VMSL: A Separation Logic for Mechanised Robust Safety of Virtual Machines Communicating over FF-A

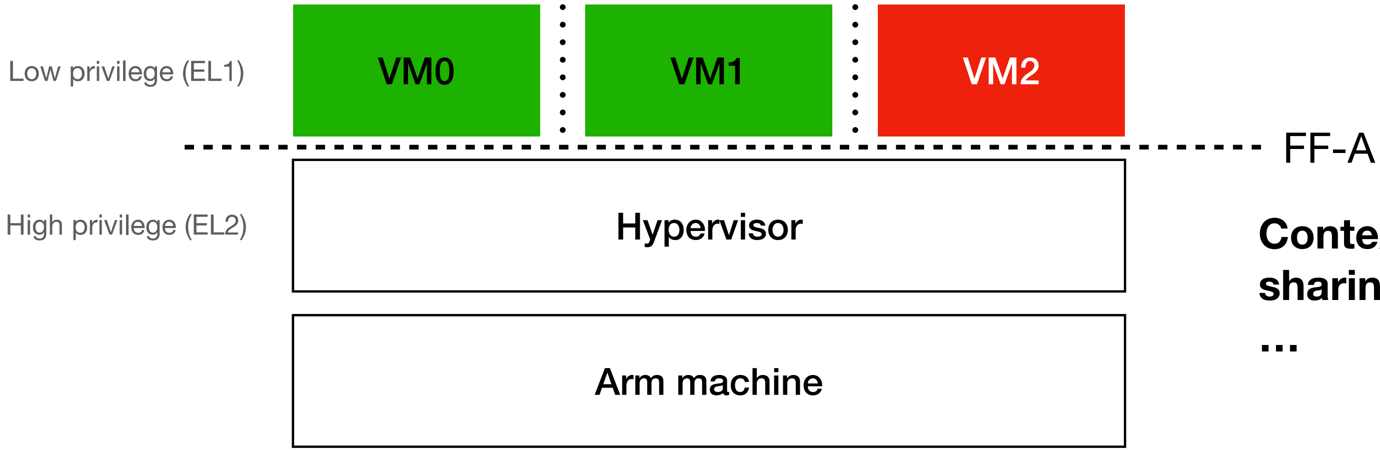
Zongyuan Liu, Sergei Stepanenko, Jean Pichon-Pharabod, Amin Timany, Aslan Askarov, Lars Birkedal

PLDI'23

AARHUS UNIVERSITY

Hypervisors and Virtual Machines

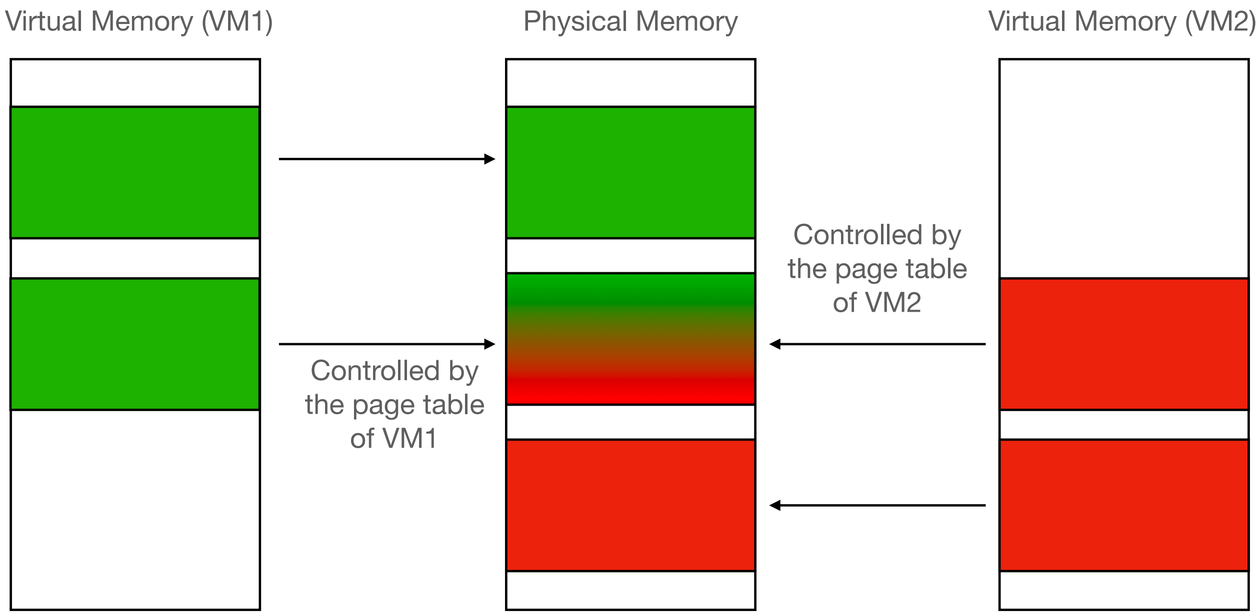
- Allows one host machine to run multiple guest VMs
- Ensures VMs run as if on bare metal, with their own CPUs, registers, memory etc.
- Provides *isolation* between VMs
- Allows controlled communication via hypercalls



Context switching, memory sharing, message passing …

Memory Management of Hypervisors

- Controlling memory access of VMs is crucial for isolation
- Access control is implemented by address translation
- Page tables of VMs are managed by the hypervisor



Verifying Communicating VMs

Separation logic nicely captures domain concepts:

Hypervisor	**Separation logic**
Communicating VMs	Cooperative threads
Permissions: access, share, …	Ownership
Sharing memory pages	Transferring ownership
Memory isolation	Separation

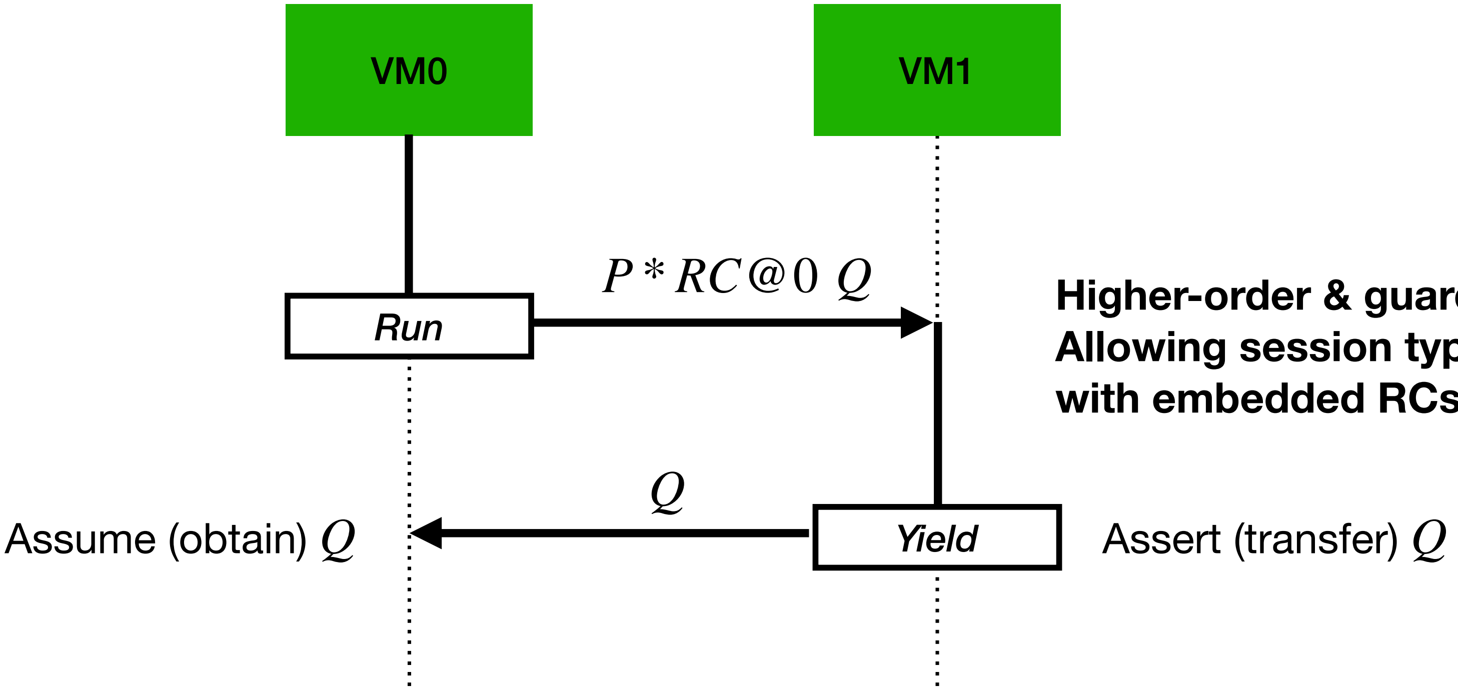
Contributions

- Formalised a substantial part of Arm's FF-A specification as an **operational semantics**
    - $\mathit{instr} ::= \texttt{mov r1} \leftarrow \texttt{r2} \mid \texttt{add r1 r2} \mid \texttt{ldr r1 [r2]} \mid \cdots \mid \texttt{hvc}$
- Developed a **separation logic** for modular reasoning about VMs with communication
    - $\{r_1 \mapsto \_ * r_2 \mapsto 42\}\ \texttt{mov r1} \leftarrow \texttt{r2}\ \{r_1 \mapsto 42 * r_2 \mapsto 42\}$
- Proved two **logical relations** to reason about combination of known and unknown VMs
- All mechanised in Coq with the **Iris** framework

VM-local Reasoning of Context Switching FF-A

```
hvc with R0 = Run, R1 = 1
```

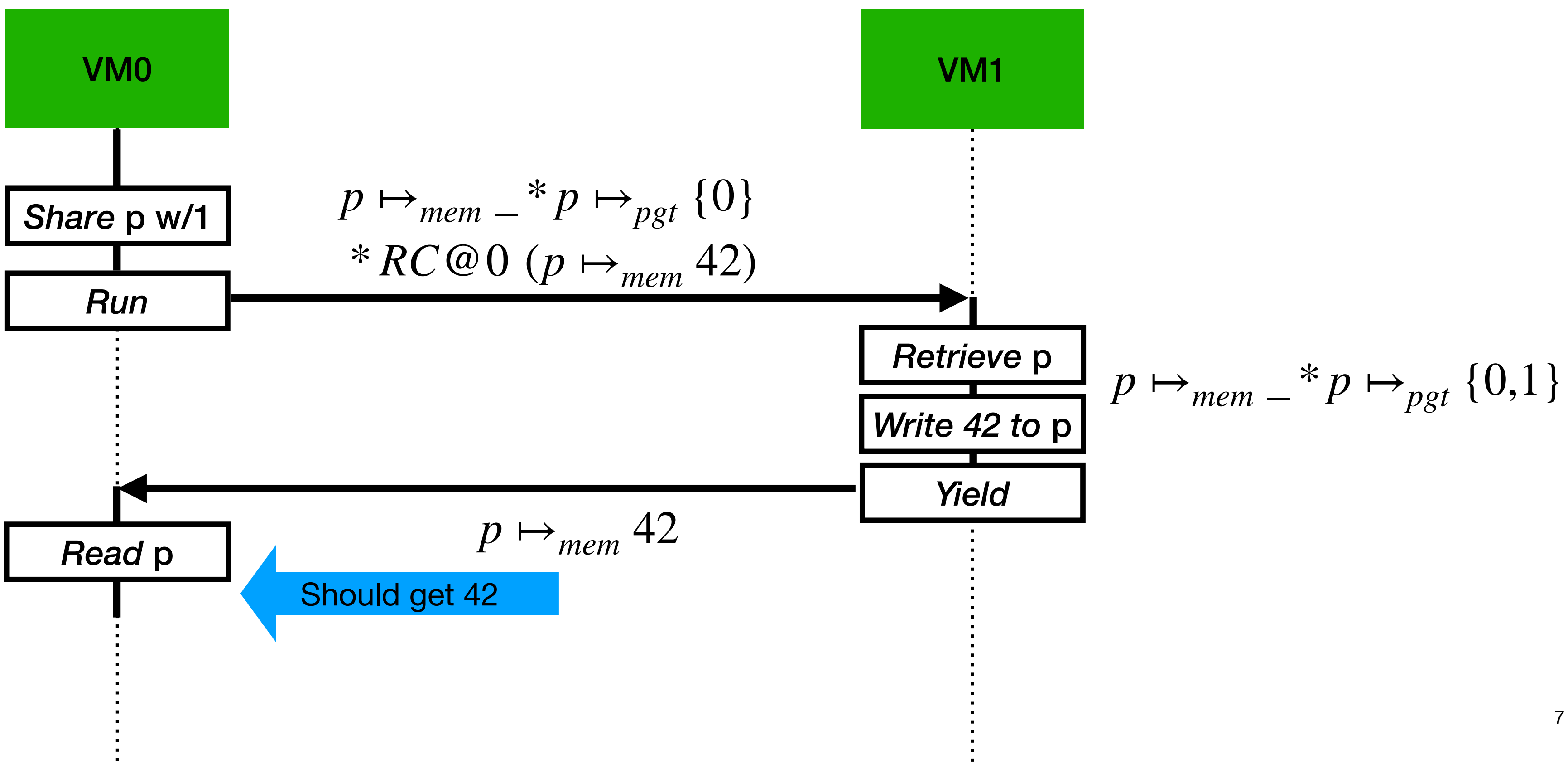
Resumption conditions for lightweight resources transfer



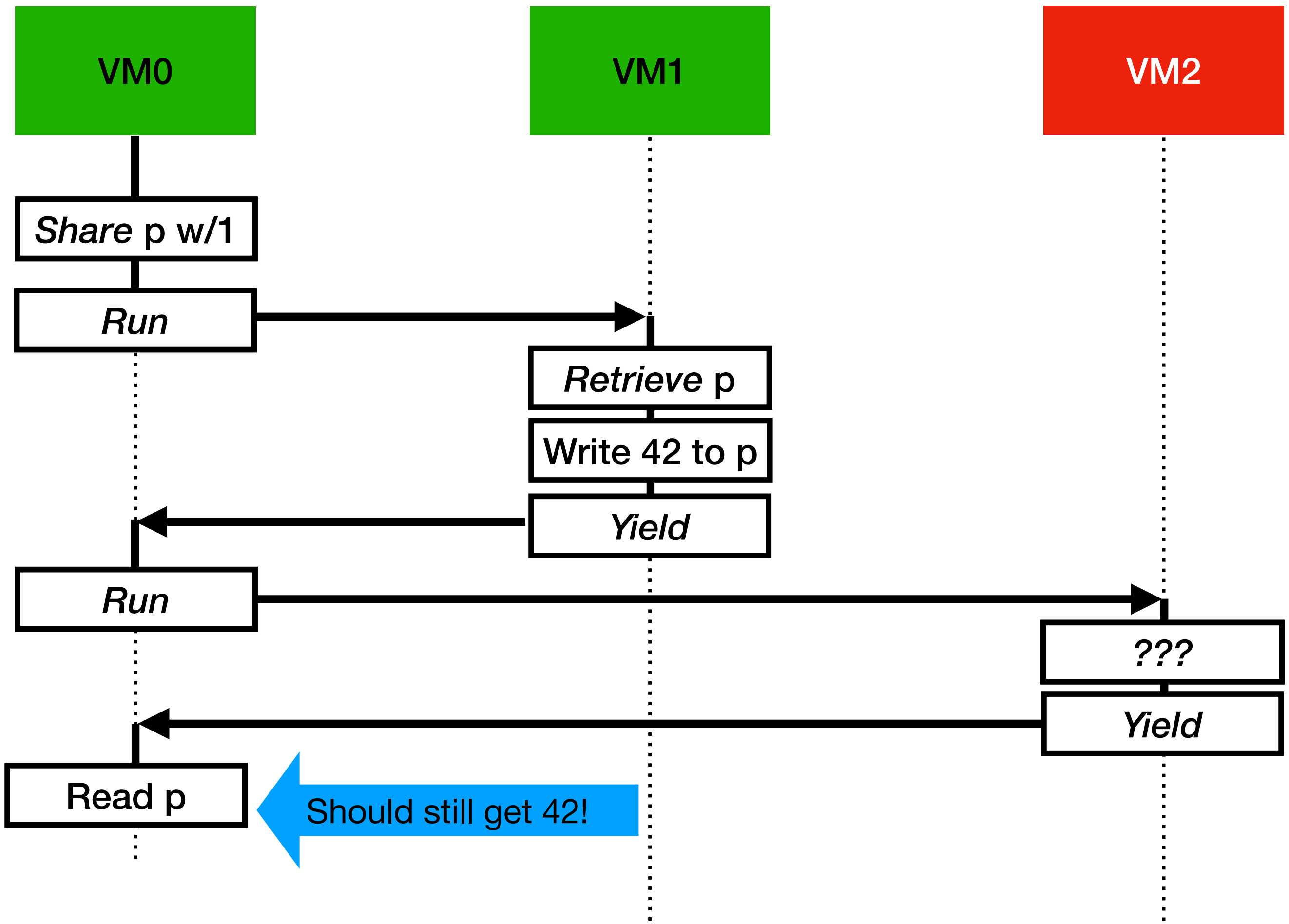
Higher-order & guarded recursion - Allowing session type like protocols with embedded RCs, e.g. ping-pong

Reasoning about Memory Sharing FF-A

Reasoning with standard points-tos



Robust Safety with Unknown VMs



VM2 doesn't have access to p

Reasoning about Unknown VMs

- Captured and proved using **logical relations**
- Intuition: a VM can only change memory it has (or can get) access to
- Shape of theorem: $\forall i, pgt, trans\,.\, \text{FootPrint}(pgt, trans) \vdash \overbrace{WP\ m@i\ \{m\,.\,\top\}}^{\text{Safe to run VM i}}$
  - Parametrised by state of the pagetable and in-flight memory sharing transactions
  - One challenge is to account for **footprint** resources required by all hypercalls
  - No assumptions on contents of memory (code is in memory)
- Two mutually compatible LRs for unknown primary and unknown secondary VMs

Conclusion

- Formalised a substantial part of FF-A specification as an **operational semantics**
  - As implemented by Google's Hafnium hypervisor
- Developed a **separation logic** for modular reasoning about VMs with communication
- Proved two **logical relations** to capture robust safety
  - Verified key scenarios of VMs using FF-A hypercalls in the presence of adversarial, unknown code
- All mechanised in Coq with the **Iris** framework