Lecture 19

- Information flow
- Basics and background
  - Entropy
- Non-lattice flow policies
- Compiler-based mechanisms
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 $\sum_i p(X = x_i) = 1$
  $H(X) = -\sum_i p(X = x_i) \log p(X = x_i)$
Heads or Tails?

- \( H(X) = - p(X = \text{heads}) \log p(X = \text{heads}) \)
  - \( - p(X = \text{tails}) \log p(X = \text{tails}) \)
  
  \[= - (1/2) \log (1/2) - (1/2) \log (1/2) \]
  \[= - (1/2) (-1) - (1/2) (-1) = 1\]

- Confirms previous intuitive result
$n$-Sided Fair Die

\[ H(X) = -\sum_i p(X = x_i) \lg p(X = x_i) \]

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

\[ H(X) = -\sum_i (1/n) \lg (1/n) = -n(1/n) (-\lg n) \]

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?

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

- So $H(W) = -\sum_i p(W = w_i) \log p(W = w_i)$
  
  $= - \left( \frac{2}{5} \right) \log \left( \frac{2}{5} \right) - \left( \frac{2}{5} \right) \log \left( \frac{2}{5} \right) - \left( \frac{1}{5} \right) \log \left( \frac{1}{5} \right)$
  
  $= - \left( \frac{4}{5} \right) + \log 5 \approx 1.52$

- If all equally likely to win, $H(W) = \log 3 = 1.58$
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:
  - $H(X, Y) = -\Sigma_j \Sigma_i p(X=x_i, Y=y_j) \log p(X=x_i, Y=y_j)$
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

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

$= -2 \left[ 6 \left[ (1/12) \log (1/12) \right] \right] = \log 12$
Conditional Entropy

• $X$ takes values from \{ $x_1$, $\ldots$, $x_n$ \}
  \[ \sum_i p(X=x_i) = 1 \]
• $Y$ takes values from \{ $y_1$, $\ldots$, $y_m$ \}
  \[ \sum_i p(Y=y_i) = 1 \]
• Conditional entropy of $X$ given $Y=y_j$ is:
  \[ H(X \mid Y=y_j) = -\sum_i p(X=x_i \mid Y=y_j) \log p(X=x_i \mid Y=y_j) \]
• Conditional entropy of $X$ given $Y$ is:
  \[ H(X \mid Y) = -\sum_j p(Y=y_j) \sum_i p(X=x_i \mid Y=y_j) \log p(X=x_i \mid Y=y_j) \]
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
- $H(X \mid Y=2) = -\sum_i p(X=x_i \mid Y=2) \log p(X=x_i \mid Y=2) = 0$
- 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
- $H(X \mid Y=7) = -\sum_i p(X=x_i \mid Y=7) \log p(X=x_i \mid Y=7)$
  $= -6 \cdot (1/6) \log (1/6) = \log 6$
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
- $H(x_s | y_t) = 0$ as if $y_t = 1$ then $x_s = 0$ and vice versa
  - Thus, $H(x_s | y_t) = 0 < 1 = H(x_s)$
- So information flowed from $x$ to $y$
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
  - $H(y_s) = H(y_t) = -8(1/8) \log_2 (1/8) = 3$
  - $H(z_s) = H(z_t) = -(1/2) \log_2 (1/2) -2(1/4) \log_2 (1/4) = 1.5$
- If you know $x_t$, $y_s$ can have at most 3 values, so
  $H(y_s \mid x_t) = -3(1/3) \log_2 (1/3) = \log_2 3$
Example 2

• Command is
  – \textbf{if} \( x = 1 \) \textbf{then} \( y := 0 \) \textbf{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

• $\_x$ means class of $x$
  – In Bell-LaPadula based system, same as “label of security compartment to which $x$ belongs”

• $x \leq y$ 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 $x$ can flow into class $y$”
Information Flow Policies

Information flow policies are usually:

• reflexive
  – So information can flow freely among members of a single class

• transitive
  – So if information can flow from class 1 to class 2, and from class 2 to class 3, then information can flow from class 1 to class 3
Non-Transitive Policies

• Betty is a confident of Anne
• Cathy is a confident of Betty
  – With transitivity, information flows from Anne to Betty to Cathy
• Anne confides to Betty she is having an affair with Cathy’s spouse
  – Transitivity undesirable in this case, probably
Transitive Non-Lattice Policies

- 2 faculty members co-PIs on a grant
  - Equal authority; neither can overrule the other
- Grad students report to faculty members
- Undergrads report to grad students
- Information flow relation is:
  - Reflexive and transitive
- But some elements (people) have no “least upper bound” element
  - What is it for the faculty members?
Confidentiality Policy Model

- Lattice model fails in previous 2 cases
- Generalize: policy $I = (SC_I, \leq_I, \text{join}_I)$:
  - $SC_I$ set of security classes
  - $\leq_I$ ordering relation on elements of $SC_I$
  - $\text{join}_I$ function to combine two elements of $SC_I$
- Example: Bell-LaPadula Model
  - $SC_I$ set of security compartments
  - $\leq_I$ ordering relation $\text{dom}$
  - $\text{join}_I$ function $\text{lub}$
Confinement Flow Model

- \((I, O, \text{confine}, \rightarrow)\)
  - \(I = (SC_I, \leq_I, \text{join}_I)\)
  - \(O\) set of entities
  - \(\rightarrow: O \times O\) with \((a, b) \in \rightarrow\) (written \(a \rightarrow b\)) iff information can flow from \(a\) to \(b\)
  - for \(a \in O\), \(\text{confine}(a) = (a_L, a_U) \in SC_I \times SC_I\) with \(a_L \leq_I a_U\)
    - Interpretation: for \(a \in O\), if \(x \leq_I a_U\), info can flow from \(x\) to \(a\), and if \(a_L \leq_I x\), info can flow from \(a\) to \(x\)
    - So \(a_L\) lowest classification of info allowed to flow out of \(a\), and \(a_U\) highest classification of info allowed to flow into \(a\)
• Assumes: object can change security classes
  – So, variable can take on security class of its data
• Object \( x \) has security class \( x \) currently
• Note transitivity *not* required
• If information can flow from \( a \) to \( b \), then \( b \) dominates \( a \) under ordering of policy \( I \):
  \[
  (\forall a, b \in O)[ a \rightarrow b \Rightarrow a_L \leq_I b_U ]
  \]
Example 1

- $SC_I = \{ U, C, S, TS \}$, with $U \leq_I C$, $C \leq_I S$, and $S \leq_I TS$
- $a$, $b$, $c \in O$
  - $\text{confine}(a) = [C, C]$
  - $\text{confine}(b) = [S, S]$
  - $\text{confine}(c) = [TS, TS]$
- Secure information flows: $a \rightarrow b$, $a \rightarrow c$, $b \rightarrow c$
  - As $a_L \leq_I b_U$, $a_L \leq_I c_U$, $b_L \leq_I c_U$
  - Transitivity holds
Example 2

- $SC_I, \leq_I$ as in Example 1
- $x, y, z \in O$
  - $\text{confine}(x) = [C, C]$
  - $\text{confine}(y) = [S, S]$
  - $\text{confine}(z) = [C, TS]$
- Secure information flows: $x \rightarrow y$, $x \rightarrow z$, $y \rightarrow z$, $z \rightarrow x$, $z \rightarrow y$
  - As $x_L \leq_I y_U$, $x_L \leq_I z_U$, $y_L \leq_I z_U$, $z_L \leq_I x_U$, $z_L \leq_I y_U$
  - Transitivity does not hold
    - $y \rightarrow z$ and $z \rightarrow x$, but $y \rightarrow x$ is false, because $y_L \leq_I x_U$ is false
Transitive Non-Lattice Policies

• $Q = (S_Q, \leq_Q)$ is a quasi-ordered set when $\leq_Q$ is transitive and reflexive over $S_Q$

• How to handle information flow?
  – Define a partially ordered set containing quasi-ordered set
  – Add least upper bound, greatest lower bound to partially ordered set
  – It’s a lattice, so apply lattice rules!
In Detail …

- $\forall x \in S_Q$: let $f(x) = \{ y \mid y \in S_Q \land y \leq_Q x \}$
  - Define $S_{QP} = \{ f(x) \mid x \in S_Q \}$
  - Define $\leq_{QP} = \{ (x, y) \mid x, y \in S_Q \land x \subseteq y \}$
    - $S_{QP}$ partially ordered set under $\leq_{QP}$
    - $f$ preserves order, so $y \leq_Q x$ iff $f(x) \leq_{QP} f(y)$

- Add upper, lower bounds
  - $S_{QP}' = S_{QP} \cup \{ S_Q, \emptyset \}$
  - Upper bound $ub(x, y) = \{ z \mid z \in S_{QP} \land x \subseteq z \land y \subseteq z \}$
  - Least upper bound $lub(x, y) = \cap ub(x, y)$
    - Lower bound, greatest lower bound defined analogously
And the Policy Is …

- Now \((S_{QP}', \leq_{QP})\) is lattice
- Information flow policy on quasi-ordered set emulates that of this lattice!
Non-transitive Flow Policies

• Government agency information flow policy (on next slide)

• Entities public relations officers PRO, analysts A, spymasters S
  – $\text{confine}(\text{PRO}) = \{ \text{public, analysis} \}$
  – $\text{confine}(A) = \{ \text{analysis, top-level} \}$
  – $\text{confine}(S) = \{ \text{covert, top-level} \}$
Information Flow

- By confinement flow model:
  - \( \text{PRO} \leq A, A \leq \text{PRO} \)
  - \( \text{PRO} \leq S \)
  - \( A \leq S, S \leq A \)

- Data cannot flow to public relations officers; not transitive
  - \( S \leq A, A \leq \text{PRO} \)
  - \( S \leq \text{PRO} \) is false
Transforming Into Lattice

• Rough idea: apply a special mapping to generate a subset of the power set of the set of classes
  – Done so this set is partially ordered
  – Means it can be transformed into a lattice

• Can show this mapping preserves ordering relation
  – So it preserves non-orderings and non-transitivity of elements corresponding to those of original set
Dual Mapping

- $R = (SC_R, \leq_R, join_R)$ reflexive info flow policy
- $P = (S_P, \leq_P)$ ordered set
  - Define dual mapping functions $l_R, h_R: SC_R \rightarrow S_P$
    - $l_R(x) = \{ x \}$
    - $h_R(x) = \{ y \mid y \in SC_R \land y \leq_R x \}$
  - $S_P$ contains subsets of $SC_R$; $\leq_P$ subset relation
  - Dual mapping function order preserving iff
    $$(\forall a, b \in SC_R)[ a \leq_R b \iff l_R(a) \leq_P h_R(b) ]$$
Theorem

Dual mapping from reflexive info flow policy $R$ to ordered set $P$ order-preserving

Proof sketch: all notation as before

$(\Rightarrow)$ Let $a \leq_R b$. Then $a \in l_R(a)$, $a \in h_R(b)$, so
\[ l_R(a) \subseteq h_R(b), \text{ or } l_R(a) \leq_P h_R(b) \]

$(\Leftarrow)$ Let $l_R(a) \leq_P h_R(b)$. Then $l_R(a) \subseteq h_R(b)$.
But $l_R(a) = \{ a \}$, so $a \in h_R(b)$, giving $a \leq_R b$
Info Flow Requirements

- Interpretation: let $\text{confine}(x) = \{ x_L, x_U \}$, consider class $\mathcal{Y}$
  - Information can flow from $x$ to element of $\mathcal{Y}$ iff $x_L \leq_R \mathcal{Y}$, or $l_R(x_L) \subseteq h_R(\mathcal{Y})$
  - Information can flow from element of $\mathcal{Y}$ to $x$ iff $\mathcal{Y} \leq_R x_U$, or $l_R(\mathcal{Y}) \subseteq h_R(x_U)$
Revisit Government Example

- Information flow policy is $R$
- Flow relationships among classes are:
  
  $\text{public} \leq_R \text{public}$
  
  $\text{public} \leq_R \text{analysis}$
  $\text{analysis} \leq_R \text{analysis}$
  
  $\text{public} \leq_R \text{covert}$
  $\text{covert} \leq_R \text{covert}$
  
  $\text{public} \leq_R \text{top-level}$
  $\text{covert} \leq_R \text{top-level}$
  $\text{analysis} \leq_R \text{top-level}$
  $\text{top-level} \leq_R \text{top-level}$
Dual Mapping of $R$

- Elements $l_R, h_R$:
  
  $l_R(\text{public}) = \{ \text{public} \}$
  
  $h_R(\text{public}) = \{ \text{public} \}$
  
  $l_R(\text{analysis}) = \{ \text{analysis} \}$
  
  $h_R(\text{analysis}) = \{ \text{public, analysis} \}$
  
  $l_R(\text{covert}) = \{ \text{covert} \}$
  
  $h_R(\text{covert}) = \{ \text{public, covert} \}$
  
  $l_R(\text{top-level}) = \{ \text{top-level} \}$
  
  $h_R(\text{top-level}) = \{ \text{public, analysis, covert, top-level} \}$
confine

- Let $p$ be entity of type PRO, $a$ of type A, $s$ of type S
- In terms of $P$ (not $R$), we get:
  - $\text{confine}(p) = \{ \{ \text{public} \}, \{ \text{public, analysis} \} \}$
  - $\text{confine}(a) = \{ \{ \text{analysis} \}, \{ \text{public, analysis, covert, top-level} \} \}$
  - $\text{confine}(s) = \{ \{ \text{covert} \}, \{ \text{public, analysis, covert, top-level} \} \}$
And the Flow Relations Are …

- \( p \to a \) as \( l_R(p) \subseteq h_R(a) \)
  - \( l_R(p) = \{ \text{public} \} \)
  - \( h_R(a) = \{ \text{public, analysis, covert, top-level} \} \)

- Similarly: \( a \to p, p \to s, a \to s, s \to a \)

- **But** \( s \to p \) is false as \( l_R(s) \not\subseteq h_R(p) \)
  - \( l_R(s) = \{ \text{covert} \} \)
  - \( h_R(p) = \{ \text{public, analysis} \} \)
Analysis

• \( (S_P, \leq_P) \) is a lattice, so it can be analyzed like a lattice policy

• Dual mapping preserves ordering, hence non-ordering and non-transitivity, of original policy
  – So results of analysis of \( (S_P, \leq_P) \) can be mapped back into \( (SC_R, \leq_R, join_R) \)
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

\begin{verbatim}
if \( x = 1 \) then \( y := a; \)
else \( y := b; \)
\end{verbatim}

- Info flows from \( x \) and \( a \) to \( y \), or from \( x \) and \( b \) to \( y \)
- Certified only if \( x \leq y \) and \( a \leq y \) and \( b \leq y \)
  - Note flows for both branches must be true unless compiler can determine that one branch will never be taken
Declarations

• Notation:

\[ x: \text{int class } \{ A, B \} \]

means \( x \) is an integer variable with security class at least \( \text{lub}\{ A, B \} \), so \( \text{lub}\{ A, B \} \leq x \)

• 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

\[ i_p : \text{type class} \{ i_p \} \]
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:
  \[ o_p: \text{type class} \{ r_1, \ldots, r_n \} \]
  where \( r_i \) is class of \( i \)th input or input/output argument
Example

```
proc sum(x: int class { A };
    var out: int class { A, B });
begin
    out := out + x;
end;
• Require $x \leq \text{out}$ and $\text{out} \leq \text{out}$
```
Array Elements

• Information flowing out:
  \[ \ldots := a[i] \]
  Value of \( i \), \( a[i] \) both affect result, so class is \( \text{lub}\{ a[i], i \} \)

• Information flowing in:
  \[ a[i] := \ldots \]
  Only value of \( a[i] \) affected, so class is \( a[i] \)
Assignment Statements

\[ x := y + z; \]

- Information flows from \( y, z \) to \( x \), so this requires \( \text{lub}(y, z) \leq x \)

More generally:

\[ y := f(x_1, \ldots, x_n) \]

- the relation \( \text{lub}(x_1, \ldots, x_n) \leq y \) must hold
Compound Statements

\[ x := y + z; \quad a := b * c - x; \]

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

More generally:

\[ S_1; \cdots ; S_n; \]

- Each individual \( S_i \) must be secure
Conditional Statements

if \( x + y < z \) then \( a := b \) else \( d := b \cdot c - x \); end

• The statement executed reveals information about \( x, y, z \), so \( lub(x, y, z) \leq glb(a, d) \)

More generally:

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

• \( S_1, S_2 \) must be secure

• \( lub(x_1, \ldots, x_n) \leq glb(y \mid y \text{ target of assignment in } S_1, S_2) \)
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:

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

- Loop must terminate;
- $S$ must be secure
- $lub(x_1, \ldots, x_n) \leq$ $glb(y | y \text{ target of assignment in } S)$
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 int class {x});
    var y: array[1..10][1..10] of int class {y});
var i, j: int {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;

May 13, 2013  ECS 235B Spring Quarter 2013  Slide #53
IFDs

• 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
  – $lub(x_{i1}, \ldots, x_{in}) \leq glb\{y \mid y\text{ target of assignment in } B_i\}$
Example of Requirements

• Within each basic block:

\[ b_1: \text{Low} \leq i \quad b_3: \text{Low} \leq j \quad b_6: \text{lub}\{\text{Low}, i\} \leq i \]

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

– Combining, \( \text{lub}(x[i][j], i, j) \leq y[j][i] \)

– From declarations, true when \( \text{lub}(x, i) \leq y \)

• \( B_2 = \{b_3, b_4, b_5, b_6\} \)

  – Assignments to \( i, j, y[j][i] \); conditional is \( i \leq 10 \)

  – Requires \( i \leq \text{glb}(i, j, y[j][i]) \)

  – From declarations, true when \( i \leq y \)
Example (continued)

• $B_4 = \{ b_5 \}$
  – Assignments to $j, y[j][i]$; conditional is $j \leq 10$
  – Requires $j \leq \text{glb}(j, y[j][i])$
  – From declarations, means $i \leq y$

• Result:
  – Combine $\text{lub}(x, i) \leq y; i \leq y; i \leq y$
  – Requirement is $\text{lub}(x, i) \leq y$