Concurrent Cerise
A program logic for multi-core capability machine

June ROUSSEAU
Supervised by Lars Birkedal and Aïna Linn Georges
LogSem, Aarhus University, Denmark

2021

Formal methods on capability machine

Architecture security-oriented

- ˜70% Microsoft security updates: memory safety
- Coarse-grained memory compartmentalization
- Capabilities Hardware-Enhanced RISC Instructions (CHERI)
- Arm Morello — Prototype January 2022

Formal methods

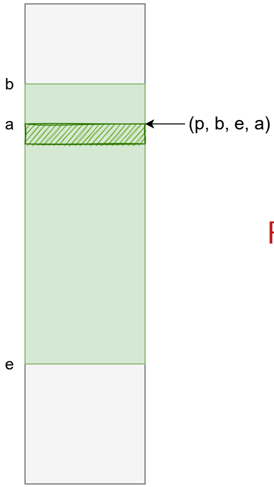
- Strong security guarantees
- Cerise: program logic & logical relation
- Huge gap between model and real-world machine

Outline

1. Context
   - Capability machines
   - Separation logic
   - Motivations
2. Contributions
   - Threat model
   - Program logic
   - Case study
3. Conclusion

Context

Capabilities machine



Hardware capability

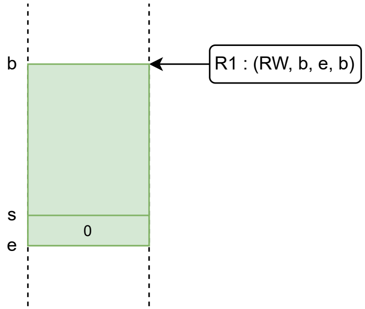
- Usual CPU: everything is integer
- Capability machine: integer for arithmetic, capability for pointers
- Unforgeable token of authority

Representation

Capability **(p,b,e,a)**

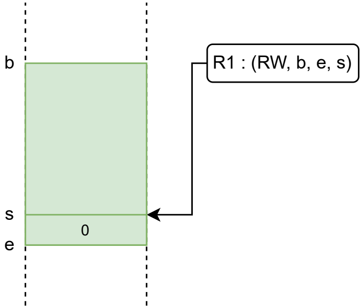
$p \in \{\mathrm{RO}, \mathrm{RW}, \mathrm{RWX}, \ldots\}$	permission
$b \in Addr$	base address
$e \in Addr$	end address
$a \in Addr$	current address

Example sub-buffer



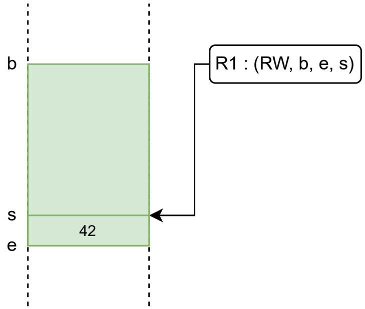
```
lea r1 [s-b]
store r1 42
lea r1 -[s-b]
subseg r1 b s
jmp radv
```

Example sub-buffer



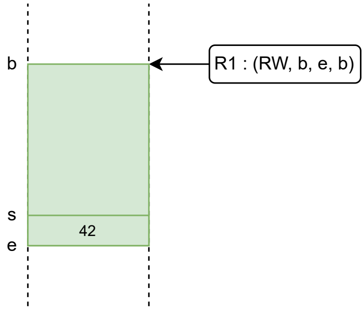
```
lea r1 [s-b]    <---
store r1 42
lea r1 -[s-b]
subseg r1 b s
jmp radv
```

Example sub-buffer



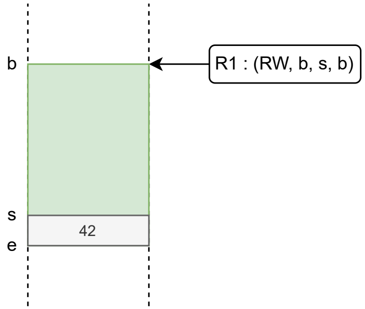
```
lea r1 [s-b]
store r1 42     <---
lea r1 -[s-b]
subseg r1 b s
jmp radv
```

Example sub-buffer



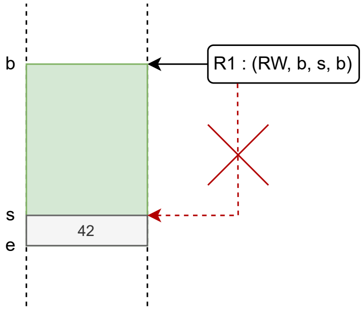
```
lea r1 [s-b]
store r1 42
lea r1 -[s-b]   <---
subseg r1 b s
jmp radv
```

Example sub-buffer



```
lea r1 [s-b]
store r1 42
lea r1 -[s-b]
subseg r1 b s   <---
jmp radv
```

Example sub-buffer



```
lea r1 [s-b]
store r1 42
lea r1 -[s-b]
subseg r1 b s
jmp radv          <---
```

Separation Logic in Iris

Iris: highly expressive framework for separation logic

Separation logic

- Logic of resources
- Separation conjunction $*$
  - exclusive ownership
  - disjoints resources
- Magic wand $-\!\!*$

$$P \triangleq (\mathsf{r}_1 \Mapsto_r (\textsc{rwx}, \mathit{init}, \mathit{end}, \mathit{init}) * \mathit{init} \mapsto_a 0)$$

Cerise

What is Cerise ?

Operational semantic

- semantic for capability machine ISA
- security properties

Program logic

- resource for machine registers and memory addresses
- program specification *à la* Hoare triples

Logical relation

- *safe-to-share*: transitive access to *safe-to-share* words
- *safe-to-execute*: execute safely in any *safe context*

Goal & Motivations

Motivations

- Single-core machine
- Concurrency orthogonal with security ?
- Reduce gap between formal model and real-world machine

Goal — Add concurrency

- Define threat model
- Update semantic, program logic, logical relation
- Case study
- Synchronization

Contributions

Assumptions

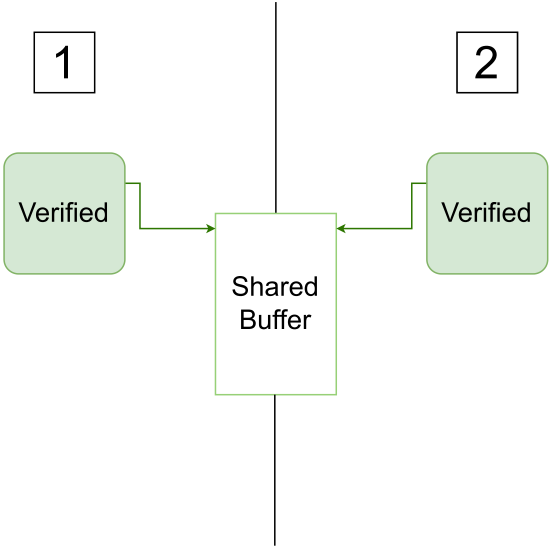
Model

- small ISA
- no virtual memory / page table
- sequentially consistent memory model

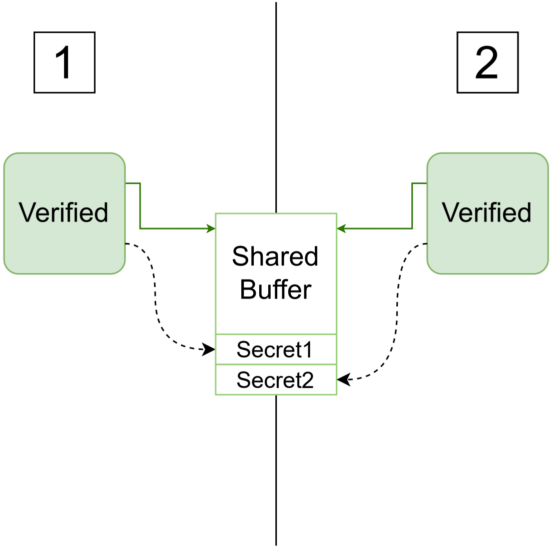
Sequentially consistent

- interleaving instructions
- non deterministic
- locally, behaves as sequential execution

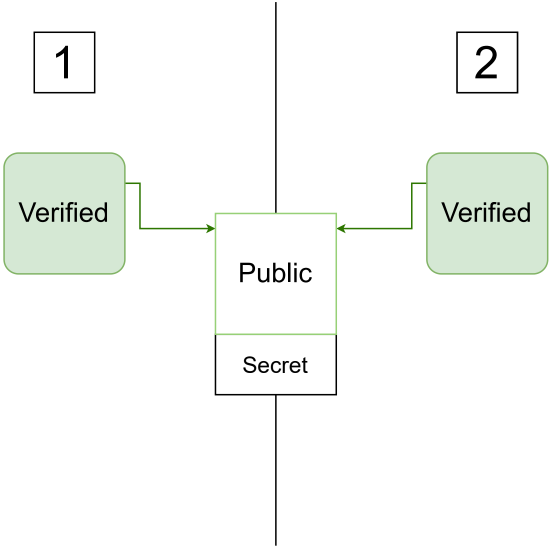
Threat model



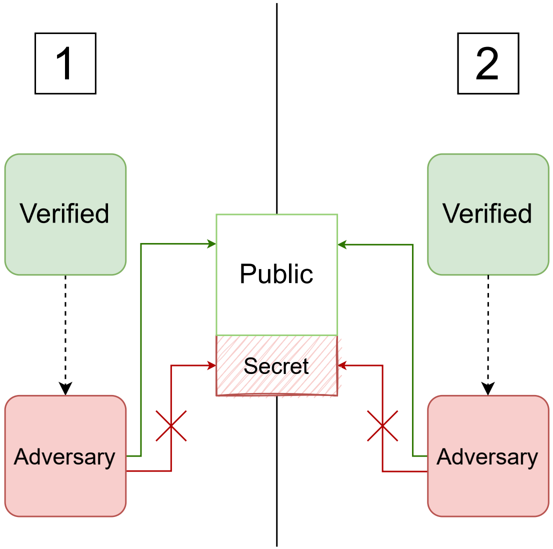
Threat model



Threat model

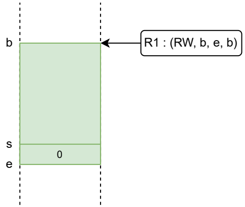


Threat model



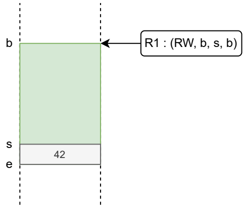
Sub-buffer — Non concurrent

$$\left\{ (\textsc{rwx}, \mathsf{init}, \mathsf{end}, \mathsf{init}); \begin{array}{l} (i, \mathsf{r}_0) \Mapsto w_{adv} \ast \\ (i, \mathsf{r}_1) \Mapsto (RW, b, e, b) \ast \\ [\mathsf{b}, \mathsf{s}) \mapsto [0 \ldots 0] \ast \\ \mathsf{s} \mapsto 0 \ast \\ [\mathsf{init}, \mathsf{end}) \mapsto \mathrm{prog\_instrs} \end{array} \right\}$$



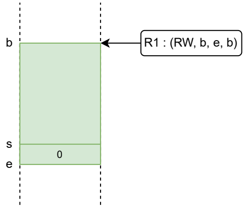
$$\overset{i}{\rightsquigarrow}$$

$$\left\{ w_{adv}; \begin{array}{l} (i, \mathsf{r}_0) \Mapsto w_{adv} \ast \\ (i, \mathsf{r}_1) \Mapsto (RW, b, \color{red}{s}, b) \ast \\ [\mathsf{b}, \mathsf{s}) \mapsto [0 \ldots 0] \ast \\ \mathsf{s} \mapsto \color{red}{42} \ast \\ [\mathsf{init}, \mathsf{end}) \mapsto \mathrm{prog\_instrs} \end{array} \right\}$$



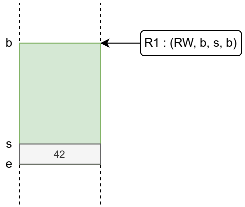
Sub-buffer — Non concurrent

$$\left\{ (\mathrm{RWX}, \mathsf{init}, \mathsf{end}, \mathsf{init});\ \begin{array}{l} (i, \mathsf{r}_0) \Rrightarrow w_{adv} * \\ (i, \mathsf{r}_1) \Rrightarrow (RW, b, e, b) * \\ {\color{red}[\mathsf{b}, \mathsf{s}) \mapsto [0 \ldots 0]} * \\ {\color{red}\mathsf{s} \mapsto 0} * \\ [\mathsf{init}, \mathsf{end}) \mapsto \mathrm{prog\_instrs} \end{array} \right\}$$



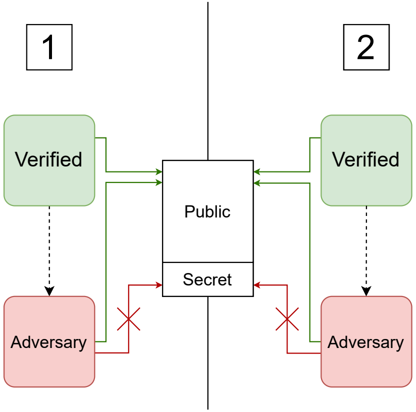
$$\overset{i}{\rightsquigarrow}$$

$$\left\{ w_{adv};\ \begin{array}{l} (i, \mathsf{r}_0) \Rrightarrow w_{adv} * \\ (i, \mathsf{r}_1) \Rrightarrow (RW, b, s, b) * \\ {\color{red}[\mathsf{b}, \mathsf{s}) \mapsto [0 \ldots 0]} * \\ {\color{red}\mathsf{s} \mapsto 42} * \\ [\mathsf{init}, \mathsf{end}) \mapsto \mathrm{prog\_instrs} \end{array} \right\}$$



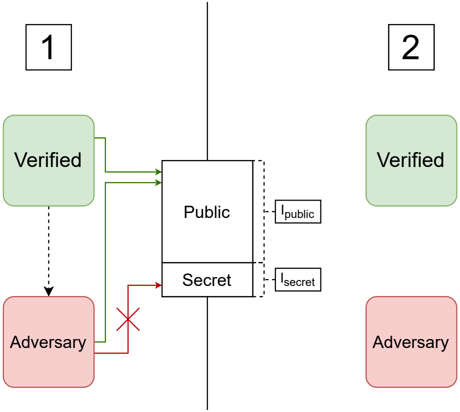
Shared sub-buffer — Invariant

I_{public}	–	only *safe-to-share* words
I_{secret}	–	secret data



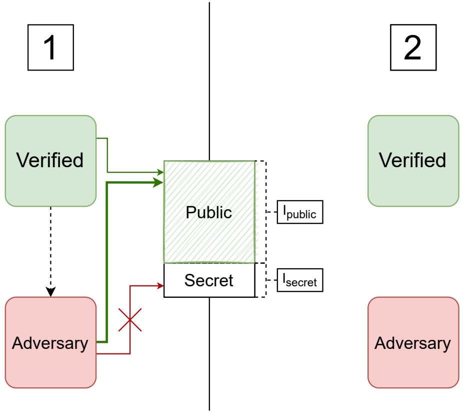
Shared sub-buffer — Invariant

I_{public}	–	only *safe-to-share* words
I_{secret}	–	secret data



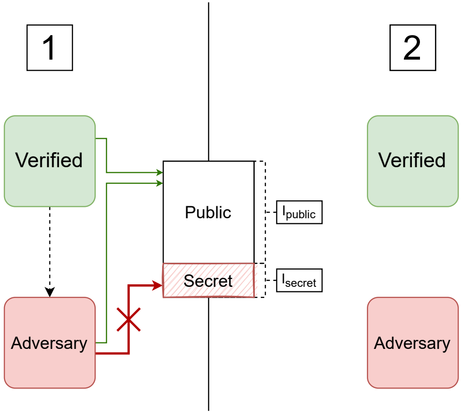
Shared sub-buffer — Invariant

$$\begin{array}{lcl} I_{public} & \triangleq & \mathop{\Large\ast}_{a \in [\mathrm{b,s})} \exists w, a \mapsto w * \mathcal{V}(w) \\ I_{secret} & - & \text{secret data} \end{array}$$



Shared sub-buffer — Invariant

$$\begin{aligned} I_{public} &\triangleq \mathop{\ast}_{a \in [\mathrm{b},\mathrm{s})} \exists w, a \mapsto w * \mathcal{V}(w) \\ I_{secret} &\triangleq (s1 \mapsto 0 \vee s1 \mapsto secret_1) \;\wedge\; (s2 \mapsto 0 \vee s2 \mapsto secret_2) \end{aligned}$$



Shared sub-buffer — Full specification

$$\boxed{I_{public}} * \boxed{I_{secret}} \vdash$$

$$\left\{ (\textsc{rwx}, \mathsf{init}, \mathsf{end}, \mathsf{init});\ \begin{array}{l} (i, \mathsf{r}_0) \Mapsto w_{adv} * \\ (i, \mathsf{r}_1) \Mapsto (RW, b, e, b) * \\ [\mathsf{init}, \mathsf{end}) \mapsto \mathrm{prog\_instrs} \end{array} \right\}$$

$$\overset{i}{\rightsquigarrow}$$

$$\left\{ w_{adv};\ \begin{array}{l} (i, \mathsf{r}_0) \Mapsto w_{adv} * \\ (i, \mathsf{r}_1) \Mapsto (RW, b, s, b) * \\ [\mathsf{init}, \mathsf{end}) \mapsto \mathrm{prog\_instrs} \end{array} \right\}$$

Complete specification

- assume the previous specification for each core
- Fundamental Theorem Logical Relation (non-trivial)
- safely complete execution

Adequacy theorem $\rightarrow$ invariant on operational semantic

Conclusion

Further works

Synchronization

- Compare and Swap instruction
- Spinlock library
- Concurrently safe macro for dynamic allocation
- Scenario involving malloc

Other

- Other scenarios
- Pedagogical exercises to learn Cerise in Coq (WIP)

Summary and Future Work

Reason formally on multi-core capability machines

Summary

- Extend Cerise with concurrency
- Define the threat model
- Add synchronization mechanism
- Design, specify and prove case studies
- Mechanized and proved with Iris and Coq

Future work

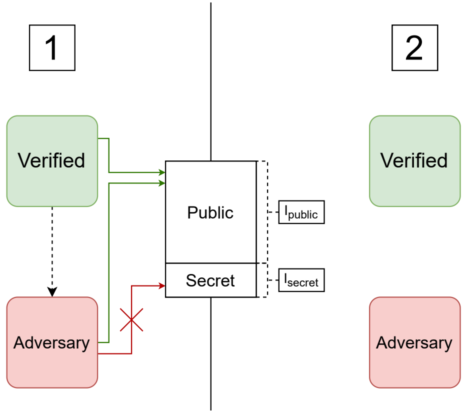
- Relaxed memory model
- Real world ISA
- Virtual memory / Page tables
- Security properties, including interrupts/exceptions

Appendix

Hoare triples — Example load

$$\frac{\mathrm{ValidPC}(p_{pc}, b_{pc}, e_{pc}, a_{pc}) \\ \mathrm{ValidLoad}(p, b, e, a) \qquad \mathrm{decode}(n) = \texttt{load}\ dst\ src}{\left\langle \begin{array}{l} (i, \mathrm{pc}) \Mapsto (p_{\mathrm{pc}}, b_{\mathrm{pc}}, e_{pc}, a_{\mathrm{pc}}) * a_{\mathrm{pc}} \mapsto n * \\ (i, dst) \Mapsto - * (i, src) \Mapsto (p, b, e, a) * a \mapsto w \end{array} \right\rangle \xrightarrow{i} \left\langle \begin{array}{l} (i, \mathrm{pc}) \Mapsto (p_{\mathrm{pc}}, b_{\mathrm{pc}}, e_{\mathrm{pc}}, a_{\mathrm{pc}} + 1) * a_{\mathrm{pc}} \mapsto n * \\ (i, dst) \Mapsto w * (i, src) \Mapsto (p, b, e, a) * a \mapsto w \end{array} \right\rangle}$$

Shared Sub-buffer — Invariant



$$\begin{aligned} I_{public} &\triangleq \mathop{\ast}_{a \in [\mathrm{p},\mathrm{s})} \exists w, a \mapsto w * \mathcal{V}(w) \\ I_{secret} &\triangleq (s1 \mapsto 0 \vee s1 \mapsto secret_1) \wedge (s2 \mapsto 0 \vee s2 \mapsto secret_2) \end{aligned}$$

Logical relation — Full definition

$$\mathcal{V}(w) \quad \begin{cases} \mathcal{V}(z), \mathcal{V}(\mathrm{O}, -, -, -) & \triangleq \text{True} \\ \mathcal{V}(\mathrm{E}, b, e, a) & \triangleq \triangleright \square\, \mathcal{E}(\mathrm{RX}, b, e, a) \\ \mathcal{V}(\mathrm{RW}/\mathrm{RWX}, b, e, -) & \triangleq \mathop{\Large\ast}_{a \in [b,e)} \boxed{\exists w,\ a \mapsto w \ast \mathcal{V}(w)} \\ \mathcal{V}(\mathrm{RO}/\mathrm{RX}, b, e, -) & \triangleq \mathop{\Large\ast}_{a \in [b,e)} \exists P,\ \boxed{\exists w,\ a \mapsto w \ast P(w)} \ast \triangleright \square\, (\forall w,\ P(w) \twoheadrightarrow \mathcal{V}(w)) \end{cases}$$

$$\mathcal{E}(w) \quad \triangleq \quad \forall i\ reg,\ \left\{ (i, w);\ \mathop{\Large\ast}_{\substack{((j,r),v) \in reg,\\ i=j \\ r \neq \mathrm{pc}}} (j, r) \Mapsto v \ast \mathcal{V}(v) \right\} \overset{i}{\rightsquigarrow} \bullet$$

Spinlock

Principle

- prevent concurrent access to a critical section
- loop and try acquire the lock since available

Acquire

```
; r0 -> capability pointing to the lock state
loop:
mov r3 0
cas r0 r3 1
jnz r3 [loop]
end:
```

Spinlock

Specification — Acquire

$$\boxed{[a_{acquire}, e_{acquire}) \mapsto instrs_{acquire} * is\_lock\ \gamma\ a_{lock}\ P}$$

$$\vdash \left\{ \begin{array}{l} (i, PC) \Rrightarrow (\mathrm{RWX}, b_{pc}, e_{pc}, a_{acquire}) * \\ (i, r_0) \Rrightarrow (p_{lock}, b_{lock}, e_{lock}, a_{lock}) * [\cdots] \end{array} \right\} \overset{i}{\rightsquigarrow} \left\{ \begin{array}{l} (i, PC) \Rrightarrow (\mathrm{RWX}, b_{pc}, e_{pc}, e_{acquire}) * \\ (i, r_0) \Rrightarrow (p_{lock}, b_{lock}, e_{lock}, a_{lock}) * [\cdots] * \\ P * locked\ \gamma \end{array} \right\}$$

Specification — Release

$$\boxed{[a_{release}, e_{release}) \mapsto instrs_{release} * is\_lock\ \gamma\ a_{lock}\ P}$$

$$\vdash \left\{ \begin{array}{l} (i, PC) \Rrightarrow (\mathrm{RWX}, b_{pc}, e_{pc}, a_{release}) * \\ (i, r_0) \Rrightarrow (p_{lock}, b_{lock}, e_{lock}, a_{lock}) * \\ P * locked\ \gamma \end{array} \right\} \overset{i}{\rightsquigarrow} \left\{ \begin{array}{l} (i, PC) \Rrightarrow (\mathrm{RWX}, b_{pc}, e_{pc}, e_{release}) * \\ (i, r_0) \Rrightarrow (p_{lock}, b_{lock}, e_{lock}, a_{lock}) \end{array} \right\}$$

Operational Semantic

Model N-cores capability machine

$$\begin{array}{llcll}
i & \in & \mathcal{N} & \triangleq & \{i \mid i \in \mathbb{N} \land i < N\} \\
m & \in & \mathrm{Mem} & \triangleq & \mathrm{Addr} \rightarrow \mathrm{Word} \\
reg & \in & \mathrm{Reg} & \triangleq & (\mathcal{N} \times \mathrm{RegName}) \rightarrow \mathrm{Word} \\
\mathcal{C} & \in & \mathrm{CoreState} & ::= & \mathsf{Running} \mid \mathsf{Halted} \mid \mathsf{Failed} \\
\mathcal{E} & \in & \mathrm{ExecState} & \triangleq & \mathcal{N} \rightarrow \mathrm{CoreState} \\
\varphi & \in & \mathrm{Conf} & \triangleq & \mathrm{ExecState} \times \mathrm{Reg} \times \mathrm{Mem}
\end{array}$$

Operational Semantic

Reduction relation

- per-core reduction:
  $(\mathrm{CoreState} \times \mathrm{Reg} \times \mathrm{Mem}) \xrightarrow[core]{i} (\mathrm{CoreState} \times \mathrm{Reg} \times \mathrm{Mem})$
- configuration reduction

$$\frac{(s, r, m) \xrightarrow[core]{i} (s', r', m')}{(\mathcal{E}[i \mapsto s], r, m) \xrightarrow[conf]{} (\mathcal{E}[i \mapsto s'], r', m')}$$

Example: `load` r_1 r_2

$$\frac{\begin{array}{cc} reg(i, r_2) = (p, b, e, a) & \mathrm{ValidLoad}(p, b, e, a) \\ mem(a) = w & reg' \triangleq \mathrm{updPC}(reg(i, r_1) \mapsto w) \end{array}}{(\mathsf{Running}, reg, mem) \xrightarrow[core]{i} (\mathsf{Running}, reg', mem)}$$

Program Logic

Syntax

$P, Q \in \mathit{iProp} ::=$

$\top \mid \bot \mid P \wedge Q \mid \ldots$	HOL
$\mid P * Q \mid P \mathrel{-\!\!*} Q$	separation logic
$\mid \lceil \varphi \rceil \mid \Box\, P \mid \triangleright P$	iris features
$\mid \boxed{P}$	iris invariant
$\mid \mathsf{a} \mapsto w \mid (i, r) \Mapsto w$	machine resources
$\mid \langle P \rangle \xrightarrow{i} \langle s.\, Q \rangle \mid \{P\} \overset{i}{\leadsto} \{s.\, Q\} \mid \{P\} \overset{i}{\leadsto} \bullet$	program logic

Program Logic

Program specification

$\langle P \rangle \xrightarrow{i} \langle s.\, Q \rangle$	single instruction
$\{P\} \overset{i}{\rightsquigarrow} \{s.\, Q\}$	code fragment
$\{P\} \overset{i}{\rightsquigarrow} \bullet$	complete safe execution.

Sequencing rule

Compose the specification together

$$\frac{\{P\} \overset{i}{\rightsquigarrow} \{Q\} \qquad \{Q\} \overset{i}{\rightsquigarrow} \{R\}}{\{P\} \overset{i}{\rightsquigarrow} \{R\}}$$

Logical Relation

Capture semantic properties of the language

Overview Definition

$$\mathcal{V}(w) \quad \begin{cases} \mathcal{V}(z), \mathcal{V}(\mathrm{O}, -, -, -) & \triangleq \text{True} \\ \mathcal{V}(\mathrm{RW}/\mathrm{RWX}, b, e, -) & \triangleq \mathop{\Huge *}_{a \in [b,e)} \boxed{\exists w,\ a \mapsto w * \mathcal{V}(w)} \\ \mathcal{V}(\mathrm{E}, b, e, a) & \triangleq \triangleright \square\, \mathcal{E}(\mathrm{RX}, b, e, a) \\ \mathcal{V}(\mathrm{RO}/\mathrm{RX}, b, e, -) & \triangleq - \end{cases}$$

$$\mathcal{E}(w) \quad \triangleq \quad \forall i\ \{w; \text{any } \textit{safe context}\} \overset{i}{\rightsquigarrow} \bullet$$

FTLR

Fundamental Theorem of Logical Relation

$$\forall w, \mathcal{V}(w) \Rightarrow \mathcal{E}(w)$$