Security and Trust I:
3. Channel Security

Dusko Pavlovic

UHM ICS 355
Fall 2014

Outline

What is a channel?

State machines and processes

Sharing

Noninterference

What did we learn?

Outline

What is a channel?

Notation

Definition

Examples

State machines and processes

Sharing

Noninterference

What did we learn?

Lists

Definition

$$\langle listsofX \rangle \quad ::= \quad () \mid x :: \langle listsofX \rangle$$

Lists

Datatype of lists

For any set X, the set of *lists*

$$X^* = \left\{(x_1\ x_2 \cdots x_n) \in X^n \mid n = 0, 1, 2, \ldots\right\}$$

is generated by

$$\begin{array}{rcl} 1 & \xrightarrow{()} & X^* \\ X \times X^* & \xrightarrow{::} & X^* \end{array}$$

$$\left\langle x_0, (y_1\ y_2 \cdots y_n)\right\rangle \mapsto \left(x_0\ y_1\ y_2 \cdots y_n\right)$$

Lists

Notation

We write lists as vectors[1]

$$\vec{x} \quad = \quad (x_1 \; x_2 \cdots x_n)$$

[1] Functional programmers write $xs \; = \; (x1 \; x2 \cdots xn)$

Lists

Concatenation

The derived structure can be defined inductively, e.g.

$$\begin{array}{rcl} X^* \times X^* & \stackrel{@}{\longrightarrow} & X^* \stackrel{()}{\longleftarrow} 1 \\ \left\langle (), \vec{x} \right\rangle & \longmapsto & \vec{x} \\ \left\langle x :: \vec{y}, \vec{z} \right\rangle & \longmapsto & x :: \left(\vec{y} @ \vec{z} \right) \end{array}$$

Lists

Prefix ordering

$$\vec{x} \sqsubseteq \vec{y} \quad \iff \quad \exists \vec{z}.\ \vec{x} @ \vec{z} = \vec{y}$$

Lists

Prefix ordering

$$\vec{x} \sqsubseteq \vec{y} \iff \exists \vec{z}.\ \vec{x} @ \vec{z} = \vec{y}$$

i.e.

$$(x_1\ x_2 \cdots x_k\ \cdots\cdots\cdots)$$
$$=$$
$$(y_1\ y_2 \cdots y_k\ y_{k+1} \cdots y_n)$$

Lists

Notation: Prepending as concatenation

Since

$$(x)@\vec{y} \quad = \quad x::\vec{y}$$

we usually identify the symbols $x \in X$ with the one-element lists $(x) \in X^*$, elide (x) to x, and write

$$x@\vec{y} \text{ instead of } x::\vec{y}$$

Strings

Strings are nonempty lists

For any set X, the set of *lists*

$$X^+ \quad = \quad \left\{(x_1 x_2 \cdots x_n) \in X^n \mid n = 1, 2, \ldots\right\}$$

is generated by

$$\begin{array}{ccc} X & \xrightarrow{(-)} & X^* \\ X \times X^* & \xrightarrow{::} & X^* \end{array}$$

$$\left\langle x_0, (y_1 y_2 \cdots y_n) \right\rangle \quad \mapsto \quad \left(x_0 y_1 y_2 \cdots y_n\right)$$

Partial functions

Notation

A partial function from A to B is written $A \rightharpoonup B$.

Domain of definition

For any partial function $A \stackrel{f}{\rightharpoonup} B$ we define

$$
\begin{aligned}
f(a)\!\downarrow \quad & \iff \quad \exists b.\ f(a) = b \\
\downarrow\! f \quad & = \quad \{a \mid f(a)\!\downarrow\}
\end{aligned}
$$

What is a channel?

Definition

A *deterministic channel* with

- the *inputs* (or *actions*) from A
- the *outputs* (or *observations*) from B

is a partial function

$$f \quad : \quad A^+ \rightharpoonup B$$

whose domain is prefix closed, i.e.

$$f(\vec{x}@a)\downarrow \implies f(\vec{x})\downarrow$$

holds for all $\vec{x} \in A^+$ and $a \in A$

What is a channel flow?

Definition

A *flow* with

- the *inputs* (or *actions*) from A
- the *outputs* (or *observations*) from B

is a partial function

$$\vec{f} \quad : \quad A^* \rightharpoonup B^*$$

which is prefix closed and monotone:

$$\vec{f}(\vec{x}@a)\downarrow \implies \vec{f}(\vec{x})\downarrow \qquad\qquad \vec{x} \sqsubseteq \vec{y} \implies \vec{f}(\vec{x}) \sqsubseteq \vec{f}(\vec{y})$$

Channels and flows are equivalent

Proposition

Every deterministic channel induces a unique flow.

Every flow arises from a unique deterministic channel.

Channels and flows are equivalent

Proof of $f \rightsquigarrow \vec{f}$

$$\vec{f}(x_1\ x_2 \cdots x_n) \quad = \quad \left(f(x_1)\ \ f(x_1 x_2) \cdots f(x_1 x_2 \cdots x_n)\right)$$

Channels and flows are equivalent

Proof of $f \rightsquigarrow \vec{f}$

$$\vec{f}(x_1\ x_2 \cdots x_n) \quad = \quad \Big(f(x_1)\ \ f(x_1 x_2) \cdots f(x_1 x_2 \cdots x_n)\Big)$$

Proof of $\vec{f} \rightsquigarrow f$

$$f(x_1\ x_2 \cdots x_n) \quad = \quad \vec{f}\Big(x_1\ x_2 \cdots x_n\Big)_n$$

where $\vec{a}_n$ denotes the n-th component of the string $\vec{a}$

What do flows and channels represent?

- Any resource use, or process in general
  - takes some inputs
  - gives some outputs .
- If we hide the internal details, we only see
  - which inputs induce
  - which outputs .

What do flows and channels represent?

- Any resource use, or process in general
  - takes some actions
  - gives some reactions, or results.
- If we hide the internal details, we only see
  - which actions induce
  - which reactions.

What do they have to do with security?

- A shared resource induces a *shared channel*.
  - Each user extracts a different *flow*
- The problems of resource security can be modeled as
  - *interferences* of the individual flows
  - in a shared channel.

Memory

- A process *with no memory* is a *function* $A \xrightarrow{f} B$.
    - It is *partial* when some inputs yield no outputs.

Memory

- A process *with no memory* is a *function* $A \stackrel{f}{\to} B$.
  - It is *partial* when some inputs yield no outputs.
- A process *with memory* is a channel $A^+ \stackrel{f}{\to} B$.
  - The outputs depend on all past inputs

More general channels

A channel can display the observable behaviors of several types of processes, such as

deterministic: partial function $A^+ \rightharpoonup B$

possibilistic: relation $A^+ \to \wp B$

probabilistic: stochastic matrix $A^+ \to \Delta B$

Examples

Channels in computation

- Any computation takes inputs and gives outputs.
  - The simplest applications are *memoryless*: they induce functions $A \to B$.
  - Some applications' outputs depend on may previous inputs: they induce proper channels $A^+ \to B$.

Example 1

Binary successor channel

$$\begin{array}{ccc} \{0,1\}^+ & \xrightarrow{+} & \{0,1\} \\ (a_1\ a_2 \cdots a_{n-1}) & \longmapsto & b_n \end{array}$$

so that

$$(b_n\ b_{n-1} \cdots b_1) \quad = \quad (a_{n-1}\ a_{n-2} \cdots a_1) + 1$$

Example 2

Binary addition channel

$$\begin{array}{ccc} \{00, 01, 10, 11\}^+ & \stackrel{+}{\longrightarrow} & \{0, 1\} \\ (a_1\ a_2 \cdots a_{n-1}) & \longmapsto & b_n \end{array}$$

so that

$$(b_n\ b_{n-1} \cdots b_1) \quad = \quad \left(a^0_{n-1}\ a^0_{n-2} \cdots a^0_1\right) + \left(a^1_{n-1}\ a^1_{n-2} \cdots a^1_1\right)$$

where $a_i = a^0_i a^1_i$.

Example 3

Remainder mod 3

$$\{0,1\}^+ \xrightarrow{\text{mod } 3} \{0,1,2\}$$
$$\vec{a} \longmapsto b$$

so that

$$b = \vec{a} \mod 3$$

Other examples of channels

Communication channels

- Radio channel
  - the inputs at transmitter are the outputs at receiver
- Social channel
  - this lecture, exam, conversation . . .
- Phone channel
  - both radio and social. . .

Other examples of channels

Traffic channels

- Shipping channel between two rivers
- Road between two cities
- Street in a town

Other examples of channels

Traffic channels

- Shipping channel between two rivers
- Road between two cities
- Street in a town

input:	vehicles enter on one end
output:	vehicles exit at the other end

Other examples of channels

Traffic channels

- Shipping channel between two rivers
- Road between two cities
- Street in a town

input:	vehicles enter on one end
output:	vehicles exit at the other end

channel with memory: How each of them comes out depends on all of them.

Examples of channels

Network channels

- network nodes: local actions
  - programmable computation
- network channels: nonlocal interactions
  - non-programmable communication

Examples of channels

Strategies

- $A =$ the moves available to the Opponent
- $B =$ the moves available to the Player
- $A^+ \xrightarrow{f} B$ tells how the Player should respond to the Opponent's strategies

Problem

- The listing of A^+ is always infinite.
- How do you specify $A^+ \xrightarrow{f} B$?

Question

- Is there a "programming language" allowing finite descriptions of infinite channels?
  - (like in Examples 1–3)

Answer

$$\frac{\textbf{machines}}{\text{channels}} = \frac{\text{programs}}{\text{computations}}$$

Outline

State machines: Mealy

Definition

A *Mealy machine* is a partial function

$$Q \times I \xrightarrow{\theta} Q \times O$$

where Q, I, O are finite sets, representing

- Q — states
- I — inputs
- O — outputs

- $Q \times I \xrightarrow{\theta_0} Q$ — next state
- $Q \times I \xrightarrow{\theta_1} O$ — observation
- $\theta_0(q, i)\downarrow \iff \theta_1(q, i)\downarrow$

State machines: Moore

Definition

A *Moore machine* is a pair of maps

$$Q \times I \xrightarrow{\theta_0} Q \xrightarrow{\theta_1} O$$

where Q, I, O are finite sets, representing

- Q — states
- I — inputs
- O — outputs

- $Q \times I \xrightarrow{\theta_0} Q$ — next state
- $Q \xrightarrow{\theta_1} O$ — observation

State machines

Notation

When no confusion is likely, the state machine is denoted by the name of its state set Q.

Processes

Definition

A *process* is a pair $\langle Q, q_0 \rangle$ where

- Q is a machine
- $q_0 \in Q$ is a chosen *initial state*.

Processes

Definition

A *process* is a pair $\langle Q, q_0 \rangle$ where

- Q is a machine
- $q_0 \in Q$ is a chosen *initial state*.

Notation

Proceeding with the abuse of notation, even a process is often called by the name of its state space, conventionally denoting the initial state by q_0, or sometimes ι.

Example 1

Binary successor channel: Implement it!

$$\begin{array}{ccc} \{0,1\}^{+} & \xrightarrow{+} & \{0,1\} \\ (a_1\ a_2 \cdots a_{n-1}) & \longmapsto & b_n \end{array}$$

so that

$$(b_n\ b_{n-1} \cdots b_1) \quad = \quad (a_{n-1}\ a_{n-2} \cdots a_1) + 1$$

Example 1: Mealy

Binary successor process

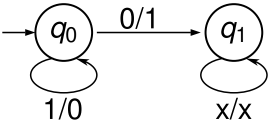
- $Q = \{q_0, q_1\}$
- $I = O = \{0, 1\}$
- θ :

	0	1
q_0	$\langle 1, q_1 \rangle$	$\langle 0, q_0 \rangle$
q_1	$\langle 0, q_1 \rangle$	$\langle 1, q_1 \rangle$

Example 1: Mealy

Binary successor process

- $Q = \{q_0, q_1\}$
- $I = O = \{0, 1\}$
- θ :



Example 2

Binary addition channel: Implement it!

$$\begin{array}{ccc} \{00,01,10,11\}^+ & \xrightarrow{+} & \{0,1\} \\ (a_1\ a_2 \cdots a_{n-1}) & \longmapsto & b_n \end{array}$$

so that

$$(b_n\ b_{n-1} \cdots b_1) \quad = \quad \left(a^0_{n-1}\ a^0_{n-2} \cdots a^0_1\right) + \left(a^1_{n-1}\ a^1_{n-2} \cdots a^1_1\right)$$

where $a_i = a^0_i a^1_i$.

Example 2: Mealy

Binary addition process

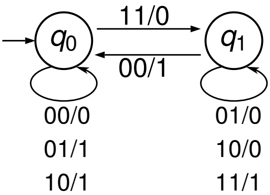
- $Q = \{q_0, q_1\}$
- $I = \{00, 01, 10, 11\}$
- $O = \{0, 1\}$
- θ :

	00	01	10	11
q_0	$\langle 0, q_0 \rangle$	$\langle 1, q_0 \rangle$	$\langle 1, q_0 \rangle$	$\langle 0, q_1 \rangle$
q_1	$\langle 1, q_0 \rangle$	$\langle 0, q_1 \rangle$	$\langle 0, q_1 \rangle$	$\langle 1, q_1 \rangle$

Example 2: Mealy

Binary addition process

- $Q = \{q_0, q_1\}$
- $I = \{00, 01, 10, 11\}$
- $O = \{0, 1\}$
- θ :



Example 3

Remainder mod 3 channel: Implement it!

$$\begin{array}{ccc} \{0,1\}^+ & \xrightarrow{\text{mod } 3} & \{0,1,2\} \\ \vec{a} & \longmapsto & b \end{array}$$

so that

$$b = \vec{a} \mod 3$$

Example 3: Task

Remainder mod 3 channel

$$\begin{array}{ccc} \{0,1\}^+ & \xrightarrow{\text{mod } 3} & \{0,1,2\} \\ (a_1\ a_2 \cdots a_n) & \longmapsto & b \end{array}$$

so that

$$b \quad = \quad (a_1\ a_2 \cdots a_n) \quad \text{mod } 3$$

Example 3: Idea

Remainder mod 3 process

$a = p \mod 3$	$a0 = 2p \mod 3$	$a1 = 2p + 1 \mod 3$
0	0	1
1	2	0
2	1	2

Example 3: Moore

Remainder mod 3 process

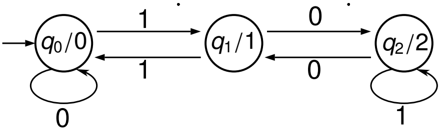
- $Q = \{q_0, q_1, q_2\}$
- $I = \{0, 1\}$
- $O = \{0, 1, 2\}$
- θ :

	0	1
$q_0/0$	q_0	q_1
$q_1/1$	q_2	q_0
$q_2/2$	q_1	q_2

Example 3: Moore

Remainder mod 3 process

- $Q = \{q_0, q_1, q_2\}$
- $I = \{0, 1\}$
- $O = \{0, 1, 2\}$
- θ :



Example 3: Mealy

Remainder mod 3 process

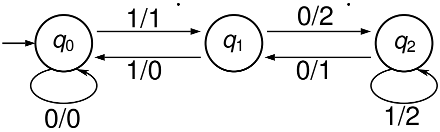
- $Q = \{q_0, q_1, q_2\}$
- $I = \{0, 1\}$
- $O = \{0, 1, 2\}$
- θ :

	0	1
q_0	$\langle q_0, 0\rangle$	$\langle q_1, 1\rangle$
q_1	$\langle q_2, 2\rangle$	$\langle q_0, 0\rangle$
q_2	$\langle q_1, 1\rangle$	$\langle q_2, 2\rangle$

Example 3: Mealy

Remainder mod 3 process

- $Q = \{q_0, q_1, q_2\}$
- $I = \{0, 1\}$
- $O = \{0, 1, 2\}$
- θ :



Mealy vs Moore

- Both Mealy and Moore machines present the state updates dependent on the inputs:

$$Q \times I \xrightarrow{\theta_0} Q$$

- Mealy machines moreover present the outputs dependent on the inputs:

$$Q \times I \xrightarrow{\theta_1} O$$

- Moore machines only present the observations of the states:

$$Q \xrightarrow{\theta_1} O$$

Mealy vs Moore

It turns out that they capture the same family of processes.

Running Mealy machines

$$\frac{Q \times I \xrightarrow{\theta_0} Q \qquad\qquad Q \times I \xrightarrow{\theta_1} O}{Q \times I^+ \quad \xrightarrow{\Theta} \quad O}$$

Running Mealy machines

$$\frac{Q \times I \xrightarrow{\theta_0} Q \qquad\qquad Q \times I \xrightarrow{\theta_1} O}{\begin{array}{rcl} Q \times I^+ & \xrightarrow{\Theta} & O \\ \langle q, x\rangle & \longmapsto & \theta_1(q,x) \\ \left\langle q,\ x @ \vec{y}\right\rangle & \longmapsto & \Theta\left(\theta_0(q,x),\ \vec{y}\right) \end{array}}$$

Running Moore machines

$$\frac{Q \times I \xrightarrow{\theta_0} Q \qquad\qquad Q \xrightarrow{\theta_1} O}{\begin{array}{rcl} Q \times I^+ & \xrightarrow{\Theta} & O \\ \langle q, x \rangle & \longmapsto & \theta_1\left(\theta_0(q,x)\right) \\ \left\langle q,\ x @ \vec{y} \right\rangle & \longmapsto & \Theta\left(\theta_0(q,x),\ \vec{y}\right) \end{array}}$$

Recall

Definition

A *process* is a state machine with a chosen initial state.

Induced channels

Running a Mealy process yields a channel

$$\frac{q \in Q \qquad\qquad Q \times I \xrightarrow{\theta_0} Q \qquad\qquad Q \times I \xrightarrow{\theta_1} O}{\begin{array}{rcl} I^+ & \xrightarrow{\Theta^q} & O \\ x & \longmapsto & \theta_1(q,x) \\ x@\vec{y} & \longmapsto & \Theta^{\theta_0(q,x)}\left(\vec{y}\right) \end{array}}$$

Induced channels

Running a Moore process yields a channel

$$\frac{q \in Q \qquad\qquad Q \times I \xrightarrow{\theta} Q \times O}{\begin{array}{rcl} I^+ & \xrightarrow{\Theta^q} & O \\ x & \longmapsto & \theta_1\left(\theta_0(q,x)\right) \\ x@\vec{y} & \longmapsto & \Theta^{\theta_0(q,x)}\left(\vec{y}\right) \end{array}}$$

...but the other way around,

every channel is a machine

$$\dfrac{\dfrac{q \in Q \qquad\qquad Q \times I \xrightarrow{\theta} Q \times O}{I^+ \xrightarrow{\Theta^q} O}}{\begin{array}{ccc} I^* \times I & \xrightarrow{\Theta^q_*} & I^* \times O \\ \langle \vec{x}, y \rangle & \longmapsto & \langle \vec{x}@y, \Theta(\vec{x}@y) \rangle \end{array}}$$

. . . but the other way around,

every channel is a machine tracing the original process

$$\begin{array}{ccc} I^* \times I & \xrightarrow{\Theta^q_*} & I^* \times O \\ \Big\downarrow{\scriptstyle \theta_0^* \times I} & & \Big\downarrow{\scriptstyle \theta_0^* \times O} \\ Q \times I & \xrightarrow{\theta^Q} & Q \times O \end{array}$$

where

$$\begin{aligned} \theta_0^*() &= q_0 \\ \theta_0^*(\vec{x}@y) &= \theta_0(\theta_0^*(\vec{x}), y) \end{aligned}$$

Observable behaviors

Definition

The *(observable) behavior* of a process is the channel that it induces.

Observable behaviors

Definition

The *(observable) behavior* of a process is the channel that it induces.

Two processes are *(observationally) indistinguishable* if they induce the same observable behaviors.

The Universal Machine

Notation: Space of channels

$$[I, O] \;=\; \left\{ I^+ \xrightarrow{f} O \mid \forall \vec{x} \forall a.\;\; f(\vec{x}@a)\downarrow \Rightarrow f(\vec{x})\downarrow \right\}$$

$$\cong\; \left\{ I^* \xrightarrow{\vec{f}} O^* \mid \forall \vec{x} \forall a.\;\; \vec{f}(\vec{x}@a)\downarrow \Rightarrow \vec{f}(\vec{x})\downarrow \right.$$

$$\left. \wedge\;\; \vec{x} \sqsubseteq \vec{y} \Rightarrow \vec{f}(\vec{x}) \sqsubseteq \vec{f}(\vec{y}) \right\}$$

The Universal Machine

Idea

The *behavior mapping*

$$Q \xrightarrow{\Theta} [I, O]$$

induces the *universal representation* of

- any Mealy Machine over the state space Q
- the canonical Mealy Machine over the state space $[I, O]$

The Universal Machine

Definition

The Universal Mealy machine over the inputs I and outputs O has

- the state space $[I, O]$ consisting of the channels $I^+ \rightharpoonup O$
- the structure map $[I, O] \times I \xrightarrow{\theta} [I, O] \times O$ where

$$\theta_0(f, x)(\vec{y}) = f(x::\vec{y}) \qquad \theta_1(f, x) = f(x)$$

The Universal Machine

Theorem

(a) For every Mealy machine Q the behavioral representation $Q \stackrel{\Theta}{\longrightarrow} [I, O]$ makes the following diagram commute

$$\begin{array}{ccc} Q \times I & \xrightarrow{\theta^Q} & Q \times O \\ \downarrow{\scriptstyle \Theta \times I} & & \downarrow{\scriptstyle \Theta \times O} \\ [I,O] \times I & \xrightarrow{\theta^{[I,O]}} & [I,O] \times O \end{array}$$

(b) $\Theta^{q'} = \Theta^{q''} \iff \langle Q, q' \rangle$ and $\langle Q, q'' \rangle$ indistinguishable

The Universal Machine

Interpretation of the Theorem

- The representation $Q \stackrel{\Theta}{\longrightarrow} [I, O]$ traces the behavior of the machine Q in the machine $[I, O]$
  - $\theta_0^{[I,O]} \circ (\Theta \times I) = \Theta \circ \theta_0^Q$ says that the next state of the representation Θ^q in $[I, O]$ is the representation of the next state in Q
  - $\theta_1^{[I,O]} \circ (\Theta \times I) = \Theta \circ \theta_1^Q$ says that the outputs at the state Θ^q in $[I, O]$ are the same as the outputs at the state q in Q.

The Universal Machine

Interpretation of the Theorem

- The representation $Q \xrightarrow{\Theta} [I, O]$ traces the behavior of the machine Q in the machine $[I, O]$
  - $\theta_0^{[I,O]} \circ (\Theta \times I) = \Theta \circ \theta_0^Q$ says that the next state of the representation Θ^q in $[I, O]$ is the representation of the next state in Q
  - $\theta_1^{[I,O]} \circ (\Theta \times I) = \Theta \circ \theta_1^Q$ says that the outputs at the state Θ^q in $[I, O]$ are the same as the outputs at the state q in Q.
- The Universal Mealy machine thus contains the behavior of any given Mealy machine!

Moore = Mealy

Proposition

Moore processes and Mealy processes are observationally equivalent.

Moore = Mealy

Proposition

Moore processes and Mealy processes are observationally equivalent.

More precisely,

- for every Mealy process, there is a Moore process implementing the same channel, and
- for every Moore process, there is a Mealy process implementing the same channel.

Moore = Mealy

Proof of Moore $\subseteq$ Mealy

Every Moore machine can be viewed as a special kind of Mealy machine, by setting

$$\theta_1^{Me}(q,x) \quad = \quad \begin{cases} \theta_1^{Mo}\big(\theta_0(q,x)\big) & \text{if } \theta_0(q,x)\downarrow \\ \uparrow & \text{otherwise} \end{cases}$$

The fact that

$$\Theta^q_{Mo}(\vec{x}) \quad = \quad \Theta^q_{Me}(\vec{x})$$

follows by the inspection of the definitions of Θ^q.

Moore = Mealy

Proof of Mealy ⊆ Moore

Given a Mealy machine $Q \times I \xrightarrow{\theta} Q \times O$, set

$$\widetilde{Q} \quad = \quad Q \times O$$

and define the induced Moore machine

$$\begin{array}{ccc} \widetilde{Q} \times I & \xrightarrow{\theta_0} & \widetilde{Q} \\ \left\langle q, y, x \right\rangle & \longmapsto & \left\langle \theta_0(q,x), \theta_1(q,x) \right\rangle \end{array} \qquad \begin{array}{ccc} \widetilde{Q} & \xrightarrow{\theta_1} & O \\ \langle q, y \rangle & \longmapsto & y \end{array}$$

The fact that

$$\Theta^q_{Me}(\vec{x}) \quad = \quad \Theta^q_{Mo}(\vec{x})$$

again follows by the inspection of the definitions.

Why **state** machines?

Stateless Mealy

- A Mealy process with 1 state is a partial function:

$$\frac{1 \times A \xrightarrow{\theta} 1 \times B}{A \xrightarrow{\theta} B}$$

Why **state** machines?

Stateless Mealy

- A Mealy process with 1 state is a partial function:

$$\frac{1 \times A \xrightarrow{\theta} 1 \times B}{A \xrightarrow{\theta} B}$$

- A Mealy machine with Q states, but where processes never change state is a Q-indexed family of partial functions:

$$\frac{Q \times A \xrightarrow{\theta_0 = \pi_0} Q \qquad Q \times A \xrightarrow{\theta_1} B}{\left\{ A \xrightarrow{\theta_q} B \mid q \in Q \right\}}$$

Why **state** machines?

Stateless Mealy

- A Mealy process with 1 state is a partial function:

$$\frac{1\times A \xrightarrow{\theta} 1\times B}{A \xrightarrow{\theta} B}$$

- A Mealy machine with Q states, but where processes never change state is a Q-indexed family of partial functions:

$$\frac{Q\times A \xrightarrow{\theta_0=\pi_0} Q \qquad Q\times A \xrightarrow{\theta_1} B}{\left\{A \xrightarrow{\theta_q} B \mid q\in Q\right\}}$$

where $\theta_0(q,x)=\pi_0(q,x)=q$ and $\theta_q(x)=\theta_1(q,x)$

Why **state** machines?

Stateless Moore machines are less useful:

- the outputs are only obtained by observing states
- if there is a single state then there is a single output

Why **state** machines?

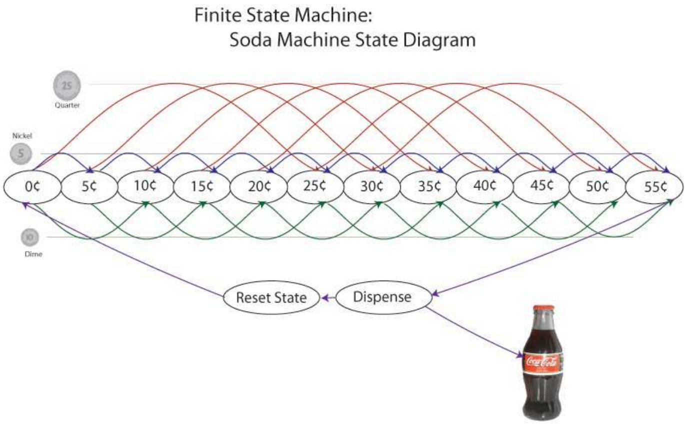
States display the *space* of a process

- computational processes: states assign values to variables
- physical processes: states are positions and momenta of objects
- social processes: states are
  - locations and types of human actors
  - locations and relations of physical actors
- the assignments of properties to entities

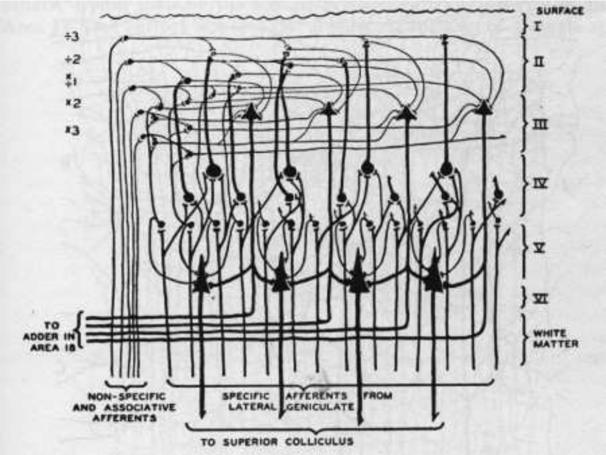
Why **state** machines?

Transitions display *dynamics* of a process

State machines model diverse processes

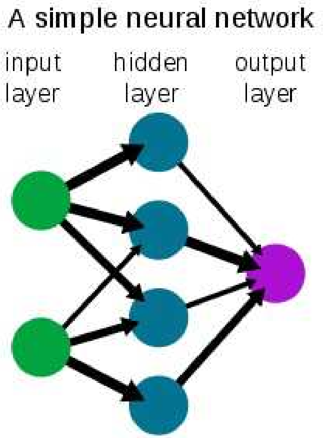


. . . going back to neural nets



Warren S. McCulloch and Walter Pitts, *A logical calculus of the ideas immanent in nervous activity*. B. Math. Biophys. 5(1943)

. . . which also implement channels



But where is **channel security** in all this?

Outline

What is a channel?

State machines and processes

Sharing

Noninterference

What did we learn?

Shared channels, processes and machines

- use of a resource induces a channel
- shared use of a resource induces a *shared channel*.

Shared channels, processes and machines

Definition

Let $\mathbb{L}$ be a security lattice.

A channel (process, machine) is said to be *shared* among the subjects with the security clearances over the lattice $\mathbb{L}$ if its set of inputs I are partitioned over $\mathbb{L}$.

Shared channels, processes and machines

Definition

Let $\mathbb{L}$ be a security lattice.

A channel (process, machine) is said to be *shared* among the subjects with the security clearances over the lattice $\mathbb{L}$ if its set of inputs I are partitioned over $\mathbb{L}$.

More precisely, a shared channel (process, machine) is simply a channel (resp. process, machine) given with a mapping

$$\ell : I \to \mathbb{L}$$

Shared channels, processes and machines

Notation

The inputs available at the security level $k \in \mathbb{L}$ are

$$I_k \quad = \quad \{x \in I \mid \ell(x) = k\}$$

Shared channels, processes and machines

Remark

A shared channel (process, machine) is thus simply a channel (resp. process, machine) where the inputs are partitioned over a security lattice $\mathbb{L}$, in the form

$$I \quad = \quad \coprod_{\ell \in \mathbb{L}} I_\ell$$

where $\coprod$ denotes the disjoint union.

Shared channels, processes and machines

Conventions

For simplicity,

- we usually assume that there is just one actor at each security level, i.e.

$$\mathcal{S} = \mathbb{L}$$

Shared channels, processes and machines

Conventions

For simplicity,

- we usually assume that there is just one actor at each security level, i.e.

$$\mathcal{S} = \mathbb{L}$$

- We sometimes assume that there are just two security levels, Hi and Lo, i.e.

$$\mathbb{L} = \{Lo < Hi\}$$

Problem of sharing

Security goal

When sharing a resource, Bob should only observe the results of the actions at his clearance level.

Problem of sharing

Security goal

When sharing a resource, Bob should only observe the outputs of the inputs at his clearance level.

Problem of sharing

Security goal

When sharing a resource, Bob should only observe the outputs of the inputs at his clearance level.

Security problem

By observing the outputs of his own inputs,
Bob can learn about Alice's inputs and outputs.

Outline

Example 4

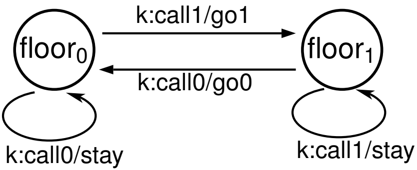
Elevator Story

- Alice and Bob are the only inhabitants of the two apartments on the first floor.
- Alice wakes up, calls the elevator and leaves.
- Bob wakes up and calls the elevator.
- Observing the elevator, Bob learns the state of the world:
  - If the elevator comes from the ground floor, Alice is gone.
  - If the elevator is already at the first floor, Alice is home.

Example 4

Elevator Model

- Q = {floor0, floor1}
- I_k = {k:call0, k:call1}, $k \in \mathbb{L}$ = {Alice, Bob}
- O = {go0, go1, stay}
- θ :



Example 4

Elevator Interference

For Bob, the histories

(Alice:call0 Bob:call1) and (Alice:call1 Bob:call1)

are

- indistinguishable through the inputs, since he only sees Bob:call1 in both of them, yet they are
- distinguishable through the outputs, since Bob's channel outputs are
  - (Alice:call0 Bob:call1) $\longmapsto$ go1
  - (Alice:call1 Bob:call1) $\longmapsto$ stay

Example 4

Remark

- The elevator and the binary addition machines have the same state/transition structure, just slightly different input/output assignments.
  - They can be made isomorphic by refining the elevator behaviors
- The binary multiplication process has the same state/transition structure, also just different input/output assignments.
  - Another elevator could be made isomorphic to the binary multiplication process.

Example 4

Remark

- The elevator and the binary addition machines have the same state/transition structure, just slightly different input/output assignments.
  - They can be made isomorphic by refining the elevator behaviors
- The binary multiplication process has the same state/transition structure, also just different input/output assignments.
  - Another elevator could be made isomorphic to the binary multiplication process.
- **You could build a pocket calculator just from the elevators in two storey buildings.**

Interference

We say that there is *interference* between Alice's and Bob's processes in a shared channel when Bob's outputs depend on Alice's inputs.

We formalize it in the rest of the lecture.

Intuition and terminology

A list of process inputs

$$\vec{x} \quad = \quad (x_1 \ x_2 \cdots x_n) \ \in \ I^*$$

has many names in many models:

- history
- trace
- state of the world

They all support useful intuitions.

Basic assumption

In an environment with a security lattice $\mathbb{L}$
a subject at the level k only sees
the actions performed at the levels $\ell \leq k$.

Basic assumption formalized

Definition

The *k-purge* $\vec{x}\restriction_k \in I_k^*$ of a history $\vec{x} \in I^*$ is defined

$$
\begin{aligned}
()\restriction_k &= () \\
(x{::}\vec{y})\restriction_k &= \begin{cases} x{::}(\vec{y})\restriction_k & \text{if } \ell(x) \leq k \\ (\vec{y})\restriction_k & \text{otherwise} \end{cases}
\end{aligned}
$$

Complement

Definition

The *k-complement* of a history $\vec{x} \in I^*$, is just the subhistory eliminated from the *k-purge*

$$
\begin{aligned}
()\restriction_{\neg k} &= () \\
(x::\vec{y})\restriction_{\neg k} &= \begin{cases} (\vec{y})\restriction_k & \text{if } \ell(x) \leq k \\ x::(\vec{y})\restriction_k & \text{otherwise} \end{cases}
\end{aligned}
$$

Local input equivalence

Definition

We say that the histories $\vec{x}, \vec{y} \in I^*$ are *k-input equivalent* when

$$\vec{x} \lfloor k \rfloor \vec{y} \iff \vec{x}\restriction_k = \vec{y}\restriction_k$$

Local input information

Definition

The *k-input information set* $\left[\vec{x}\right]_k$ is the set of all states of the world that are k-input equivalent with $\vec{x}$, i.e.

$$\left[\vec{x}\right]_k \quad = \quad \left\{\vec{y} \in I^* \mid \vec{x}\restriction_k = \vec{y}\restriction_k\right\}$$

Local input information

Comment

A subject at the level k

- sees *only* a local input history $\vec{x}_k \in I_k^*$ directly
- considers *all* nonlocal input histories *possible*, and its information set is thus

$$\left[\vec{x}_k\right] \quad = \quad \left\{\vec{y} \in I^* \mid \vec{x}_k \lfloor k \rfloor \vec{y}\right\}$$

Local input information

Comment

A subject at the level k

- sees *only* a local input history $\vec{x}_k \in I_k^*$ directly
- considers *all* nonlocal input histories *possible*, and its information set is thus

$$\left[\vec{x}_k\right] = \left\{\vec{y} \in I^* \mid \vec{x}_k \lfloor k \rfloor \vec{y}\right\}$$

- the elements of the information set $\left[\vec{x}_k\right]$ are often called *possible worlds* consistent with $\vec{x}_k$

Digression: Quotients

Lemma

For any set A, there is a one-to-one correspondence between

- equivalence relations $(e) \subseteq A \times A$ and
- partitions $\mathcal{E} \subseteq \wp A$

where

- $(e) \mapsto \mathcal{E} = \big\{\{y \in A \mid x(e)y\} \mid x \in A\big\}$ and
- $\mathcal{E} \mapsto (e)$ where $x(e)y \iff \exists U \in \mathcal{E}.\ x, y \in U$

Digression: Quotients

Lemma

Any function $A \xrightarrow{f} B$ induces the *kernel* equivalence on A

$$x(f)y \iff f(x) = f(y)$$

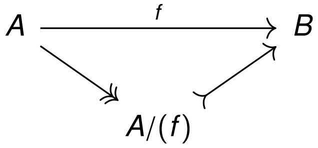
Digression: Quotients

Lemma

Any function $A \stackrel{f}{\to} B$ induces the *kernel* equivalence on A

$$x(f)y \iff f(x) = f(y)$$

The partition $A/(f)$ is the *quotient* of A along f, which factors f through a surjection followed by an injection

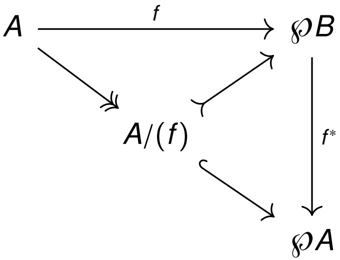


Digression: Quotients

By post-composing the partial function $A \xrightarrow{f} B$ or relation $A \xrightarrow{f} \wp B$ with

$$\begin{array}{ccc} \wp B & \xrightarrow{f^*} & \wp A \\ V & \longmapsto & \bigcup\{U \subseteq A \mid f(U) \subseteq V\} \end{array}$$

the quotient is constructed as follows



Local input views

Terminology

We call *local input views* either of the following equivalent data

- local input equivalences

$$\lfloor k \rfloor \quad \subseteq \quad I^* \times I^*$$

- the partitions into the local input information sets

$$\mathcal{J}_k \quad = \quad \left\{ \left[\vec{x} \right]_k \subseteq I^* \mid \vec{x} \in I^* \right\}$$

for $k \in \mathbb{L}$.

Local channel views

Idea

- When Alice and Bob share a channel $I^+ \xrightarrow{f} O$, then in addition to his inputs, Bob also sees the corresponding outputs

- If Alice's inputs change the state of the process, then the same inputs from Bob may result in different outputs.

Local channel equivalence

Definition

We say that the histories $\vec{x}, \vec{y} \in I^*$ are k-equivalent in the channel $I^* \xrightarrow{f} O^*$ if

$$\vec{x} \lceil f_k \rceil \vec{y} \iff f_k(\vec{x}) = f_k(\vec{y})$$

where

$$f_k() = ()$$

$$f_k(x@\vec{y}) = \begin{cases} f(x)@f_k(\vec{y}) & \text{if } \ell(x) \leq k \\ f_k(\vec{y}) & \text{otherwise} \end{cases}$$

Local channel information

Definition

The *k-information set* with respect to the channel $I^* \xrightarrow{f} O^*$ is the set $\left[\vec{x}\right]_{f_k}$ all histories that yield the same k-outputs, i.e.

$$\left[\vec{x}\right]_{f_k} = \left\{\vec{y} \in I^* \mid f_k(\vec{x}) = f_k(\vec{y})\right\}$$

Local channel views

Terminology

We call *local channel views* either of the following equivalent data

- local channel equivalences

$$\lceil f_k \rceil \quad \subseteq \quad I^* \times I^*$$

- the partitions into the channel information sets

$$\mathcal{J}_{f_k} \quad = \quad \left\{ \left[\vec{x} \right]_{f_k} \subseteq I^* \mid \vec{x} \in I^* \right\}$$

for $k \in \mathbb{L}$.

Noninterference

Definition

A shared channel $I^+ \xrightarrow{f} O$ satisfies the *noninterference* requirement at the level k if for all states of the world $\vec{x}, \vec{y} \in I^*$ holds

$$\vec{x} \lfloor k \rfloor \vec{y} \implies \vec{x} \lceil f_k \rceil \vec{y}$$

Noninterference

Definition

A shared channel $I^+ \xrightarrow{f} O$ satisfies the *noninterference* requirement at the level k if for all states of the world $\vec{x}, \vec{y} \in I^*$ holds

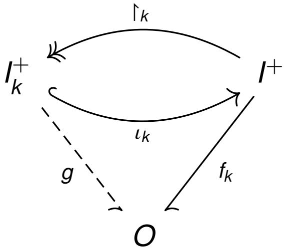
$$\text{whenever} \quad \vec{x}\restriction_k = \vec{y}\restriction_k \quad \text{then} \quad f_k(\vec{x}) = f_k(\vec{y})$$

Noninterference

Proposition 1

The channel $I^+ \xrightarrow{f} O$ satisfies the noninterference requirement if and only if

$$f_k \quad = \quad f_k \circ \iota_k \circ \upharpoonright_k$$



Noninterference

Proof of Proposition 1

- The definition of noninterference says that the kernel of f_k must be at least as large as the kernel of $\restriction_k$.
- It follows that there must be g such that $f_k = g \circ \restriction_k$.
- Since $\restriction_k \circ \iota_k = \mathrm{id}$, we have $g = g \circ \restriction_k \circ \iota_k = f_k \circ \iota_k$.

Noninterference

Proposition 2

For every deterministic channel the following conditions are equivalent

(a) for all $\vec{x}, \vec{y} \in I^*$ holds

$$\vec{x}\restriction_k = \vec{y}\restriction_k \implies f_k(\vec{x}) = f_k(\vec{y})$$

(b) forall $\vec{x} \in I^*$ holds

$$f_k(\vec{x}) = f(\vec{x}\restriction_k)$$

(c) for all $\vec{x}, \vec{z} \in I^*$ there is $\vec{y} \in I^*$

$$\vec{x}\restriction_k = \vec{y}\restriction_k \wedge \vec{y}\restriction_{\neg k} = \vec{z}\restriction_{\neg k} \wedge f_k(\vec{x}) = f_k(\vec{y})$$

Noninterference

Proof of Proposition 2 (c)$\Longrightarrow$(b)

Take in (c) any given $\vec{x}$ and $\vec{z} = ()$.

Then (c) gives $\vec{y}$ such that

(i) $\vec{y}\upharpoonright_{\neg k} = ()$

(ii) $\vec{x}\upharpoonright_k = \vec{y}\upharpoonright_k$

(iii) $f_k(\vec{x}) = f_k(\vec{y})$

which imply

(i) $\vec{y} = \vec{y}\upharpoonright_k$,

(ii) $\vec{x}\upharpoonright_k = \vec{y}$

(iii) $f_k(\vec{x}) = f_k(\vec{x}\upharpoonright_k)$

This yields (b), since $f_k(\vec{x}\upharpoonright_k) = f(\vec{x}\upharpoonright_k)$ is obvious from the definition of f_k.

Noninterference

Proof of Proposition 2 (b)$\Longrightarrow$(a)

If $\vec{x}\restriction_k = \vec{y}\restriction_k$ then

$$f_k(\vec{x}) \stackrel{(b)}{=} f_k(\vec{x}\restriction_k) = f_k(\vec{y}\restriction_k) \stackrel{(b)}{=} f_k(\vec{y})$$

Noninterference

Proof of Proposition 2 (a)$\Longrightarrow$(c)

Given $\vec{x}, \vec{z} \in I^*$, set

$$\vec{y} \quad = \quad \vec{x}\restriction_k \ @ \ \vec{z}\restriction_{\neg k}$$

Then obviously

$$\vec{x}\restriction_k = \vec{y}\restriction_k \quad \wedge \quad \vec{y}\restriction_{\neg k} = \vec{z}_{\neg k}$$

But the first conjunct and (a) imply

$$f_k(\vec{x}) = f_k(\vec{y})$$

Noninterference for processes

Definition

A process satisfies the noninterference property if and only if the induced channel does.

Noninterference in shared processes

Proposition 3

For every deterministic process $\langle Q, q\rangle$ the following conditions are equivalent

(a) for all $\vec{x}, \vec{y} \in I^*$ holds

$$\vec{x}\restriction_k = \vec{y}\restriction_k \implies \Theta_k^q(\vec{x}) = \Theta_k^q(\vec{y})$$

(b) forall $\vec{x} \in I^*$ holds

$$\Theta_k^q(\vec{x}) = \Theta^q(\vec{x}\restriction_k)$$

(c) for all $\vec{x}, \vec{z} \in I^*$ there is $\vec{y} \in I^*$

$$\vec{x}\restriction_k = \vec{y}\restriction_k \wedge \vec{y}\restriction_{\neg k} = \vec{z}\restriction_{\neg k} \wedge \Theta_k^q(\vec{x}) = \Theta_k^q(\vec{y})$$

Noninterference in shared processes

A more informative characterization requires a couple of definitions.

Noninterference in shared processes

Definition

In a process $\langle Q, q_0 \rangle$ we define

- $q' \xrightarrow{\ell} q''$ if there is $a \in I_\ell$ such that $\theta(q', a) = q''$

Noninterference in shared processes

Definition

In a process $\langle Q, q_0 \rangle$ we define

- $q' \stackrel{\ell}{\to} q''$ if there is $a \in I_\ell$ such that $\theta(q', a) = q''$
- $q' \stackrel{\ell}{\twoheadrightarrow} q''$ is the transitive closure of $q' \stackrel{\ell}{\to} q''$
  - $q' \stackrel{M}{\twoheadrightarrow} q''$ for $M \subseteq \mathbb{L}$ is the transitive closure of $\bigcup_{\ell \in M} \stackrel{\ell}{\to}$.

Noninterference in shared processes

Definition

In a process $\langle Q, q_0 \rangle$ we define

- $q' \stackrel{\ell}{\to} q''$ if there is $a \in I_\ell$ such that $\theta(q', a) = q''$
- $q' \stackrel{\ell}{\twoheadrightarrow} q''$ is the transitive closure of $q' \stackrel{\ell}{\to} q''$
  - $q' \stackrel{M}{\twoheadrightarrow} q''$ for $M \subseteq \mathbb{L}$ is the transitive closure of $\bigcup_{\ell \in M} \stackrel{\ell}{\to}$.
- $q' \stackrel{\ell}{\sim} q''$ is the equivalence relation over $q' \stackrel{\ell}{\twoheadrightarrow} q''$
  - $q' \stackrel{M}{\sim} q''$ for $M \subseteq \mathbb{L}$ is the transitive closure of $\bigcup_{\ell \in M} \underset{\ell}{\sim}$.

Noninterference in shared processes

Definition

$q \in Q$ is *reachable* if $q_0 \overset{\mathbb{L}}{\twoheadrightarrow} q$.

Noninterference in shared processes

Proposition 4

A process Q satisfies the noninterference property at the level $k \in \mathbb{L}$ if and only if for all reachable states q', q'' and all histories $\vec{x}$ holds

$$\begin{array}{ccc} q' & \stackrel{\neg k}{\sim} & q'' \\ & \Downarrow & \\ \Theta_k^{q'}(\vec{x}) & = & \Theta_k^{q''}(\vec{x}) \end{array}$$

where $\neg k = \{\ell \not\leq k\}$.

Proof of Proposition 4

We prove

$$\Theta_k^q(\vec{x}) \stackrel{(*)}{=} \Theta_k^{\theta(q,a)}(\vec{x})$$

$$\Updownarrow$$

$$\Theta_k(\vec{x}) \stackrel{3(b)}{=} \Theta\left(\vec{x}\restriction_k\right)$$

Proof of Proposition 4

$3(b) \Longrightarrow (*)$

$$\begin{array}{rcl}\Theta_k^{\theta(q,a)}(\vec{x}) & = & \Theta_k^q(a::\vec{x}) \\ & \stackrel{3(b)}{=} & \Theta^q\left((a::\vec{x})\restriction_k\right) \\ & = & \Theta^q\left(\vec{x}\restriction_k\right) \\ & \stackrel{3(b)}{=} & \Theta_k^q\left(\vec{x}\right)\end{array}$$

Proof of Proposition 4

$(*) \Longrightarrow 3(b)$

Induction along $\vec{x} \in I^*$. The critical case is $\vec{x} = a::\vec{y}$ when $a \in I_{\neg k}$.

$$
\begin{aligned}
\Theta_k^q(a::\vec{y}) &= \Theta_k^{\theta_0(q,a)}(\vec{y}) \\
&\overset{(*)}{=} \Theta_k^q\left(\vec{y}\right) \\
&\overset{(IH)}{=} \Theta^q\left(\vec{y}\restriction_k\right) \\
&\overset{a\in I_{\neg k}}{=} \Theta^q\left((a::\vec{y})\restriction_k\right)
\end{aligned}
$$

Noninterference in shared processes

Interpretation of Proposition 4

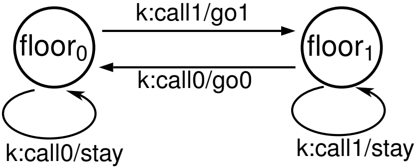
Here are several ways to rephrase the characterization of the processes satisfying k-noninterference:

- At any reachable state q, the state changes induced by the actions of $\neg k$ must be unobservable for k.
- Any pair of states connected by the actions from $\neg k$ must be observationally indistinguishable for k.
- The processes of k-actions starting from any pair of $\neg k$-connected states mustinduce the same channel.

Application of Proposition 4: Example 4

Remember the elevator model

- Q = {floor0, floor1}
- I_k = {k:call0, k:call1}, $k \in \mathbb{L}$ = {Alice, Bob}
- O = {go0, go1, stay}
- θ :



Application of Proposition 4: Example 4

Remember the elevator interference

The histories

$$(\text{Alice:call0} \;\; \text{Bob:call1}) \quad \text{and} \quad (\text{Alice:call1} \;\; \text{Bob:call1})$$

are for Bob

- indistinguishable by the inputs, since he only sees Bob:call1 in both of them, yet they are
- distinguishable by the outputs, since Bob's channel outputs are
  - $(\text{Alice:call0} \;\; \text{Bob:call1}) \longmapsto \text{go1}$
  - $(\text{Alice:call1} \;\; \text{Bob:call1}) \longmapsto \text{stay}$

Application of Proposition 4: Example 4

Question: How should the elevator be modified to assure the noninterference requirement for Bob?

Application of Proposition 4: Example 4

Question: How should the elevator be modified to assure the noninterference requirement for Bob?

Answer: After each action, the elevator should (unobservably) return to the same state.

Application of Proposition 4: Example 4

Question: How should the elevator be modified to assure the noninterference requirement for Bob?

Answer: After each action, the elevator should (unobservably) return to the same state.

- The outputs need to be redefined to implement this.

Outline

What did we learn?

- Resources are modeled using channels, as history dependent functions
- Channels are described ("programmed") using state machines
- Resource security processes are modeled using shared channels
- The simplest and the strongest channel security requirement is *noninterference*.

Access Control vs Noninterference

Bell-LaPadula (from Lecture 2)

The no-read-up condition prevents

- k-subjects' accesses to ℓ-objects for $\ell \not\leq k$
- along any of the *provided system* channels

Access Control vs Noninterference

Bell-LaPadula (from Lecture 2)

The no-read-up condition prevents

- k-subjects' accesses to ℓ-objects for $\ell \not\leq k$
- along any of the *provided system* channels

Noninterference (from this Lecture)

The noninterference condition prevents

- k-subjects' accesses to ℓ-objects for $\ell \not\leq k$
- along any *unspecified covert* channels

Huh?

- But what are covert channels?

Huh?

- But what are covert channels?

- We'll deal with them next time.

Trouble

Covert channels can never be completely eliminated.

Trouble

Covert channels can never be completely eliminated.

In practice, **noninterference is usually impossible.**

Noninterference is almost never satisfied

- trying a password releases some information
- voting releases some information

Declassification problem