Katamaran

Semi-Automated Verification of Instruction Set Architectures

Steven Keuchel

Georgy Lukyanov

Dominique Devriese

June 16, 2020

SOFTWARE LANGUAGES LAB

Newcastle University

Hardware (Assisted) Security

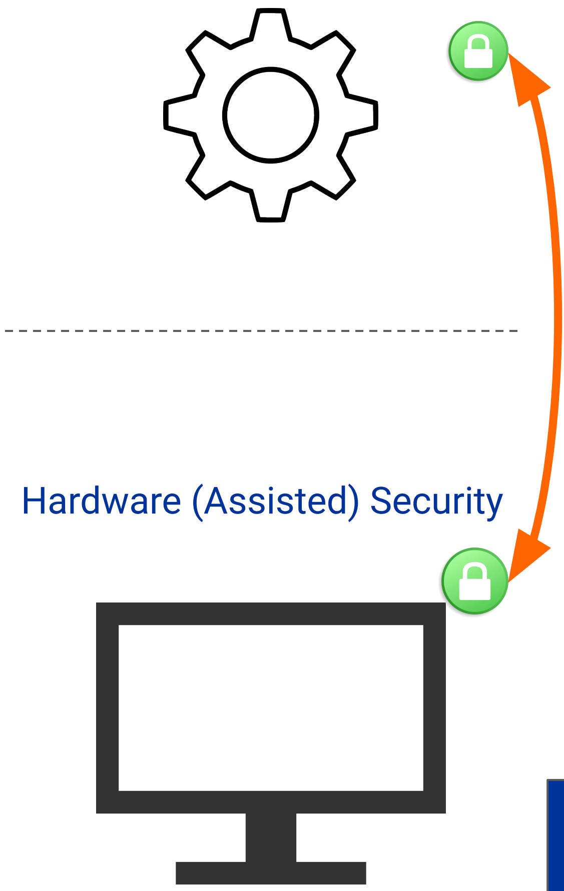
CHERI

Intel SGX

ARM TrustZone

Program Security

Hardware (Assisted) Security



“Robust and compositional verification of object capability patterns.”
Swasey, Garg & Dreyer. *OOPSLA’17.*

“Reasoning about object capabilities with logical relations and effect parametricity.”
Devriese, Birkedal & Piessens. *EuroS&P’16.*

“Linear Capabilities for Fully Abstract Compilation of Separation-Logic-Verified Code.”
Van Strydonck, Piessens & Devriese. *ICFP’19.*

“Beyond good and evil: Formalizing the security guarantees of compart-mentalizing compilation”.
Juglaret et al. *CSF’16.*

"Reasoning about a Machine with Local Capabilities.”
Skorstengaard et al, *TOPLAS 42.1 (2019).*

CHERI
“Rigorous engineering for hardware security: Formal modelling and proof [..].”
Nienhuis et al. *IEEE S&P’20.*

PUMP
“Micro-policies: Formally verified, tag-based security monitors.”
De Amorim et al. *IEEE S&P’15.*

Program Security

Hardware (Assisted) Security

"Robust and compositional verification of object capability patterns."
Swasey, Garg & Dreyer. *OOPSLA'17.*

"Reasoning about object capabilities with logical relations and effect parametricity."
Devriese, Birkedal & Piessens. *EuroS&P'16.*

"Linear Capabilities for Fully Abstract Compilation of Separation-Logic-Verified Code."
Van Strydonck, Piessens & Devriese. *ICFP'19.*

"Beyond good and evil: Formalizing the security guarantees of compart-mentalizing compilation".
Juglaret et al. *CSF'16.*

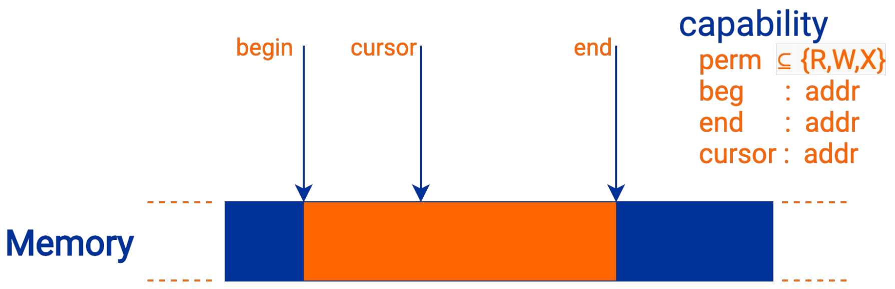
"Reasoning about a Machine with Local Capabilities."
Skorstengaard et al, *TOPLAS 42.1 (2019).*

CHERI
"Rigorous engineering for hardware security: Formal modelling and proof [..]."
Nienhuis et al. *IEEE S&P'20.*

PUMP
"Micro-policies: Formally verified, tag-based security monitors."
De Amorim et al. *IEEE S&P'15.*

Capability Safety for Capability Machines

Capability Machines



Hardware Guarantees

- Capabilities are unforgeable
- Permissions are checked
- Capability manipulation is safe

Can we verify this on the spec?

Example: Store Instruction

```
unit execute_store (d : reg, s : reg) :=
  let: c  : cap  := call read_reg_cap d in
  let: wa : bool := call write_allowed c.perm in
  let: wb : bool := call within_bounds c in
 assert (wa && wb) ;;
  let: w : cap + int := call read_reg_word s in
 call write_mem c.cursor w ;; call update_pc
```

μSail code for `execute_store`

Universal Contract

$$\{\mathop{\ast}_{r \in \mathbf{reg}} r \mapsto w_r * \mathrm{safe}(w_r)\}$$

`execute_store d s`

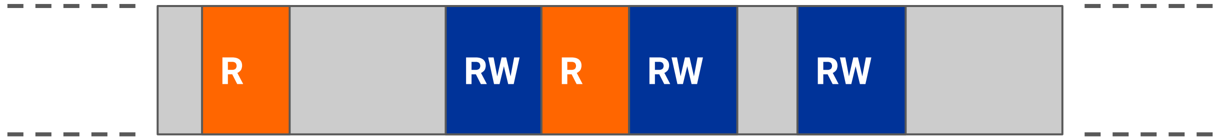
$$\{\mathop{\ast}_{r \in \mathbf{reg}} r \mapsto w_r * \mathrm{safe}(w_r)\}$$

Checks are critical!!

"Reasoning about a Machine with Local Capabilities."
Skorstengaard et al, *TOPLAS 42.1 (2019).*

Universal Safety Contract

Memory subset m



```
safe(cap(p,b,e,a))
  ⇔ [b,e[ ⊆ dom(m)
  ∧ (R ⊑ p =>
      ∀ a ∈ [b,e[. safe(m(a)))
  ∧ ...
```

$$\boxed{\{\mathop{\ast}_{r \in \mathbf{reg}} r \mapsto w_r \ast \mathrm{safe}(c_r)\}}$$

```
unit execute_store (d : reg, s : reg) :=
  let: c  : cap  := call read_reg_cap d in
```

$$\boxed{\{(\mathop{\ast}_{r \in \mathbf{reg}} r \mapsto w_r \ast \mathrm{safe}(c_r)) \ast \ulcorner c = w_d \wedge c = \mathrm{cap}(p, b, e, a)\urcorner\}}$$

```
  let: wa : bool := call write_allowed c.perm in
  let: wb : bool := call within_bounds c in
  assert (wa && wb) ;;
```

$$\boxed{\{\ldots c = \mathrm{cap}(p, b, e, a) \wedge p \sqsupseteq \mathrm{W} \wedge b \leq a < e\urcorner\}}$$

```
  let: w : cap + int := call read_reg_word s in
```

$$\boxed{\{\ldots c = \mathrm{cap}(p, b, e, a) \wedge p \sqsupseteq \mathrm{W} \wedge b \leq a < e \wedge w = w_s\urcorner\}}$$

```
  call write_mem c.cursor w ;; call update_pc
```

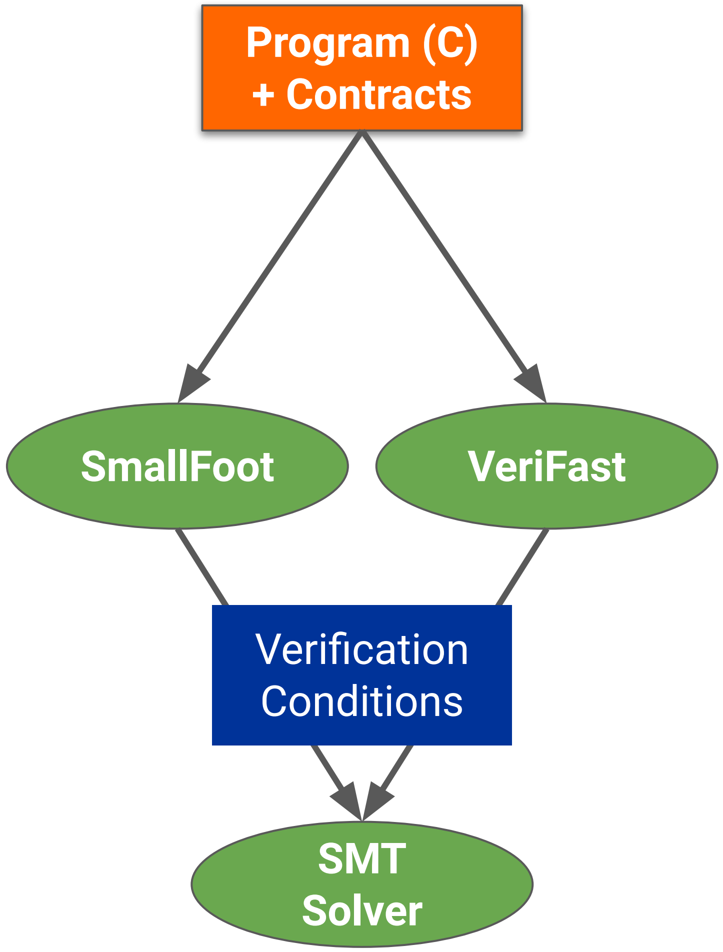
$$\boxed{\{\mathop{\ast}_{r \in \mathbf{reg}} r \mapsto w_r \ast \mathrm{safe}(c_r)\}}$$

$$\{\mathrm{safe}(\mathrm{cap}(p, b, e, -)) \ast \mathrm{safe}(w) \ast \ulcorner b \leq a < e \wedge p \sqsupseteq \mathrm{W}\urcorner\}$$

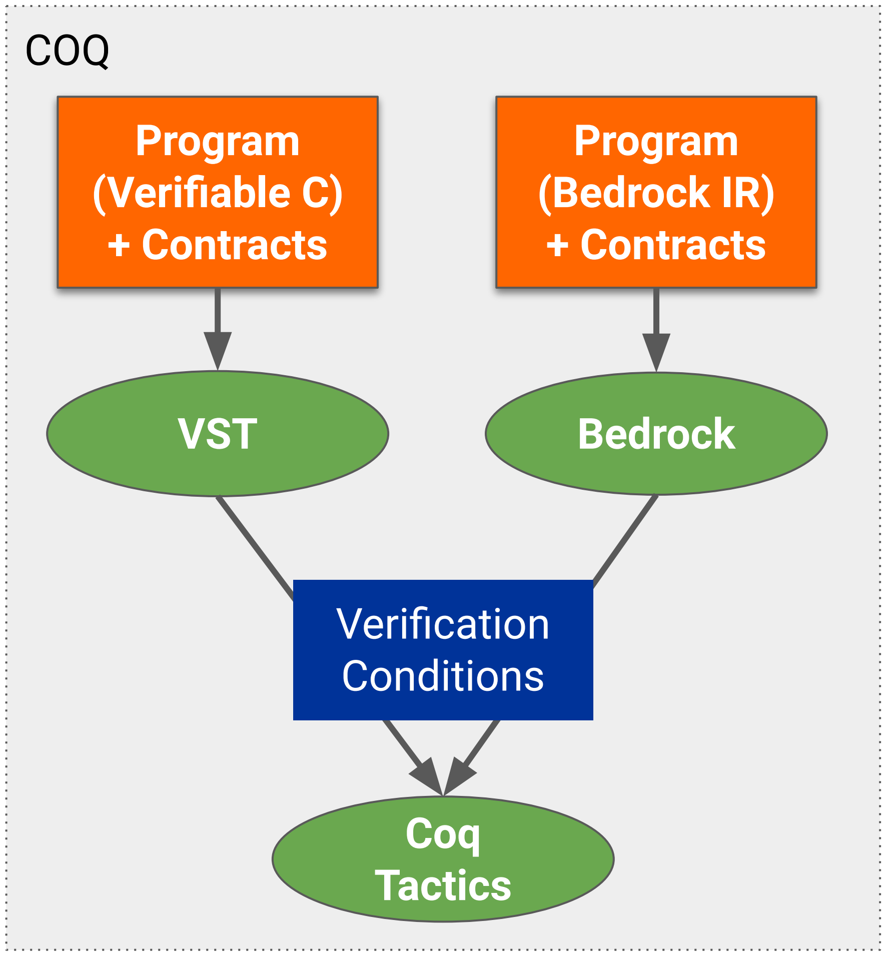
```
  write_mem a w
```

$$\{\mathrm{safe}(\mathrm{cap}(p, b, e, -)) \ast \mathrm{safe}(w)\}$$

Verifiers



Verified Verifiers

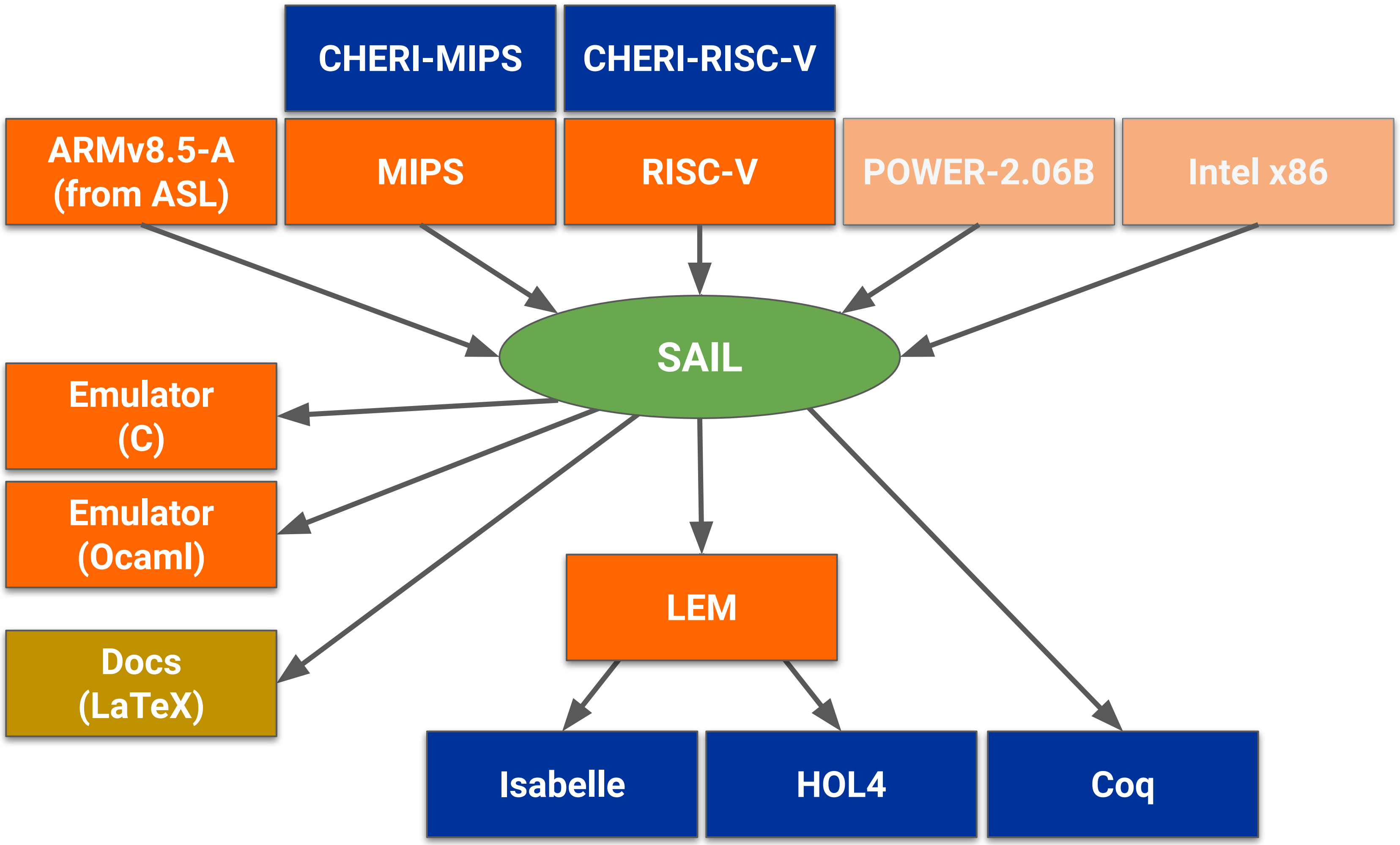


COQ
Program (Verifiable C) + Contracts
Program (Bedrock IR) + Contracts
VST
Bedrock
Verification Conditions
Coq Tactics

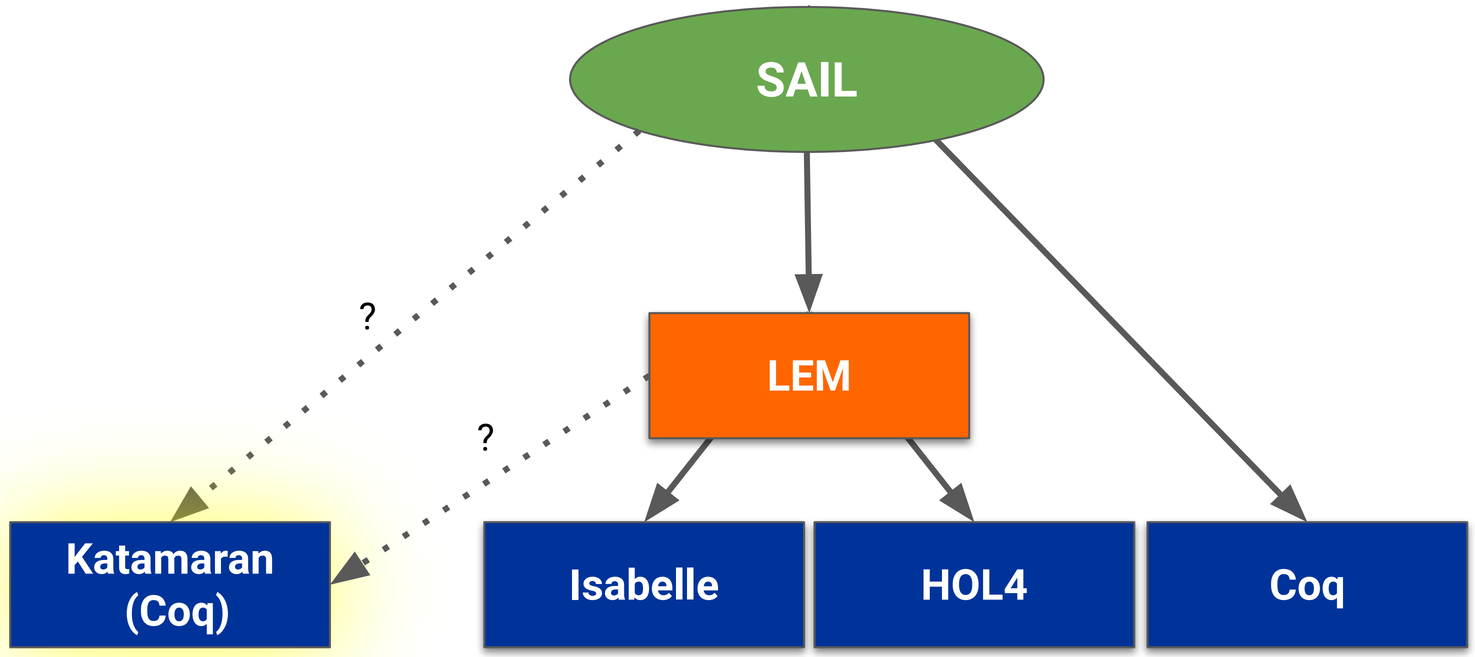
Katamaran

Verified Semi-Automated Separation Logic Verifier for Sail

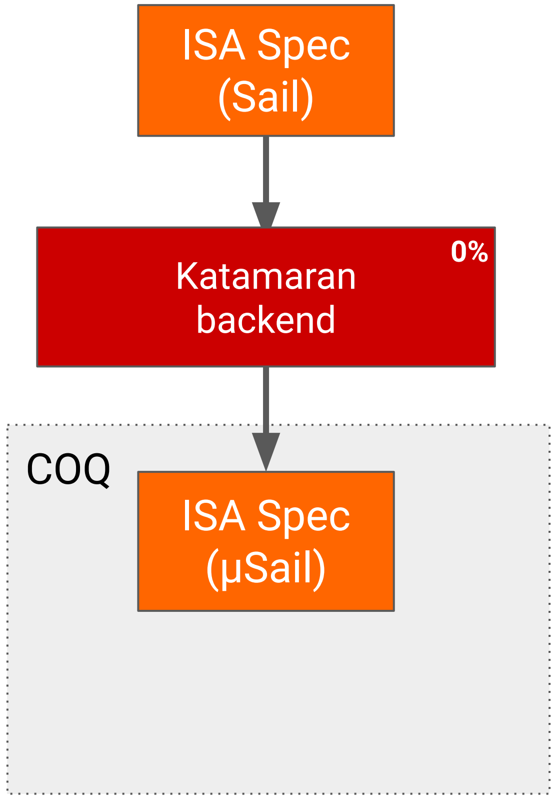
Sail DSL for ISA Specifications



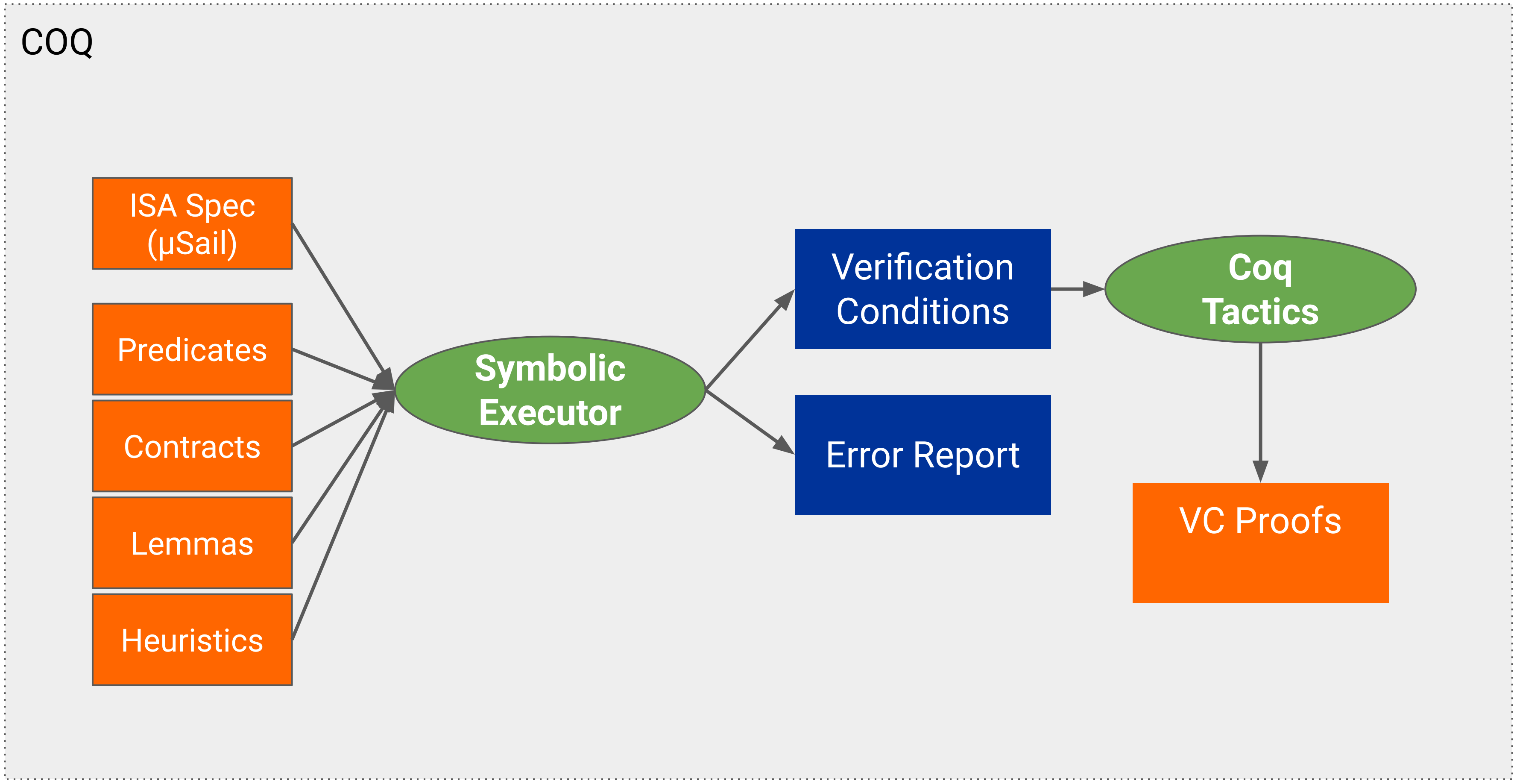
Sail DSL for ISA Specifications



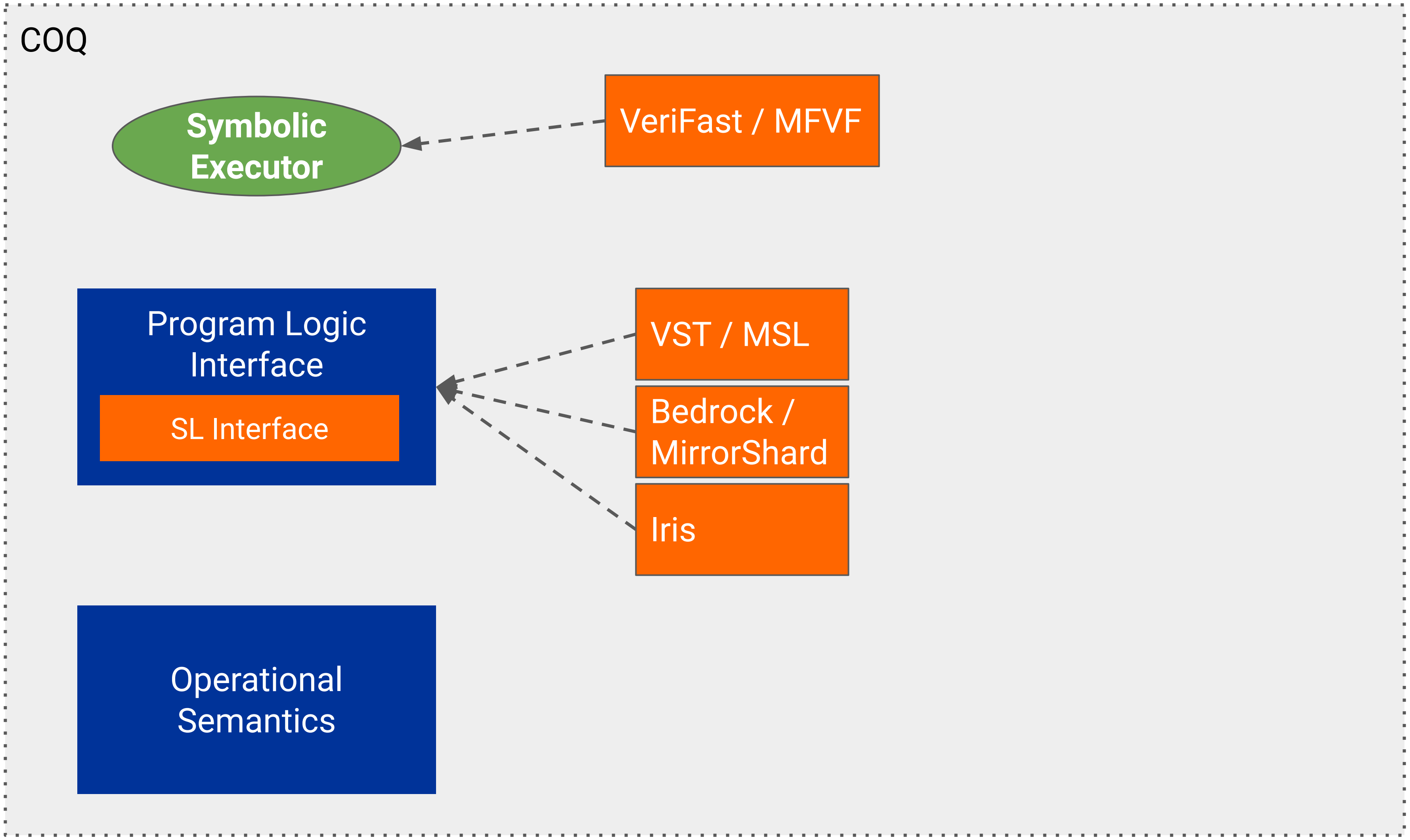
Katamaran



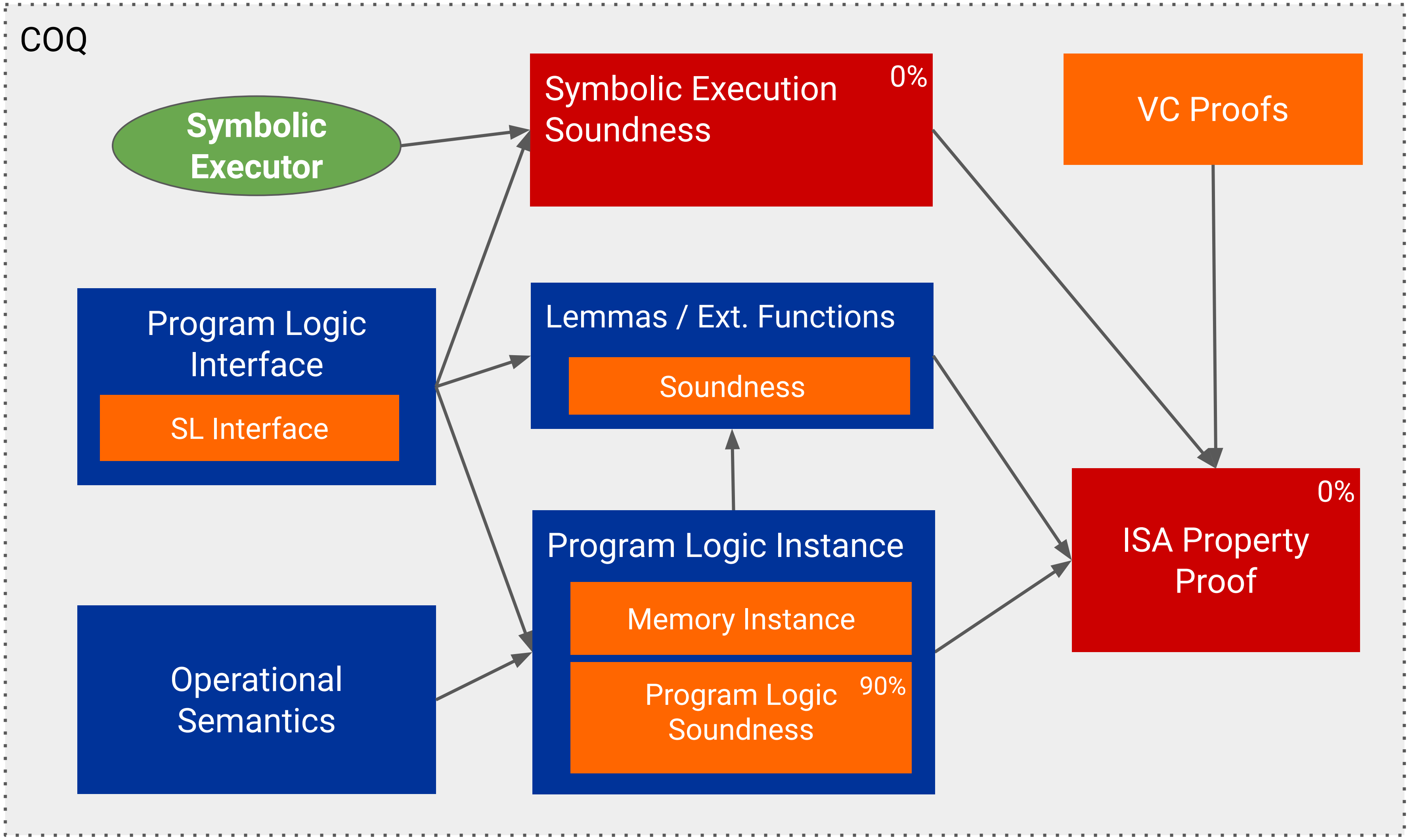
Katamaran Workflow



Katamaran Structure



Katamaran Structure



μSail Language Features

🟩 Mutable variables	🟩 Registers	🟩 External functions
🟩 Primitive types (Bool, enum, int, string,...)	🟩 Structured types (List, records, unions)	🟩 Type polymorphism
🟥 Bitvectors	🟥 Int/bool/order polymorphism	🟥 Return, exceptions, while-loops
▨ Scattered Definitions	▨ Bidirectional mappings	▨ Complex l-values

🟩 Supported   🟥 Unsupported / Maybe planned   ▨ Not planned

Sail Proof Support	Katamaran
• Shallow embedded syntax	• Deeply embedded syntax
• Monadic semantics (free prompt monad / state monad)	• Operational semantics
• Prover's assertion logic	• Embedded separation logic
• LTac / Eisbach	• Gallina (reflective proofs)

Future Work

Short Term Future

Program Logic Soundness

Symbolic Execution Soundness

Automation

Case Study: Register only capabilities

Mid Term Future

Linear capabilities
Skorstengaard, Devriese & Birkedal. "StkTokens: Enforcing well-bracketed control flow and stack encapsulation using linear capabilities." *POPL'19.*

Program Logic Instance
Iris?

Local capabilities
Skorstengaard, Devriese & Birkedal. "Reasoning about a Machine with Local Capabilities." *TOPLAS* 42.1 (2019).

Uninitialized capabilities
Huyghebaert, Van Strydonck, Keuchel & Devriese. "Uninitialized Capabilities." *arXiv:2006.01608 (2020)*.

Language Features
Bitvectors

REDFIN - REDuced instruction set for Fixed-point & INteger arithmetic
Mokhov, Lukyanov & Lechner. "Formal Verification of Spacecraft Control Programs." *Haskell'19.*

Long Term Future

Sail Integration

CHERI
Woodruff et al. “The CHERI capability model: Revisiting RISC in an age of risk.” ISCA’14.

Intel SGX
McKeen et al. “Innovative instructions and software model for isolated execution.” HASP’13.

Secure Compilation
Patrignani, Ahmed & Clarke. “Formal approaches to secure compilation: A survey of fully abstract compilation and related work.” *CSUR 51.6 (2019).*

Thanks for your Attention!

https://github.com/skeuchel/katamaran