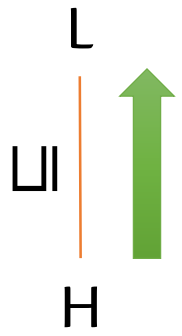
CS 5430
Expressive Information Flow Labels

Elisavet Kozyri

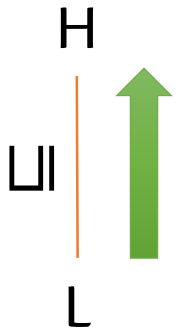
Spring 2019

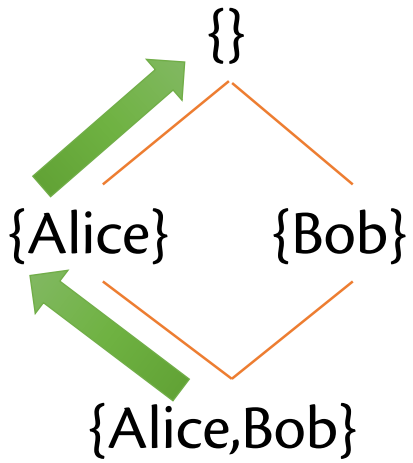
A lattice for integrity

- Information is allowed to flow from data of high integrity (H) to data of low integrity (L).
- Low integrity (e.g., corrupted) data is not allowed to flow to high integrity data.



Lattices for confidentiality





Defining allowed flows based only on a lattice might be too conservative or too permissive!

Examples for confidentiality

Allowing a flow may depend on **operations**

Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`res:=maj(vote, vote', …)`			

`vote` is high, but result of `maj` can be low.

Allowing a flow may depend on **operations**

Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`res:=maj(vote, vote', …)`	YES		

`vote` is high, but result of `maj` can be low.

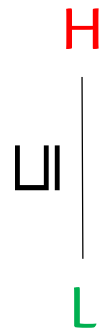
Declassification: A desirable flow from H to L.

Allowing a flow may depend on **operations**

Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`res:=maj(vote, vote', …)`	YES	NO	is too conservative

`vote` is high,
but result of
`maj` can be low.

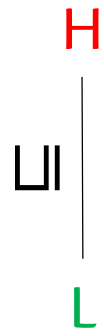
Allowing a flow may depend on **operations**



Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`c:=Enc(msg,key)`			

`msg` and `key` are high, but result of `Enc` can be low.

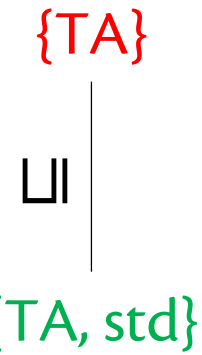
Allowing a flow may depend on **operations**



Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`c:=Enc(msg,key)`	YES	NO	is too conservative

`msg` and `key` are high, but result of `Enc` can be low.

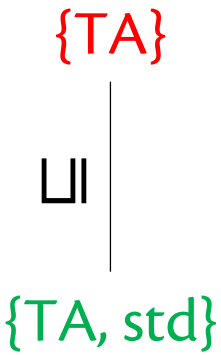
Allowing a flow may depend on **state**



Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`if end_of_semester() then out:=release(grade)`			

`grade` can flow only to TA, but at the end of the semester it can also flow to std.

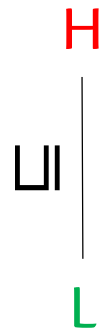
Allowing a flow may depend on **state**



Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`if end_of_semester() then out:=release(grade)`	YES	NO	is too conservative

`grade` can flow only to TA, but at the end of the semester it can also flow to std.

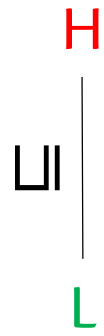
Allowing a flow may depend on **ownership**



Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`if pgm_executed_by_FBI()` `  out:=release(xfile)`			

`xfile` is high and owned by FBI. Only the FBI can make it low.

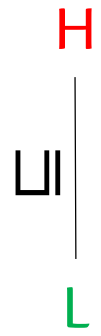
Allowing a flow may depend on **ownership**



Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`if pgm_executed_by_FBI() out:=release(xfile)`	YES	NO	is too conservative

`xfile` is high and owned by FBI. Only the FBI can make it low.

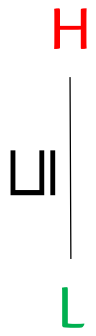
Allowing a flow may depend on **operations**



Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`a:=asgn(papers,reviewers)`			

`reviewers` is low, but result of `asgn` should be high.

Allowing a flow may depend on **operations**



Example for confidentiality	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`a:=asgn(papers,reviewers)`	NO	YES	is too permissive

`reviewers` is low, but result of `asgn` should be high.

Classification: A mandatory flow from L to H.

Examples for integrity

Allowing a flow may depend on **operations**

Example for integrity	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`entry:=sanitize(userIN)`			

`userIN` is low, but result of `sanitize` can be high.

Allowing a flow may depend on **operations**

Example for integrity	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`entry:=sanitize(userIN)`	YES	NO	is too conservative

`userIN` is low, but result of `sanitize` can be high.

Endorsement: A desirable flow from L to H.

Allowing a flow may depend on **operations**

Example for integrity	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`doc' := excerpt(doc)`			

`doc` is high, but result of `excerpt` should be low.

Allowing a flow may depend on **operations**

Example for integrity	Is flow desirable?	Is flow allowed by lattice?	Defining allowed flows based on lattice
`doc' :=excerpt(doc)`	NO	YES	is too permissive

`doc` is high, but result of `excerpt` should be low.

Deprecation: A mandatory flow from H to L.

Reclassifiers cause restrictions to change

Reclassifier	Restriction	Change
Operations	Confidentiality	Declassification Classification
State Ownership	Integrity	Endorsement Deprecation

Reclassifiers cause *reclassifications*!

We need information flow labels that specify reclassifications.

Reactive Information Flow (RIF) labels

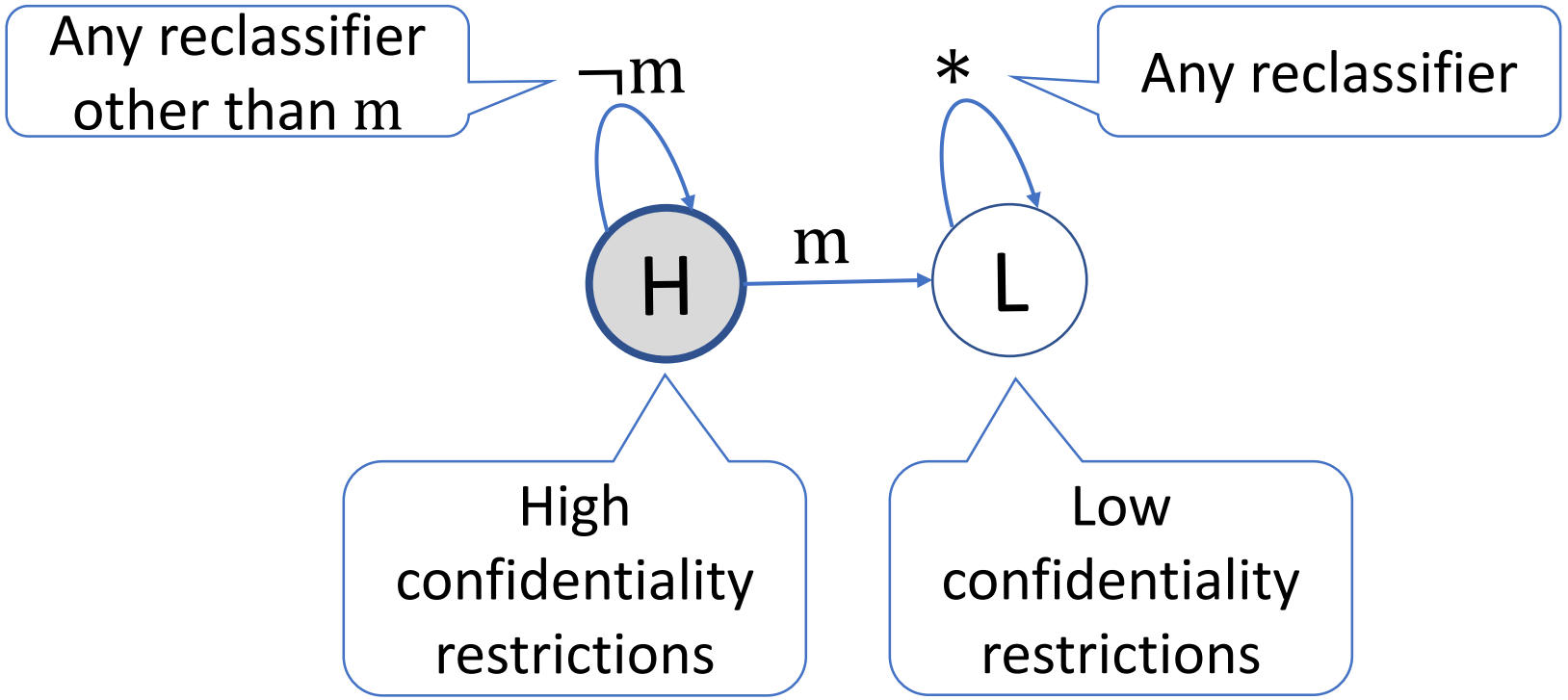
- A *RIF label* maps sequences of operations to restrictions.
  - Restrictions are taken from a partially ordered set $\langle R, \sqsubseteq_R \rangle$.
  - For confidentiality, restrictions can be sets of principals allowed to read.
- *Reclassifiers* abstract operations applied on inputs:
  - $[op(e_1, \ldots, e_n)]_{f_1,..,f_n}$

Confidentiality example

$$\texttt{res} := [\texttt{maj}(\texttt{vote}, \texttt{vote}', \ldots)]_{m}$$

Reclassifier

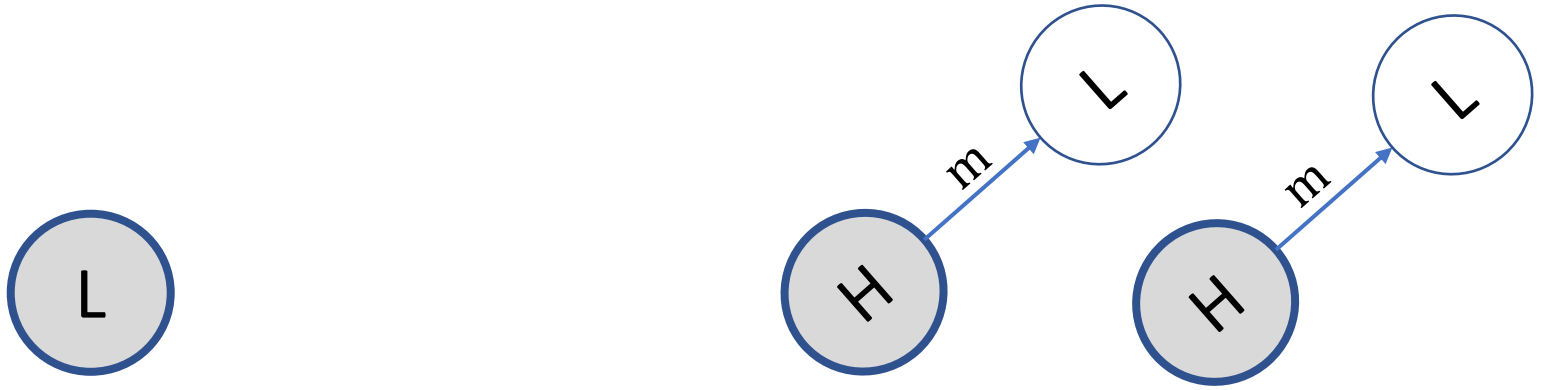
RIF automaton for a vote



$L, H \in R$

$L \sqsubseteq_R H$

RIF automata: Confidentiality example



$$\texttt{res} := [\texttt{maj}(\texttt{vote}, \texttt{vote}', \ldots)]_m$$

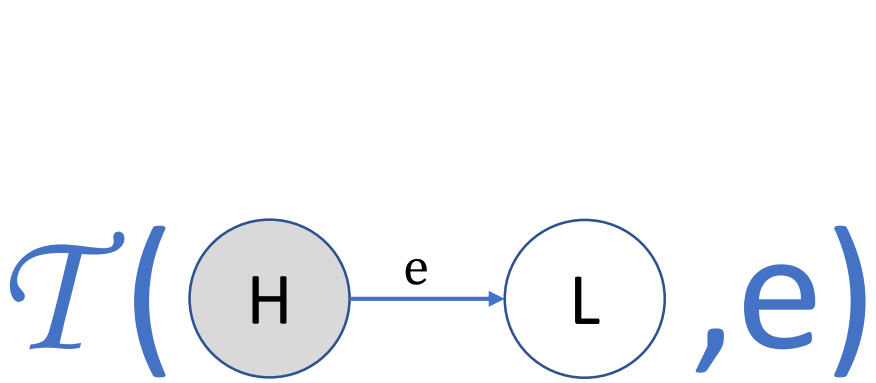
Reclassifier m triggers a declassification.

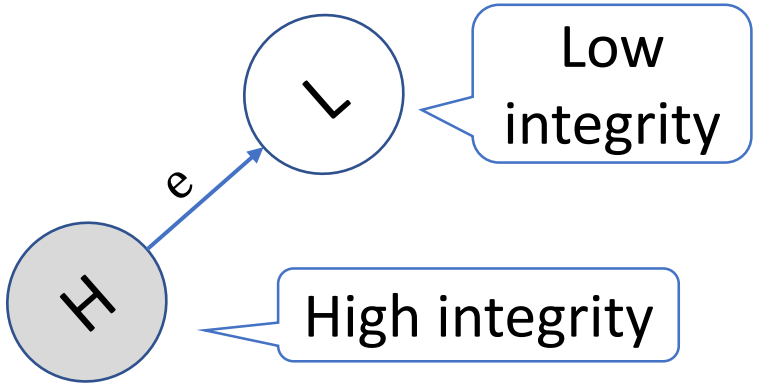
RIF automata

- An instantiation of RIF labels.
- A *RIF automaton* is a finite state automaton whose states correspond to restrictions and transitions correspond to reclassifiers.
- Formally, a RIF automaton λ_a is $\langle Q, F, \delta, q_0, r \rangle$:
  - Q is a finite set of automaton states,
  - F is the finite set of reclassifiers,
  - $\delta: Q \times F \rightarrow Q$ is a (deterministic) next-state transition function,
  - $q_0 \in Q$ is the current state of the RIF automaton,
  - $r: Q \rightarrow R$ gives the restrictions associated with each automaton state.

Function $\mathcal{T}$ for transition

L, H $\in R$
H $\sqsubseteq_R$ L





$$\texttt{doc}' := [\texttt{excerpt}(\texttt{doc})]_{\boldsymbol{e}}$$

Function $\mathcal{T}$ for transition

$$\texttt{doc}' := [\texttt{excerpt}(\texttt{doc})]_{e}$$

Function $\mathcal{R}$ for current restrictions

$$\texttt{doc}' := [\texttt{excerpt}(\texttt{doc})]_{e}$$

Reclassifier e triggers a deprecation.

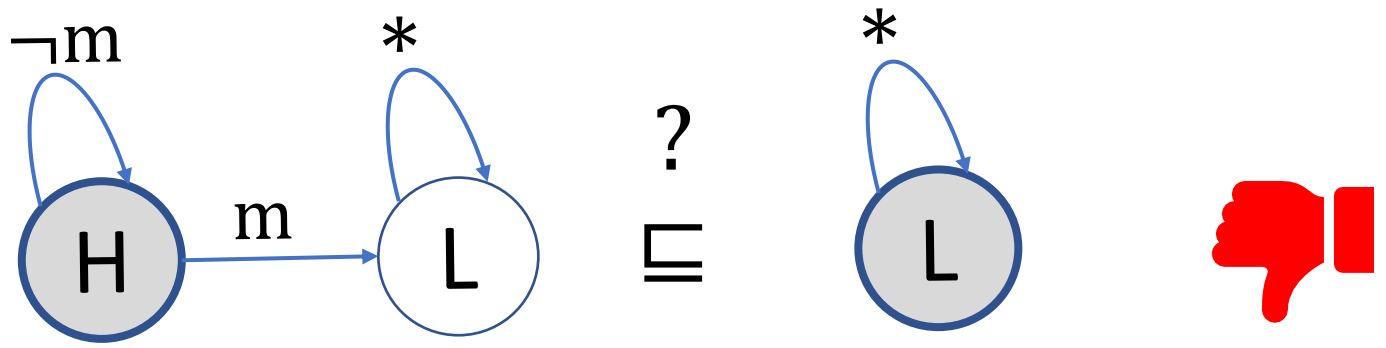
Functions $\mathcal{R}$ and $\mathcal{T}$

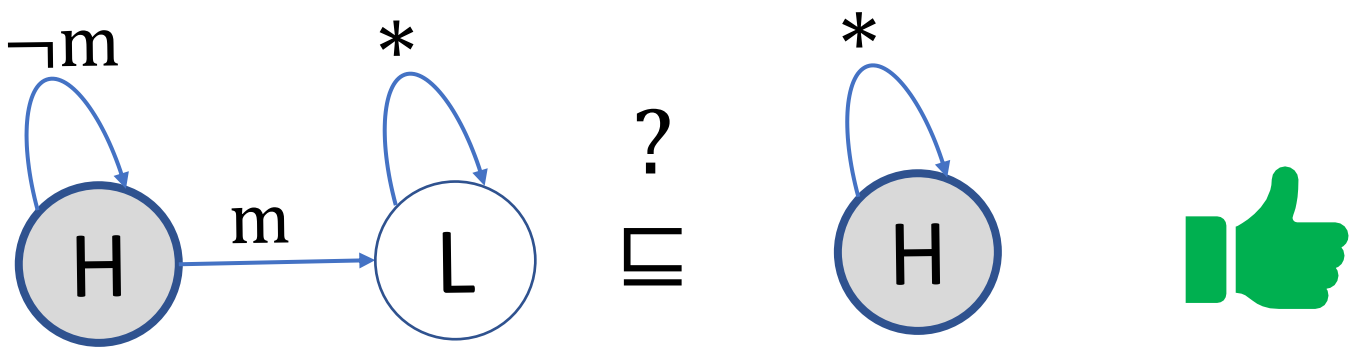
- $\mathcal{R}$ maps $\lambda_a = \langle Q, F, \delta, q_0, r \rangle$ to the restrictions $r \in R$ that λ_a currently imposes:
  - $\mathcal{R}(\lambda_a) \triangleq r(q_0)$
- $\mathcal{T}$ maps $\lambda_a = \langle Q, F, \delta, q_0, r \rangle$ and reclassifier $f \in \mathcal{F}$ to a RIF automaton that should be associated with the value produced by an operation abstracted by f:
  - $\mathcal{T}(\lambda_a) \triangleq \langle Q, F, \delta, \delta(q_0, f), r \rangle$
  - For sequence F of reclassifiers, $\mathcal{T}(\lambda_a, \mathrm{F}f) \triangleq \mathcal{T}(\mathcal{T}(\lambda_a, \mathrm{F}), f)$.

RIF automata form a lattice

$$\lambda \sqsubseteq \lambda' \triangleq \forall \mathrm{F} \in \mathcal{F}^*: \mathcal{R}(\mathcal{T}(\lambda, \mathrm{F})) \sqsubseteq_R \mathcal{R}(\mathcal{T}(\lambda', \mathrm{F}))$$

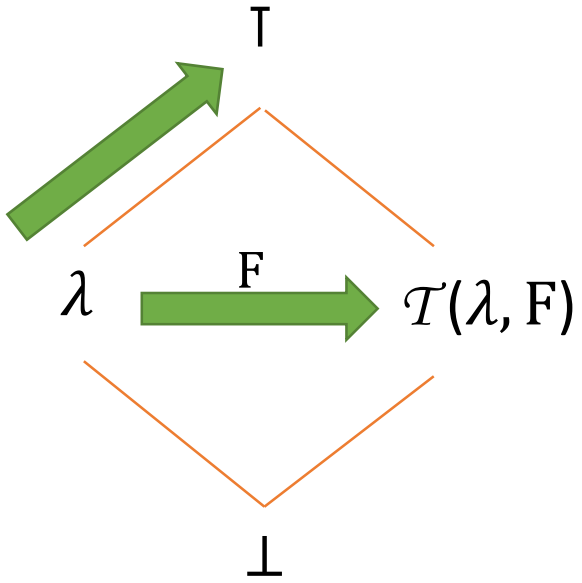
$\mathsf{L}, \mathsf{H} \in R$
$\mathsf{L} \sqsubseteq_R \mathsf{H}$





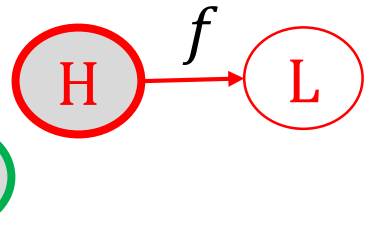
RIF automata form a lattice

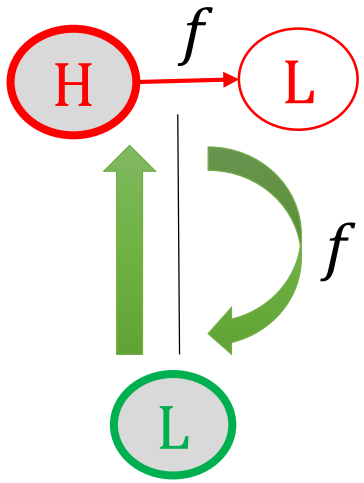
$$\lambda \sqsubseteq \lambda' \triangleq \forall \mathrm{F} \in \mathcal{F}^*: \mathcal{R}(\mathcal{T}(\lambda, \mathrm{F})) \sqsubseteq_R \mathcal{R}(\mathcal{T}(\lambda', \mathrm{F}))$$



When is a flow allowed?

Assume variable y is tagged with H and variable x is tagged with L





☺ `y := x`

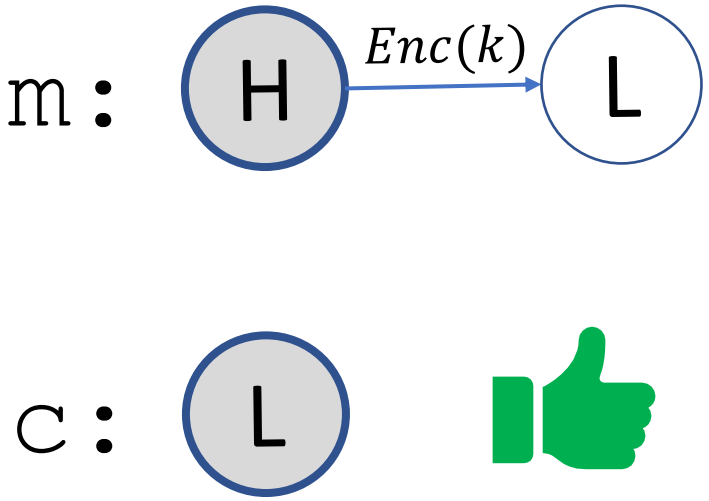
😯 `x := y mod 2`

☺ $x := [y \bmod 2]_f$

Can RIF automata handle the encryption example?

$\mathsf{L}, \mathsf{H} \in R$
$\mathsf{L} \sqsubseteq_R \mathsf{H}$

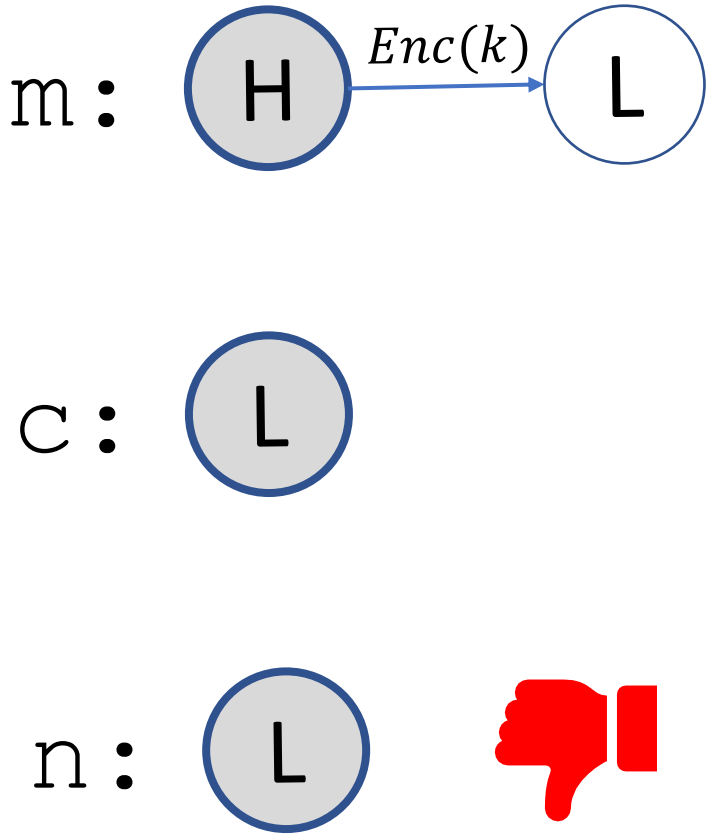
$\mathtt{c} := [\mathtt{Enc}(\mathtt{m}, \mathtt{k})]_{Enc(k)}$



Can RIF automata handle the encryption example?

$\texttt{c} := [\texttt{Enc(m, k)}]_{Enc(k)}$

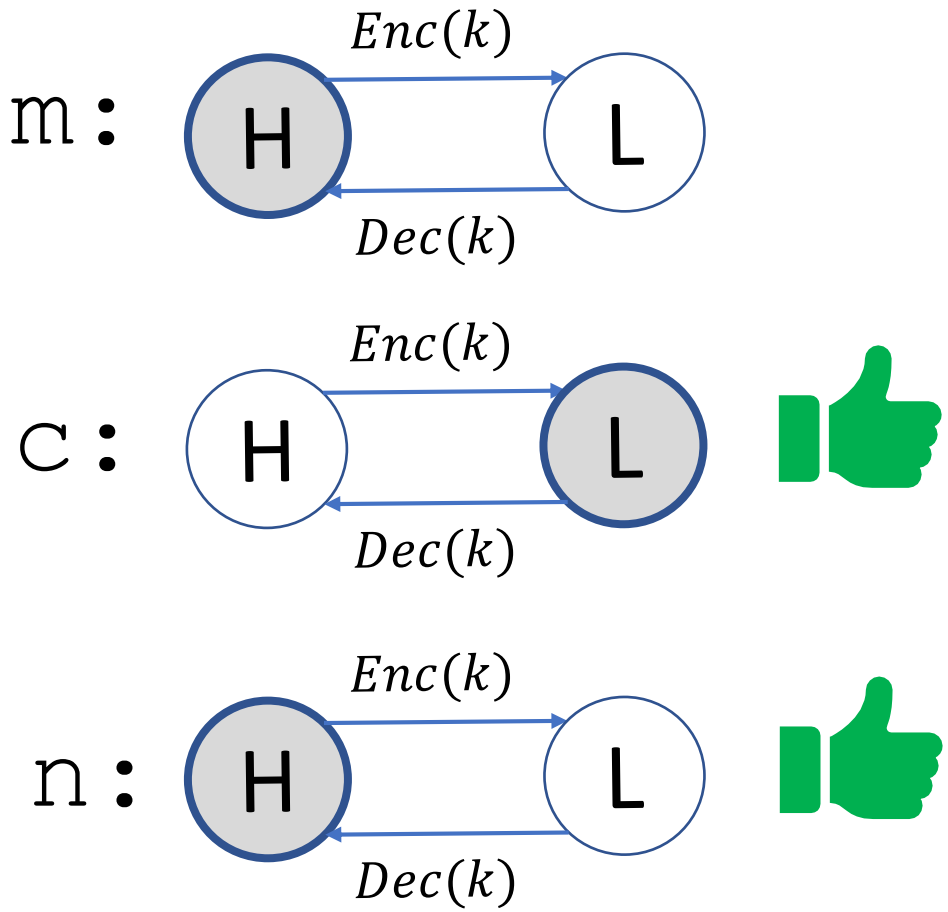
$\texttt{n} := [\texttt{Dec(c, k)}]_{Dec(k)}$



Can RIF automata handle the encryption example?

$c := [\texttt{Enc(m,k)}]_{Enc(k)}$

$n := [\texttt{Dec(c,k)}]_{Dec(k)}$

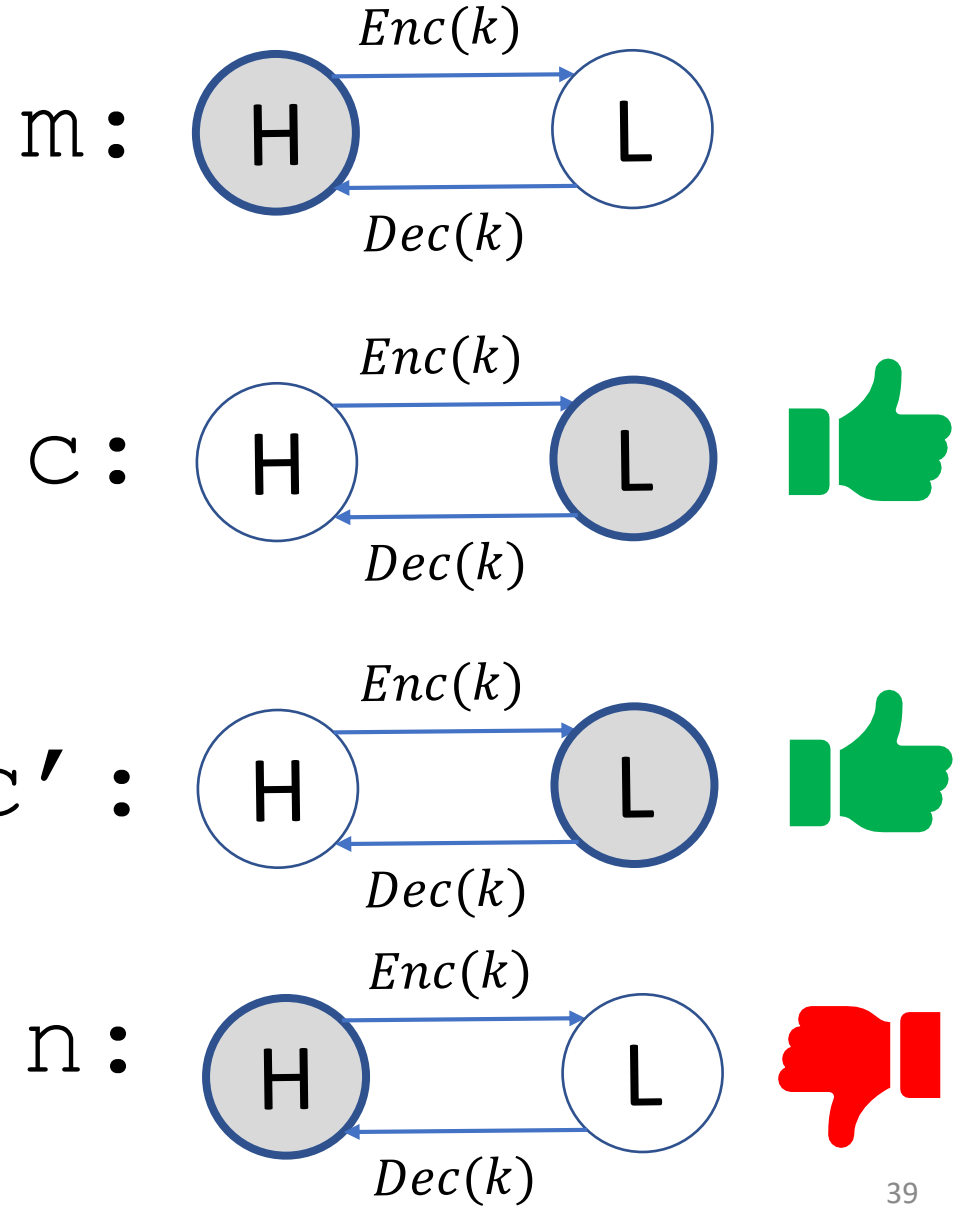


Can RIF automata handle the encryption example?

$c := [\mathtt{Enc}(m, k)]_{Enc(k)}$

$c' := [\mathtt{Enc}(c, k)]_{Enc(k)}$

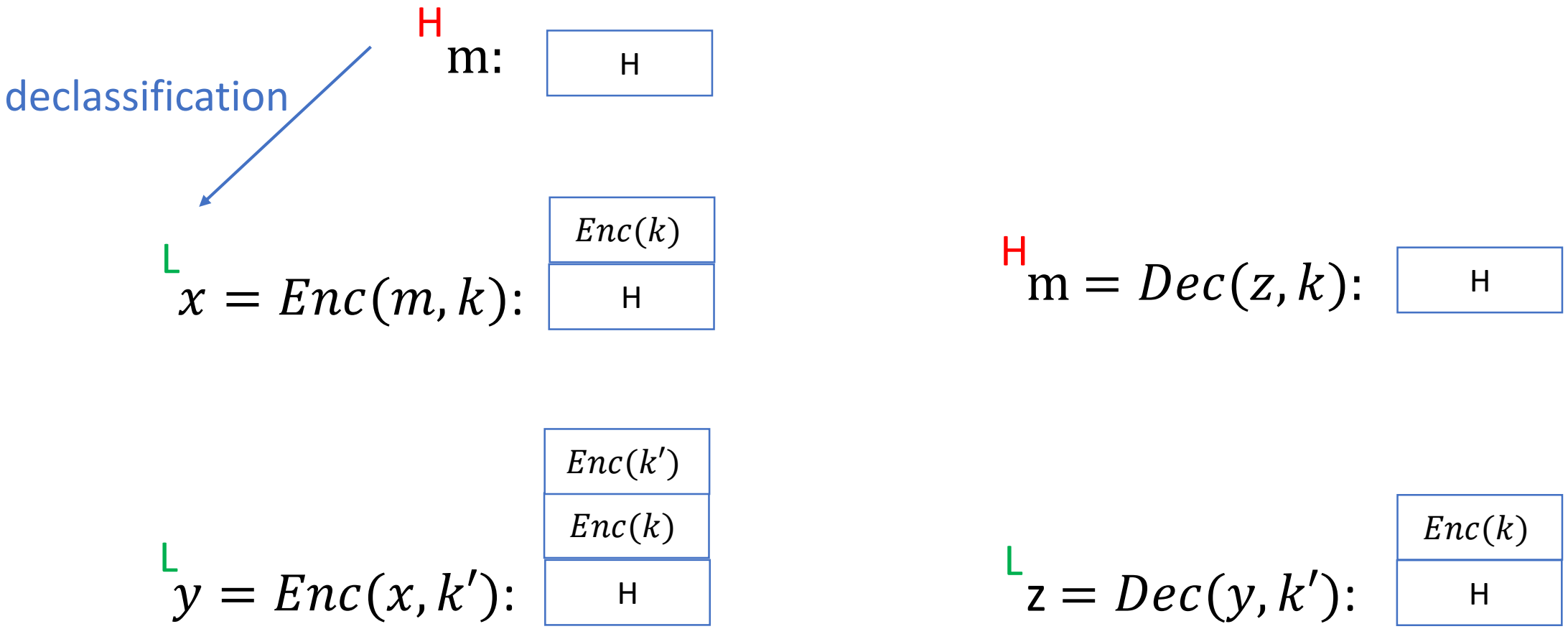
$n := [\mathtt{Dec}(c', k)]_{Dec(k)}$



Can RIF automata handle the encryption example?

- Examples:
  - m is H
  - $\mathrm{Enc}(\mathrm{Enc}(\mathrm{m},\mathrm{k}),\mathrm{k})$ is L
  - $\mathrm{Dec}(\mathrm{Enc}(\mathrm{Enc}(\mathrm{m},\mathrm{k}),\mathrm{k}),\mathrm{k})$ is L
  - $\mathrm{Dec}(\mathrm{Dec}(\mathrm{Enc}(\mathrm{Enc}(\mathrm{m},\mathrm{k}),\mathrm{k}),\mathrm{k}),\mathrm{k})$ is H
- If the number of consecutive Dec equals the number of consecutive Enc, then the restriction is H; otherwise the restriction is L.
- RIF automata cannot be used to count an unbounded number of applied cryptographic operations.

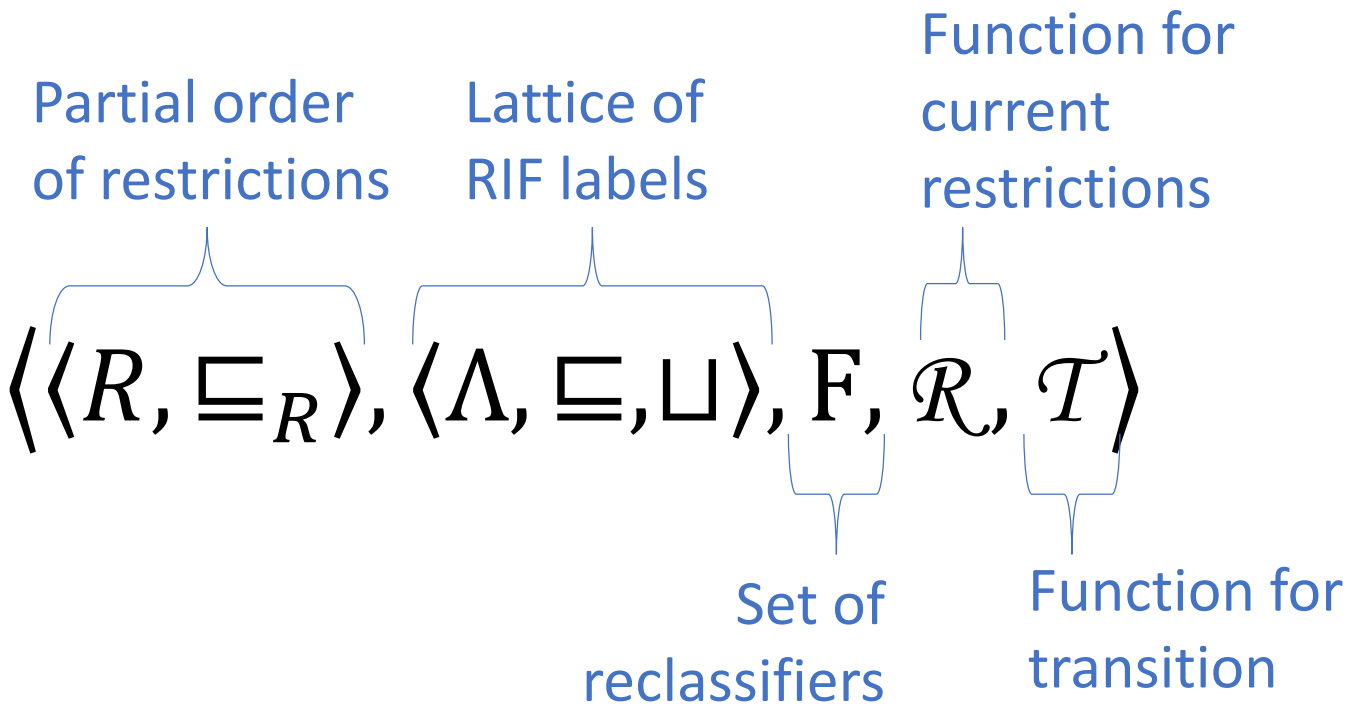
κ-labels


H
m:
H
declassification
L
$x = Enc(m, k)$:
$Enc(k)$
H
H
m = $Dec(z, k)$:
H
L
$y = Enc(x, k')$:
$Enc(k')$
$Enc(k)$
H
L
z = $Dec(y, k')$:
$Enc(k)$
H

κ-labels

- A κ-label λ_κ is a stack (i.e., a simplifies push-down automaton).
- Assume restrictions $\langle \{\mathsf{H}, \mathsf{L}\}, \sqsubseteq_R \rangle$.
- $\mathcal{R}(\lambda_\kappa) \triangleq$
  - H, if λ_κ is empty (stores no reclassifier)
  - L, otherwise.
- $\mathcal{T}(\lambda_\kappa, f) \triangleq$
  - pop(λ_κ), if f is the inverse of the top element in λ_κ
  - push(λ_κ, f), otherwise.

A class of RIF labels



RIF labels

$v_1 := [op(v_0)]_{f_1}$ $v_n := [op(v_{n-1})]_{f_n}$