

# ECS 235B Module 9

## HRU Result

# What Is “Secure”?

- Adding a generic right  $r$  where there was not one is “leaking”
  - In what follows, a right leaks if it was not present *initially*
  - Alternately: not present *in the previous state* (not discussed here)
- If a system  $S$ , beginning in initial state  $s_0$ , cannot leak right  $r$ , it is *safe with respect to the right  $r$* 
  - Otherwise it is called *unsafe with respect to the right  $r$*

# Safety Question

- Is there 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  $c_1, \dots, c_k$  to leak the right.

- Can omit **delete**, **destroy** (with some rewriting)

- Can merge all **creates** into one

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

# General Case

- Answer: *no*

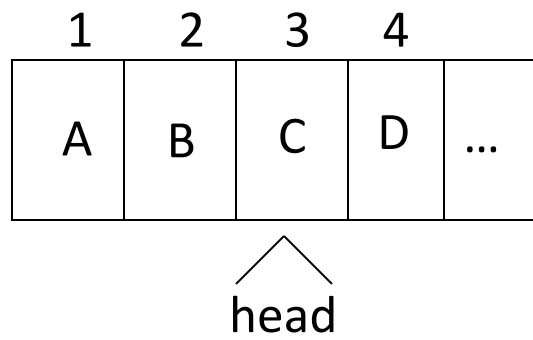
- Sketch of proof:

Reduce halting problem to safety problem

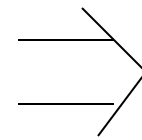
Turing Machine review:

- Infinite tape in one direction
- States  $K$ , symbols  $M$ ; distinguished blank  $b$
- Transition function  $\delta(k, m) = (k', m', L)$  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

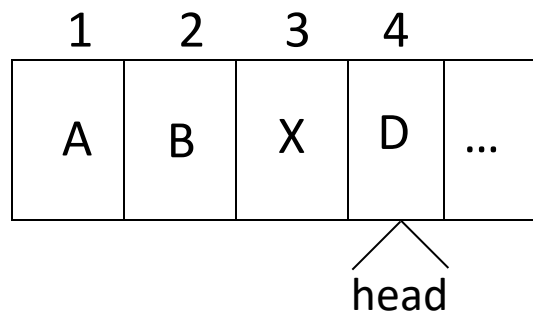


Current state is  $k$



	$s_1$	$s_2$	$s_3$	$s_4$	
$s_1$	A	<i>own</i>			
$s_2$		B	<i>own</i>		
$s_3$			C $k$	<i>own</i>	
$s_4$				D <i>end</i>	

# Mapping



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

A transition table with 6 columns and 6 rows. The first row contains labels  $s_1, s_2, s_3, s_4$  in the second through fifth columns. The first column contains labels  $s_1, s_2, s_3, s_4$  in the second through fifth rows. The table content is as follows:

	$s_1$	$s_2$	$s_3$	$s_4$	
$s_1$	A	<i>own</i>			
$s_2$		B	<i>own</i>		
$s_3$			X	<i>own</i>	
$s_4$				D $k_1$ <i>end</i>	

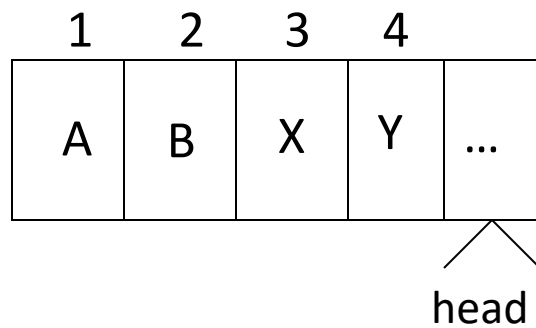
# Command Mapping

- $\delta(k, C) = (k_1, X, R)$  at intermediate becomes

```
command  $c_{k,C}(s_3, s_4)$   
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



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

	$s_1$	$s_2$	$s_3$	$s_4$	$s_5$
$s_1$	A	<i>own</i>			
$s_2$		B	<i>own</i>		
$s_3$			X	<i>own</i>	
$s_4$				Y	<i>own</i>
$s_5$					<i>b k_2 end</i>

# Command Mapping

- $\delta(k_1, D) = (k_2, Y, R)$  at end becomes

```
command crightmostk,c(s4, s5)  
if end in A[s4, s4] and k1 in A[s4, s4]  
    and D in A[s4, s4]  
then  
    delete end from A[s4, s4];  
    delete k1 from A[s4, s4];  
    delete D from A[s4, s4];  
    enter Y into A[s4, s4];  
    create subject s5;  
    enter own into A[s4, s5];  
    enter end into A[s5, s5];  
    enter k2 into A[s5, s5];  
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 biconditional protection systems is decidable
- Safety question for monoconditional, monotonic protection systems is decidable
- Safety question for monoconditional protection systems with **create, enter, delete** (and no **destroy**) is decidable.

# Quiz

The Harrison-Ruzzo-Ullman result says that the security question is undecidable. But it is also said we can determine whether a Linux system is secure for a given security policy. How would you resolve this apparent contradiction?

- The Linux claim is false for all security policies, just as the HRU result says.
- The Linux system is more general than the system in the HRU model, so the HRU result does not apply.
- There is no contradiction, as security policies for Linux systems are not so general as the security policy in the HRU model, and Linux systems are not as general as the system used in the HRU model.