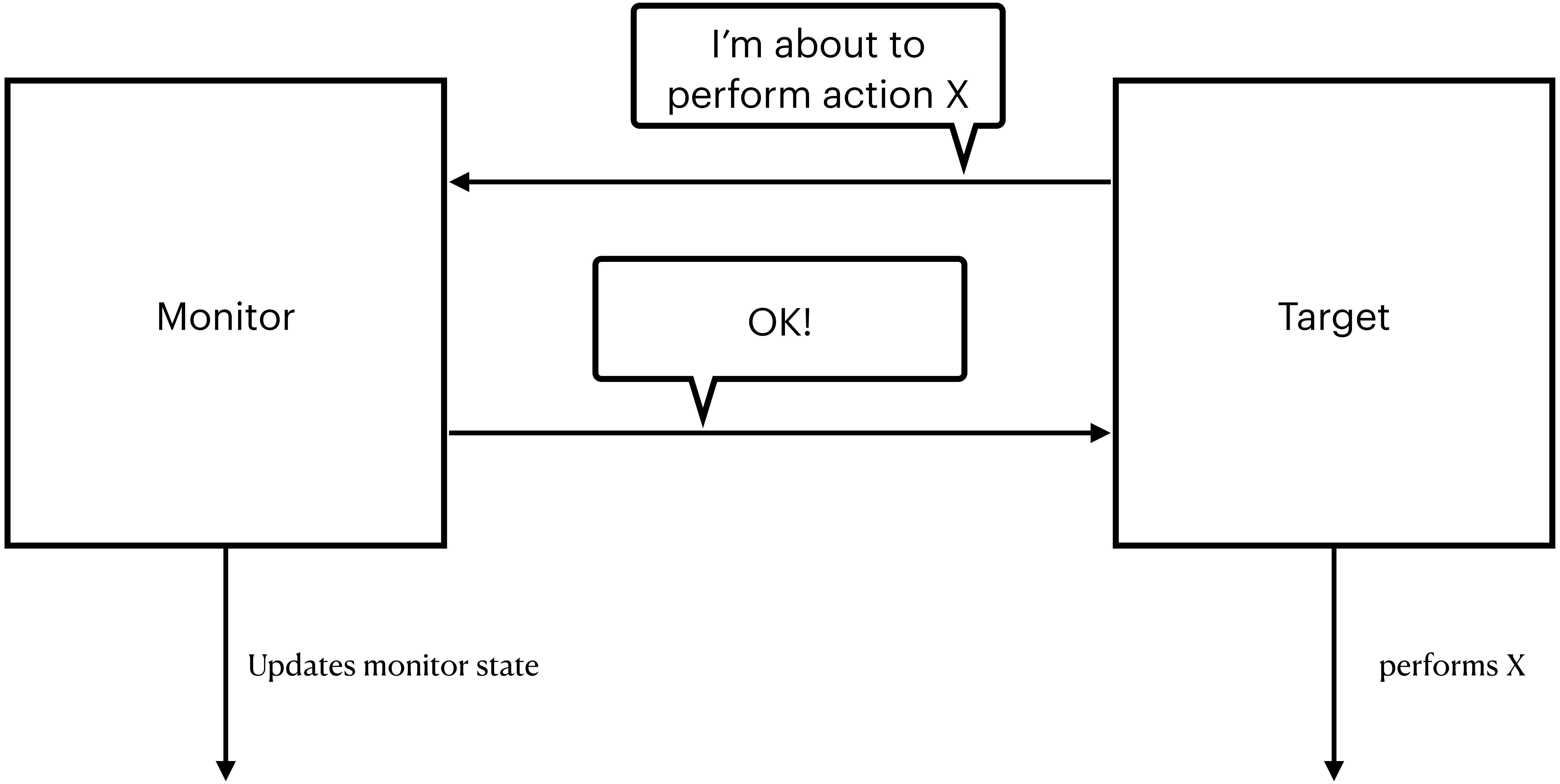
Execution Monitoring

Language-Based Security 2023
aslan@cs.au.dk

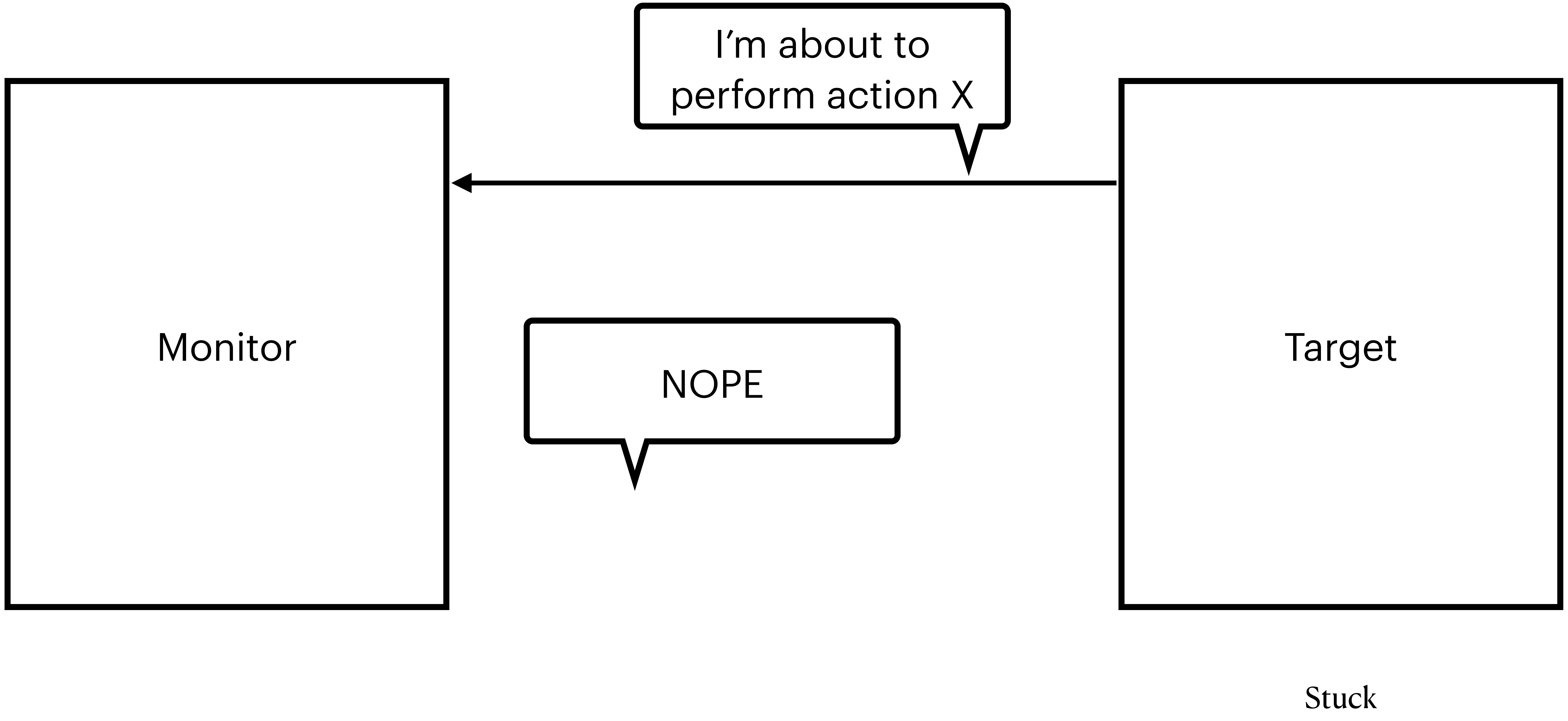
Monitoring for Security

- Monitoring for security is a very intuitive mechanism
  - Ubiquitous in applications; often to enforce a form of access control
  - Easy to deploy without deep understanding
- Today's focus
  - What can we actually enforce with monitoring?

Monitoring


I’m about to perform action X
Monitor
OK!
Target
Updates monitor state
performs X

Monitoring



Definition of Security Policy: A *security policy* is specified by giving a predicate on sets of executions. A target S *satisfies* security policy $\mathcal{P}$ if and only if $\mathcal{P}(\Sigma_S)$ equals $true$.

```
e ::= n | x | e₁ op e₂

c ::= skip
    | x := e
    | c₁; c₂
    | if e then c₁ else c₂
    | while e do c
    | input (channel)
    | out (channel)

channel ::= secret | public
```

Example setting: simple imperative language with I/O

- Standard semantics, recording I/O events
- Execution: sequence of I/O events

```
Input(Secret, 42); Output (Public, 0); …
```

Example policies:

- No public output after secret input
- Output to public channel must be copied to secret channel
- Public output does not depend on secret input

Exercise: specify the policy predicate for the above examples

What can we enforce with Execution Monitoring?

1/3

By definition, enforcement mechanisms in EM work by monitoring execution of the target. Thus, any security policy $\mathcal{P}$ that can be enforced using a mechanism from EM must be specified by a predicate of the form

$$\mathcal{P}(\Pi)\colon (\forall \sigma \in \Pi\colon \hat{\mathcal{P}}(\sigma)) \tag{1}$$

where $\hat{\mathcal{P}}$ is a predicate on (individual) executions.

Note: this eliminates some of the example policies we discussed; which ones?

What can we enforce with Execution Monitoring?

2/3

Monitor cannot foresee the future – places additional constraint on the policies

finite prefixes

$$(\forall \tau' \in \Psi^{-}\colon \neg \hat{\mathscr{P}}(\tau') \Rightarrow (\forall \sigma \in \Psi\colon \neg \hat{\mathscr{P}}(\tau' \sigma))) \tag{2}$$

If security policy $\hat{\mathscr{P}}$ considers prefix τ as insecure, then $\hat{\mathscr{P}}$ must deem all extensions of τ also insecure

Note: this eliminates some other example policies we discussed; which ones?

What can we enforce with Execution Monitoring?

3/3

Execution rejected by an enforcement mechanism must be rejected after a finite period

prefix of σ involving i steps

$$(\forall \sigma \in \Psi\colon \neg \hat{\mathscr{P}}(\sigma) \Rightarrow (\exists i\colon \neg \hat{\mathscr{P}}(\sigma[..i]))) \tag{3}$$

Properties satisfying (1), (2), and (3) are safety properties

What does that mean?

Safety property ~ no "bad things" happen during any execution [Lamport 1977]

If security policy $\mathscr{P}$ is not a safety policy, it is *not* enforceable by an execution monitor

Contra-positive:

Execution monitors enforce security policies that are safety properties

But not all safety properties are monitorable (limited monitor memory)

Consequences:

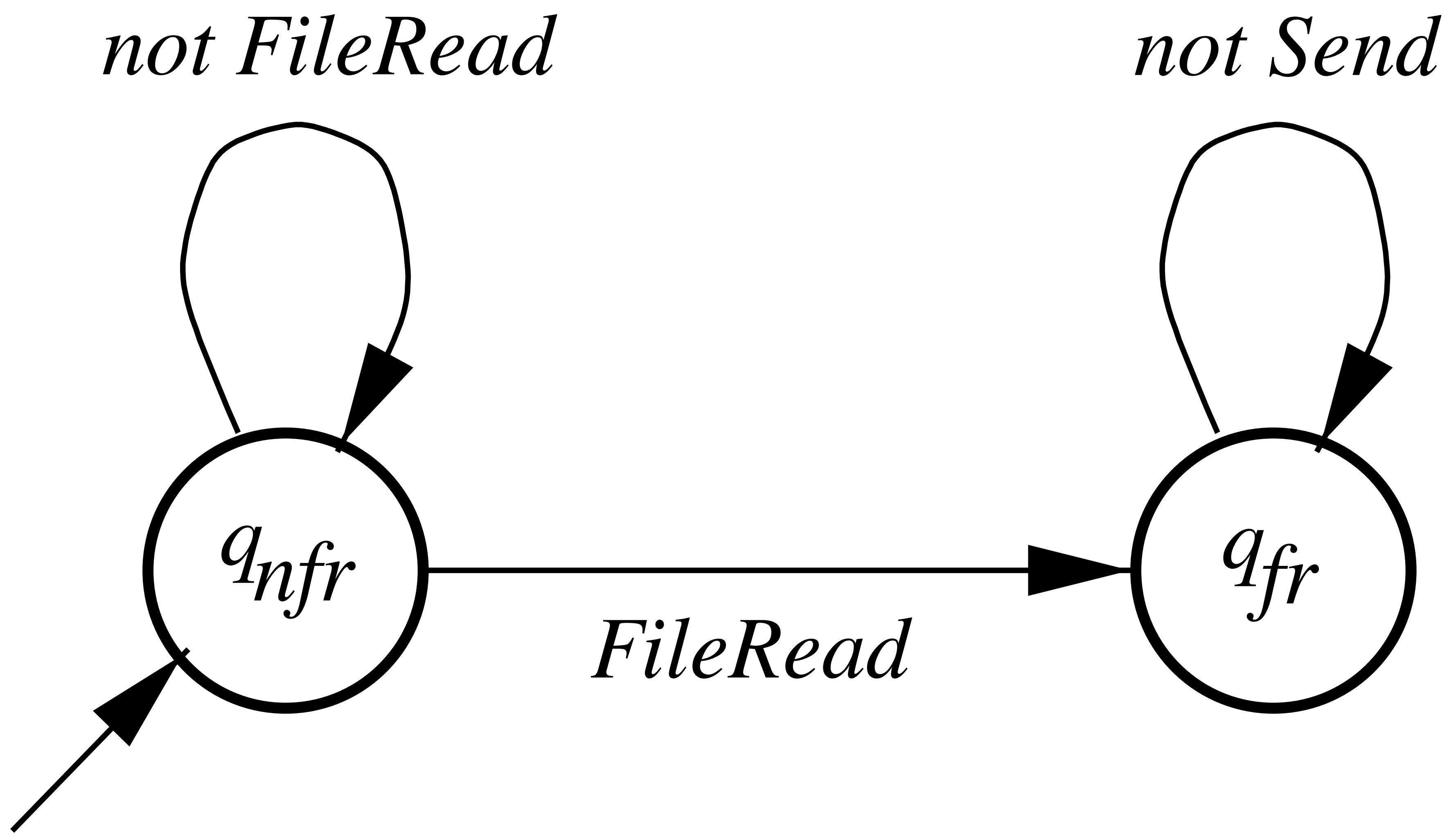
1) We can enforce $\mathscr{P}$ by enforcing a stronger policy $\mathscr{P}'$ such that $\mathscr{P}' \implies \mathscr{P}$

2) Monitors are composable

Security Automata

Monitoring can be implemented as a security automata ~ an NFA-like automata

Expressive enough for many access control policies



No *Send* after *FileRead*

Implementing monitoring?

1) Monitor as part of the runtime

2) Inlining monitoring

    - Rewriting code to encode the state of the monitor

What can monitors do?

- Schneider's definition: only fail-stop monitoring
- Extensions
  - Edit automata [Ligatti, 2005]: suppress/insert additional actions
  - [Basin et al, 2013] Distinction between observable and controllable events

- These decisions are relevant when *sandboxing*

Summary

- Security policies as predicates on sets of executions (very general definition)
- Monitoring can only enforce safety properties