Copy Right

- Allows possessor to give rights to another
- Often attached to a right, so only applies to that right
  - *r* is read right that cannot be copied
  - *rc* is read right that can be copied
- Is copy flag copied when giving *r* rights?
  - Depends on model, instantiation of model

Own Right

- Usually allows possessor to change entries in ACM column
  - So owner of object can add, delete rights for others
  - May depend on what system allows
    - Can't give rights to specific (set of) users
    - Can't pass copy flag to specific (set of) users

Attenuation of Privilege

- Principle says you can’t give rights you do not possess
  - Restricts addition of rights within a system
  - Usually *ignored* for owner
    - Why? Owner gives herself rights, gives them to others, deletes her rights.

Foundational Results

- Overview
- Harrison-Ruzzo-Ullman result
  - Corollaries
- Take-Grant Protection Model
- SPM and successors

Overview

- Safety Question
- HRU Model
- Take-Grant Protection Model
- SPM, ESPM
  - Multiparent joint creation
- Expressive power
- Typed Access Matrix Model

What Is "Secure"?

- Adding a generic right r where there was not one is "leaking"
- If a system S, beginning in initial state s_0, cannot leak right r, it is *safe with respect to the right r*.

Safety Question

- Does there exist an algorithm for determining whether a protection system S with initial state s_0 is safe with respect to a generic right r?
  - Here, “safe” = “secure” for an abstract model

Mono-Operational Commands

- Answer: *yes*
- Sketch of proof:

  Consider minimal sequence of commands $c_1, \ldots, c_k$ to leak the right.

  – Can omit **delete**, **destroy**

  – Can merge all **create**s into one

  Worst case: insert every right into every entry; with s subjects and o objects initially, and n rights, upper bound is $k \le n(s+1)(o+1)$

General Case

- Answer: *no*
- Sketch of proof:

  Reduce halting problem to safety problem

  Turing Machine review:

  – Infinite tape in one direction
  – States K, symbols M; distinguished blank b
  – Transition function $\delta(k, m) = (k', m', \mathrm{L})$ means in state k, symbol m on tape location replaced by symbol m', head moves to left one square, and enters state k'
  – Halting state is q_f; TM halts when it enters this state

Mapping

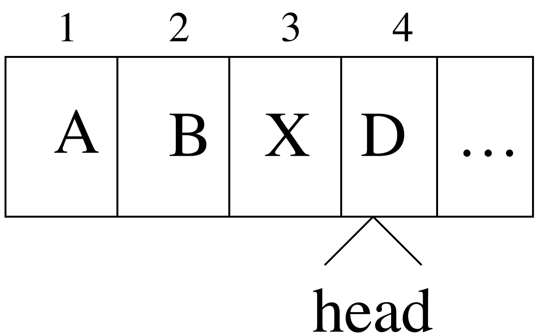
1	2	3	4	
A	B	C	D	…

head

Current state is k

	s_1	s_2	s_3	s_4	
s_1	A	*own*			
s_2		B	*own*		
s_3			C k	*own*	
s_4				D end	

Mapping



	s_1	s_2	s_3	s_4	
s_1	A	*own*			
s_2		B	*own*		
s_3			X	*own*	
s_4				D k_1 end	

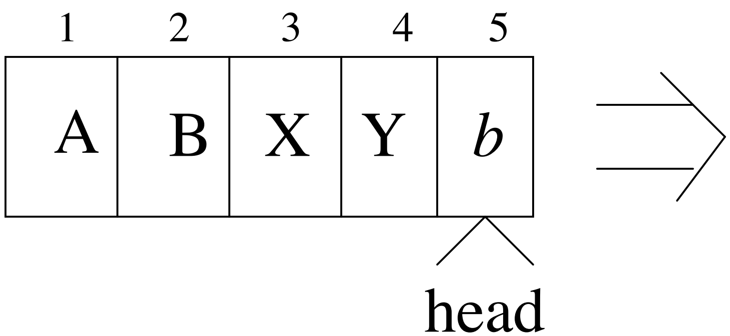
After $\delta(k, C) = (k_1, X, R)$ where k is the current state and k_1 the next state

Command Mapping

$\delta(k, \mathrm{C}) = (k_1, \mathrm{X}, \mathrm{R})$ at intermediate becomes

command $\mathtt{c}_{k,\mathtt{c}}(s_3, s_4)$
if *own* **in** $A[s_3, s_4]$ **and** k **in** $A[s_3, s_3]$
      and C **in** $A[s_3, s_3]$
then
  delete k **from** $A[s_3, s_3]$;
  delete C **from** $A[s_3, s_3]$;
  enter X **into** $A[s_3, s_3]$;
  enter k_1 **into** $A[s_4, s_4]$;
end

Mapping



	s_1	s_2	s_3	s_4	s_5
s_1	A	*own*			
s_2		B	*own*		
s_3			X	*own*	
s_4				Y	*own*
s_5					b k_2 end

After $\delta(k_1, \mathrm{D}) = (k_2, \mathrm{Y}, \mathrm{R})$ where k_1 is the current state and k_2 the next state

Command Mapping

$\delta(k_1, \mathrm{D}) = (k_2, \mathrm{Y}, \mathrm{R})$ at end becomes

command crightmost$_{k,\mathrm{c}}$(s_4, s_5)
if *end* **in** $A[s_4, s_4]$ **and** k_1 **in** $A[s_4, s_4]$
    and D **in** $A[s_4, s_4]$
then
  delete *end* **from** $A[s_4, s_4]$;
  create subject s_5;
  enter *own* **into** $A[s_4, s_5]$;
  enter *end* **into** $A[s_5, s_5]$;
  delete k_1 **from** $A[s_4, s_4]$;
  delete D **from** $A[s_4, s_4]$;
  enter Y **into** $A[s_4, s_4]$;
  enter k_2 **into** $A[s_5, s_5]$;
end

Rest of Proof

- Protection system exactly simulates a TM
  - Exactly 1 *end* right in ACM
  - 1 right in entries corresponds to state
  - Thus, at most 1 applicable command
- If TM enters state q_f, then right has leaked
- If safety question decidable, then represent TM as above and determine if q_f leaks
  - Implies halting problem decidable
- Conclusion: safety question undecidable

Other Results

- Set of unsafe systems is recursively enumerable
- Delete **create** primitive; then safety question is complete in **P-SPACE**
- Delete **destroy**, **delete** primitives; then safety question is undecidable
  - Systems are monotonic
- Safety question for monoconditional, monotonic protection systems is decidable
- Safety question for monoconditional protection systems with **create**, **enter**, **delete** (and no **destroy**) is decidable.

Take-Grant Protection Model

- A specific (not generic) system
  - Set of rules for state transitions
- Safety decidable, and in time linear with the size of the system
- Goal: find conditions under which rights can be transferred from one entity to another in the system

System

- ○ objects (files, …)
- ● subjects (users, processes, …)
- ⊗ don't care (either a subject or an object)

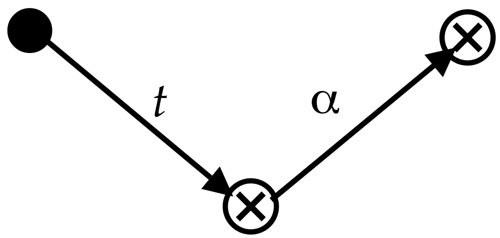
$G \vdash_x G'$ apply a rewriting rule x (witness) to G to get G'

$G \vdash^* G'$ apply a sequence of rewriting rules (witness) to G to get G'

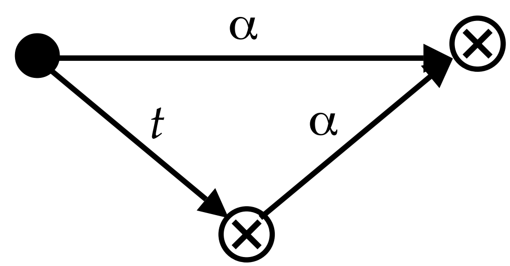
$R = \{ t, g, r, w, \ldots \}$ set of rights

Rules

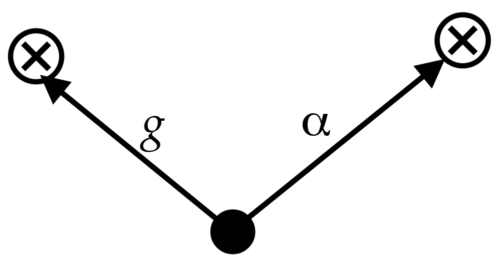
take



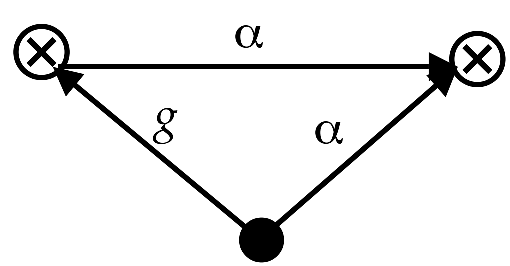
|-



grant



|-



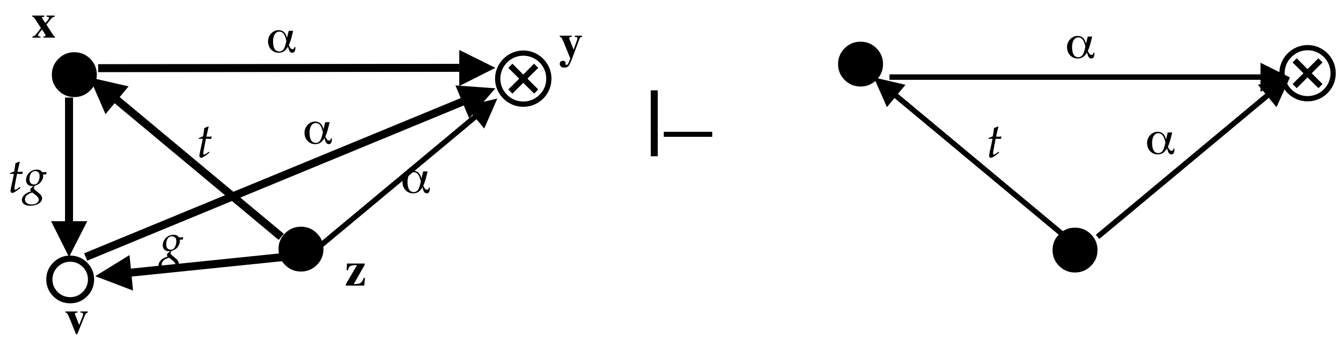
More Rules

create $\bullet \quad \vdash \quad \bullet \xrightarrow{\alpha} \otimes$

remove $\bullet \xrightarrow{\alpha} \otimes \quad \vdash \quad \bullet \xrightarrow{\alpha - \beta} \otimes$

These four rules are called the *de jure* rules

Symmetry



1. **x** creates (*tg* to new) **v**
2. **z** takes (*g* to **v**) from **x**
3. **z** grants (α to **y**) to **v**
4. **x** takes (α to **y**) from **v**

Similar result for grant

Islands

- *tg*-path: path of distinct vertices connected by edges labeled *t* or *g*
  – Call them “tg-connected”
- island: maximal *tg*-connected subject-only subgraph
  – Any right one vertex has can be shared with any other vertex

Initial, Terminal Spans

- *initial span* from **x** to **y**
  - **x** subject
  - *tg*-path between **x**, **y** with word in { $\overrightarrow{t^*}\overrightarrow{g}$ } ∪ { ν }
  - Means **x** can give rights it has to **y**
- *terminal span* from **x** to **y**
  - **x** subject
  - *tg*-path between **x**, **y** with word in { $\overrightarrow{t^*}$ } ∪ { ν }
  - Means **x** can acquire any rights **y** has