Chapter 3: Foundational Results

- Overview
- Harrison-Ruzzo-Ullman result
  - Corollaries

Overview

- Safety Question
- HRU 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, …, 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 $k \leq n(s+1)(o+1)$

General Case

- Answer: *no*
- Sketch of proof:

  Reduce halting problem to safety problem

  Turing Machine review:

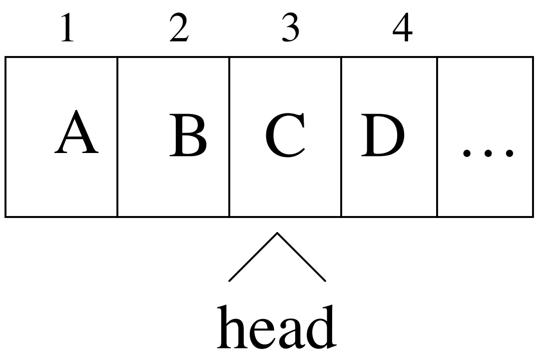
  – Infinite tape in one direction

  – States K, symbols M; distinguished blank b

  – Transition function $\delta(k, m) = (k', m', \text{L})$ 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

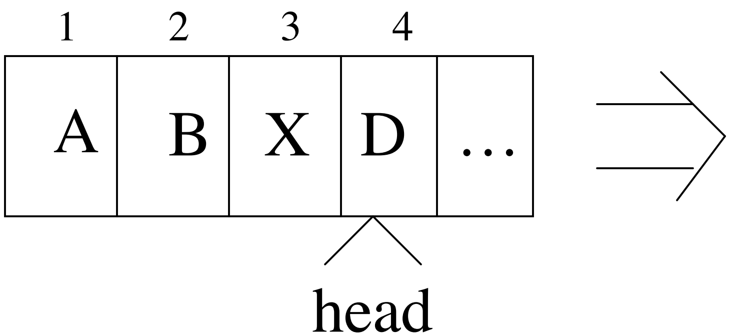
Mapping



Current state is *k*

	s_1	s_2	s_3	s_4	
s_1	A	*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 $\delta(k, \text{C}) = (k_1, \text{X}, \text{R})$ 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** $A[s_3, s_3]$;

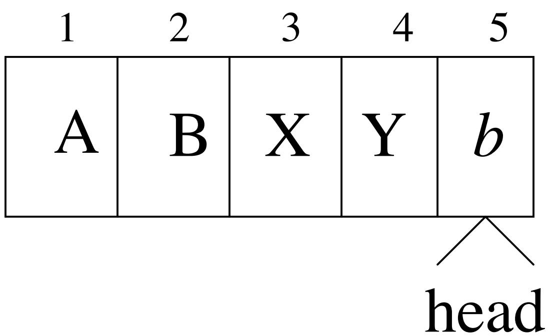
  delete C **from** $A[s_3, s_3]$;

  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 $\delta(k_1, \text{D}) = (k_2, \text{Y}, \text{R})$ 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,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.

Key Points

- Safety problem undecidable
- Limiting scope of systems can make problem decidable