Lecture 13
Region-Based Analysis

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

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

Review: Dominance

A region in a flow graph is a set of nodes with a header that dominates all other nodes in a region

A region in a flow graph is a set of nodes with a header that dominates all other nodes in a region
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
    – out[$B$] = $F_{R,B}(v)$
    – in[$B$] = $A_{B'}$ 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

• Transfer function over a block:
  $$ F(x) = Gen \cup (x - Kill) $$

• Resulting transfer functions (after operations) must be consistent with this form:
  – same equation
  – updated values for $Gen$ and $Kill$ set parameters

Operations on Transfer Functions: Composition

$$ F_2(F_1(x)) = Gen_2 \cup (F_1(x) - Kill_2) $$

$$ = Gen_2 \cup (Gen_1 \cup (x - Kill_1)) - Kill_2 $$

$$ = Gen_2 \cup (Gen_1 \cup Kill_2 \cup (x - (Kill_1 \cup Kill_2))) $$

Input parameters

$F_1$ $F_2$ $F_3$ $F_2(F_1(x))$ $F_1$ $Gen_1$ $Kill_1$ $Gen_2$ $Kill_2$ $Gen_3$ $Kill_3$ $F_3(F_2(F_1(x)))$

Gen set after composition Kill set after composition
Operations on Transfer Functions: **Meet**

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

(Recall that for Reaching Definitions, \( \land = \cup \).)

**Example:**

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

Gen set after \( \land \) \hspace{1cm} Kill set after \( \land \)

Operations on Transfer Functions: **Closure**

For Reaching Definitions:

- Including the possible effects of the back edge, it may iterate 0, 1, 2, ..., \( \infty \) times.

\[
F^*(x) = \bigcup_{n=0}^{\infty} F(x) = x \land F(0) \land F(F(0)) \land \ldots
\]

What is the value at the input of the block?

**Example:**

\[
F^*(x) = x \land (\text{Gen} \cup \{x - \text{Kill}\}) \cup (\text{Gen} \cup \{\text{Gen} \cup \{x - \text{Kill}\} - \text{Kill}\}) \cup \ldots
\]

Gen set after closure \hspace{1cm} Kill set (after closure)

Recap of Operations on Transfer Functions

**For Reaching Definitions:**

- **Transfer Function** \( F(x) \):
  \[
  F(x) = \text{Gen} \cup \{x - \text{Kill}\}
  \]

- **Composition** \( F_2(F_1(x)) \):
  \[
  \text{Gen} = \text{Gen}_2 \cup \{\text{Gen}_1 - \text{Kill}_2\}
  \text{Kill} = \text{Kill}_1 \land \text{Kill}_2
  \]

- **Meet** \( F_1(x) \land F_2(x) \):
  \[
  \text{Gen} = \text{Gen}_1 \lor \text{Gen}_2
  \text{Kill} = \text{Kill}_1 \cap \text{Kill}_2
  \]

- **Closure** \( F^*(x) \):
  \[
  \text{Gen} = \text{Gen}
  \text{Kill} = \emptyset
  \]

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) for Reducible Flow Graphs
  - **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 \( m \) 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
  – reducible flow graph: limit flow graph has a single vertex

Transfer Functions for T1 Rule

R: new region (subsumes back edges from \( R_j \rightarrow R_i \))

Observations:
  – the header of \( R_j \) (i.e. \( H \)) is also the header of \( R \)
  – we already know how to get from \( H \) to \( B \) for every block \( B \) in \( R_j \): i.e. \( F_{R_j,B} \)
    - this will be the last step in getting from the new \( R \) to \( B \) (composition)
  – what’s new: we need to get from \( R \) to the input of \( H \), including back edges!
    - this involves both meet (\( \land \)) and closure (\( * \)) operations

Transfer Functions for T2 Rule

• Transfer function \( F_{R,B} \):
  - \( F_{R,B} \) summarizes the effect from beginning of \( R \) to end of \( B \)
  - \( F_{R,H2} \) summarizes the effect from beginning of \( R \) to beginning of \( H2 \)
    - Unchanged for blocks \( B \) in region \( R \): \( F_{R,B} = F_{R_1,B} \)
    - \( F_{R,H2} = F_{R_1,H2} \) where \( p \) is a predecessor of \( H_2 \)
    - For blocks \( B \) in region \( R \): \( F_{R,B} = F_{R_2,B} \cdot F_{R,H2} \)
Let $R, T_1, T_2, B_1, B_2, R', B_3, B_4, B_5$ be regions.

**Example**

- **Rule $R'$**: $R, in(R')$.
- **Data structure keeps “header” relationship**.
- **Practical algorithm**: $O(m \log n)$.
- **Complexity**: $O(m \mu(m, n))$, $\mu$ is inverse Ackermann function.

### Optimization

- Let $m = \text{number of edges}$, $n = \text{number of nodes}$.
- **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 $F_{E,R}$ for every $B$.
    - There are many common subexpressions between $F_{E,B_1}$ and $F_{E,B_2}$.
    - Number of $F_{E,B}$ calculated is $m$.
  - Also, we need to compute $F_{R',B}$, where $R'$ represents the region whose header is subsumed.
    - Number of $F_{R,B}$ calculated, where $R$ is not final is $n$.
  - Total number of $F_{R,B}$ calculated: $(m + n)$.
    - Data structure keeps “header” relationship.
    - Practical algorithm: $O(m \log n)$.
    - Complexity: $O(m \mu(m, n))$, $\mu$ is inverse Ackermann function.

### III. Complexity of Algorithm

- Worst case: exponential.
- Most graphs (including GOTO programs) are reducible.

### Reducibility

- If no $T_1$, $T_2$ is applicable before graph is reduced to single node, then split node (make $k$ copies of node, one per predecessor) and continue.
- T1: Remove a n->n loop.
- T2: Remove a vertex w/unique predecessor.
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

---

**Wednesday's Class**

- Register Allocation [ALSU 8.8]
- Assignment #2 due Wednesday midnight