Non-interference

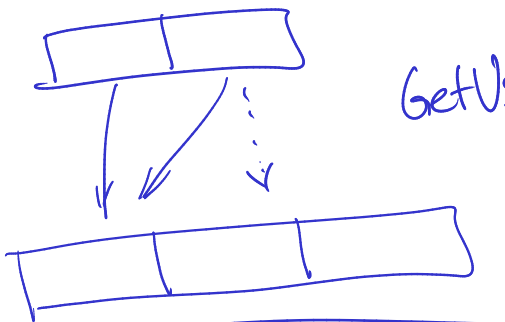
So far: func. correctness
≈ integrity.

Keeping secrets:
confidentiality.

"Open" problem.

Other APIs

Dedup:

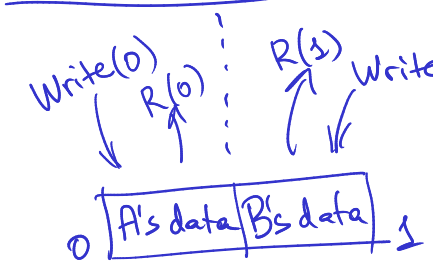


GetUsedBlocks() → 1.
→ 2.

Confid. not just about Read()

OneDisk API

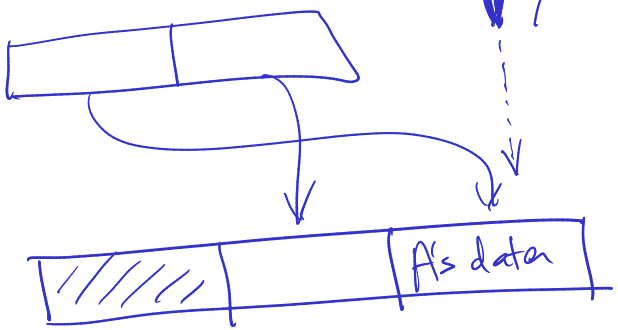
A B



Rule: B never calls R(0).
OK?

Non-determinism

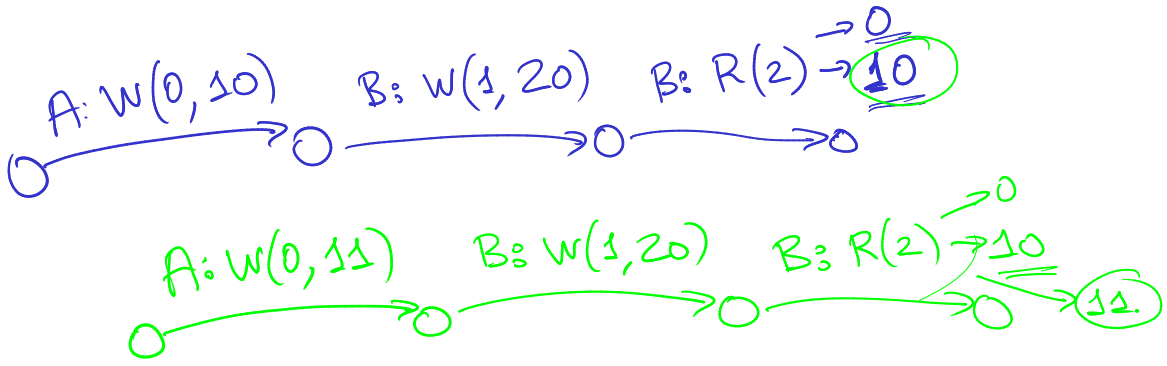
Spec is non-det.
Exec is non-det.: crashes.



☆ Confidentiality vs. non-determinism

Non-interference spec

Intuition: B's exec should not be affected by A. ←



Non-interference:
prop. about 2 exec.
"2-safety prop".

What data is confidential?

What data is observed/exposed?

Approach

Deterministic spec.
Deterministic code.
Observation func.

$O(\text{principal}, \text{state}) \rightarrow$ subset of state that princ. can observe.

Spec. obs.

$O(p, s_{spec}) \rightarrow$ p should be able to read this part of s_{spec} (is OK)

Implies confid. for $p' \neq p$.

Code obs.

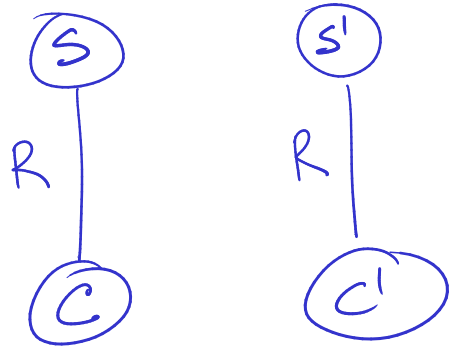
$O(p, s_{code}) \rightarrow$ externally observable state.

Output to p's terminal

Packets send to p's computer.

Spec. vs. code obs.

$O(s_{code}) \subseteq O(s_{spec})$.



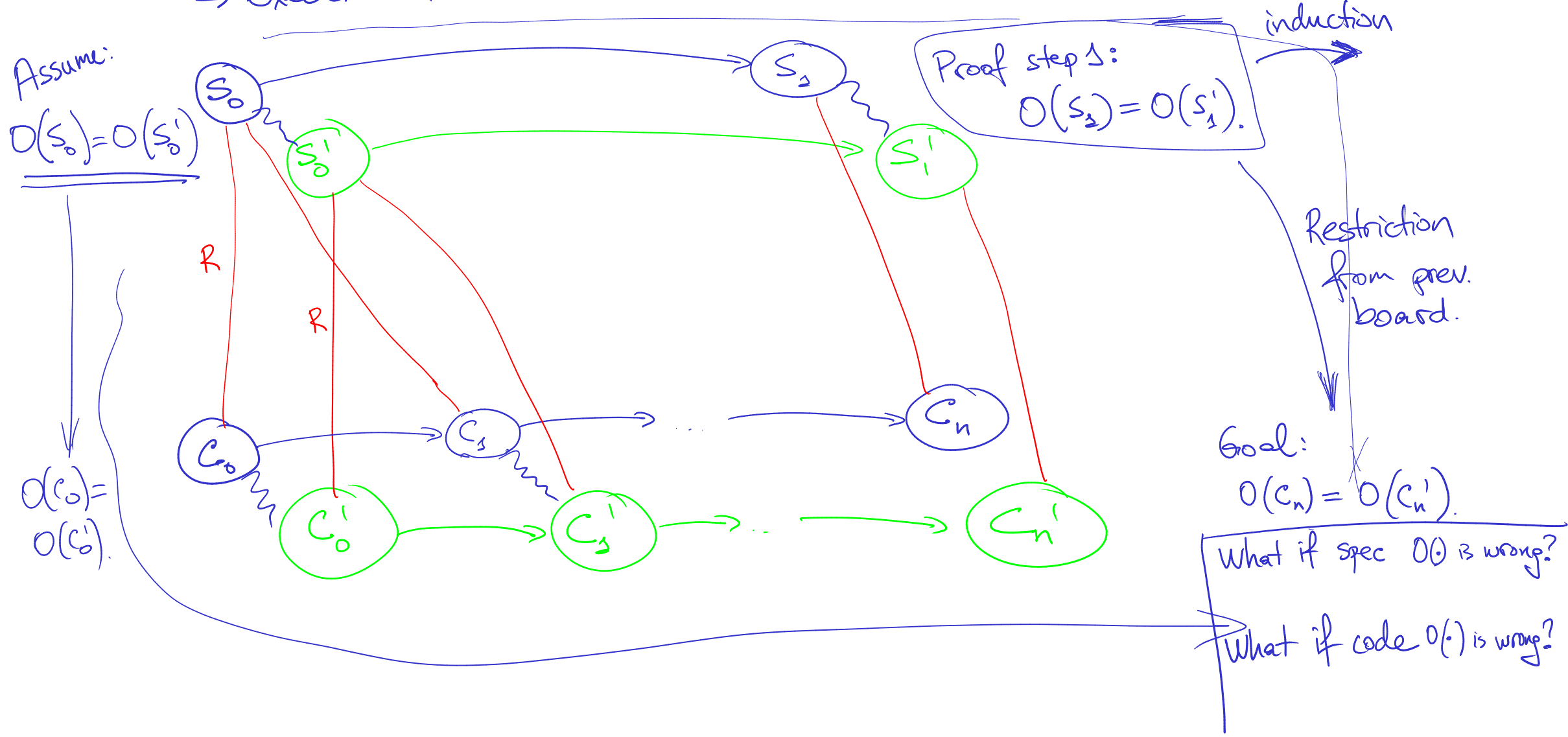
$O(s) = O(s')$

$\downarrow$

$O(c) = O(c')$.

Proof of non-interference

Goal: start out in states that have same spec. obs
→ execute * → observe same results at code level.



Proving CertiKOS

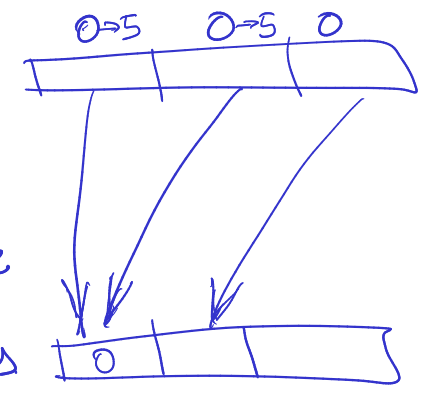
Principal = process in OS.
Output terminal per process.
Obs (p, s): virtual mem
regs. (if running)
saved regs. (if not running).
Output to terminal.

API Issue: PIDs

fork() → pid
global counter

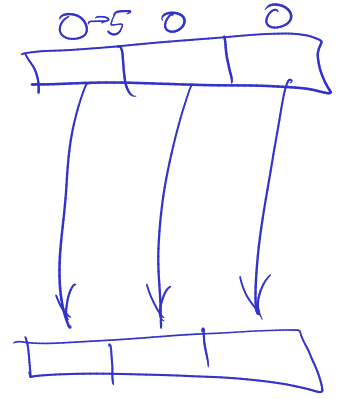
Proof Issue: VM.

spec P:



impl: page table
phys pages

write(0, 5)



Spec. Issue: local semantics