Lecture 12

- Policies that change over time
- Policy composition
- Deducible security
- Generalized noninterference
- Restrictiveness
- Information flow
- Entropy

Policies Changing Over Time

- Problem: previous analysis assumes static system
  - In real life, ACM changes as system commands issued
- Example: $w \in C^*$ leads to current state
  - $cando(w, s, z)$ holds if s can execute z in current state
  - Condition noninterference on *cando*
  - If $\neg cando(w, \text{Lara}, \text{“write } f\text{”})$, Lara can’t interfere with any other user by writing file f

Generalize Noninterference

- $G \subseteq S$ group of subjects, $A \subseteq Z$ set of commands, p predicate over elements of C^*
- $c_s = (c_1, \ldots, c_n) \in C^*$
- $\pi''(\nu) = \nu$
- $\pi''((c_1, \ldots, c_n)) = (c_1', \ldots, c_n')$
  - $c_i' = \nu$ if $p(c_1', \ldots, c_{i-1}')$ and $c_i = (s, z)$ with $s \in G$ and $z \in A$
  - $c_i' = c_i$ otherwise

Intuition

- $\pi''(c_s) = c_s$
- But if p holds, and element of c_s involves both command in A and subject in G, replace corresponding element of c_s with empty command ν
  - Just like deleting entries from c_s as $\pi_{A,G}$ does earlier

Noninterference

- $G, G' \subseteq S$ groups of subjects, $A \subseteq Z$ set of commands, p predicate over C^*
- Users in G executing commands in A are noninterfering with users in G' under condition p iff, for all $c_s \in C^*$, all $s \in G'$, $proj(s, c_s, \sigma_i) = proj(s, \pi''(c_s), \sigma_i)$
  - Written $A,G :| G'$ **if** p

Example

- From earlier one, simple security policy based on noninterference:

  $\forall(s \in S)\ \forall(z \in Z)$

  $$[\ \{z\}, \{s\} :\mid S \textbf{ if } \neg cando(w, s, z)\ ]$$

- If subject can't execute command (the $\neg$*cando* part), subject can't use that command to interfere with another subject

Another Example

- Consider system in which rights can be passed
  - $pass(s, z)$ gives s right to execute z
  - $w_n = v_1, \ldots, v_n$ sequence of $v_i \in C^*$
  - $prev(w_n) = w_{n-1}$; $\text{last}(w_n) = v_n$

Policy

- No subject s can use z to interfere if, in previous state, s did not have right to z, and no subject gave it to s

$\{ z \}, \{ s \}$:| S **if**

$$[\ \neg cando(prev(w), s, z) \wedge$$
$$[\ cando(prev(w), s', pass(s, z)) \Rightarrow$$
$$\neg last(w) = (s', pass(s, z))\ ]\ ]$$

Effect

- Suppose $s_1 \in S$ can execute $pass(s_2, z)$
- For all $w \in C^*$, $cando(w, s_1, pass(s_2, z))$ true
- Initially, $cando(\nu, s_2, z)$ false
- Let $z' \in Z$ be such that (s_3, z') noninterfering with (s_2, z)
  - So for each w_n with $v_n = (s_3, z')$,

$$cando(w_n, s_2, z) = cando(w_{n-1}, s_2, z)$$

Effect

- Then policy says for all $s \in S$

$$proj(s, ((s_2, z), (s_1, pass(s_2, z)), (s_3, z'), (s_2, z)), \sigma_i)$$
$$= proj(s, ((s_1, pass(s_2, z)), (s_3, z'), (s_2, \mathrm{z})), \sigma_i)$$

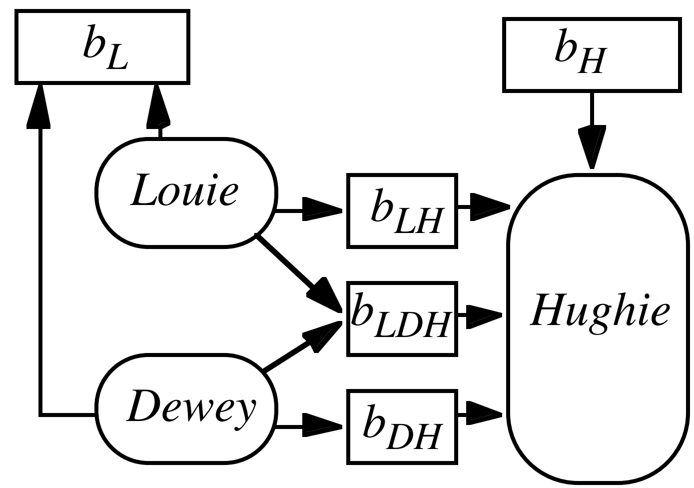
- So s_2's first execution of z does not affect any subject's observation of system

Policy Composition I

- Assumed: Output function of input
  - Means deterministic (else not function)
  - Means uninterruptability (differences in timings can cause differences in states, hence in outputs)
- This result for deterministic, noninterference-secure systems

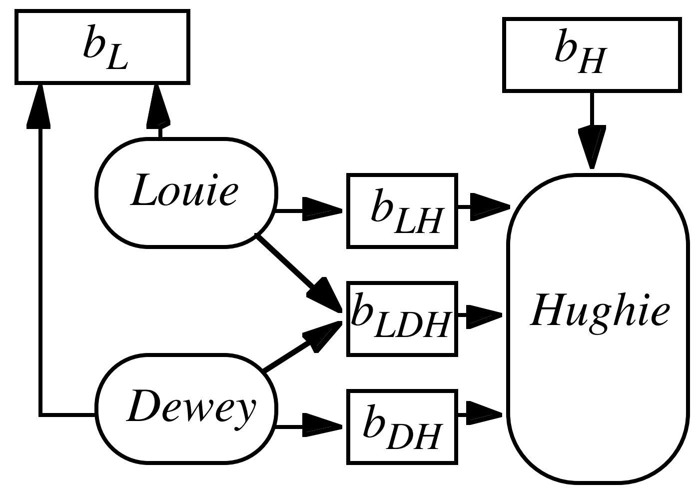
Compose Systems

- Louie, Dewey LOW
- Hughie HIGH
- b_L output buffer
  - Anyone can read it
- b_H input buffer
  - From HIGH source
- Hughie reads from:
  - b_{LH} (Louie writes)
  - b_{LDH} (Louie, Dewey write)
  - b_{DH} (Dewey writes)



Systems Secure

- All noninterference-secure
  - Hughie has no output
    - So inputs don't interfere with it
  - Louie, Dewey have no input
    - So (nonexistent) inputs don't interfere with outputs



Security of Composition

- Buffers finite, sends/receives blocking: composition *not* secure!
  - Example: assume b_{DH}, b_{LH} have capacity 1
- Algorithm:
  1. Louie (Dewey) sends message to b_{LH} (b_{DH})
     - Fills buffer
  2. Louie (Dewey) sends second message to b_{LH} (b_{DH})
  3. Louie (Dewey) sends a 0 (1) to b_L
  4. Louie (Dewey) sends message to b_{LDH}
     - Signals Hughie that Louie (Dewey) completed a cycle

Hughie

- Reads bit from b_H
  - If 0, receive message from b_{LH}
  - If 1, receive message from b_{DH}
- Receive on b_{LDH}
  - To wait for buffer to be filled

Example

- Hughie reads 0 from b_H
  - Reads message from b_{LH}
- Now Louie's second message goes into b_{LH}
  - Louie completes setp 2 and writes 0 into b_L
- Dewey blocked at step 1
  - Dewey cannot write to b_L
- Symmetric argument shows that Hughie reading 1 produces a 1 in b_L
- So, input from b_H copied to output b_L

Nondeducibility

- Noninterference: do state transitions caused by high level commands interfere with sequences of state transitions caused by low level commands?
- Really case about inputs and outputs:
  – Can low level subject deduce *anything* about high level outputs from a set of low level outputs?

Example: 2-Bit System

- *High* operations change only *High* bit
  - Similar for *Low*
- $\sigma_0 = (0, 0)$
- Commands (Heidi, xor_1), (Lara, xor_0), (Lara, xor_1), (Lara, xor_0), (Heidi, xor_1), (Lara, xor_0)
  - Both bits output after each command
- Output is: 00 10 10 11 11 01 01

Security

- Not noninterference-secure w.r.t. Lara
  - Lara sees output as 0001111
  - Delete *High* and she sees 00111
- But Lara still cannot deduce the commands deleted
  - Don't affect values; only lengths
- So it is deducibly secure
  - Lara can't deduce the commands Heidi gave

Event System

- 4-tuple (E, I, O, T)
  - E set of events
  - $I \subseteq E$ set of input events
  - $O \subseteq E$ set of output events
  - T set of all finite sequences of events legal within system
- E partitioned into H, L
  - H set of *High* events
  - L set of *Low* events

More Events …

- $H \cap I$ set of *High* inputs
- $H \cap O$ set of *High* outputs
- $L \cap I$ set of *Low* inputs
- $L \cap O$ set of *Low* outputs
- T_{Low} set of all possible sequences of *Low* events that are legal within system
- π_L: $T \rightarrow T_{Low}$ projection function deleting all *High* inputs from trace
  - *Low* observer should not be able to deduce anything about *High* inputs from trace $t_{Low} \in T_{low}$

Deducibly Secure

- System deducibly secure if, for every trace $t_{Low} \in T_{Low}$, the corresponding set of high level traces contains every possible trace $t \in T$ for which $\pi_L(t) = t_{Low}$
  - Given any t_{Low}, the trace $t \in T$ producing that t_{Low} is equally likely to be *any* trace with $\pi_L(t) = t_{Low}$

Example

- Back to our 2-bit machine
  - Let xor0, xor1 apply to both bits
  - Both bits output after each command
- Initial state: (0, 1)
- Inputs: $1_H 0_L 1_L 0_H 1_L 0_L$
- Outputs: 10 10 01 01 10 10
- Lara (at *Low*) sees: 001100
  - Does not know initial state, so does not know first input; but can deduce fourth input is 0
- Not deducibly secure

Example

- Now xor_0, xor_1 apply only to state bit with same level as user
- Inputs: $1_H 0_L\ 1_L 0_H\ 1_L 0_L$
- Outputs: 10 11 11 10 11
- Lara sees: 01101
- She cannot deduce *anything* about input
  - Could be $0_H 0_L\ 1_L 0_H\ 1_L 0_L$ or $0_L 1_H\ 1_L 0_H\ 1_L 0_L$ for example
- Deducibly secure

Security of Composition

- In general: deducibly secure systems not composable
- *Strong noninterference*: deducible security + requirement that no *High* output occurs unless caused by a *High* input
  - Systems meeting this property *are* composable

Example

- 2-bit machine done earlier does not exhibit strong noninterference
  - Because it puts out *High* bit even when there is no *High* input
- Modify machine to output only state bit at level of latest input
  - *Now* it exhibits strong noninterference

Problem

- Too restrictive; it bans some systems that are *obviously* secure
- Example: System *upgrade* reads *Low* inputs, outputs those bits at *High*
  - Clearly deducibly secure: low level user sees no outputs
  - Clearly does not exhibit strong noninterference, as no high level inputs!

Remove Determinism

- Previous assumption
  - Input, output synchronous
  - Output depends only on commands triggered by input
    - Sometimes absorbed into commands …
  - Input processed one datum at a time
- Not realistic
  - In real systems, lots of asynchronous events

Generalized Noninterference

- Nondeterministic systems meeting noninterference property meet *generalized noninterference-secure property*
  - More robust than deducible security because minor changes in assumptions affect whether system is deducibly secure

Example

- System with *High* Holly, *Low* lucy, text file at *High*
  - File fixed size, symbol b marks empty space
  - Holly can edit file, Lucy can run this program:

```
while true do begin
     n := read_integer_from_user;
     if n > file_length or char_in_file[n] = b then
              print random_character;
     else
              print char_in_file[n];
end;
```

Security of System

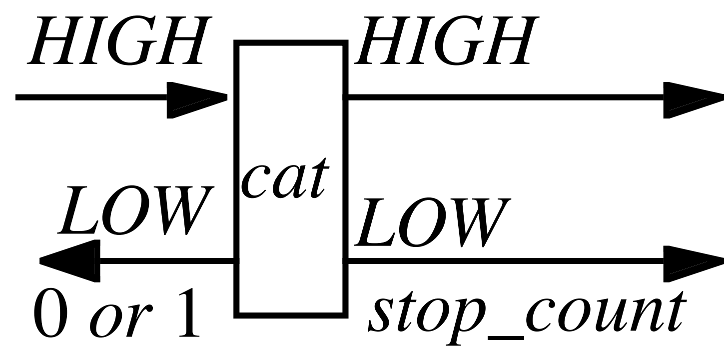
- Not noninterference-secure
  - High level inputs—Holly's changes—affect low level outputs
- *May* be deducibly secure
  - Can Lucy deduce contents of file from program?
  - If output meaningful ("This is right") or close ("Thes is riqht"), yes
  - Otherwise, no
- So deducibly secure depends on which inferences are allowed

Composition of Systems

- Does composing systems meeting generalized noninterference-secure property give you a system that also meets this property?
- Define two systems (*cat*, *dog*)
- Compose them

First System: *cat*

- Inputs, outputs can go left or right
- After some number of inputs, *cat* sends two outputs
  - First *stop_count*
  - Second parity of *High* inputs, outputs

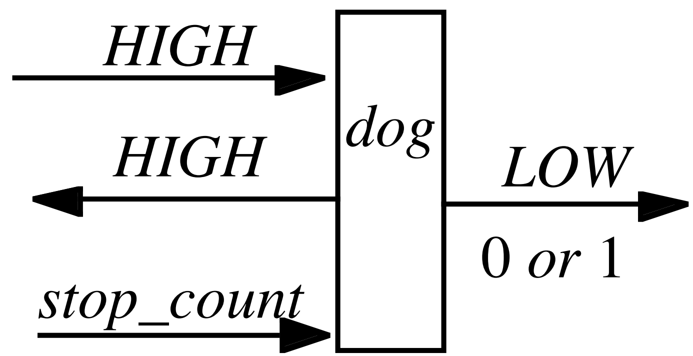


Noninterference-Secure?

- If even number of *High* inputs, output could be:
  - 0 (even number of outputs)
  - 1 (odd number of outputs)
- If odd number of *High* inputs, output could be:
  - 0 (odd number of outputs)
  - 1 (even number of outputs)
- High level inputs do not affect output
  - So noninterference-secure

Second System: *dog*

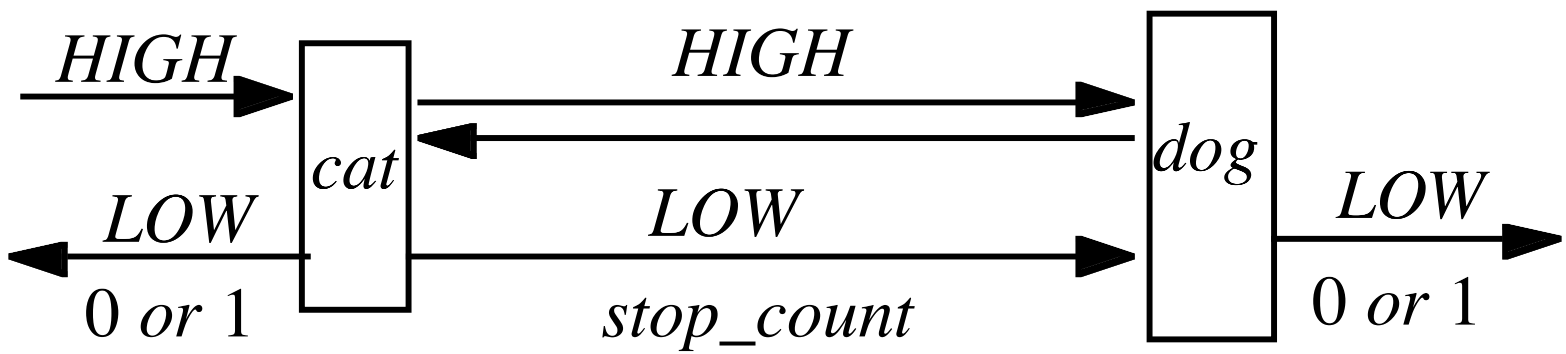
- High outputs to left
- Low outputs of 0 or 1 to right
- *stop_count* input from the left
  - When it arrives, *dog* emits 0 or 1



Noninterference-Secure?

- When *stop_count* arrives:
  - May or may not be inputs for which there are no corresponding outputs
  - Parity of *High* inputs, outputs can be odd or even
  - Hence *dog* emits 0 or 1
- High level inputs do not affect low level outputs
  - So noninterference-secure

Compose Them



- Once sent, message arrives
  - But *stop_count* may arrive before all inputs have generated corresponding outputs
  - If so, even number of *High* inputs and outputs on *cat*, but odd number on *dog*
- Four cases arise

The Cases

- *cat*, odd number of inputs, outputs; *dog*, even number of inputs, odd number of outputs
  - Input message from *cat* not arrived at *dog*, contradicting assumption
- *cat*, even number of inputs, outputs; *dog*, odd number of inputs, even number of outputs
  - Input message from *dog* not arrived at *cat*, contradicting assumption

The Cases

- cat, odd number of inputs, outputs; dog, odd number of inputs, even number of outputs
  – dog sent even number of outputs to cat, so cat has had at least one input from left
- cat, even number of inputs, outputs; dog, even number of inputs, odd number of outputs
  – dog sent odd number of outputs to cat, so cat has had at least one input from left

The Conclusion

- Composite system *catdog* emits 0 to left, 1 to right (or 1 to left, 0 to right)
  - Must have received at least one input from left
- Composite system *catdog* emits 0 to left, 0 to right (or 1 to left, 1 to right)
  - Could not have received any from left
- So, *High* inputs affect *Low* outputs
  - Not noninterference-secure

Feedback-Free Systems

- System has n distinct components
- Components c_i, c_j connected if any output of c_i is input to c_j
- System is *feedback-free* if for all c_i connected to c_j, c_j not connected to any c_i
  - Intuition: once information flows from one component to another, no information flows back from the second to the first

Feedback-Free Security

- *Theorem*: A feedback-free system composed of noninterference-secure systems is itself noninterference-secure

Some Feedback

- *Lemma*: A noninterference-secure system can feed a high level output *o* to a high level input *i* if the arrival of *o* at the input of the next component is delayed until *after* the next low level input or output
- *Theorem*: A system with feedback as described in the above lemma and composed of noninterference-secure systems is itself noninterference-secure

Why Didn't They Work?

- For compositions to work, machine must act same way regardless of what precedes low level input (high, low, nothing)
- *dog* does not meet this criterion
  - If first input is *stop_count*, *dog* emits 0
  - If high level input precedes *stop_count*, *dog* emits 0 or 1

State Machine Model

- 2-bit machine, levels *High*, *Low*, meeting 4 properties:

1. For every input i_k, state σ_j, there is an element $c_m \in C^*$ such that $T^*(c_m, \sigma_j) = \sigma_n$, where $\sigma_n \neq \sigma_j$
   – T^* is total function, inputs and commands always move system to a different state

Property 2

- There is an equivalence relation $\equiv$ such that:
  - If system in state σ_i and high level sequence of inputs causes transition from σ_i to σ_j, then $\sigma_i \equiv \sigma_j$
  - If $\sigma_i \equiv \sigma_j$ and low level sequence of inputs $i_1, \ldots, i_n$ causes system in state σ_i to transition to σ_i', then there is a state σ_j' such that $\sigma_i' \equiv \sigma_j'$ and the inputs $i_1, \ldots, i_n$ cause system in state σ_j to transition to σ_j'
- $\equiv$ holds if low level projections of both states are same

Property 3

- Let $\sigma_i \equiv \sigma_j$. If high level sequence of outputs $o_1, \ldots, o_n$ indicate system in state σ_i transitioned to state σ_i', then for some state σ_j' with $\sigma_j' \equiv \sigma_i'$, high level sequence of outputs $o_1', \ldots, o_m'$ indicates system in σ_j transitioned to σ_j'
  – High level outputs do not indicate changes in low level projection of states

Property 4

- Let $\sigma_i \equiv \sigma_j$, let c, d be high level output sequences, e a low level output. If ced indicates system in state σ_i transitions to σ_i', then there are high level output sequences c' and d' and state σ_j' such that $c'ed'$ indicates system in state σ_j transitions to state σ_j'
  – Intermingled low level, high level outputs cause changes in low level state reflecting low level outputs only

Restrictiveness

- System is *restrictive* if it meets the preceding 4 properties

Composition

- Intuition: by 3 and 4, high level output followed by low level output has same effect as low level input, so composition of restrictive systems should be restrictive

Composite System

- System M_1's outputs are M_2's inputs
- μ_{1i}, μ_{2i} states of M_1, M_2
- States of composite system pairs of M_1, M_2 states (μ_{1i}, μ_{2i})
- e event causing transition
- e causes transition from state (μ_{1a}, μ_{2a}) to state (μ_{1b}, μ_{2b}) if any of 3 conditions hold

Conditions

1. M_1 in state μ_{1a} and e occurs, M_1 transitions to μ_{1b}; e not an event for M_2; and $\mu_{2a} = \mu_{2b}$
2. M_2 in state μ_{2a} and e occurs, M_2 transitions to μ_{2b}; e not an event for M_1; and $\mu_{1a} = \mu_{1b}$
3. M_1 in state μ_{1a} and e occurs, M_1 transitions to μ_{1b}; M_2 in state μ_{2a} and e occurs, M_2 transitions to μ_{2b}; e is input to one machine, and output from other

Intuition

- Event causing transition in composite system causes transition in at least 1 of the components
- If transition occurs in exactly one component, event must not cause transition in other component when not connected to the composite system

Equivalence for Composite

- Equivalence relation for composite system
  $(\sigma_a, \sigma_b) \equiv_C (\sigma_c, \sigma_d)$ iff $\sigma_a \equiv \sigma_c$ and $\sigma_b \equiv \sigma_d$
- Corresponds to equivalence relation in property 2 for component system

Information Flow

- Basics and background
  - Entropy
- Nonlattice flow policies
- Compiler-based mechanisms
- Execution-based mechanisms
- Examples
  - Security Pipeline Interface
  - Secure Network Server Mail Guard

Basics

- Bell-LaPadula Model embodies information flow policy
  - Given compartments A, B, info can flow from A to B iff B *dom* A
- Variables x, y assigned compartments $\underline{x}$, $\underline{y}$ as well as values
  - If $\underline{x} = A$ and $\underline{y} = B$, and A *dom* B, then $x := y$ allowed but not $y := x$

Quick Review of Entropy

- Random variables
- Joint probability
- Conditional probability
- Entropy (or uncertainty in bits)
- Joint entropy
- Conditional entropy
- Applying it to secrecy of ciphers

Random Variable

- Variable that represents outcome of an event
  - X represents value from roll of a fair die; probability for rolling n: $p(X = n) = 1/6$
  - If die is loaded so 2 appears twice as often as other numbers, $p(X = 2) = 2/7$ and, for $n \neq 2$, $p(X = n) = 1/7$
- Note: $p(X)$ means specific value for X doesn't matter
  - Example: all values of X are equiprobable

Joint Probability

- Joint probability of X and Y, $p(X, Y)$, is probability that X and Y simultaneously assume particular values
  - If X, Y independent, $p(X, Y) = p(X)p(Y)$
- Roll die, toss coin
  - $p(X = 3, Y = \text{heads}) = p(X = 3)p(Y = \text{heads}) = 1/6 \times 1/2 = 1/12$

Two Dependent Events

- X = roll of red die, Y = sum of red, blue die rolls

$p(Y=2) = 1/36$	$p(Y=3) = 2/36$	$p(Y=4) = 3/36$	$p(Y=5) = 4/36$
$p(Y=6) = 5/36$	$p(Y=7) = 6/36$	$p(Y=8) = 5/36$	$p(Y=9) = 4/36$
$p(Y=10) = 3/36$	$p(Y=11) = 2/36$	$p(Y=12) = 1/36$	

- Formula:
  - $p(X=1, Y=11) = p(X=1)p(Y=11) = (1/6)(2/36) = 1/108$

Conditional Probability

- Conditional probability of X given Y, written $p(X \mid Y)$, is probability that X takes on a particular value given Y has a particular value
- Continuing example …
  - $p(Y = 7 \mid X = 1) = 1/6$
  - $\mathrm{p}(Y = 7 \mid X = 3) = 1/6$

Relationship

- $p(X, Y) = p(X \mid Y)\, p(Y) = p(X)\, p(Y \mid X)$
- Example:
  - $p(X = 3, Y = 8) = p(X = 3 \mid Y = 8)\, p(Y = 8) = (1/5)(5/36) = 1/36$
- Note: if X, Y independent:
  - $p(X \mid Y) = p(X)$

Entropy

- Uncertainty of a value, as measured in bits
- Example: X value of fair coin toss; X could be heads or tails, so 1 bit of uncertainty
  - Therefore entropy of X is $H(X) = 1$
- Formal definition: random variable X, values $x_1, \ldots, x_n$; so Σ_i p$(X = x_i) = 1$

  $H(X) = -\Sigma_i \, p(X = x_i) \lg p(X = x_i)$

Heads or Tails?

- $H(X) = -p(X = \text{heads}) \lg p(X = \text{heads})$
  $- \text{p}(X = \text{tails}) \lg p(X = \text{tails})$
  $= -(1/2) \lg (1/2) - (1/2) \lg (1/2)$
  $= -(1/2)(-1) - (1/2)(-1) = 1$
- Confirms previous intuitive result

n-Sided Fair Die

$H(X) = -\Sigma_i\, p(X = x_i) \lg p(X = x_i)$

As $p(X = x_i) = 1/n$, this becomes

$H(X) = -\Sigma_i\, (1/n) \lg (1/n) = -n(1/n)(-\lg n)$

so

$H(X) = \lg n$

which is the number of bits in n, as expected

Ann, Pam, and Paul

Ann, Pam twice as likely to win as Paul

W represents the winner. What is its entropy?

- w_1 = Ann, w_2 = Pam, w_3 = Paul
- $p(W = w_1) = p(W = w_2) = 2/5$, $p(W = w_3) = 1/5$

- So $H(W) = -\Sigma_i\, p(W = w_i) \lg p(W = w_i)$

  $= -(2/5) \lg (2/5) - (2/5) \lg (2/5) - (1/5) \lg (1/5)$

  $= -(4/5) + \lg 5 \approx 1.52$
- If all equally likely to win, $H(W) = \lg 3 = 1.58$

Joint Entropy

- X takes values from $\{ x_1, \ldots, x_n \}$
  - $\Sigma_i \, p(X = x_i) = 1$
- Y takes values from $\{ y_1, \ldots, y_m \}$
  - $\Sigma_i \, p(Y = y_i) = 1$
- Joint entropy of X, Y is:
  - $H(X, Y) = -\Sigma_j \, \Sigma_i \, p(X{=}x_i, Y{=}y_j) \lg p(X{=}x_i, Y{=}y_j)$

Example

X: roll of fair die, Y: flip of coin

$p(X=1, Y=\text{heads}) = p(X=1)\ p(Y=\text{heads}) = 1/12$

- As X and Y are independent

$H(X, Y) = -\Sigma_j\ \Sigma_i\ p(X=x_i, Y=y_j) \lg p(X=x_i, Y=y_j)$

$= -2\ [\ 6\ [\ (1/12) \lg (1/12)\ ]\ ] = \lg 12$

Conditional Entropy

- X takes values from $\{ x_1, \ldots, x_n \}$
  - $\Sigma_i\, p(X=x_i) = 1$
- Y takes values from $\{ y_1, \ldots, y_m \}$
  - $\Sigma_i\, p(Y=y_i) = 1$
- Conditional entropy of X given $Y=y_j$ is:
  - $H(X \mid Y=y_j) = -\Sigma_i\, p(X=x_i \mid Y=y_j) \lg p(X=x_i \mid Y=y_j)$
- Conditional entropy of X given Y is:
  - $H(X \mid Y) = -\Sigma_j\, p(Y=y_j)\, \Sigma_i\, p(X=x_i \mid Y=y_j) \lg p(X=x_i \mid Y=y_j)$

Example

- X roll of red die, Y sum of red, blue roll
- Note $p(X=1 \mid Y=2) = 1$, $p(X=i \mid Y=2) = 0$ for $i \neq 1$
  - If the sum of the rolls is 2, both dice were 1
- $H(X|Y=2) = -\Sigma_i\, p(X=x_i \mid Y=2) \lg p(X=x_i \mid Y=2) = 0$
- Note $p(X=i\ , Y=7) = 1/6$
  - If the sum of the rolls is 7, the red die can be any of 1, …, 6 and the blue die must be 7–roll of red die
- $H(X|Y=7) = -\Sigma_i\, p(X=x_i \mid Y=7) \lg p(X=x_i \mid Y=7)$

$$= -6\,(1/6) \lg (1/6) = \lg 6$$

Perfect Secrecy

- Cryptography: knowing the ciphertext does not decrease the uncertainty of the plaintext
- $M = \{ m_1, \ldots, m_n \}$ set of messages
- $C = \{ c_1, \ldots, c_n \}$ set of messages
- Cipher $c_i = E(m_i)$ achieves *perfect secrecy* if $H(M \mid C) = H(M)$

Entropy and Information Flow

- Idea: info flows from x to y as a result of a sequence of commands c if you can deduce information about x before c from the value in y after c
- Formally:
  - s time before execution of c, t time after
  - $H(x_s \mid y_t) < H(x_s \mid y_s)$
  - If no y at time s, then $H(x_s \mid y_t) < H(x_s)$

Example 1

- Command is $x := y + z$; where:
  - $0 \leq y \leq 7$, equal probability
  - $z = 1$ with prob. 1/2, $z = 2$ or 3 with prob. 1/4 each
- s state before command executed; t, after; so
  - $H(y_s) = H(y_t) = -8(1/8) \lg (1/8) = 3$
  - $H(z_s) = H(z_t) = -(1/2) \lg (1/2) - 2(1/4) \lg (1/4) = 1.5$
- If you know x_t, y_s can have at most 3 values, so $H(y_s \mid x_t) = -3(1/3) \lg (1/3) = \lg 3$

Example 2

- Command is
  - **if** $x = 1$ **then** $y := 0$ **else** $y := 1$;

  where:
  - x, y equally likely to be either 0 or 1
- $H(x_s) = 1$ as x can be either 0 or 1 with equal probability
- $H(x_s \mid y_t) = 0$ as if $y_t = 1$ then $x_s = 0$ and vice versa
  - Thus, $H(x_s \mid y_t) = 0 < 1 = H(x_s)$
- So information flowed from x to y