5.5 DATA-FLOW ANALYSIS OF STRUCTURED FLOW GRAPHS

Introduction

Data-Flow Analysis is an important technique used in compiler design to collect information about how data values are defined, used, and changed during program execution.

It analyzes the flow of data through different basic blocks in a Control Flow Graph (CFG). The information obtained is mainly used for code optimization.

A structured flow graph represents the control flow of a structured program containing sequences, conditional statements, and loops.

5.5.1 Data-Flow Analysis

In data-flow analysis, a program is divided into basic blocks. These basic blocks are connected according to the flow of control.

For each basic block, two important values are calculated:

- IN[B]: Information available before entering basic block B.
- OUT[B]: Information available after executing basic block B.

The general data-flow equation is:

\[OUT[B] = GEN[B] \cup (IN[B] - KILL[B]) \]

Where:

- GEN[B]: Information generated by block B.
- KILL[B]: Information made invalid by block B.
- IN[B]: Information entering the block.
- OUT[B]: Information leaving the block.

Example:

Consider the following basic blocks:

```
    B1
a = 10
b = 20
  |
  v
  B2
```

c = a + b

|

v

B3

d = c * 2

|

v

Exit

Explanation:

- In B1, the values of a and b are generated.
- These values flow into B2, where they are used to calculate c.
- The value of c flows into B3 and is used to calculate d.

Thus, data-flow analysis tracks how values move from one basic block to another.

5.5.2 Data Flow in Structured Flow Graphs

Structured flow graphs mainly contain three types of control structures.

1. Sequential Flow

In sequential flow, one basic block is executed after another.

B1

|

v

B2

|

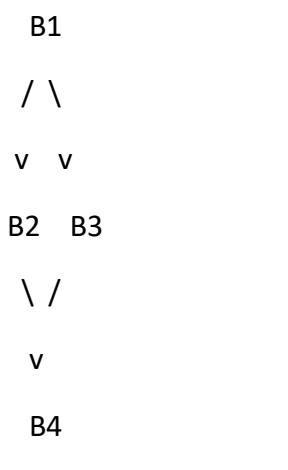
v

B3

- The output of B1 becomes the input of B2.
- The output of B2 becomes the input of B3.
- Blocks are executed in a fixed order.

2. Conditional Flow

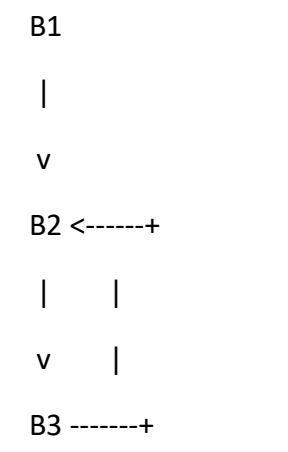
In conditional flow, the program can follow different paths depending on a condition.



- B1 branches into B2 and B3.
- Depending on the condition, either B2 or B3 is executed.
- The information from both paths is combined before entering B4.

3. Loop Flow

A loop contains a backward control-flow edge.



- The program executes the loop body repeatedly.
- Data-flow information may pass through the loop several times.
- Analysis continues until the information becomes stable.

5.5.3 Types of Data-Flow Analysis

Some important types of data-flow analysis are:

1. Reaching Definitions: Determines which definitions of variables can reach a particular statement.
2. Available Expressions: Determines which expressions have already been computed and are still valid.

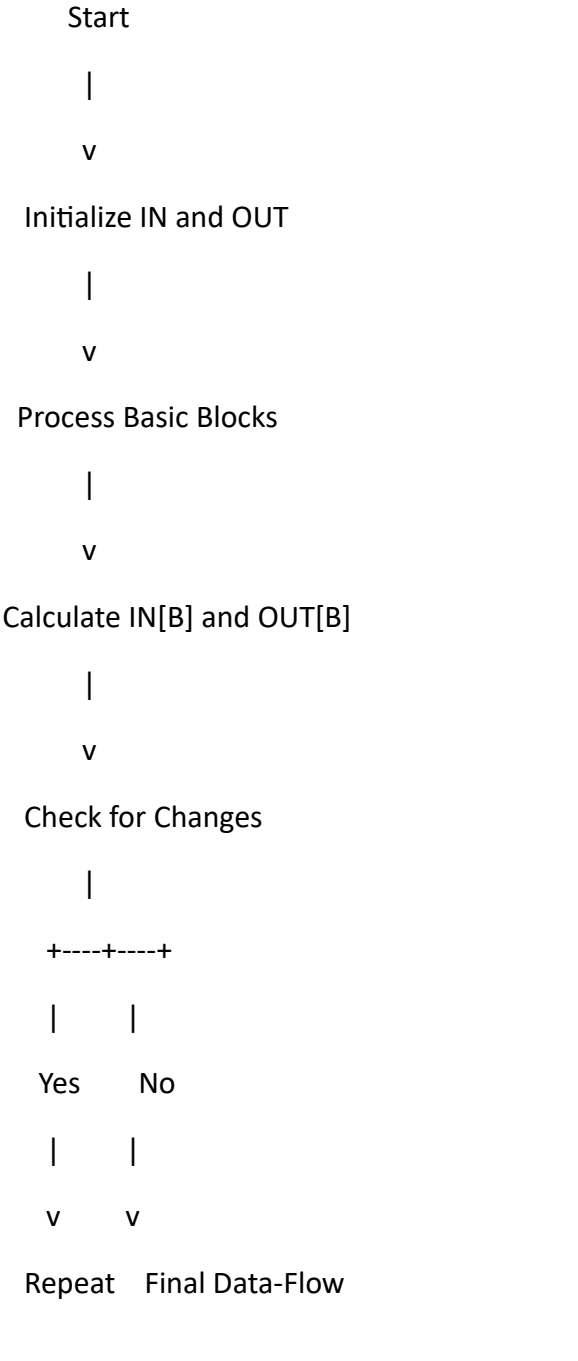
3. Live Variable Analysis: Determines whether the value of a variable will be used later in the program.
4. Constant Propagation: Determines variables whose values are known constants.
5. Copy Propagation: Tracks values copied from one variable to another.

5.5.4 Iterative Data-Flow Analysis

Data-flow information is usually calculated using an iterative process.

Initially, the IN and OUT values are assigned suitable starting values. Then each basic block is processed repeatedly until no further changes occur.

Steps:



Information

Explanation:

1. Initialize the IN and OUT values of all basic blocks.
2. Process each basic block.
3. Calculate the new IN and OUT values.
4. Check whether any values have changed.
5. If changes occur, repeat the process.
6. If no changes occur, stop the analysis.

The process stops when there is no further change in the data-flow information. This condition is called a Fixed Point.

Applications of Data-Flow Analysis

Data-flow analysis is useful for:

1. Dead Code Elimination.
2. Common Subexpression Elimination.
3. Constant Propagation.
4. Copy Propagation.
5. Register Allocation.
6. Loop Optimization.
7. Code Improvement.