fine-grained refines coarse-grained (Frumin+, 2021)

Distributed programming, CRDTs, ...

Trillium.

- concrete implementation refines abstract model (Timany+, 2021)

Compiler verification. Rich area for future work?

Simulinis.

- compiled program refines source program (Gäher+, 2022)

Security.

Tiniris.

- noninterference (Gregersen+, 2021)

Actris (Hinrichsen+, 2020, 2022).

- communication via a two-way channel

Effect handlers (de Vilhena+, 2021). Hazel. @Inria

- communication via jumps between handlee and handler

Aneris (Krogh-Jespersen+, 2020; Gondelman+ (2021)

- communication via message-passing

Charguéraud+ (2019), Mével+ (2019), Moine+ (2022). @Inria

- time credits 1$
- big-O reasoning: Guéneau+ (2018, 2019). @Inria

Madiot+ (2022), Moine+ (2023). @Inria

- space credits 1$\diamond$
- tracing GC, implicit deallocation — ghost operation!
- pointed-by-heap and pointed-by-thread assertions + concurrency!
  (reverse edges) (roots)

Establishing Type Soundness

Timany+ (2022).

- after 20 years of domination by the syntactic approach,
- the semantic approach to type soundness proofs emerges again.

type = set of values

Val → iProp

Examples:

- λRust: Jung+ (2018) RustBelt.
- gDOT: Giarrusso+ (2020)

Some Challenges and Future Directions

Termination and liveness.

- Transfinite Iris (Spies+, 2021)
- TaDA Live (D'Osualdo+, 2022)
- notify/wait (Hamin+, 2018; Reinhard+, 2021)

Improving automation.

- VeriFast, Viper, diaframe (Mulder +, 2021)
- RefinedC (Sammler+, 2021)

Reasoning about I/O and systems code.

Reasoning about probabilistic programs.