Control Flow in Programs
(Part I: Sequential and Conditional control)

Instructor

Institute

Outline

Topic

What is a Program?

- **Program**: A sequence **instructions** interleaved with locations

```
0
  a = 3
1
  b = read()
2
  c = a // b
3
  d = c - a
4
  # end
```

What is control flow analysis?

```
0
  a = 3
1
  b = read()
2
  c = a // b
3
  d = c - a
4
  # end
```

- Control Flow analysis is the examination of possible paths a program can take when it runs. These paths are called **executions**.
- Control Flow analysis is done **without** running the program, i.e., statically.

Structural Abstraction

Program

```
0
  a = 3
1
  b = read()
2
  c = a // b
3
  d = c - a
4
  # end
```

Structural Abstraction

```
0
  expression assignment
1
  expression assignment
2
  expression assignment
3
  expression assignment
4
  # end
```

Topic

Control Transfer Functions

Structural Abstraction

```
0
  expression assignment
1
  expression assignment
2
  expression assignment
3
  expression assignment
4
  # end
```

Control Transfer Functions

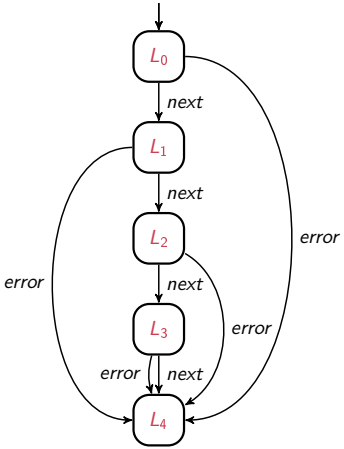
i	next	error
0	1	4
1	2	4
2	3	4
3	4	4
4		

Control Flow Graph

Control Transfer Functions

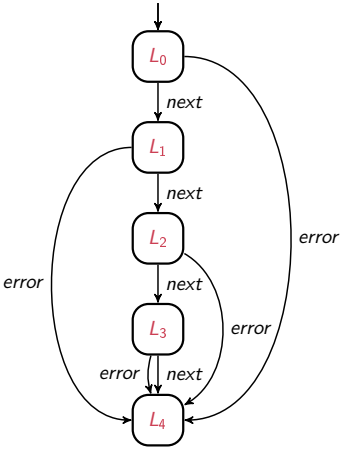
i	next	error
0	1	4
1	2	4
2	3	4
3	4	4
4		

Control Flow Graph

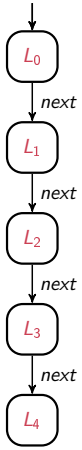


Implicit Error edges

Control Flow Graph

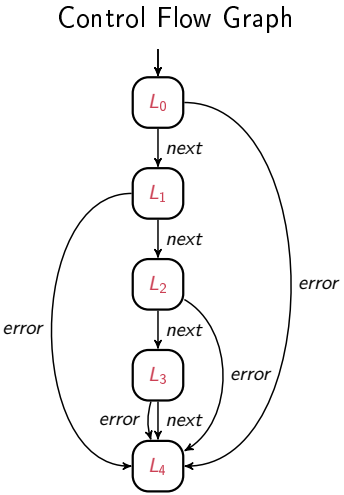


Control Flow Graph with error edges implicit



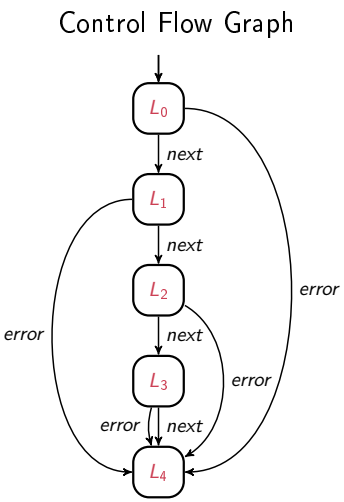
Topic

What is an execution?



An execution is a labelled path in the Control Flow graph starting from L_0 and ending at the last location L_N.

Structurally Feasible Executions



Structurally feasible executions:

1. $$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{next} L_3 \xrightarrow{next} L_4$$

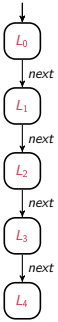
2. All error executions (executions containing an error edge).

Logically Feasible Executions

Program:

```
0
  a = 3
1
  b = read()
2
  c = a // b
3
  d = c - a
4
  # end
```

Control Flow Graph:



Logically feasible executions:

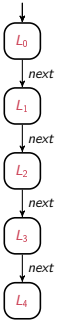
- b is a number not equal to 0:
  $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{next} L_3 \xrightarrow{next} L_4$
- $b = 0$:
  $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{error} L_4$
- b is not a number:
  $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{error} L_4$

Actual Execution

Program:

```
0
  a = 3
1
  b = read()
2
  c = a // b
3
  d = c - a
4
  # end
```

Corresponding CFG:



Actual execution given that $b = 5$:

$$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{next} L_3 \xrightarrow{next} L_4$$

Topic

Conditional Control Flow

```
0
a = read()
1
if a < 5:
2
    b = 7
3
else:
4
    b = 2 * a
5
c = a + b
6
# end
```

1. A block is a sequence of instructions.
2. An If-else statement ($L_1 - L_4$) has three parts:
   - A 'test' expression (L_1)
   - A 'then' block ($L_2 - L_2$)
   - An 'else' block ($L_4 - L_4$)
   - The 'else' keyword (L_3) is punctuation.
3. In the concrete syntax, the then and else blocks are indented.

Structural Abstraction

Program

```
0 a = read()
1 if a < 5:
2     b = 7
3 else:
4     b = 2 * a
5 c = a + b
6 # end
```

Structural Abstraction

```
0 expression assignment
1 if:
2     expression assignment
3 else:
4     expression assignment
5 expression assignment
6 # end
```

Control Transfer Functions

Structural Abstraction

```
0
  expression assignment
1
  if:
2
      expression assignment
3
  else:
4
      expression assignment
5
  expression assignment
6
  # end
```

Control Transfer Functions

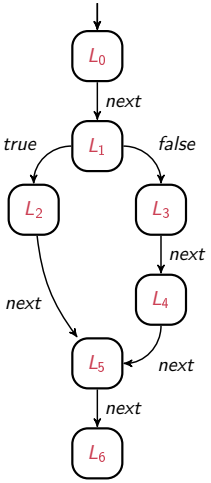
i	next	true	false	error
0	1			6
1		2	3	6
2	5			6
3	4			
4	5			6
5	6			6
6				

Control Flow Graph

Control Transfer Functions

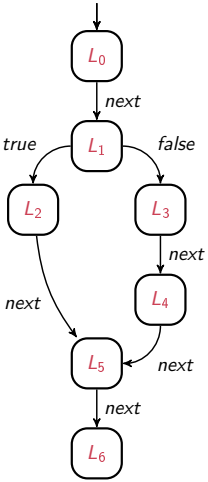
i	next	true	false	error
0	1			6
1		2	3	6
2	5			6
3	4			
4	5			6
5	6			6
6				

Control Flow Graph with error edges implicit



Structurally Feasible Executions

CFG:



Structurally Feasible Executions:

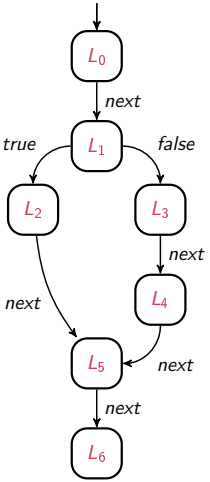
1. $L_0 \xrightarrow{next} L_1 \xrightarrow{true} L_2 \xrightarrow{next} L_5 \xrightarrow{next} L_6$
2. $L_0 \xrightarrow{next} L_1 \xrightarrow{false} L_3 \xrightarrow{next} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_6$
3. All error executions.

Logically Feasible Executions

Program:

```
0
  a = read()
1
  if a < 5:
2
    b = 7
3
  else:
4
    b = 2 * a
5
  c = a + b
6
  # end
```

CFG:



Logically Feasible Executions:

1. $a < 5$:
   $L_0 \xrightarrow{next} L_1 \xrightarrow{true} L_2 \xrightarrow{next}$
   $L_5 \xrightarrow{next} L_6$

2. $a \geq 5$:
   $L_0 \xrightarrow{next} L_1 \xrightarrow{false} L_3 \xrightarrow{next}$
   $L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_6$

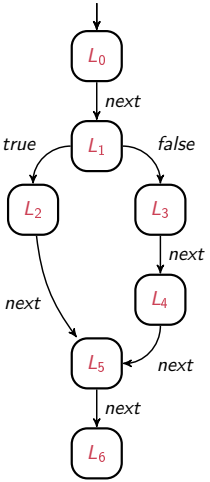
3. a is not a number:
   $L_0 \xrightarrow{next} L_1 \xrightarrow{error} L_6$

Actual execution

Program:

```
0
a = read()
1
if a < 5:
2
    b = 7
3
else:
4
    b = 2 * a
5
c = a + b
6
# end
```

CFG:



Actual execution, given that value read at L_0 is the number 3.

$$L_0 \xrightarrow{next} L_1 \xrightarrow{true} L_2 \xrightarrow{next} L_5 \xrightarrow{next} L_6$$

Topic

Program and its structural abstraction

1. **Location**: a natural number between 0 and N
2. **Instruction**: A basic executional unit of a program
3. **Program**: A map from $[0..N-1]$ to instructions
4. **Structural abstraction** of an **instruction**: Expression assignment, `if` or `else`
5. **Structural abstraction** of **a program**: structural abstractions of all its instructions

Control transfer functions and CFG

1. **Control Transfer Functions**: partial functions from locations to locations
2. **next**, **error**, **true** and **false**: control transfer functions
3. **Control Flow Graph**: A diagram representing the control transfer functions
4. **Control Flow Graph with implicit error edges**: error edges suppressed to reduce clutter

Executions

1. **Execution**: A labelled path from L_0 to L_N.
2. **Structurally feasible executions**: Possible executions inferable from the structural abstraction of a program
3. **Logically feasible executions**: Possible executions inferable from the actual program
4. **Actual Execution**: The single execution when the results of all the read() expressions, if any, in the program are known