CS 428/528 Lecture 14: A Lattice of Information & Robust Declassification

Feb 29, 2024

Based on the CSFW01 paper/slides by Zdancewic & Myers and the CSFW93 paper by Landauer and Redmond

Information Flow Security

Information flow policies are a natural way to specify precise, system-wide, multi-level security requirements.

Enforcement mechanisms are often too restrictive – prevent desired policies.

Information flow controls provide declassification mechanisms to accommodate intentional leaks.

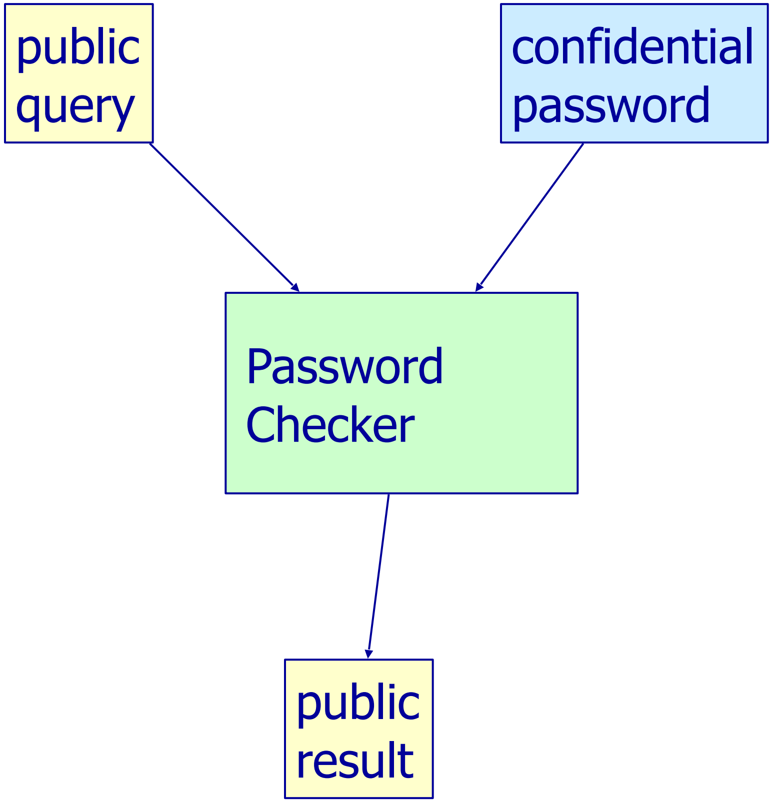
But… hard to understand end-to-end system behavior.

Declassification

Declassification (downgrading) is the intentional release of confidential information.

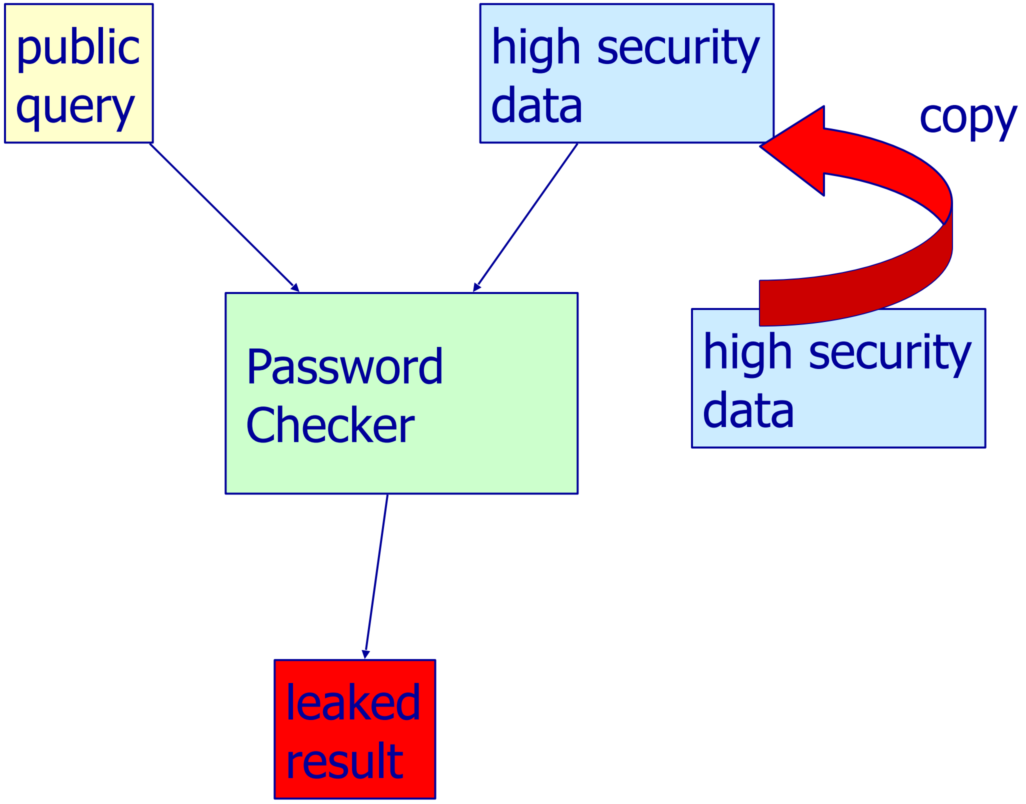
Policy governs use of declassification operation.

Password Example



This system includes declassification.

Attack: Copy data into password



Attacker can launder data through the password checker.

Robust Declassification

Goal:

Formalize the intuition that an attacker should not be able to abuse the downgrade mechanisms provided by the system to cause more information to be declassified than intended.

How to Proceed?

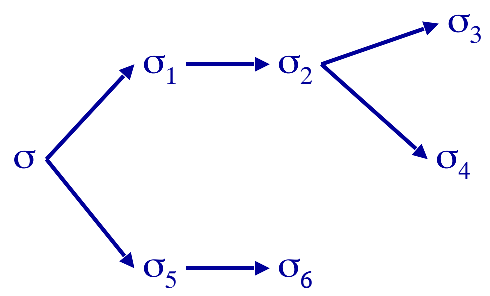
- Characterize what information is declassified.
- Make a distinction between "intentional" and "unintentional" information flow.
- Explore some of the consequences of robust declassification.

A Simple System Model

A system S is a pair:

Σ is a set of states: $\sigma_1, \sigma_2, \ldots$

$\rightarrow$ is a transition relation in $\Sigma \times \Sigma$



Views of a System

A view of $(\Sigma, \rightarrow)$ is an equivalence relation on Σ.

Example: $\Sigma = \mathrm{String} \times \mathrm{Integer}$
"integer component is visible"

$$(x,i) \approx_I (y,j) \text{ iff } i = j$$

("attack at dawn", 3) $\approx_I$ ("retreat", 3)

("attack at dawn", 3) $\not\approx_I$ ("retreat", 4)

Example Views

Example: Σ = String × Integer

"string component is visible"

$(x,i) \approx_S (y,j)$ iff $x = y$

"integer is even or odd"

$(x,i) \approx_E (y,j)$ iff $i\%2 = j\%2$

"complete view"

$(x,i) \approx_T (y,j)$ iff $(x,i) = (y,j)$

Passive Observers

A view induces an observation of a trace:

τ_1 = ("x",1)→("y",1)→("z",2)→("z",3)

τ_1 through view $\approx_I$

1 → 1 → 2 → 3

τ_2 = ("a",1)→("b",2)→("z",2)→("c",3)

τ_2 through view $\approx_I$

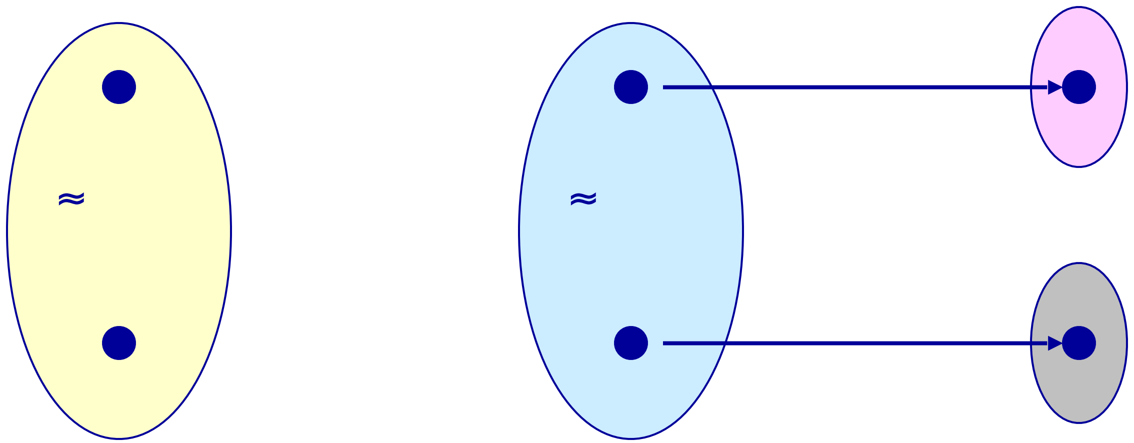
1 → 2 → 2 → 3

Observational Equivalence

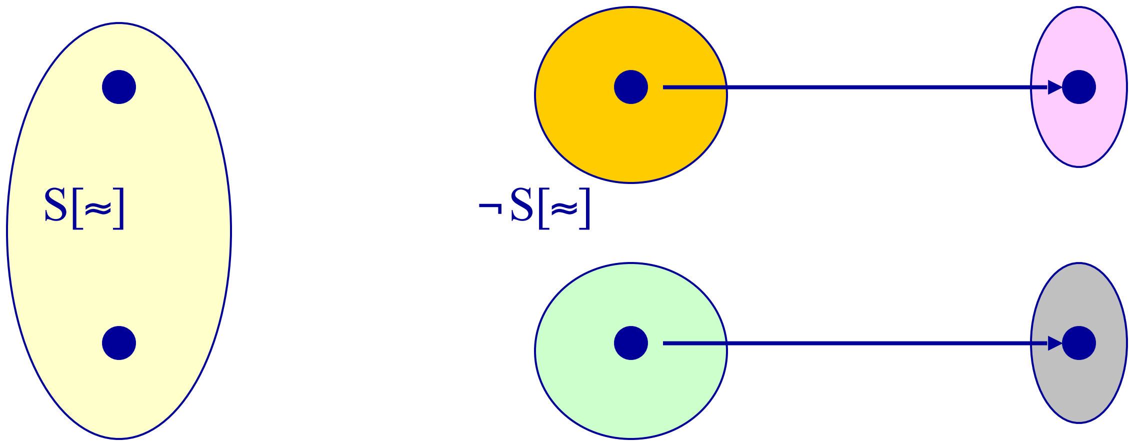
The induced observational equivalence is $S[\approx]$:

$\sigma \; S[\approx] \; \sigma'$ if the traces from σ look the same as the traces from σ' through the view $\approx$.

Simple Example



Simple Example



Observational Equivalence

σ

$S[\approx]$?

σ'

Are σ and σ' observationally equivalent with respect to $\approx$?

Observational Equivalence

$Trc_{\sigma}(S)$

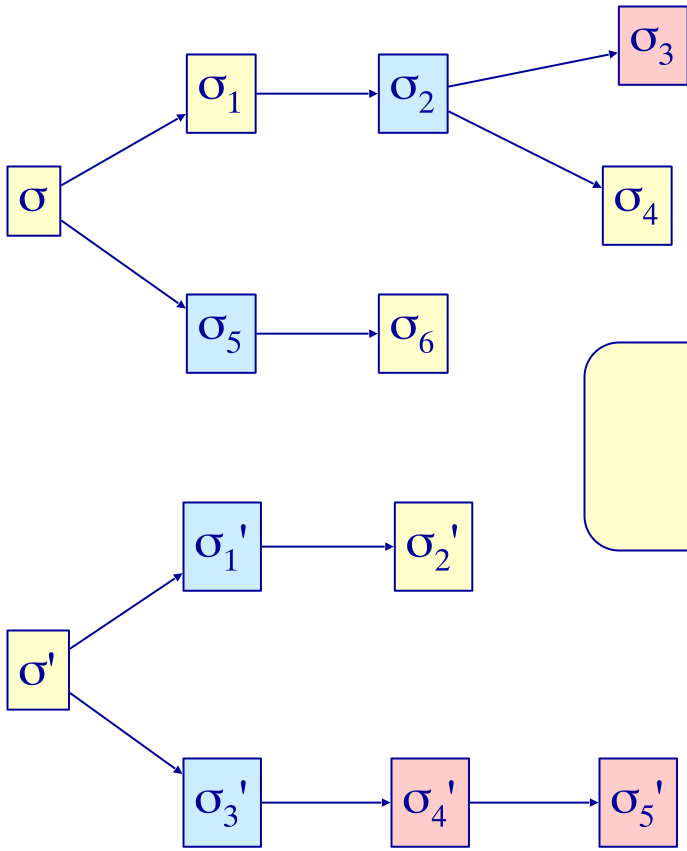
$$\begin{tikzcd}
 & & & \sigma_3 \\
 & \sigma_1 \arrow[r] & \sigma_2 \arrow[ur] \arrow[dr] & \\
\sigma \arrow[ur] \arrow[dr] & & & \sigma_4 \\
 & \sigma_5 \arrow[r] & \sigma_6 &
\end{tikzcd}$$

Take the sets of traces starting at σ and σ'.

$Trc_{\sigma'}(S)$

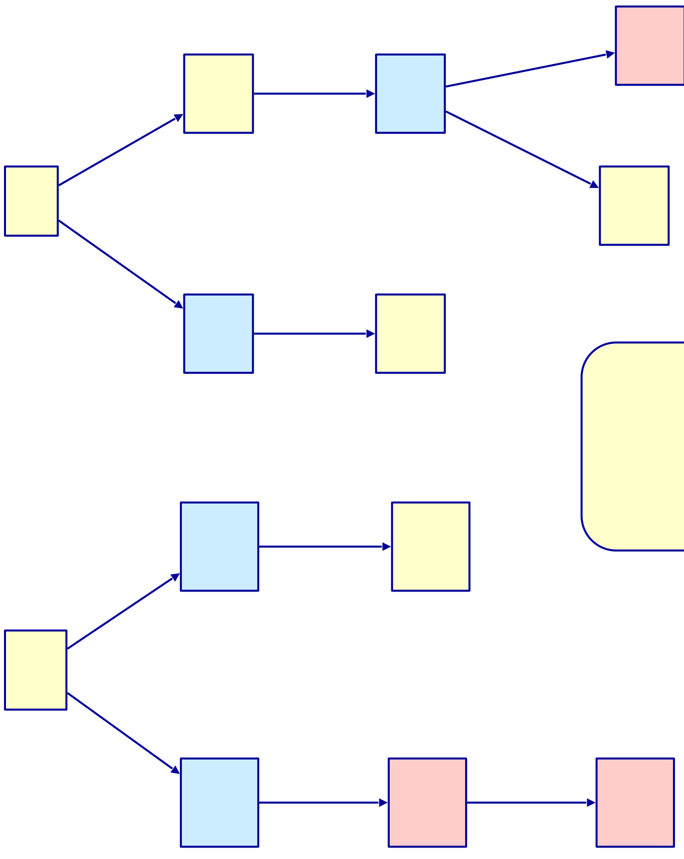
$$\begin{tikzcd}
 & \sigma_1' \arrow[r] & \sigma_2' & \\
\sigma' \arrow[ur] \arrow[dr] & & & \\
 & \sigma_3' \arrow[r] & \sigma_4' \arrow[r] & \sigma_5'
\end{tikzcd}$$

Observational Equivalence



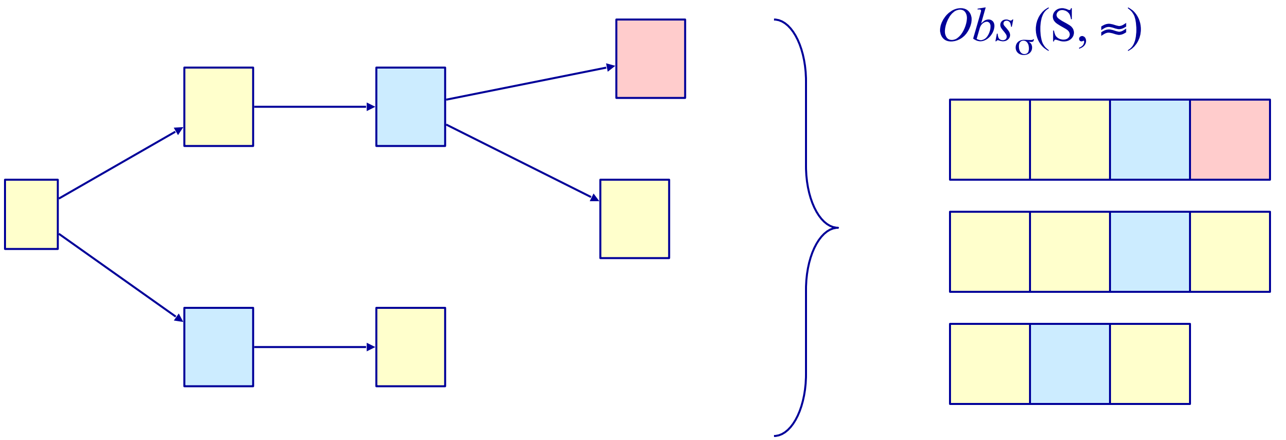
Look at the traces modulo ≈.

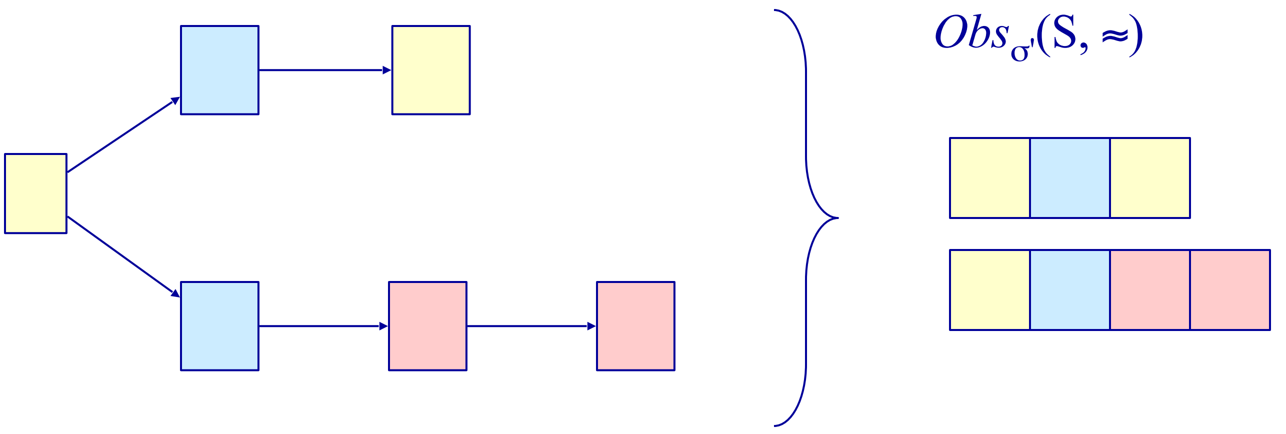
Observational Equivalence



Look at the traces
modulo ≈.

Observational Equivalence

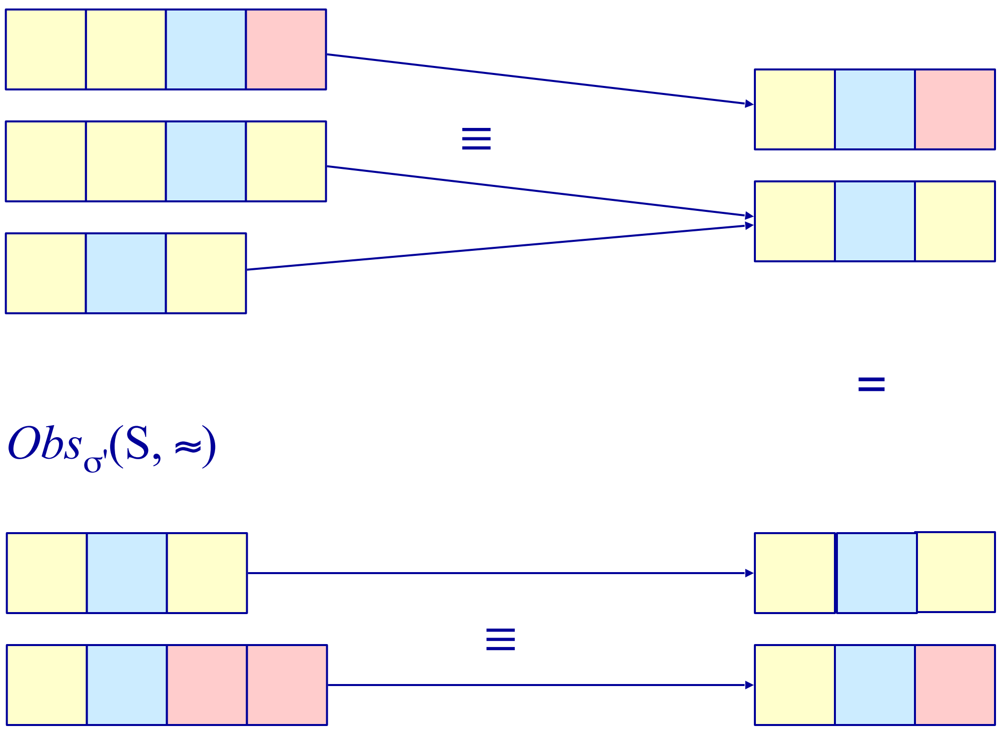




Observational Equivalence

$Obs_{\sigma}(S, \approx)$

$Obs_{\sigma'}(S, \approx)$



Are they stutter equivalent?

Yes!

$\sigma \quad S[\approx] \quad \sigma'$

Why Did We Do This?

$\approx$ is a view of Σ indicating what
an observer sees directly.

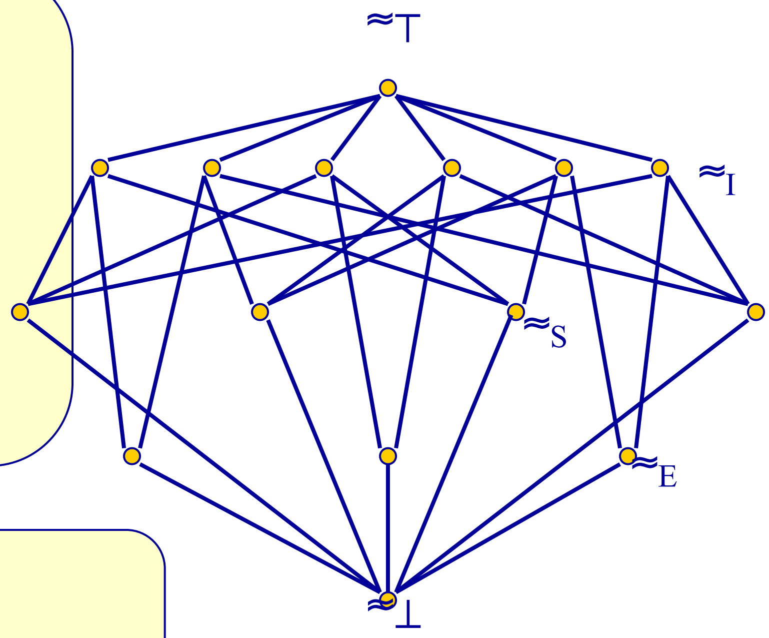
$S[\approx]$ is a view of Σ indicating what
an observer learns by watching S evolve.

Need some way to compare them…

An Information Lattice

The views of a system S form a lattice.

Order: $\approx_A \sqsubseteq \approx_B$

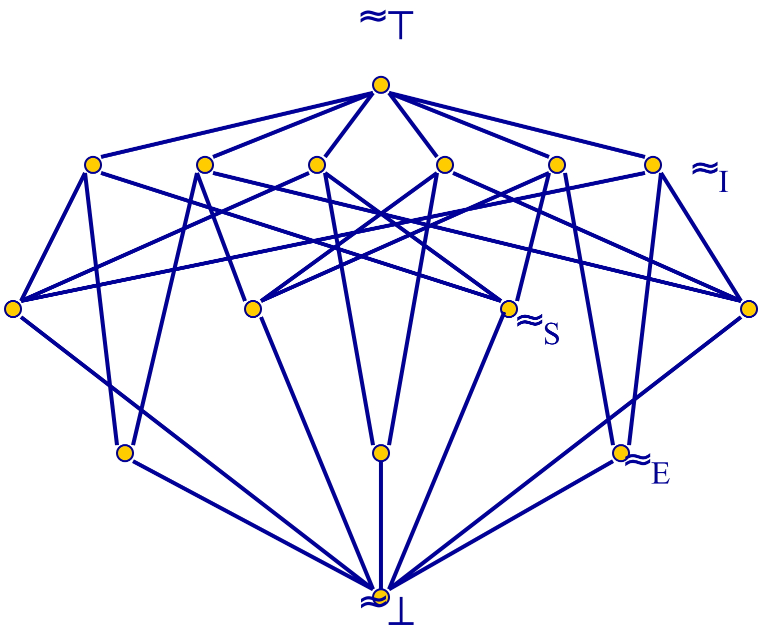


Intuition:
Higher = "more information"

Lattice Order

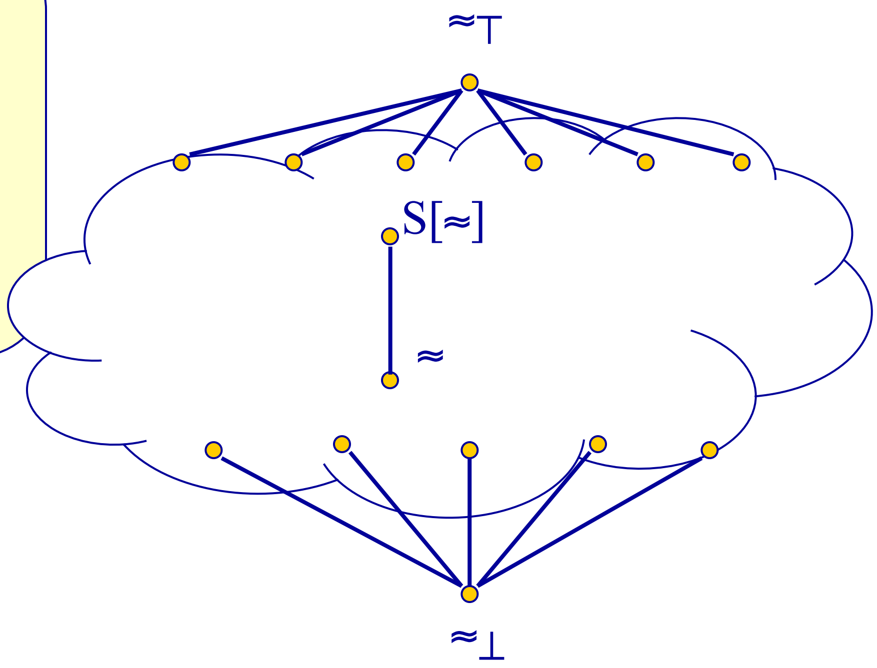
$\approx_\top$ is the identity relation
$\approx_\bot$ is the total relation

Join: $\approx_A \sqcup \approx_B$
Meet: $\approx_A \sqcap \approx_B$



Information Learned via Observation

Intuition: An observer only learns information by watching the system. (Can't lose information.)

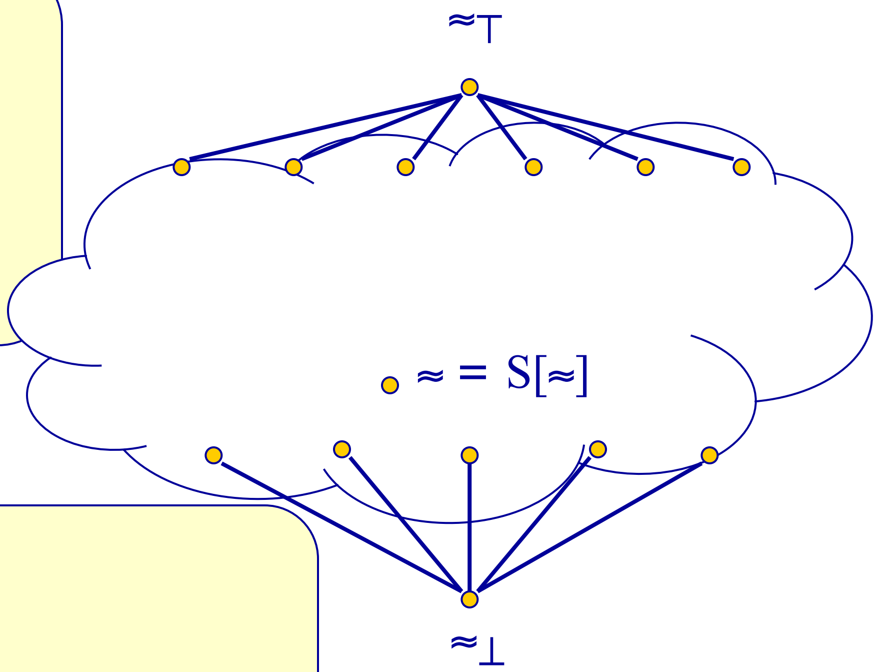


Lemma: $\approx \sqsubseteq S[\approx]$

Natural Security Condition

Definition: A system $\mathcal{S}$ is $\approx$-secure whenever

$$\approx \; = \; \mathcal{S}[\approx]$$



Closely related to standard definitions of non-interference style properties.

Example: A Password System

State of a 5-tuple: <t, h, p, q, r>

- t: 0 or 1 (0 the password checker has not run yet, 1 the checker completed)
- h: a bit representing the high security data
- p: a bit representing the password
- Q: the query submitted by the external user
- r: toggles the boolean value if p and q match

State transition relation S:

<0,h,p,p,0> —> <1,h,p,p,1>
<0,h,p,p,1> —> <1,h,p,p,0>
<0,h,p,q,0> —> <1,h,p,p,0>
<0,h,p,q,1> —> <1,h,p,p,1>

$$\langle t,h,p,q,r\rangle \approx \langle t',h',p',q',r'\rangle \iff (t=t')\wedge(q=q')\wedge(r=r')$$

$$\langle t,h,p,q,r\rangle \; S[\approx] \; \langle t',h',p',q',r'\rangle \iff (t=t')\wedge(q=q')\wedge(r=r')\wedge(t=0\Rightarrow(p=p'))$$

Example: A Password System

State transition relation S:

<0,h,p,p,0> —> <1,h,p,p,1>
<0,h,p,p,1> —> <1,h,p,p,0>
<0,h,p,q,0> —> <1,h,p,p,0>
<0,h,p,q,1> —> <1,h,p,p,1>

Attack transition relation A:

<0,h,p,q,r> —>A <0,h,h,q,r>

S′ = S ∪ A

$$\langle t,h,p,q,r\rangle \; S'[\approx] \; \langle t',h',p',q',r'\rangle$$
$$\Leftrightarrow$$
$$(t=t')\wedge(q=q')\wedge(r=r')\wedge$$
$$(t=0\Rightarrow p=p'\vee h=h'\vee p=h'\vee h=p')$$

Example: A Password System

State transition relation S:

<0,h,p,p,0> —> <1,h,p,p,1>
<0,h,p,p,1> —> <1,h,p,p,0>
<0,h,p,q,0> —> <1,h,p,p,0>
<0,h,p,q,1> —> <1,h,p,p,1>

Attack transition relation A:

<0,h,p,q,0> —>A <0,h,h,q,1>
<0,h,p,q,1> —>A <0,h,h,q,0>

S′ = S ∪ A

$$\langle t, h, p, q, r\rangle \, \mathsf{S'}[\approx] \, \langle t', h', p', q', r'\rangle$$
$$\Leftrightarrow$$
$$(t = t') \wedge (q = q') \wedge (r = r') \wedge$$
$$(t = 0 \Rightarrow (p = p') \vee (h = h'))$$

Declassification

Declassification is intentional leakage of information.

Implies that $\approx \neq S[\approx]$

We want to characterize unintentional declassification.

A Simple Attack Model

An $\approx_A$-attack is a system $A = (\Sigma, \rightarrow_A)$
such that $\approx_A = A[\approx_A]$

$\approx_A$ is the attacker's view

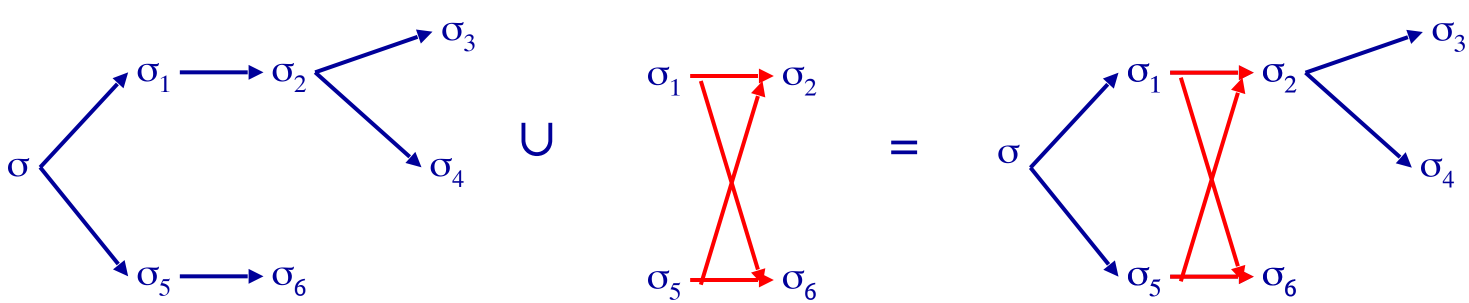
$\rightarrow_A$ is a set of additional transitions introduced by the attacker

$\approx_A = A[\approx_A]$ means "fair environment"

Attacked Systems

Given a system $S = (\Sigma, \rightarrow)$
and attack $A = (\Sigma, \rightarrow_A)$
the attacked system is:

$S \cup A = (\Sigma, \rightarrow \cup \rightarrow_A)$

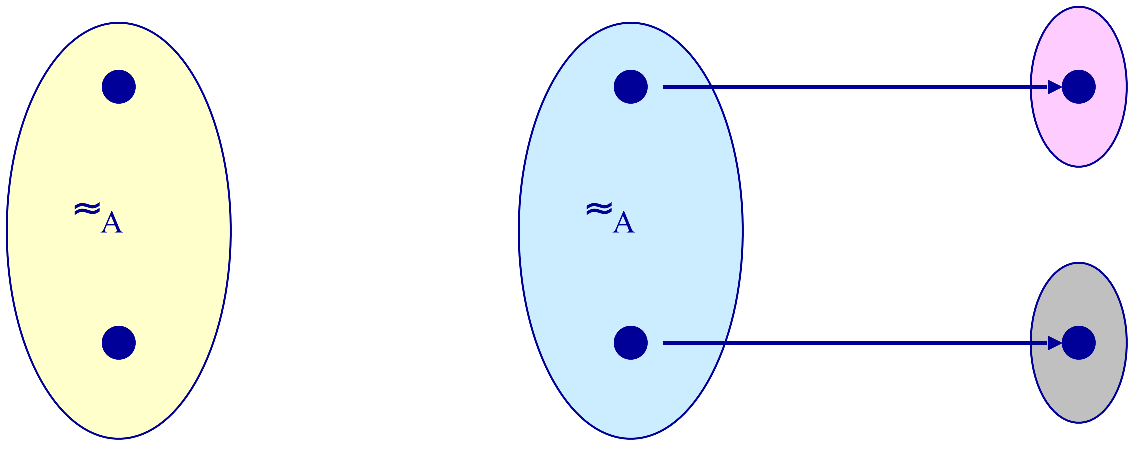


More Intuition

$S[\approx]$ describes the information intentionally declassified by the system – a specification for how S ought to behave.

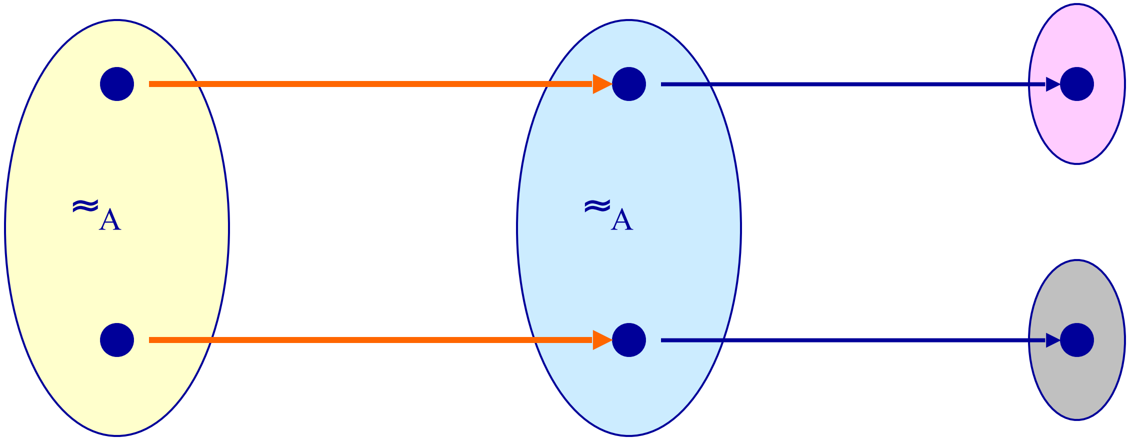
$(S \cup A)[\approx_A]$ describes the information obtained by an attack A.

Example Attack



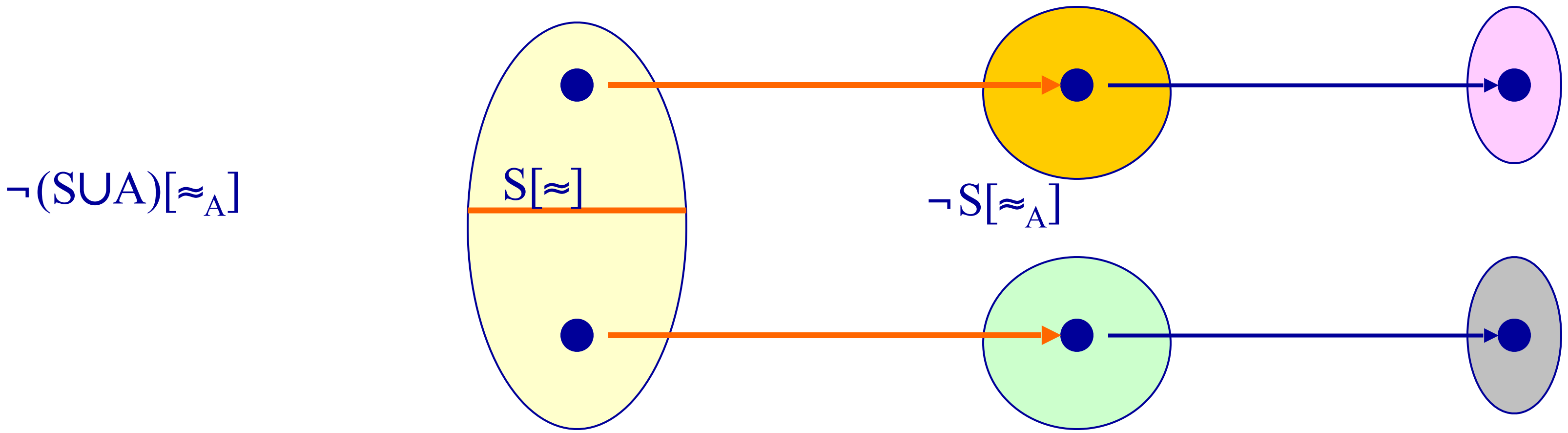
Example Attack

Attack transitions affect the system.



Example Attack

Attacked system may reveal more.

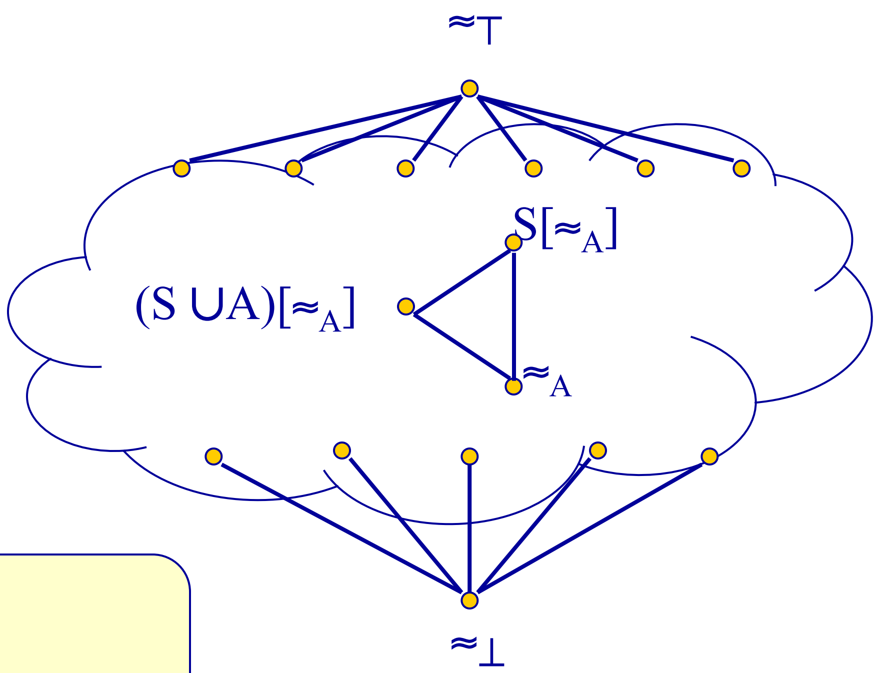


Robust Declassification

A system S is robust with respect to attack A whenever

$(S \cup A)[\approx_A] \sqsubseteq S[\approx_A]$.

Intuition: Attack reveals no additional information.



Secure Systems are Robust

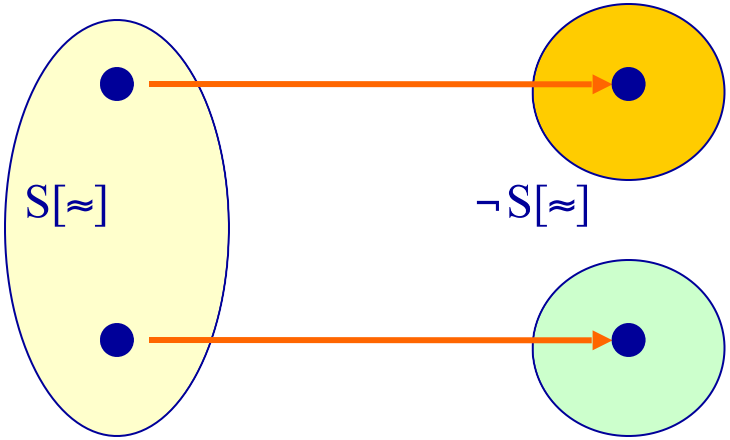
Theorem:
If S is $\approx_A$-secure then S is $\approx_A$-robust with respect to all $\approx_A$-attacks.

Intuition: S doesn't leak any information to $\approx_A$-observer, so no declassifications to exploit.

Characterizing Attacks

Given a system S and a view $\approx_A$, for what $\approx_A$-attacks is S robust?

Need to rule out
Attack transitions
Like this:



Providing Robustness

- Identify a class of relevant attacks
  - Example: attacker may modify / copy files
  - Example: untrusted host may send bogus requests
- Try to verify that the system is robust vs. that class of attacks.
- Proof fails… system is insecure.

- Possible to provide enforcement mechanisms against certain classes of attacks
  - Protecting integrity of downgrade policy

Conclusions

It's critical to prevent downgrading mechanisms in a system from being abused.

Robustness is an attempt to capture this idea in a formal way.

Suggests that integrity and confidentiality are linked in systems with declassification.

A Lattice of Information [Landauer & Redmond CSFW93]

We will now construct of the lattice. We first define a set, $\mathcal{I}(\Sigma)$, to be the set of all equivalence relations on the set Σ. We will define an ordering on this set that makes it a complete lattice. The ordering on $\mathcal{I}(\Sigma)$ is defined as follows

$$\approx \leq \sim \;\; \leftrightarrow \;\; \forall \sigma_1, \sigma_2 \;\; (\sigma_1 \sim \sigma_2 \Rightarrow \sigma_1 \approx \sigma_2) \qquad (1)$$

where $\approx$ and $\sim$ are elements of the set $\mathcal{I}$.

We will now demonstrate why the ordering (1) makes the information set on Σ into a complete lattice. It is sufficient to show that for any set, $P \subseteq \mathcal{I}(\Sigma)$, there exists a least upper bound for that set [1, 2]. It follows from lattice theory that this is enough to guarantee that the information set is a lattice. It is not difficult to see that the least upper bound of the set P is the the equivalence relation, $\sim$, given by

$$\forall x, y \in \Sigma \;\; (x \sim y \leftrightarrow \forall \approx \in P \;\; x \approx y)$$

A Lattice of Information [Landauer & Redmond CSFW93]

as follows: for any function, $f : \Sigma \rightarrow X$, we will define $||f||$ to be the element of $\mathcal{I}(\Sigma)$ for which

$$\forall \sigma, \sigma' \in \Sigma \ \ (\sigma \ ||f|| \ \sigma' \ \leftrightarrow \ f(\sigma) = f(\sigma'))$$

Theorem 1 *For any set Σ, the following properties hold:*

- *any element of $\mathcal{I}(\Sigma)$ can be represented as $||f||$ for some set, X, and some function $f : \Sigma \rightarrow X$.*
- *$||f|| = ||g||$ iff there exists a set isomorphism, ϕ, from the range of f to the range of g such that $g = \phi \circ f$.*
- *$||g|| \leq ||f||$ iff there exists a function, ϕ, such that $g = \phi \circ f$.*
- *if $f : \Sigma \rightarrow X$ and $g : \Sigma \rightarrow Y$ then*

  $$||f|| \vee ||g|| = ||h||$$

  where $h : \Sigma \rightarrow X \times Y$ is defined by

  $$\forall \sigma \in \Sigma \quad h(\sigma) = (f(\sigma), g(\sigma))$$

A Lattice of Information [Landauer & Redmond CSFW93]

The most basic property of the lattice is the manner in which a function $f : \Sigma_1 \rightarrow \Sigma_2$ induces a function $f_\# : \mathcal{I}(\Sigma_2) \rightarrow \mathcal{I}(\Sigma_1)$. The function $f_\#$ can be defined by the equation

$$f_\#(||g||) = ||g \circ f||$$

Equivalently, if $\sim \in \mathcal{I}(\Sigma_2)$, then $f_\#(\sim)$ is the equivalence relation given by

$$x \quad f_\#(\sim) \quad y \qquad \leftrightarrow \qquad f(x) \quad \sim \quad f(y)$$

An important property of this induced function is that for $f : \Sigma_1 \rightarrow \Sigma_2$ and $g : \Sigma_2 \rightarrow \Sigma_3$, we have,

$$f_\# \circ g_\# = (g \circ f)_\#$$

Also if $id : \Sigma_1 \rightarrow \Sigma_1$ is the identity map, then $id_\#$ denotes the identity map on $\mathcal{I}(\Sigma_1)$.

A Lattice of Information [Landauer & Redmond CSFW93]

The practical significance of the induced function $f_\#$ is that it provides a formalism for determining the source of updated information after a state change. To elaborate, we formalize the notion of state change. Let $R : \Sigma \rightarrow \Sigma$ be a transition function. Let $f : \Sigma \rightarrow X$ be a view of the state space. If σ is the state before the transition, then the value of f after the transition is $f \circ R(\sigma)$. Thus the information in f after the transition can be determined from knowing the information in

$$||f \circ R|| = R_\#(||f||)$$

before the transition.

A Lattice of Information [Landauer & Redmond CSFW93]

A second important concept is the notion of a function leaving certain information invariant. If $R : \Sigma \rightarrow \Sigma$ is a function then we define $\mathbf{fix}(R)$ to be the greatest element of $\mathcal{I}(\Sigma)$ such that

$$\forall \sigma \in \Sigma \;\; \sigma \quad \mathbf{fix}(R) \quad R(\sigma)$$

The equivalence relation $\mathbf{fix}(R)$ can be formed by constructing the reflexive transitive closure of the symmetric relation that identifies σ and $R(\sigma)$ for all $\sigma \in \Sigma$.

This is important for expressing a requirement that a high process does not write down. If $\sim \in \mathcal{I}(\Sigma)$ represents information with a low sensitivity label and R represents a transition that is being executed by a high process, then we require that the high transition leave the low information invariant. Using the above notation this can be expressed as $\sim \leq \mathbf{fix}(R)$. This

A Lattice of Information [Landauer & Redmond CSFW93]

We will suppose the existence of a distributive lattice, L, representing sensitivity levels. We will suppose that we have a state machine consisting of an initial state, $\sigma_0 \in \Sigma$, a transition function

$$R : \Sigma \times I \to \Sigma$$

and output functions $o_\lambda : \Sigma \to O_L$ for each sensitivity level $\lambda \in L$. We will assume also that the set of inputs I is partitioned into disjoint sets, I_λ, where $\lambda \in L$.

The transition function, R, can be used to define a function

$$R^\star : \Sigma \times I^\star \to \Sigma$$

where $I^\star$ is the set of sequences of elements of I as follows:

$$R^\star(\sigma, ()) = \sigma$$

$$R^\star(\sigma, (i_0, \ldots, i_{n+1})) = R(R^\star(\sigma, (i_0, \ldots, i_n)), i_{n+1})$$

For each sensitivity level $\lambda \in L$ we will form a purge function, $p_\lambda : I^\star \to I^\star$, that takes a sequence of elements of I and returns the sequence formed by removing all the elements not in some $I_{\lambda'}$ where $\lambda' \leq \lambda$.

The non-interference property states that for all sensitivity levels, $\lambda \in L$, and all input sequences $(i_0, \ldots, i_n) \in I^\star$, we have

$$o_\lambda(R^\star(\sigma_0, (i_0, \ldots, i_n)) = o_\lambda(R^\star(\sigma_0, p_\lambda(i_0, \ldots, i_n))$$

A Lattice of Information [Landauer & Redmond CSFW93]

Theorem 3 Haigh-Young Unwinding *Suppose that all states are reachable and let $R_i(\sigma) = R(\sigma, i)$ for all $\sigma \in \Sigma$ and $i \in I$. The non-interference property is satisfied if and only if there exists a function, $lvl : L \longrightarrow \mathcal{I}(\Sigma)$ such that*

- *(Information flows up) For all $i \in I$*

$$(R_i)_\#(lvl(\lambda)) \leq \bigcup_{\lambda' \leq \lambda} lvl(\lambda')$$

- *(Processes only write up) For all λ, λ' such that λ' is not greater than λ, and all $i \in I_\lambda$,*

$$lvl(\lambda') \leq \mathbf{fix}((R_i)_\#)$$

- *(Output is determined by the information at a level) For all λ,*

$$||o_\lambda|| \leq lvl(\lambda)$$