Lecture 12

Region-Based Analysis

I. Basic Idea
II. Algorithm
III. Optimization and Complexity
IV. Comparing region-based analysis with iterative algorithms

Reading: ALSU 9.7
Motivation for Studying Region-Based Analysis

- **Exploit the structure of block-structured programs in data flow**
- **Tie in several concepts studied:**
  - Use of structure in induction variables, loop invariant
    - motivated by nature of the problem
    - *This lecture: can we use structure for speed?*
  - Iterative algorithm for data flow
    - *This lecture: an alternative algorithm*
  - Reducibility
    - all retreating edges of DFST are back edges
    - reducible graphs converge quickly
    - *This lecture: algorithm exploits & requires reducibility*
- **Usefulness in practice**
  - Faster for “harder” analyses
  - Useful for analyses related to structure
- **Theoretically interesting: better understanding of data flow**
I. Big Picture

Diagram of interconnected nodes labeled B3, B1, B2, and B4.
Basic Idea

• **In Iterative Analysis:**
  - DEFINITION: Transfer function $F_B$:
    summarize effect from beginning to end of basic block $B$

• **In Region-Based Analysis:**
  - DEFINITION: Transfer function $F_{R,B}$:
    summarize effect from beginning of $R$ to end of basic block $B$

• Recursively
  - construct a larger region $R$ from smaller regions
  - construct $F_{R,B}$ from transfer functions for smaller regions
    until the program is one region

• Let $P$ be the region for the entire program, and $v$ be initial value at entry node
  - $\text{out}[B] = F_{P,B}(v)$
  - $\text{in}[B] = \land_{B'} \text{out}[B']$, where $B'$ is a predecessor of $B$
II. Algorithm

1. Operations on transfer functions

2. How to build nested regions?

3. How to construct transfer functions that correspond to the larger regions?
1. Operations on Transfer Functions

- **Example: Reaching Definitions**

- \( F(x) = \text{Gen} \cup (x - \text{Kill}) \)

- \( F_2(F_1(x)) = \text{Gen}_2 \cup (F_1(x) - \text{Kill}_2) \)
  = \( \text{Gen}_2 \cup (\text{Gen}_1 \cup (x - \text{Kill}_1)) - \text{Kill}_2 \)
  = \( \text{Gen}_2 \cup (\text{Gen}_1 - \text{Kill}_2) \cup (x - (\text{Kill}_1 \cup \text{Kill}_2)) \)

- \( F_1(x) \land F_2(x) = \text{Gen}_1 \cup (x - \text{Kill}_1) \cup \text{Gen}_2 \cup (x - \text{Kill}_2) \)
  = \( (\text{Gen}_1 \cup \text{Gen}_2) \cup (x - (\text{Kill}_1 \cap \text{Kill}_2)) \)

- \( F^*(x) \leq F^n(x), \forall \ n \geq 0 \)
  = \( x \cup F(x) \cup F(F(x)) \cup ... \)
  = \( x \cup (\text{Gen} \cup (x - \text{Kill})) \cup (\text{Gen} \cup ((\text{Gen} \cup (x - \text{Kill})) - \text{Kill})) \cup ... \)
  = \( \text{Gen} \cup (x - \emptyset) \)
2. Structure of Nested Regions (An Example)

• A **region** in a flow graph is a set of nodes that
  – includes a **header**, which **dominates all other nodes in a region**

• **T1-T2 rule** (Hecht & Ullman)
  • **T1**: Remove a loop
    If \( n \) is a node with a loop, i.e. an edge \( n \rightarrow n \), delete that edge

  • **T2**: Remove a vertex
    If there is a node \( n \) that has a **unique predecessor**, \( m \),
    then \( m \) may consume \( n \) by
    deleting \( n \) and making **all successors of \( n \)** be successors of \( m \).
In reduced graph:
- each vertex represents a subgraph of original graph (a region).
- each edge represents an edge in original graph

Limit flow graph: result of exhaustive application of T1 and T2
- independent of order of application.
- if limit flow graph has a single vertex $\Rightarrow$ reducible

Can define larger regions (e.g. Allen&Cocke’s intervals)
- simple regions $\Rightarrow$ simple composition rules for transfer functions
3. Transfer Functions for T2 Rule

- **Transfer function**
  
  - $F_{R,B}$: summarizes the effect from beginning of $R$ to end of $B$
  
  - $F_{R,in(H2)}$: summarizes the effect from beginning of $R$ to beginning of $H2$
    
    - Unchanged for blocks $B$ in region $R_1$ ($F_{R,B} = F_{R_1,B}$)
    
    - $F_{R,in(H2)} = \land_p F_{R,P}$, where $p$ is a predecessor of $H_2$
    
    - For blocks $B$ in region $R_2$: $F_{R,B} = F_{R_2,B} \cdot F_{R,in(H2)}$
Transfer Functions for T1 Rule

- **Transfer Function** $F_{R,B}$
  - $F_{R,\text{in}(H)} = (\land_p F_{R_1,p})^*$, where $p$ is a predecessor of $H$ in $R$
  - $F_{R,B} = F_{R_1,B} \cdot F_{R,\text{in}(H)}$
First Example

<table>
<thead>
<tr>
<th>R</th>
<th>T_1/T_2</th>
<th>R'</th>
<th>F_{R,in(R')}</th>
<th>F_{R,B1}</th>
<th>F_{R,B2}</th>
<th>F_{R,B3}</th>
<th>F_{R,B4}</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_1</td>
<td>T_2</td>
<td>B_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R_2</td>
<td>T_2</td>
<td>R_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R_3</td>
<td>T_1</td>
<td>R_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R_4</td>
<td>T_2</td>
<td>B_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **R**: region name
- **R'**: region whose header will be subsumed
First Example

- **R**: region name
- **R'**: region whose header will be subsumed

<table>
<thead>
<tr>
<th>R</th>
<th>T₁/T₂</th>
<th>R'</th>
<th>F_{R,in}(R')</th>
<th>F_{R,B1}</th>
<th>F_{R,B2}</th>
<th>F_{R,B3}</th>
<th>F_{R,B4}</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₁</td>
<td>T₂</td>
<td>B₂</td>
<td>F_{B1}</td>
<td>F_{B1}</td>
<td>F_{B2}·F_{R,in(B2)}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R₂</td>
<td>T₂</td>
<td>R₁</td>
<td>F_{B3}</td>
<td>F_{R1,B1}·F_{R,in(R1)}</td>
<td>F_{R1,B2}·F_{R,in(R1)}</td>
<td>F_{B3}</td>
<td></td>
</tr>
<tr>
<td>R₃</td>
<td>T₁</td>
<td>R₂</td>
<td>(F_{R2B1}∧F_{R2B2})*</td>
<td>F_{R2,B1}·F_{R,in(R2)}</td>
<td>F_{R2,B2}·F_{R,in(R2)}</td>
<td>F_{R2,B3}·F_{R,in(R2)}</td>
<td></td>
</tr>
<tr>
<td>R₄</td>
<td>T₂</td>
<td>B₄</td>
<td>F_{R3B3}∧F_{R3B2}</td>
<td>F_{R3,B1}</td>
<td>F_{R3,B2}</td>
<td>F_{R3,B3}</td>
<td>F_{B₄}·F_{R,in(B₄)}</td>
</tr>
</tbody>
</table>
### III. Complexity of Algorithm

![Diagram of algorithm complexity]

<table>
<thead>
<tr>
<th>R</th>
<th>$T_1/T_2$</th>
<th>$R'$</th>
<th>$F_{R,in(R')}$</th>
<th>$F_{R,B1}$</th>
<th>$F_{R,B2}$</th>
<th>$F_{R,B3}$</th>
<th>$F_{R,B4}$</th>
<th>$F_{R,B5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$T_2$</td>
<td>$B_2$</td>
<td>$F_{B2}$</td>
<td>$F_{B1} \cdot F_{B2}$</td>
<td>$F_{B2}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_2$</td>
<td>$T_2$</td>
<td>$R_1$</td>
<td>$F_{B3}$</td>
<td>$F_{R1,B1} \cdot F_{B3}$</td>
<td>$F_{R1,B2} \cdot F_{B3}$</td>
<td>$F_{B3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R_3$</td>
<td>$T_2$</td>
<td>$R_2$</td>
<td>$F_{B4}$</td>
<td>$F_{R2,B1} \cdot F_{B4}$</td>
<td>$F_{R2,B2} \cdot F_{B4}$</td>
<td>$F_{R2,B3} \cdot F_{B4}$</td>
<td>$F_{B4}$</td>
<td></td>
</tr>
<tr>
<td>$R_4$</td>
<td>$T_2$</td>
<td>$R_3$</td>
<td>$F_{B5}$</td>
<td>$F_{R3,B1} \cdot F_{B5}$</td>
<td>$F_{R3,B2} \cdot F_{B5}$</td>
<td>$F_{R3,B3} \cdot F_{B5}$</td>
<td>$F_{R4} \cdot F_{B5}$</td>
<td>$F_{B5}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>$F_{R4,in(R)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_4$</td>
<td>$I$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$F_{B5} \cdot F_{R4,in(R4)}$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$F_{B4} \cdot F_{R4,in(R3)}$</td>
</tr>
<tr>
<td>$R_1$</td>
<td>$F_{B3} \cdot F_{R4,in(R2)}$</td>
</tr>
<tr>
<td>$B_1$</td>
<td>$F_{B2} \cdot F_{R4,in(R1)}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>$F_{R4,B}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_5$</td>
<td>$F_{B5} \cdot I$</td>
</tr>
<tr>
<td>$B_4$</td>
<td>$F_{B4} \cdot F_{R4,in(R3)}$</td>
</tr>
<tr>
<td>$B_3$</td>
<td>$F_{B3} \cdot F_{R4,in(R2)}$</td>
</tr>
<tr>
<td>$B_2$</td>
<td>$F_{B2} \cdot F_{R4,in(R1)}$</td>
</tr>
<tr>
<td>$B_1$</td>
<td>$F_{B1} \cdot F_{R4,in(B1)}$</td>
</tr>
</tbody>
</table>
Optimization

• Let $m =$ number of edges, $n =$ number of nodes

• Ideas for optimization
  – If we compute $F_{R,B}$ for every region $B$ is in, then it is very expensive
  – We are ultimately only interested in the entire region $(E)$; we need to compute only $F_{E,B}$ for every $B$.
    • There are many common subexpressions between $F_{E,B_1}$, $F_{E,B_2}$, ...
    • Number of $F_{E,B}$ calculated = $m$
  – Also, we need to compute $F_{R,in(R')}$ where $R'$ represents the region whose header is subsumed.
    • Number of $F_{R,B}$ calculated, where $R$ is not final = $n$

• Total number of $F_{R,B}$ calculated: $(m + n)$
  – Data structure keeps “header” relationship
    • Practical algorithm: $O(m \log n)$
    • Complexity: $O(m\alpha(m,n))$, $\alpha$ is inverse Ackermann function
Reducibility

- If no T1, T2 is applicable before graph is reduced to single node, then **split node** and continue
- Worst case: exponential
- Most graphs (including GOTO programs) are reducible
IV. Comparison with Iterative Data Flow

• **Applicability**
  – Definitions of $F^*$ can make technique more powerful than iterative algorithms
  – **Backward flow**: reverse graph is not typically reducible.
    • Requires more effort to adapt to backward flow than iterative algorithm
  – More important for interprocedural optimization

• **Speed**
  – **Irreducible graphs**
    • Iterative algorithm can process irreducible parts uniformly
    • Serious “irreducibility” can be slow with region-based analysis
  – **Reducible graph & Cycles do not add information** (common)
    • Iterative: $(\text{depth} + 2)$ passes
      depth is 2.75 average, independent of code length
    • Region-based analysis: Theoretically almost linear, typically $O(m \log n)$
  – **Reducible & Cycles add information**
    • Iterative takes longer to converge
    • Region-based analysis remains the same