Lecture 23
November 22, 2023

Entropy for Information Flow

- 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:

  $p(X=1, Y=11) = p(X=1)p(Y=11) = (1/6)(2/36) = 1/108$
- But if the red die (X) rolls 1, the most their sum (Y) can be is 7
- The problem is X and Y are dependent

Conditional Probability

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

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; then entropy is:

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

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, $H(W) = \lg 3 \approx 1.58$

Joint Entropy

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

$$H(X, Y) = -\sum_j \sum_i p(X=x_i, Y=y_j) \lg p(X=x_i, Y=y_j)$$

Example

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

As X, Y are independent:

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

and

$$H(X, Y) = -\sum_j \sum_i p(X=x_i, Y=y_j) \lg p(X=x_i, Y=y_j)$$
$$= -2\, [\, 6\, [\, (1/12) \lg (1/12)\,]\,] = \lg 12$$

Conditional Entropy (Equivocation)

- X takes values from $\{ x_1, \ldots, x_n \}$ and $\Sigma_i \, p(X{=}x_i) = 1$
- Y takes values from $\{ y_1, \ldots, y_m \}$ and $\Sigma_i \, p(Y{=}y_i) = 1$
- 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 $p(X=1 | Y=2) = 1$, $p(X=i | Y=2) = 0$ for $i \neq 1$
  - If the sum of the rolls is 2, both dice were 1
- Thus

$$H(X | Y=2) = -\Sigma_i \, p(X=x_i | Y=2) \lg p(X=x_i | Y=2) = 0$$

Example (*con't*)

- 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
- $H(X|Y=7) = -\sum_i p(X=x_i|Y=7) \lg p(X=x_i|Y=7)$

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

Example: 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 $H(M \mid C) = H(M)$

Basics of Information Flow

- Bell-LaPadula Model embodies information flow policy
  - Given compartments *A*, *B*, info can flow from *A* to *B* iff *B dom A*
- So does Biba Model
  - Given compartments *A*, *B*, info can flow from *A* to *B* iff *A dom B*
- Variables *x*, *y* assigned compartments $\underline{x}$, $\underline{y}$ as well as values
  - Confidentiality (Bel-LaPadula): if $\underline{x} = A$, $\underline{y} = B$, and *B dom A*, then $y := x$ allowed but not $x := y$
  - Integrity (Biba): if $\underline{x} = A$, $\underline{y} = B$, and *A dom B*, then $x := y$ allowed but not $y := x$
- For now, focus on confidentiality (Bell-LaPadula)
  - We'll get to integrity later

Entropy and Information Flow

- Idea: information 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:
  - x does not exist initially (that is, has no value)
  - $0 \le y \le 7$, equal probability
  - $z = 1$ with probability 1/2, $z = 2$ or 3 with probability 1/4 each
- s state before command executed; t, after; so
  - $H(y_s) = H(y_t) = -8(1/8) \lg (1/8) = 3$
- You can show that $H(y_s \mid x_t) = (3/32) \lg 3 + 9/8 \approx 1.274 < 3 = H(y_s)$
  - Thus, information flows from y to x

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

Implicit Flow of Information

- Information flows from x to y without an *explicit* assignment of the form $y := f(x)$
  - $f(x)$ an arithmetic expression with variable x
- Example from previous slide:

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

- So must look for implicit flows of information to analyze program

Notation

- $\underline{x}$ means class of x
  - In Bell-LaPadula based system, same as “label of security compartment to which x belongs”
- $\underline{x} \leq \underline{y}$ means “information can flow from an element in class of x to an element in class of y
  - Or, “information with a label placing it in class $\underline{x}$ can flow into class $\underline{y}$”

Compiler-Based Mechanisms

- Detect unauthorized information flows in a program during compilation
- Analysis not precise, but secure
  - If a flow *could* violate policy (but may not), it is unauthorized
  - No unauthorized path along which information could flow remains undetected
- Set of statements *certified* with respect to information flow policy if flows in set of statements do not violate that policy

Example

```
if x = 1 then y := a;
else y := b;
```

- Information flows from *x* and *a* to *y*, or from *x* and *b* to *y*
- Certified only if $\underline{x} \le \underline{y}$ and $\underline{a} \le \underline{y}$ and $\underline{b} \le \underline{y}$
  - Note flows for *both* branches must be true unless compiler can determine that one branch will *never* be taken

Declarations

- Notation:

  `x: int class { A, B }`

  means x is an integer variable with security class at least *lub*{ A, B }, so *lub*{ A, B } $\leq \underline{x}$
- Distinguished classes *Low*, *High*
  - Constants are always *Low*

Input Parameters

- Parameters through which data passed into procedure
- Class of parameter is class of actual argument

$$i_p\texttt{: } \textbf{\textit{type}}\ \textbf{class}\ \{\ \mathrm{i_p}\ \}$$

Output Parameters

- Parameters through which data passed out of procedure
  - If data passed in, called input/output parameter
- As information can flow from input parameters to output parameters, class must include this:

$$o_p\texttt{: } \boldsymbol{\mathit{type}}\ \mathbf{class}\ \{\ r_1,\ \ldots,\ r_n\ \}$$

where r_i is class of ith input or input/output argument

Example

```
proc sum(x: int class { A };
     var out: int class { A, B });
begin
     out := out + x;
end;
```

- Require <u>*x*</u> ≤ <u>*out*</u> and <u>*out*</u> ≤ <u>*out*</u>

Array Elements

- Information flowing out:

```
… := a[i]
```

Value of *i*, *a*[*i*] both affect result, so class is lub{ <u>*a*[*i*]</u>, <u>*i*</u> }

- Information flowing in:

```
a[i] := …
```

- Only value of *a*[*i*] affected, so class is <u>*a*[*i*]</u>

Assignment Statements

x := *y* + *z*;

- Information flows from *y*, *z* to *x*, so this requires lub{ $\underline{y}$, $\underline{z}$ } ≤ $\underline{x}$

More generally:

$\underline{y}$:= $f(x_1, \ldots, x_n)$

- the relation lub{ $\underline{x}_1$, …, x_n } ≤ $\underline{y}$ must hold

Compound Statements

$x := y + z;\ a := b * c - x;$

- First statement: lub$\{\ \underline{y}, \underline{z}\ \} \leq \underline{x}$
- Second statement: lub$\{\ \underline{b}, \underline{c}, \underline{x}\ \} \leq \underline{a}$
- So, both must hold (i.e., be secure)

More generally:

$S_1;\ \ldots\ S_n;$

- Each individual S_i must be secure

Conditional Statements

`if` x `+` y `<` z `then` a `:=` b `else` d `:=` b `*` c `–` x`; end`

- Statement executed reveals information about x, y, z, so lub{ $\underline{x}$, $\underline{y}$, $\underline{z}$ } ≤ glb{ $\underline{a}$, $\underline{d}$ }

More generally:

`if` $f(x_1, \ldots, x_n)$ `then` S_1 `else` S_2`; end`

- S_1, S_2 must be secure
- lub{ $\underline{x}_1$, …, $\underline{x}_n$ } ≤ glb{$\underline{y}$ | y target of assignment in S_1, S_2 }

Iterative Statements

```
while i < n do begin a[i] := b[i]; i := i + 1; end
```

- Same ideas as for “if”, but must terminate

More generally:

while $f(x_1, \ldots, x_n)$ do S;

- Loop must terminate;
- *S* must be secure
- lub{ $\underline{x}_1, \ldots, \underline{x}_n$ } ≤ glb{$\underline{y}$ | y target of assignment in S }

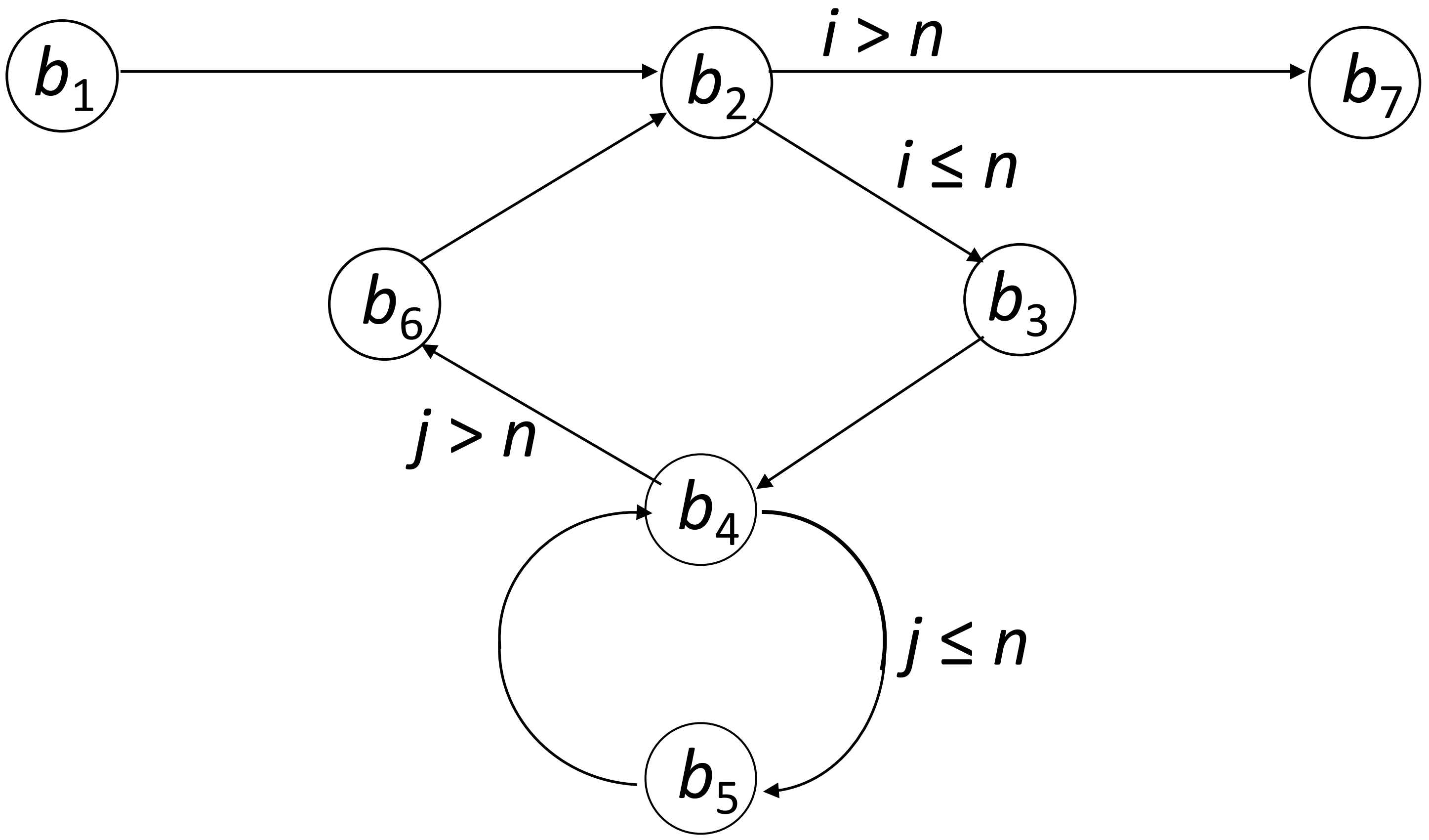
Goto Statements

- No assignments
  - Hence no explicit flows
- Need to detect implicit flows
- *Basic block* is sequence of statements that have one entry point and one exit point
  - Control in block *always* flows from entry point to exit point

Example Program

```
proc tm(x: array[1..10][1..10] of integer class {x};
                 var y: array[1..10][1..10] of integer class {y});
var i, j: integer class {i};
begin
b1     i := 1;
b2 L2: if i > 10 goto L7;
b3     j := 1;
b4 L4: if j > 10 then goto L6;
b5     y[j][i] := x[i][j]; j := j + 1; goto L4;
b6 L6: i := i + 1; goto L2;
b7 L7:
end;
```

Flow of Control



Immediate Forward Dominators

- Idea: when two paths out of basic block, implicit flow occurs
  - Because information says *which* path to take
- When paths converge, either:
  - Implicit flow becomes irrelevant; or
  - Implicit flow becomes explicit
- *Immediate forward dominator* of basic block *b* (written IFD(*b*)) is first basic block lying on all paths of execution passing through *b*

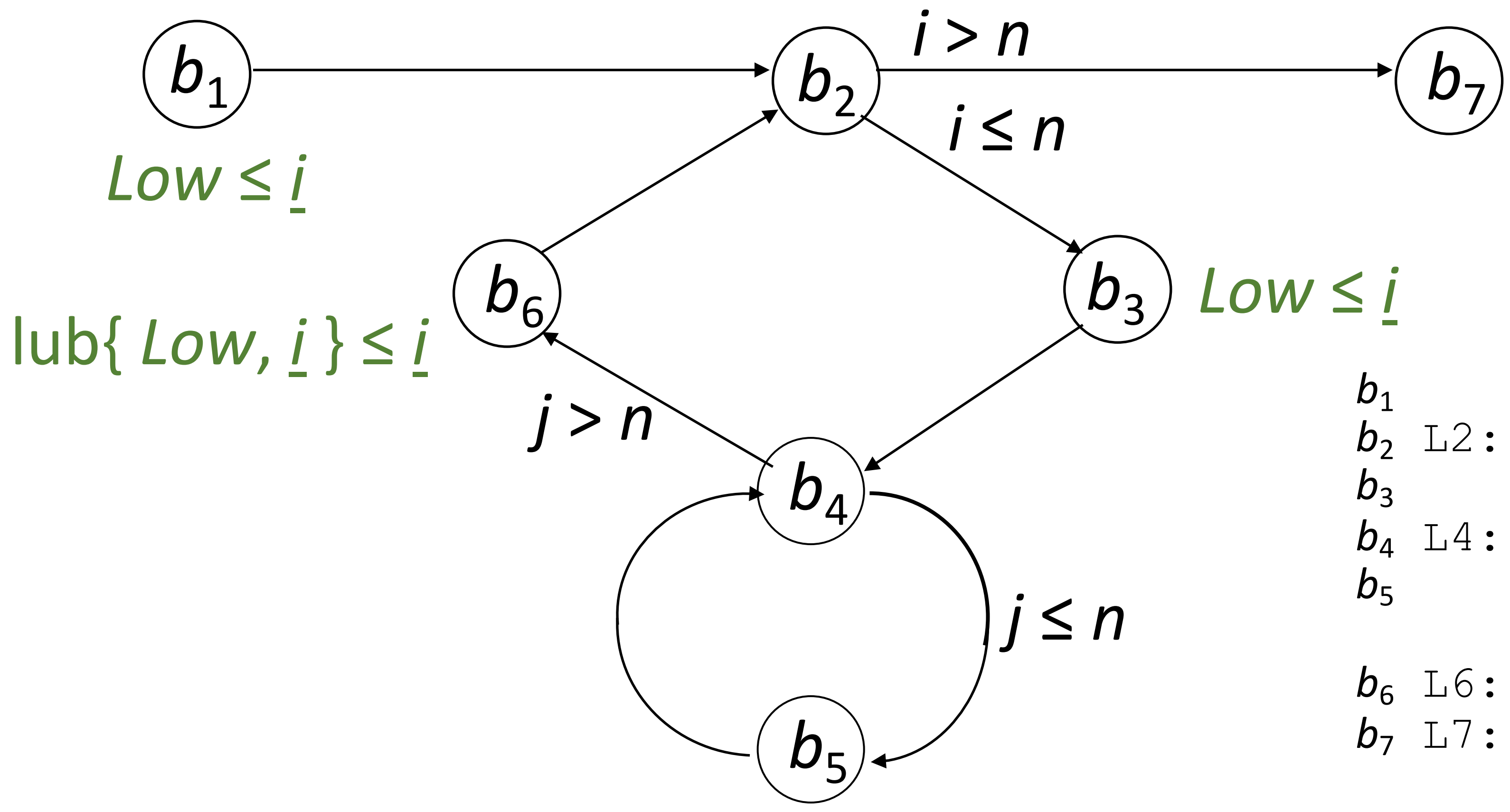
IFD Example

- In previous procedure:
  - IFD(b_1) = b_2 one path
  - IFD(b_2) = b_7 $b_2 \rightarrow b_7$ or $b_2 \rightarrow b_3 \rightarrow b_6 \rightarrow b_2 \rightarrow b_7$
  - IFD(b_3) = b_4 one path
  - IFD(b_4) = b_6 $b_4 \rightarrow b_6$ or $b_4 \rightarrow b_5 \rightarrow b_6$
  - IFD(b_5) = b_4 one path
  - IFD(b_6) = b_2 one path

Requirements

- B_i is set of basic blocks along an execution path from b_i to IFD(b_i)
  - Analogous to statements in conditional statement
- $x_{i1}, \ldots, x_{in}$ variables in expression selecting which execution path containing basic blocks in B_i used
  - Analogous to conditional expression
- Requirements for secure:
  - All statements in each basic blocks are secure
  - lub{ $\underline{x}_{i1}, \ldots, \underline{x}_{in}$ } ≤ glb{ $\underline{y}$ | y target of assignment in B_i }

Example of Requirements




```
b1        i := 1;
b2  L2:   if i > 10 goto L7;
b3        j := 1;
b4  L4:   if j > 10 then goto L6;
b5                  y[j][i] := x[i][j];
                         j := j + 1; goto L4;
b6  L6:   i := i + 1; goto L2;
b7  L7:
```

lub{ $\underline{x[i][j]}$, $\underline{i}$, $\underline{j}$ } $\le \underline{y[j][i]}$ }; lub{ Low, $\underline{j}$ } $\le \underline{j}$

Example of Requirements

- Within each basic block:

  b_1: $Low \leq \underline{i}$ b_3: $Low \leq \underline{j}$ b_6: $\text{lub}\{ Low, \underline{i} \} \leq \underline{i}$

  b_5: $\text{lub}\{ \underline{x[i][j]}, \underline{i}, \underline{j} \} \leq \underline{y[j][i]}$ }; $\text{lub}\{ Low, \underline{j} \} \leq \underline{j}$

  - Combining, $\text{lub}\{ \underline{x[i][j]}, \underline{i}, \underline{j} \} \leq \underline{y[j][i]}$ }
  - From declarations, true when $\text{lub}\{ \underline{x}, \underline{i} \} \leq \underline{y}$
- $B_2 = \{b_3, b_4, b_5, b_6\}$
  - Assignments to i, j, $y[j][i]$; conditional is $i \leq 10$
  - Requires $\underline{i} \leq \text{glb}\{ \underline{i}, \underline{j}, \underline{y[j][i]} \}$
  - From declarations, true when $\underline{i} \leq \underline{y}$

Example (continued)

- $B_4 = \{ b_5 \}$
  - Assignments to j, $y[j][i]$; conditional is $j \leq 10$
  - Requires $\underline{j} \leq \text{glb}\{ \underline{j}, \underline{y[j][i]} \}$
  - From declarations, means $\underline{i} \leq \underline{y}$
- Result:
  - Combine $\text{lub}\{ \underline{x}, \underline{i} \} \leq \underline{y}$; $\underline{i} \leq \underline{y}$; $\underline{i} \leq \underline{y}$
  - Requirement is $\text{lub}\{ \underline{x}, \underline{i} \} \leq \underline{y}$

Procedure Calls

$tm(a, b);$

From previous slides, to be secure, lub{ $\underline{x}$, $\underline{i}$ } ≤ $\underline{y}$ must hold

- In call, *x* corresponds to *a*, *y* to *b*
- Means that lub{ $\underline{a}$, $\underline{i}$ } ≤ $\underline{b}$, or $\underline{a} \le \underline{b}$

More generally:

proc *pn*(i_1, …, i_m: **int**; **var** o_1, …, o_n: **int**); **begin** *S* **end**;

- *S* must be secure
- For all *j* and *k*, if $\underline{i}_j \le \underline{o}_k$, then $\underline{x}_j \le \underline{y}_k$
- For all *j* and *k*, if $\underline{o}_j \le \underline{o}_k$, then $\underline{y}_j \le \underline{y}_k$