ECS 235B Module 35
Introduction to Noninterference

Interference

- Think of it as something used in communication
  - Holly/Lara example: Holly interferes with the CPU utilization, and Lara detects it — communication
- Plays role of writing (interfering) and reading (detecting the interference)

Model

- System as state machine
  - Subjects $S = \{ s_i \}$
  - States $\Sigma = \{ \sigma_i \}$
  - Outputs $O = \{ o_i \}$
  - Commands $Z = \{ z_i \}$
  - State transition commands $C = S \times Z$
- Note: no inputs
  - Encode either as selection of commands or in state transition commands

Functions

- State transition function *T*: $C \times \Sigma \to \Sigma$
  - Describes effect of executing command *c* in state σ
- Output function *P*: $C \times \Sigma \to O$
  - Output of machine when executing command *c* in state σ
- Initial state is σ_0

Example: 2-Bit Machine

- Users Heidi (high), Lucy (low)
- 2 bits of state, *H* (high) and *L* (low)
  - System state is (*H*, *L*) where *H*, *L* are 0, 1
- 2 commands: *xor0*, *xor1* do *xor* with 0, 1
  - Operations affect *both* state bits regardless of whether Heidi or Lucy issues it

Example: 2-bit Machine

- S = { Heidi, Lucy }
- Σ = { (0,0), (0,1), (1,0), (1,1) }
- C = { *xor0*, *xor1* }

	Input States (H, L)			
	(0,0)	(0,1)	(1,0)	(1,1)
xor0	(0,0)	(0,1)	(1,0)	(1,1)
xor1	(1,1)	(1,0)	(0,1)	(0,0)

Outputs and States

- T is inductive in first argument, as

  $T(c_0, \sigma_0) = \sigma_1$; $T(c_{i+1}, \sigma_{i+1}) = T(c_{i+1}, \mathrm{T}(c_i, \sigma_i))$
- Let C^* be set of possible sequences of commands in C
- T^*: $C^* \times \Sigma \rightarrow \Sigma$ and

  $c_s = c_0 \ldots c_n \Rightarrow T^*(c_s, \sigma_i) = T(c_n, \ldots, T(c_0, \sigma_i) \ldots)$
- P similar; define P^*: $C^* \times \Sigma \rightarrow O$ similarly

Projection

- $T^*(c_s,\sigma_i)$ sequence of state transitions
- $P^*(c_s,\sigma_i)$ corresponding outputs
- *proj*(s, c_s, σ_i) set of outputs in $P^*(c_s,\sigma_i)$ that subject s authorized to see
  - In same order as they occur in $P^*(c_s,\sigma_i)$
  - Projection of outputs for s
- Intuition: list of outputs after removing outputs that s cannot see

Purge

- $G \subseteq S$, G a group of subjects
- $A \subseteq Z$, A a set of commands
- $\pi_G(c_s)$ subsequence of c_s with all elements (s,z), $s \in G$ deleted
- $\pi_A(c_s)$ subsequence of c_s with all elements (s,z), $z \in A$ deleted
- $\pi_{G,A}(c_s)$ subsequence of c_s with all elements (s,z), $s \in G$ and $z \in A$ deleted

Example: 2-bit Machine

- Let $\sigma_0 = (0,1)$
- 3 commands applied:
  - Heidi applies *xor0*
  - Lucy applies *xor1*
  - Heidi applies *xor1*
- c_s = ((Heidi, *xor0*), (Lucy, *xor1*), (Heidi, *xor1*))
- Output is 011001
  - Shorthand for sequence (0,1) (1,0) (0,1)

Example

- $proj(\text{Heidi}, c_s, \sigma_0) = 011001$
- $proj(\text{Lucy}, c_s, \sigma_0) = 101$
- $\pi_{\text{Lucy}}(c_s) = (\text{Heidi}, \mathit{xor0}), (\text{Heidi}, \mathit{xor1})$
- $\pi_{\text{Lucy},\mathit{xor1}}(c_s) = (\text{Heidi}, \mathit{xor0}), (\text{Heidi}, \mathit{xor1})$
- $\pi_{\text{Heidi}}(c_s) = (\text{Lucy}, \mathit{xor1})$
- $\pi_{\text{Lucy},\mathit{xor0}}(c_s) = (\text{Heidi}, \mathit{xor0}), (\text{Lucy}, \mathit{xor1}), (\text{Heidi}, \mathit{xor1})$
- $\pi_{\text{Heidi},\mathit{xor0}}(c_s) = \pi_{\mathit{xor0}}(c_s) = (\text{Lucy}, \mathit{xor1}), (\text{Heidi}, \mathit{xor1})$
- $\pi_{\text{Heidi},\mathit{xor1}}(c_s) = (\text{Heidi}, \mathit{xor0}), (\text{Lucy}, \mathit{xor1})$
- $\pi_{\mathit{xor1}}(c_s) = (\text{Heidi}, \mathit{xor0})$

Noninterference

- Intuition: If set of outputs Lucy can see corresponds to set of inputs she can see, there is no interference
- Formally: G, $G' \subseteq$ S, $G \neq G'$; $A \subseteq Z$; users in G executing commands in A are *noninterfering* with users in G' iff for all $c_s \in C^*$, and for all $s \in G'$,

$$proj(s, c_s, \sigma_i) = proj(s, \pi_{G,A}(c_s), \sigma_i)$$

  - Written A,G :| G'

Example: 2-Bit Machine

- Let c_s = ((Heidi, *xor0*), (Lucy, *xor1*), (Heidi, *xor1*)) and $\sigma_0 = (0, 1)$
  - As before
- Take G = { Heidi }, G' = { Lucy }, $A = \varnothing$
- $\pi_{\text{Heidi}}(c_s)$ = (Lucy, *xor1*)
  - So *proj*(Lucy, $\pi_{\text{Heidi}}(c_s)$, σ_0) = 0
- *proj*(Lucy, c_s, σ_0) = 101
- So { Heidi } :| { Lucy } is false
  - Makes sense; commands issued to change *H* bit also affect *L* bit

Example

- Same as before, but Heidi's commands affect *H* bit only, Lucy's the *L* bit only
- Output is $0_H 0_L 1_H$
- $\pi_{\text{Heidi}}(c_s)$ = (Lucy, *xor1*)
  - So $proj(\text{Lucy}, \pi_{\text{Heidi}}(c_s), \sigma_0) = 0$
- $proj(\text{Lucy}, c_s, \sigma_0) = 0$
- So { Heidi } :| { Lucy } is true
  - Makes sense; commands issued to change *H* bit now do not affect *L* bit