Technische Universiteit
Eindhoven
University of Technology

Modeling Information Routing with Noninterference

Ruud Koolen Julien Schmaltz

January 19, 2016

Formal verification of MILS systems

MILS systems consist of a high-assurance separation kernel, high-assurance applications, and low-assurance applications.

Verified system requirements generally rely on the correctness of both separation kernel and *some* applications.

Formal verification strategy:

- Prove separation kernel correctness
- Prove correctness of vital applications
- Infer global system properties

Formal verification of MILS systems — example

- Network manager forwards messages between domains to authorized receivers only
- Global security requirement: low-assurance domains never receive secret information

Security property relies on correctness of network manager!

Noninterference (Van der Meyden variation)

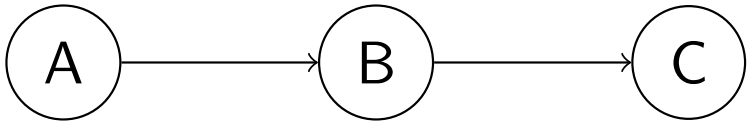
Intransitive noninterference is a formal specification of the security properties of separation kernels:

- The system contains a set of *domains*;
- Domains can *observe* certain things about the system;
- Each domain can perform *actions* that change the system state;
- Valid domain interactions are described by a *security policy* $\rightsquigarrow$.
- Security requirement:

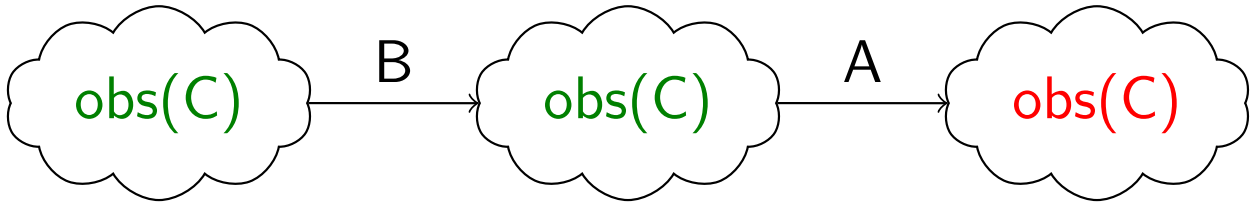
"Action a of domain d may have observable consequences for domain e only after execution of actions a_1 of d_1, ..., a_n of d_n such that $d \rightsquigarrow d_1$, $d_1 \rightsquigarrow d_2$, ..., $d_n \rightsquigarrow e$."

Noninterference — example

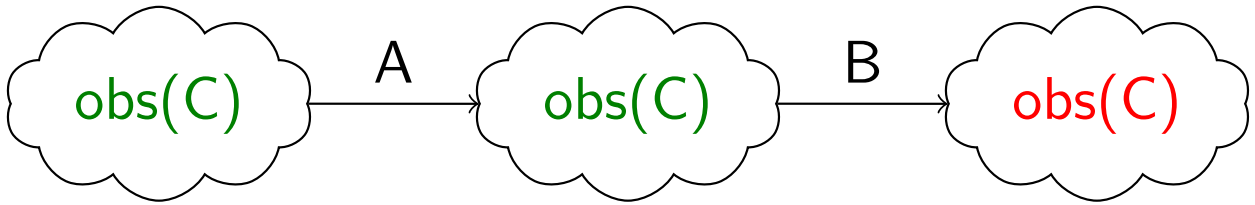
Security policy:



Not allowed — direct influence from A to C:



Allowed — influence flows via B:



System verification with noninterference

Noninterference describes the security properties of a separation kernel for *unspecified abstract observation functions* and for arbitrary actions by domains.

- How do we formalize and reason about the exact information accessible to domains?
- How do we specify that high-assurance domains behave in particular ways?

If we want to formally verify properties for systems that include information-handling domains, we need to solve both problems!

Modelling information

Noninterference assumes an abstract observation function obs(domain, state) that cannot be changed by noninterfering domains.

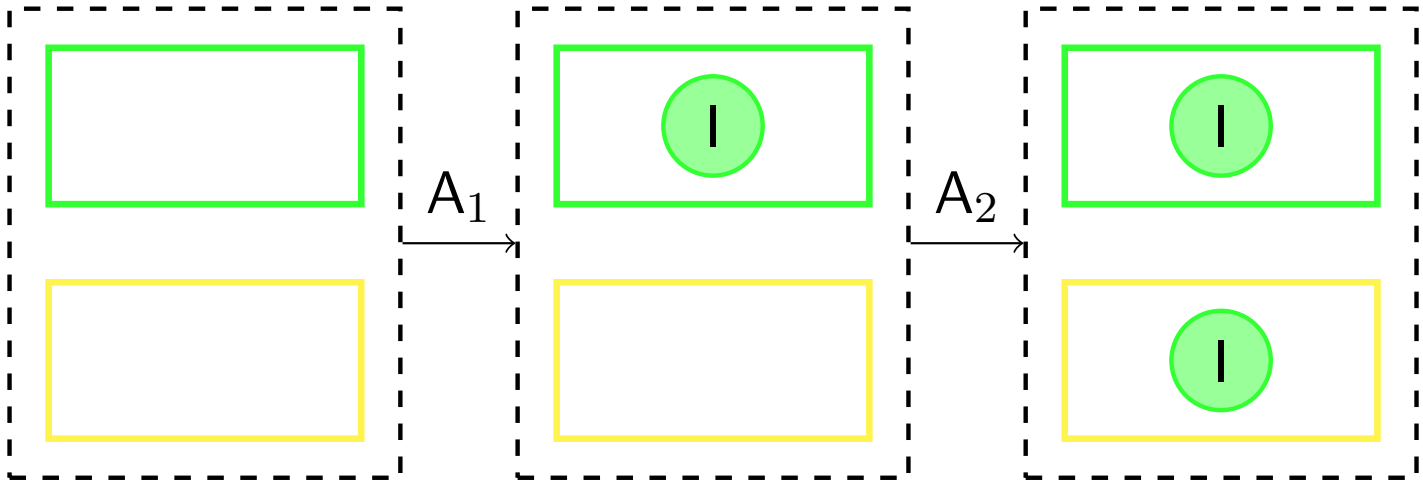
But noninterference doesn't say anything about the *contents* of this observation function.

Thus, we need to:

- encode domain-accessible information in the obs function
- formulate axioms on the behavior of this information

Informal information model

- A domain can observe *chunks of information about its own state*
- A domain can *forward observable chunks of information* to other domains



Formal information model

Observation function returns a *set of chunks of information*:

- obs(domain, state) $\in \mathcal{P}(I)$

Each chunk of information is *about* a certain domain:

- subject$(i) \in$ Domains

A domain can get access to an information chunk if the information is about the domain, or the information was forwarded by a domain with access:

- $(i \in \text{obs}(d, \text{step}(a, e, s)) \wedge i \notin \text{obs}(d, s)) \Rightarrow$
  $(\text{subject}(i) = d \vee i \in \text{obs}(e, s))$

Domain behavior

Noninterference models the security properties guaranteed by the separation kernel:

- *For all possible domain behaviors*: certain bad things do not happen

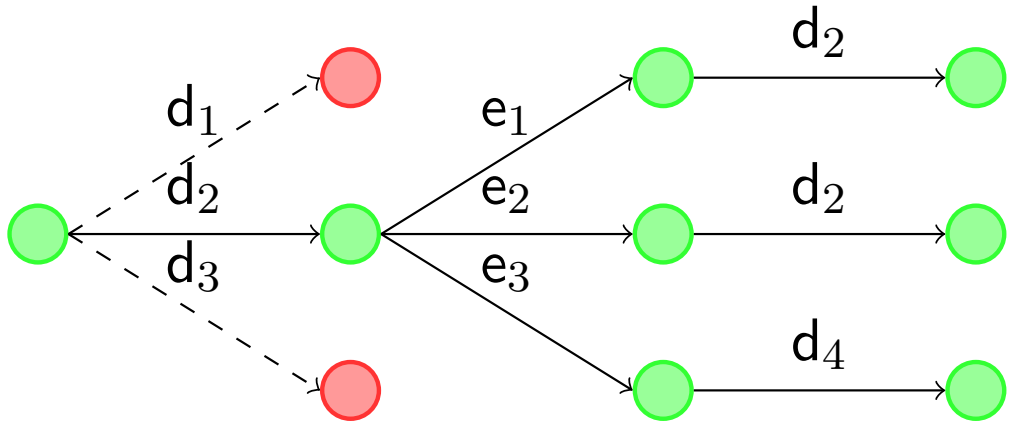
Real domains do not take arbitrary actions. They take the right action at the right time to achieve some purpose.

But noninterference does not give an obvious way to describe the behaviors of specific domains. . .

Domain program specifications

Noninterference quantifies over *all action sequences by all domains*.

A *domain specification* should specify: "domain d only takes actions that satisfy some requirement R".



Domain programs

A *program for domain* d P is a function from states to *actions of* d

- Semantics: "domain d will only perform action $P(s)$ in state s"

We can specify domain behavior using domain programs:

- "The action $P(s)$ will always satisfy some requirement $R(s)$"

System-global requirements can be verified using domain programs:

- Verify that for all action sequences α that obey program P, the global requirement holds over α.

Example formalization: network manager

Consider the network manager MILS system:

- There is a piece of information, *Secret*, known to *High*.
- The Network Manager relays information between domains, and is designed to never leak the *Secret* to either *Low* domain.
- Goal: Verify that *Secret* can never reach *Low* domains.

Network manager — formal verification

The global correctness goal should follow from the correctness of the network manager. We should be able to prove:

- Assuming that noninterference holds for the given information flow policy;
- Assuming that information in general behaves as specified by the informatiom axioms;
- Assuming that P is a network manager program that behaves as specified;
- Assuming that *Secret* is unknown to *Low* in the initial state s_0;
- We should be able to conclude: for all action sequences α that obey P, $\mathit{Secret} \notin \mathsf{obs}(\mathit{Low}, \mathsf{run}(\alpha, s_0))$

Network manager formal verification — results

The requirement on the network manager is a bit more subtle than naively expected:

- For all domains d for which an authorized information flow path exists to a *Low* domain without passing through the *NetworkManager*:
  if *Secret* $\notin$ obs(d, s); then
  Secret $\notin$ obs$(d,$ step$($*NetworkManager*$, P(s), s))$.
- Otherwise, the *Secret* can reach *Low* indirectly!

The initial state should likewise cover all such domains.

For this version: global correctness proved and verified using the Isabelle/HOL proof assistant :)

Conclusions and future work

Fully formal verification of the high-assurance parts of a MILS system is feasible. Formal specification and verification of separation kernel and high-assurance applications can be used to verify the complete system.

However, accurate specification of the desired behavior of components remains difficult. The information model of the network manager is a good start; but it is not a good end.

- Much more research can be done to improve on this!

The end

Questions?