Lesson 17

• Loops in Control Flow Graphs
  – Convergence speed of data-flow analysis
• Region based Data-Flow analysis
• Symbolic analysis
Determining Loops in Flow Graphs: Dominators

• **Dominator**: \( d \text{ dom} \ n \)
  
  – Node \( d \) of a CFG dominates node \( n \) if every path from the entry node to \( n \) goes through \( d \)
  
  – The loop entry dominates all nodes in the loop

• The **immediate dominator** \( m \) of a node \( n \) is the last dominator of \( n \) on any path from the initial node to \( n \)
  
  – If \( d \neq n \) and \( d \text{ dom} \ n \) then \( d \text{ dom} \ m \)

• Since each node has a unique **immediate dominator**, dominators form a tree
Dominator Tree of a CFG
Data-Flow Analysis for Dominators

• The set of dominators for each node $n$, $D(n)$, can be computed with dataflow analysis

• Fact: $d \, \text{dom} \, n$ iff $d = n$ or $d \, \text{dom} \, m$ for all $m$ in $\text{pred}(n)$
  – Direction: $forwards$
  – Semilattice: powerset of CFG nodes
  – Transfer function: $f_{B}(x) = x \cup \{B\}$
  – Meet operator: intersection ($must$)
  – Boundary: $\text{OUT}[ENTRY] = \{ENTRY\}$
  – Initialization: $\text{OUT}[B] = \text{all Nodes}$
Natural Loops

• A *back edge* in a CFG is an edge $a \rightarrow b$ where $b$ dominates $a$

• A *natural loop*:
  – has a single-entry node $d$, the *header*, which dominates all nodes in the loop
  – has a *back edge* that enters node $d$

• Given a back edge $n \rightarrow d$
  – Its *natural loop* consists of $d$ plus the nodes that can reach $n$ without going through $d$
  – The *loop header* is node $d$
Reducible Flow Graphs

• A flow graph is *reducible* if and only if deleting all back edges the resulting graph is acyclic.

• **We consider only CFGs which are reducible.**

Example of a reducible CFG

Example of a nonreducible CFG
(not a natural loop: no back edge to dominator 1)
Natural Inner/Outer Loops

• In reducible CFGs, unless two loops have the same header, they are disjoint or one is nested within the other
• A nested loop is an \textit{inner loop} if it contains no other loops
• A loop is an \textit{outer loop} if it is not contained within another loop
Natural Inner/Outer Loops Example

CFG

Dominator tree

Natural loop for 1 \textit{dom} 9

Natural loop for 3 \textit{dom} 4

Natural loop for 4 \textit{dom} 7

Natural loop for 7 \textit{dom} 10

Natural loop for 3 \textit{dom} 8

CFG
Depth of a Control Flow Graph

- Depth: *largest number of back edges in any acyclic path in the graph*
- Intuition: not larger than the maximal nesting of loops
Speed of convergence of data-flow analysis

• Maximum number of iterations: (height of the lattice) \( x \) (number of nodes)

• If value of interest can be propagated along acyclic path (like for *reaching definitions, available expressions, live variables*), few passes are sufficient in general, depending on number of loop nesting (typically, depth of CFG + 1).

• Otherwise, several iterations in loops might be needed: eg. constant folding

```
L:  x = y;
y = z;
z = 1;
goto L
```
Region-Based Analysis

• In dataflow analysis, transfer functions are associated with basic blocks
• Here we associate them with regions, which provide a hierarchical view on the program
• Proceeds from smaller to larger regions, up to entire procedures
• Need more algebraic structure:
  – Semilattice of values
  – Semilattice of transfer functions with meet, composition and closure operator
Regions

• A *region* is a portion of the flow graph with a single entry point.
  – A single statement of a high-level language is a region
  – Each block or other form of statement nesting is a region
• A *region* is a collection of nodes $N$ and edges $E$ such that
  – $N$ has a dominator $h$
  – No node external of $N$ can reach a node $m$ in $N$ without passing through $h$
  – $E$ contains all edges between nodes in $N$ (but, possibly, for some to $h$)
• Note: *natural loops* are regions, but regions may not contain loops
Region hierarchies

• Assumption: the CFG is reducible (thus natural loops are disjoint or nested)

• Building the region hierarchy for the CFG:
  – Every block is a leaf region
  – For each natural loop $L$, starting from the innermost, replace the body (all nodes and edges but for back edges to the header) of $L$ with a new node representing a region $R$. All edges from $L$ become edges from $R$, possibly loops. $R$ is a body region.
  – Construct the loop region $R'$, that is identical to $R$ but without the loop: it represents the whole $L$
  – Finally construct one region for the resulting acyclic flow graph, if needed
Example: the region hierarchy of a CFG
Region based analysis: idea

• We define transfer functions for regions, exploiting the hierarchy

• One transfer function for each region $R$ and subregion $R'$: $f_{R, \text{IN}[R']}$ summarizes the effect of all possible paths from the entry of $R$ to $R'$.

• One transfer function for each exit block $B$ in $R$: $f_{R, \text{OUT}[B]}$ summarizes all paths from the entry of $R$ to the exit of $B$.

• Move upwards:
  – For leaf regions, $f_{B, \text{IN}[B]}$ is the identity and $f_{B, \text{OUT}[B]}$ is the transfer function of the block
  – For body regions, they are an acyclic graph of subregions: compose the transfer functions in topological order (see later)
  – For loop regions, one has to take into account only the back edges to the header (see later)

• For the data-flow values, proceed from the top region to the leaves. Compute values at entry, then to the entry of sub regions, and so on.
Needed properties of transfer functions

• **Examples from reaching definitions:**
  \[ f(x) = \text{gen} \cup (x - \text{kill}) \]

• **Composition**, for block/region sequences
  \[
  f_2 \circ f_1(x) = \text{gen}_2 \cup \left( (\text{gen}_1 \cup (x - \text{kill}_1)) - \text{kill}_2 \right) \\
  = (\text{gen}_2 \cup (\text{gen}_1 - \text{kill}_2)) \cup (x - (\text{kill}_1 \cup \text{kill}_2))
  \]

• **Meet**, to combine transfer function along different paths to the same point
  \[
  (f_1 \land f_2)(x) = f_1(x) \land f_2(x) \\
  = (\text{gen}_1 \cup (x - \text{kill}_1)) \cup (\text{gen}_2 \cup (x - \text{kill}_2)) \\
  = (\text{gen}_1 \cup \text{gen}_2) \cup (x - (\text{kill}_1 \cap \text{kill}_2))
  \]
Needed properties of transfer functions

- **Closure** is needed for loops. Represents the effect of going around the cycle any number of times

\[ f^* = I \land \left( \bigwedge_{n>0} f^n \right). \]

- For reaching definitions:

\[
\begin{align*}
f^2(x) &= f(f(x)) \\
&= \left( \text{gen} \cup ((\text{gen} \cup (x - \text{kill})) - \text{kill}) \right) \\
&= \text{gen} \cup (x - \text{kill})
\end{align*}
\]

\[
\begin{align*}
f^3(x) &= f(f^2(x)) \\
&= \text{gen} \cup (x - \text{kill})
\end{align*}
\]

\[
\begin{align*}
f^*(x) &= I \land f^1(x) \land f^2(x) \land \ldots \\
&= x \cup (\text{gen} \cup (x - \text{kill})) \\
&= \text{gen} \cup x
\end{align*}
\]
Composing transfer functions in body regions

• If $R$ is a body region:

1) for (each subregion $S$ immediately contained in $R$, in topological order) {

2) $f_{R,\text{IN}[S]} = \bigwedge$ predecessors $B$ in $R$ of the header of $S$ $f_{R,\text{OUT}[B]}$;

/* if $S$ is the header of region $R$, then $f_{R,\text{IN}[S]}$ is the meet over nothing, which is the identity function */

3) for (each exit block $B$ in $S$)

4) $f_{R,\text{OUT}[B]} = f_{S,\text{OUT}[B]} \circ f_{R,\text{IN}[S]}$;

}
Region-based analysis of reaching definitions

\[ d_1: i = m-1 \]
\[ d_2: j = n \]
\[ d_4: i = i+1 \]
\[ d_5: a = u2 \]
\[ d_6: j = u3 \]

Transfer Function

|       | \( f_{R_6, \text{IN}[R_2]} \) | \( f_{R_6, \text{OUT}[R_2]} \) | \( f_{R_6, \text{IN}[R_3]} \) | \( f_{R_6, \text{OUT}[R_3]} \) | \( f_{R_6, \text{IN}[R_4]} \) | \( f_{R_6, \text{OUT}[R_4]} \) | \( f_{R_7, \text{IN}[R_6]} \) | \( f_{R_7, \text{OUT}[R_6]} \) | \( f_{R_7, \text{IN}[R_8]} \) | \( f_{R_8, \text{OUT}[R_8]} \) | \( f_{R_8, \text{IN}[R_1]} \) | \( f_{R_8, \text{OUT}[R_1]} \) | \( f_{R_8, \text{IN}[R_7]} \) | \( f_{R_8, \text{OUT}[R_7]} \) | \( f_{R_8, \text{IN}[R_5]} \) | \( f_{R_8, \text{OUT}[R_5]} \) | \( \emptyset \) |
|-------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| \( R_6 \) | \( I \)       | \( f_{R_2, \text{OUT}[R_2]} \) \( f_{R_6, \text{IN}[R_2]} \) | \( f_{R_6, \text{OUT}[R_2]} \) | \( f_{R_6, \text{IN}[R_3]} \) | \( f_{R_6, \text{OUT}[R_3]} \) | \( f_{R_6, \text{IN}[R_4]} \) | \( f_{R_6, \text{OUT}[R_4]} \) | \( f_{R_7, \text{IN}[R_6]} \) | \( f_{R_7, \text{OUT}[R_6]} \) | \( f_{R_7, \text{IN}[R_8]} \) | \( f_{R_8, \text{OUT}[R_8]} \) | \( f_{R_8, \text{IN}[R_1]} \) | \( f_{R_8, \text{OUT}[R_1]} \) | \( f_{R_8, \text{IN}[R_7]} \) | \( f_{R_8, \text{OUT}[R_7]} \) | \( f_{R_8, \text{IN}[R_5]} \) | \( f_{R_8, \text{OUT}[R_5]} \) | \( \emptyset \) |
| \( R_7 \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) |
| \( R_8 \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) | \( \emptyset \) |
Composing transfer functions in loop regions

• If $R$ is a loop region:

1) let $S$ be the body region immediately nested within $R$; that is, $S$ is $R$ without back edges from $R$ to the header of $R$;
2) $f_{R,\text{IN}[S]} = (\bigwedge\text{predecessors } B \text{ in } R \text{ of the header of } S \ f_{S,\text{OUT}[B]})^*$;
3) for (each exit block $B$ in $R$)
4) $f_{R,\text{OUT}[B]} = f_{S,\text{OUT}[B]} \circ f_{R,\text{IN}[S]}$;

• $f_{R,\text{IN}[S]}$ represents the effect of executing any path from the entry of $R$ to the entry of $S$, after executing any number of times the loop (possibly 0)
Computing data-flow values in a top-down pass

- The transfer functions associated with regions are used to compute values at the beginning of each region:

  (a) \( \text{IN}[R_n] = \text{IN}[\text{ENTRY}] \).  
  
  (b) For each region \( R \) in \( \{R_1, \ldots R_{n-1}\} \), in the top-down order, compute \( \text{IN}[R] = f_{R', \text{IN}[R]}(\text{IN}[R']) \), where \( R' \) is the immediate enclosing region of \( R \).
The reaching definition running example

\[
\begin{align*}
R_8 & \rightarrow R_1 \\
R_6 & \rightarrow R_2\rightarrow R_3\rightarrow R_4\rightarrow R_5
\end{align*}
\]

Values computed by Region Based reaching definition analysis
Region-Based Symbolic Analysis

• The analysis identifies program variables whose value can be expressed as affine expressions (~ linear combinations) of certain reference variables, and it returns such expressions

• Reference variables can be
  – loop control variables
  – variables holding values returned by functions or read from input

• Affine expressions can also refer to iteration counts

• Induction variables are those expressible as $a*i + b$, with $i =$ count of iterations
Symbolic Analysis: Motivations

• The identification of induction variables and of enables various kinds of optimizations
  – Values can be computed with addition or shift (not multiplication)
  – Access to array elements can be parallelized if they are distinct
  – Loop invariants and constants can be identified as degenerate affine expression over loop indexes

1) \( x = \text{input}(); \)
2) \( y = x-1; \)
3) \( z = y-1; \)
4) \( A[x] = 10; \)
5) \( A[y] = 11; \)
6) \( \text{if} \ (z > x) \)
7) \( z = x; \)

• \( x \) only reference variable
• With symbolic analysis we learn that \( y = x-1 \) and \( z = x-2 \)
• \textit{Thus the assignments to} \( A \) \textit{are at distinct locations}
• \textit{And the last statement is never executed}
Sample program and corresponding CFG (with regions, after some transformations)

1) \( a = 0; \)
2) \( \text{for } (f = 100; f < 200; f++) \) {
   3) \( a = a + 1; \)
   4) \( b = 10 \times a; \)
   5) \( c = 0; \)
   6) \( \text{for } (g = 10; g < 20; g++) \) {
      7) \( d = b + c; \)
      8) \( c = c + 1; \)
   }

- \( f \) and \( g \) are inductive variables.
- Can be expressed as
  - \( f = i + 99 \)
  - \( g = j + 9 \)
  with \( i, j \) iteration counters

- The control variables \( f \) and \( g \) are replaced by iteration counters
- \( for \) loops are transformed in \textit{repeat until
Data-flow analysis: the domain

• **Domain** of data-flow values: the map lattice 
  \((Vars \rightarrow AffExp, \wedge_s)\) of *Symbolic maps*, where
  – \(Vars\) is the set of variable of the program
  – \((AffExp, \wedge_s)\) is the flat semilattice of all affine expressions of reference variables with NAA as bottom (representing “non-affine expression”)

• By definition of *map lattice* we have that
  – The meet is defined by \((f \wedge f')(x) = f(x) \wedge f'(x)\)
  – The ordering is \(f \leq f' \iff \forall x, f(x) \leq f'(x)\)
  – The bottom value is the map mapping all variables to NAA
Data-flow analysis:
Transfer functions of statements

• Transform a symbolic map, according to the semantics of the statement

1. If $s$ is not an assignment statement, then $f_s$ is the identity function.
2. If $s$ is an assignment statement to variable $x$, then

$$f_s(m)(x) = \begin{cases} 
m(v) & \text{for all variables } v \neq x \\
c_0 + c_1 m(y) + c_2 m(z) & \text{if } x \text{ is assigned } c_0 + c_1 y + c_2 z, \\
& (c_1 = 0, \text{ or } m(y) \neq \text{NAA}), \text{ and} \\
& (c_2 = 0, \text{ or } m(z) \neq \text{NAA}) \\
\text{NAA} & \text{otherwise.}
\end{cases}$$

"$c_0 + c_1 y + c_2 z$" represents any affine expression involving variables of the program.
Data-flow analysis:
Composition of transfer functions

• Standard composition of linear combinations, if defined, otherwise NAA values are propagated

1. If \( f_2(m)(v) = \text{NAA} \), then \( (f_2 \circ f_1)(m)(v) = \text{NAA} \).

2. If \( f_2(m)(v) = c_0 + \sum_i c_i m(v_i) \), then

\[
(f_2 \circ f_1)(m)(v) = \begin{cases} 
\text{NAA,} & \text{if } f_1(m)(v_i) = \text{NAA for some } i \neq 0, c_i \neq 0 \\
c_0 + \sum_i c_i f_1(m)(v_i) & \text{otherwise}
\end{cases}
\]
Region Based Analysis: meet and closure of transfer functions

- For Region Based Analysis we should define the *meet* and *closure* of transfer functions

- **Meet**: The value of a variable for the meet of two functions is NAA unless it has the same value for both functions:

\[
(f_1 \land f_2)(m)(v) = \begin{cases} 
  f_1(m)(v) & \text{if } f_1(m)(v) = f_2(m)(v) \\
  \text{NAA} & \text{otherwise}
\end{cases}
\]

- **Closure**: Given a symbolic map \( m \) and a transfer function \( f \) (representing a single execution of a loop) we can summarize the effect of 0 or any number of execution of a loop only if \( m \) is loop invariant:

\[
f^*(m)(v) = \begin{cases} 
  m(v) & \text{if } f(m)(v) = m(v) \\
  \text{NAA} & \text{otherwise}
\end{cases}
\]
Region Based Analysis: closure is not enough

• For symbolic analysis, the closure operator on transfer functions, summarizing 0 or any number of execution of a loop is not informative enough
  – The symbolic map needs to be parametrized by the number of times a loop is executed
  – When the loop terminates, the number of iterations is used to determine the value of induction variables after the loop

• We need to compute the effect of composing a transfer function $g$ (the effect of one iteration) a fixed number of times: $g^0 = I, g^{i+1} = g \cdot g^i$

• Therefore we introduce parametrized (transfer) function composition
Region Based Analysis: parametrized function composition

- For a transfer function $g$ representing the execution of one cycle, we determine $g^i$ for all $i >= 0$
- Potential induction variables are of three kinds:
  1. if $g(m)(x) = m(x) + c$ with $c$ constant, then $g^i(m)(x) = m(x) + c$ $i$ [x is a basic induction variable]
  2. if $g(m)(x) = m(x)$, then $g^i(m)(x) = m(x)$ [x is a symbolic constant]
  3. if $g(m)(x) = c_o + c_1 m(x_1) + ... + c_n m(x_n)$ where each $x_k$ is basic induction variable or symbolic constant, then $g^i(m)(x) = c_o + c_1 g^i(m)(x_1) + ... + c_n g^i(m)(x_n)$ [x is a (non basic) induction variable]
  4. In all other cases $g^i(m)(x) = NAA$
Region Based Symbolic Analysis: the modified algorithm

• The algorithm is still made of two passes:
  – A bottom-up pass to compute the transfer functions for all regions
  – A top-down pass to compute the symbolic map at the entry of each region

• But for each loop region $R$ and body sub-region $S$, instead of $f_{R,IN[S]}$ (representing the effect of executing any number of times the loop) one computes $f_{R,i,IN[S]}$ (representing the effect of executing exactly $i$ times the loop)

$$f_{R,i,IN[S]} = \left( \bigwedge_{\text{predecessors } B \text{ in } R \text{ of the header of } S} f_{S,OUT[B]} \right)^{i-1}$$
Region Based Symbolic Analysis: the modified algorithm (2)

- If the number of iterations of a loop is known, the summary of the region is computed by replacing \( i \) with the actual count.
- In the top-down pass \( f_{R,i,\text{IN}[S]} \) is used to compute the symbolic map at the entry of the \( i \)-th iteration of a loop.
- To avoid that NAA variables penetrate into inner loops, if \( m(v) \) is used in an assignment in region \( R \) but \( m(v) = \text{NAA} \) at the entry of \( R \), the assignment \( t = v \) is added, and \( m(v) \) is replaced by \( t \) everywhere. Example:

```c
for (i = 1; i < n; i++) {
    a = input();
    for (j = 1; j < 10; j++) {
        a = a - 1;
        b = j + a;
        a = a + 1;
    }
}
```

```c
for (i = 1; i < n; i++) {
    a = input();
    t = a;
    for (j = 1; j < 10; j++) {
        a = t - 1;
        b = t - 1 + j;
        a = t;
    }
}
```
Sample program and corresponding CFG (with regions, after some transformations)

1)   a = 0;
2)   for (f = 100; f < 200; f++) {
3)       a = a + 1;
4)       b = 10 * a;
5)       c = 0;
6)   for (g = 10; g < 20; g++) {
7)           d = b + c;
8)           c = c + 1;
9)   }

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>$m(a)$</td>
<td>$m(b)$</td>
<td>$m(c)$</td>
</tr>
<tr>
<td>IN$[B_1]$</td>
<td>NAA</td>
<td>NAA</td>
<td>NAA</td>
</tr>
<tr>
<td>OUT$[B_1]$</td>
<td>0</td>
<td>NAA</td>
<td>NAA</td>
</tr>
<tr>
<td>IN$[B_2]$</td>
<td>$i - 1$</td>
<td>NAA</td>
<td>NAA</td>
</tr>
<tr>
<td>OUT$[B_2]$</td>
<td>$i$</td>
<td>$10i$</td>
<td>0</td>
</tr>
<tr>
<td>IN$[B_3]$</td>
<td>$i$</td>
<td>$10i$</td>
<td>$j - 1$</td>
</tr>
<tr>
<td>OUT$[B_3]$</td>
<td>$i$</td>
<td>$10i$</td>
<td>$j$</td>
</tr>
<tr>
<td>IN$[B_4]$</td>
<td>$i$</td>
<td>$10i$</td>
<td>$j$</td>
</tr>
<tr>
<td>OUT$[B_4]$</td>
<td>$i - 1$</td>
<td>$10i - 10$</td>
<td>$j$</td>
</tr>
</tbody>
</table>
Result of Region Based Analysis

1) $a = 0;$
2) for ($f = 100; f < 200; f++$) {
3)    $a = a + 1;$
4)    $b = 10 * a;$
5)    $c = 0;$
6)    for ($g = 10; g < 20; g++$) {
7)        $d = b + c;$
8)        $c = c + 1;$
7) }
8) }

1) $a = 0;$
2) for ($i = 1; i <= 100; i++$) {
3)    $a = i;$
4)    $b = 10*i;$
5)    $c = 0;$
6)    for ($j = 1; j <= 10; j++$) {
7)        $d = 10*i + j - 1;$
8)        $c = j;$
7) }
8) }

Modified program

<table>
<thead>
<tr>
<th>line</th>
<th>var</th>
<th>$i = 1$</th>
<th>$i = 2$</th>
<th>$1 \leq i \leq 100$</th>
<th>$i = 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>a</td>
<td>1</td>
<td>2</td>
<td>$i$</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>10</td>
<td>20</td>
<td>$10i$</td>
<td>1000</td>
</tr>
<tr>
<td>7</td>
<td>d</td>
<td>10,...,19</td>
<td>20,...,29</td>
<td>$10i,...,10i+9$</td>
<td>1000,...,1009</td>
</tr>
<tr>
<td>8</td>
<td>c</td>
<td>1,...,10</td>
<td>1,...,10</td>
<td>1,...,10</td>
<td>1,...,10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$m$</th>
<th>$m(a)$</th>
<th>$m(b)$</th>
<th>$m(c)$</th>
<th>$m(d)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$IN[B1]$</td>
<td>NAA</td>
<td>NAA</td>
<td>NAA</td>
<td>NAA</td>
</tr>
<tr>
<td>$OUT[B1]$</td>
<td>0</td>
<td>NAA</td>
<td>NAA</td>
<td>NAA</td>
</tr>
<tr>
<td>$IN[B2]$</td>
<td>$i - 1$</td>
<td>NAA</td>
<td>NAA</td>
<td>NAA</td>
</tr>
<tr>
<td>$OUT[B2]$</td>
<td>$i$</td>
<td>$10i$</td>
<td>0</td>
<td>NAA</td>
</tr>
<tr>
<td>$IN[B3]$</td>
<td>$i$</td>
<td>$10i$</td>
<td>$j - 1$</td>
<td>NAA</td>
</tr>
<tr>
<td>$OUT[B3]$</td>
<td>$i$</td>
<td>$10i$</td>
<td>$j$</td>
<td>$10i + j - 1$</td>
</tr>
<tr>
<td>$IN[B4]$</td>
<td>$i$</td>
<td>$10i$</td>
<td>$j$</td>
<td>$10i + j - 1$</td>
</tr>
<tr>
<td>$OUT[B4]$</td>
<td>$i - 1$</td>
<td>$10i - 10$</td>
<td>$j$</td>
<td>$10i + j - 11$</td>
</tr>
</tbody>
</table>

Sequence of values for the variables

Symbolic maps