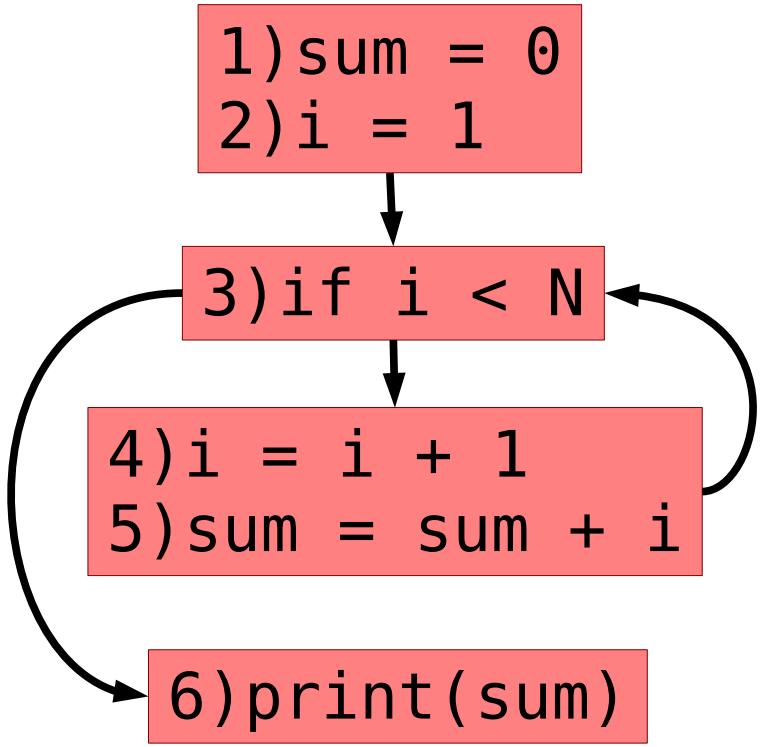
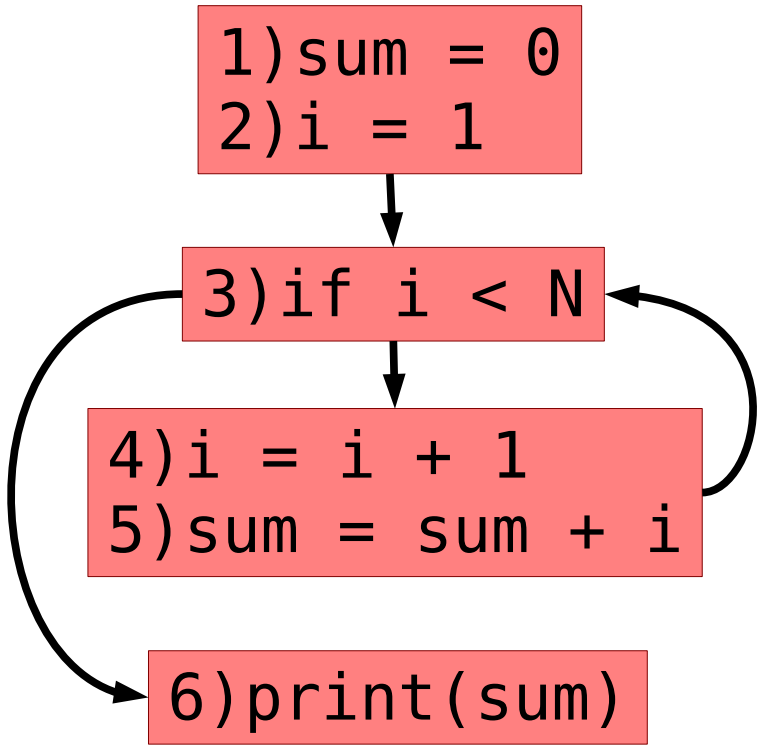
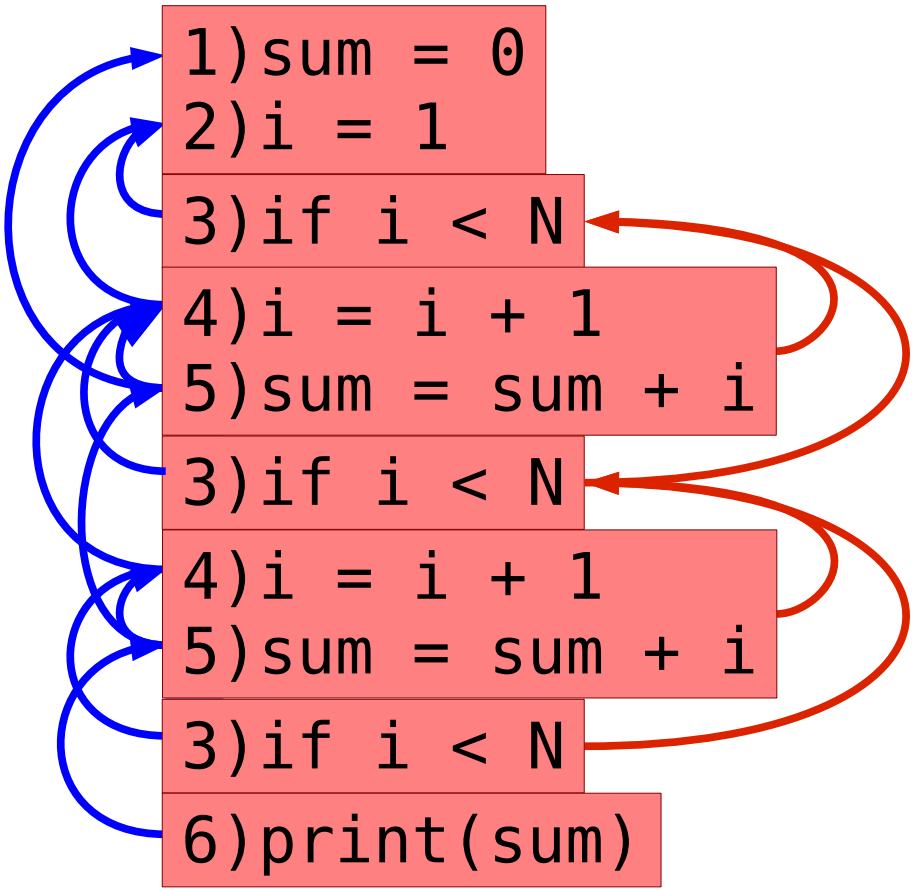
Slicing

Remember the Pain of Dependencies

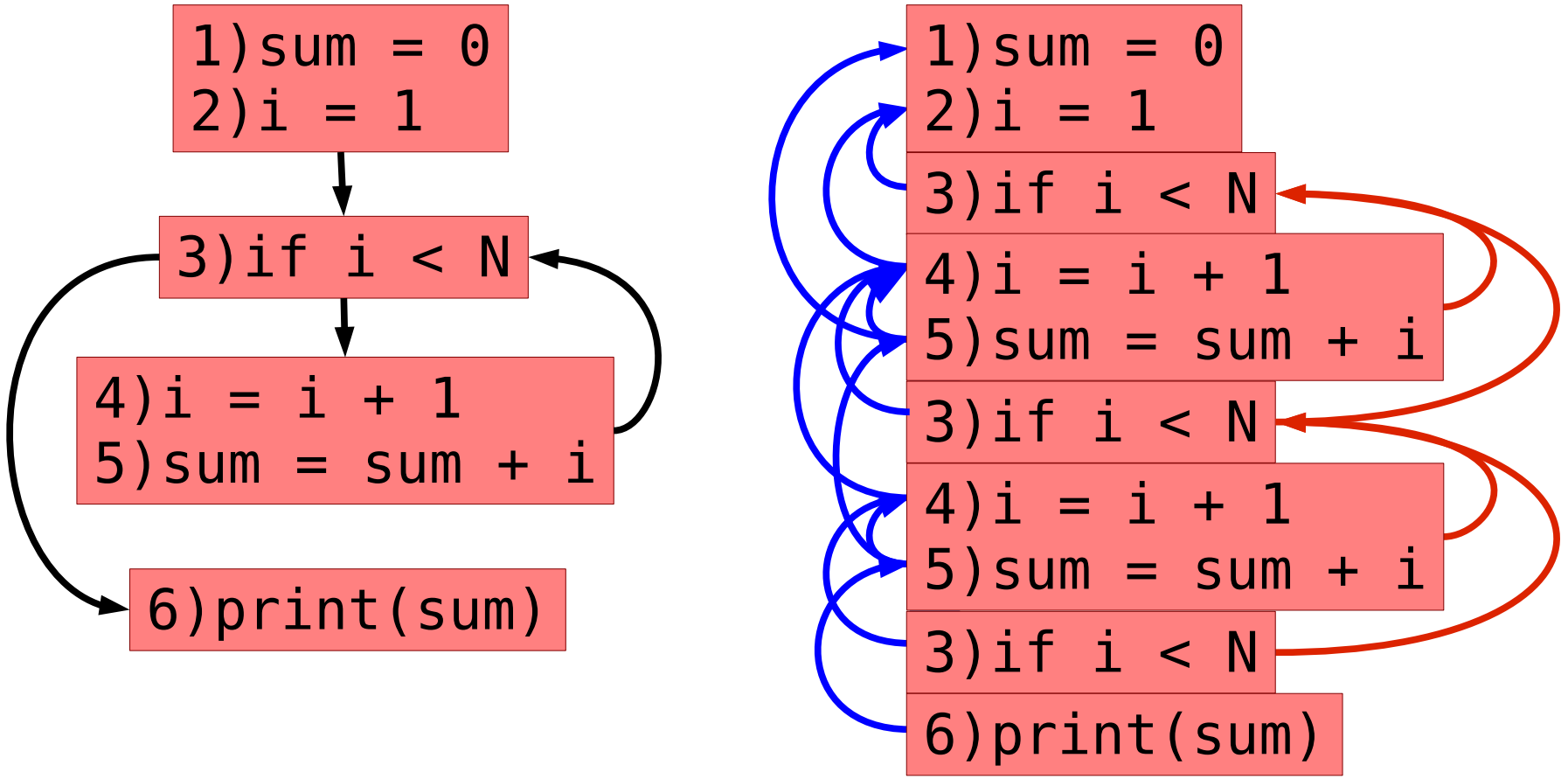


Remember the Pain of Dependencies



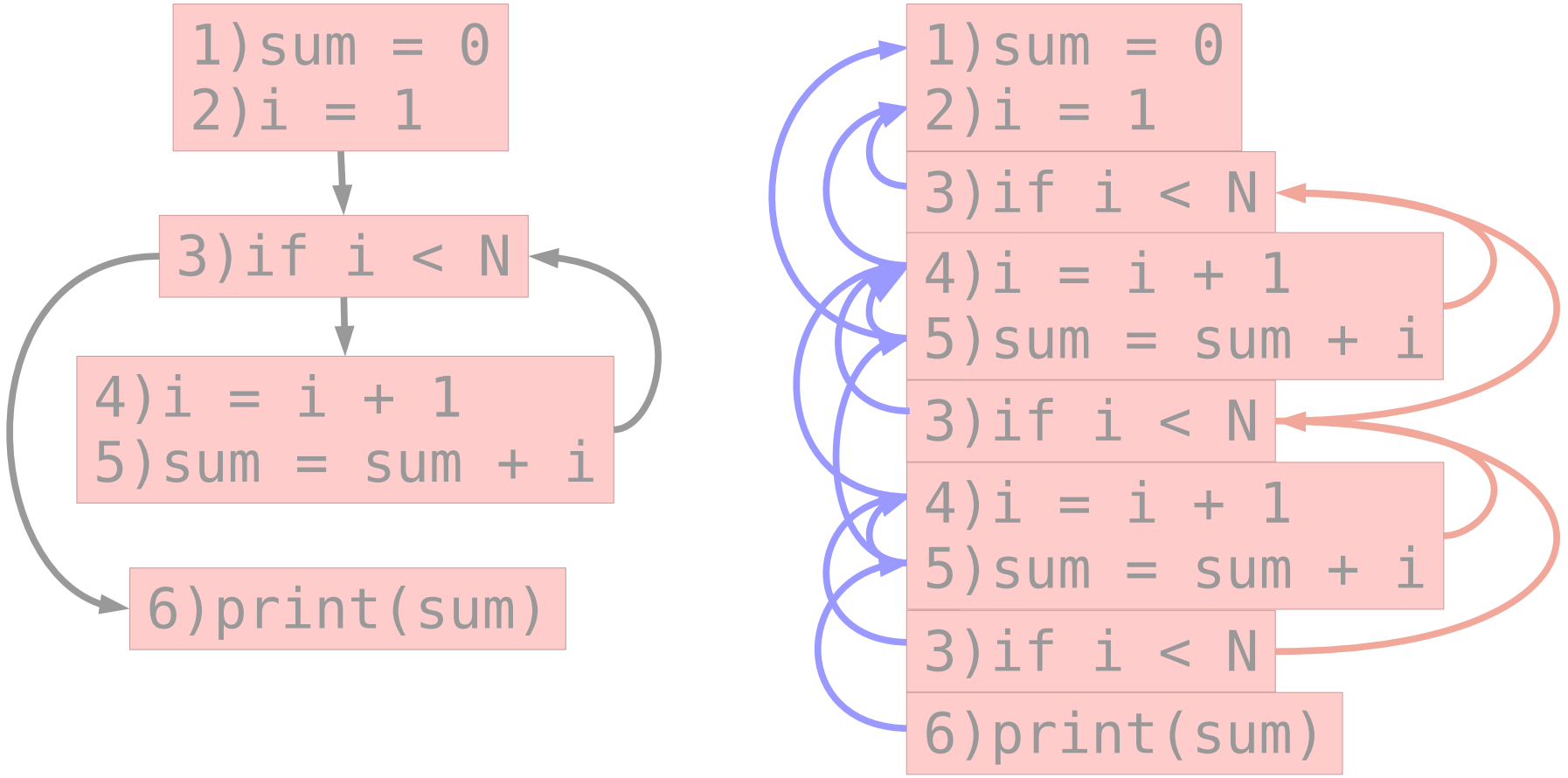


Remember the Pain of Dependencies



If only we could focus on the parts that interest us…

Remember the Pain of Dependencies



If only we could focus on the parts that interest us…

Slicing is a technique for identifying interesting parts of a program/execution

Program Slicing

- The *slice* of a value v at a statement s is:

Program Slicing

- The *slice* of a value v at a statement s is:
  - the set of statements involved in computing v's value at s. [Weiser 82]

Program Slicing

- The *slice* of a value v at a statement s is:
  - the set of statements involved in computing v's value at s. [Weiser 82]

```
1)sum = 0
2)i = 1
3)while i < N:
4)   i = i + 1
5)   sum = sum + i
6)print(sum)
7)print(i)
```

Program Slicing

- The *slice* of a value v at a statement s is:
  - the set of statements involved in computing v's value at s. [Weiser 82]

```
1)sum = 0
2)i = 1
3)while i < N:
4)   i = i + 1
5)   sum = sum + i
6)print(sum)
7)print(i)
```

Program Slicing

- The *slice* of a value v at a statement s is:
  - the set of statements involved in computing v's value at s. [Weiser 82]

```
1)sum = 0
2)i = 1
3)while i < N:
4)   i = i + 1
5)   sum = sum + i
6)print(sum)
7)print(i)
```

How does this relate to our representations?

Program Slicing

- The *slice* of a value v at a statement s is:
  - the set of statements involved in computing v's value at s. [Weiser 82]
  - The statements that may influence v…

Program Slicing

- The *slice* of a value v at a statement s is:
  - the set of statements involved in computing v's value at s. [Weiser 82]
  - The statements that may influence v…
    - Data dependence
    - Control dependence
    - Compute using the PDG!

Program Slicing Uses

- Debugging
- Testing
- Reverse Engineering
- Optimization
- Design Profiling
- Malware analysis
- …

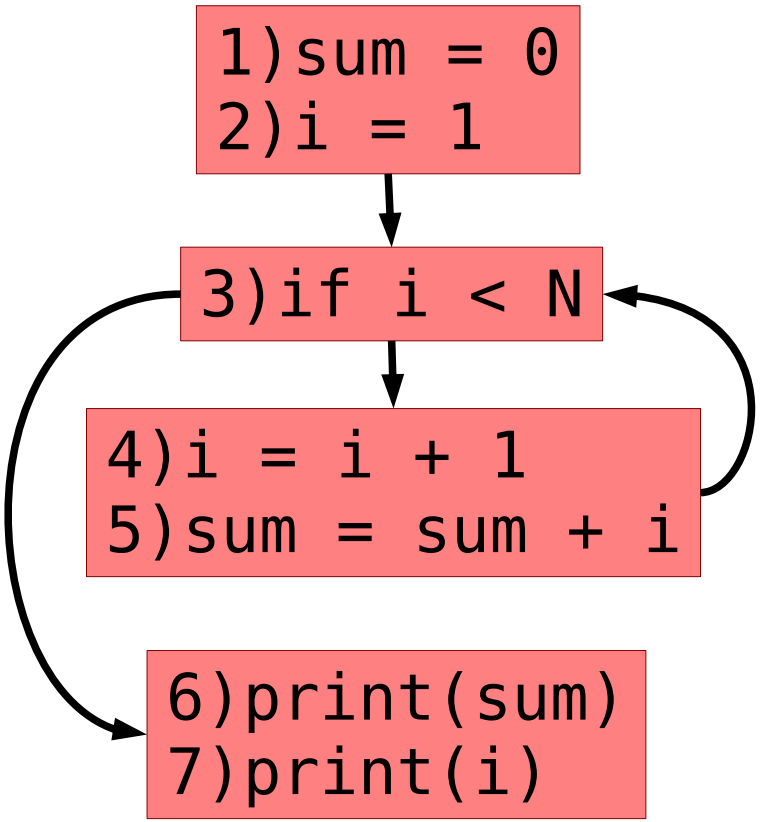
How to Slice?

- Transitive closure of edges in the PDG
  - Start from v and just follow edges backward

How to Slice?

- Transitive closure of edges in the PDG
  - Start from v and just follow edges backward

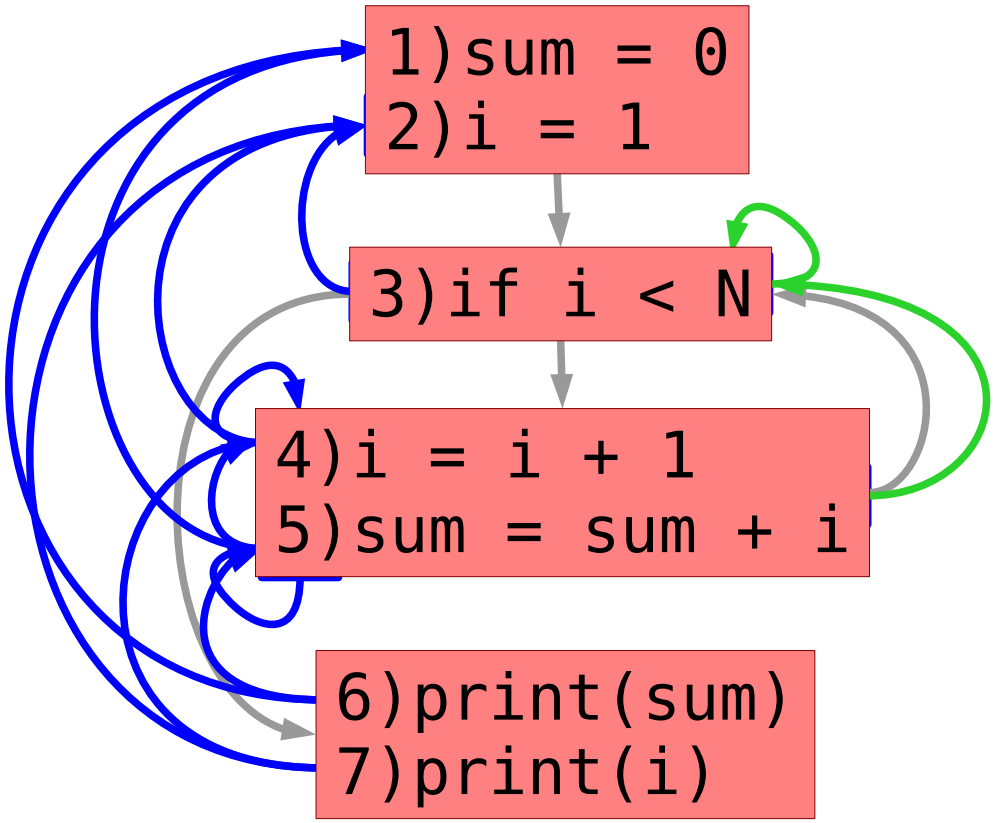
CFG:



How to Slice?

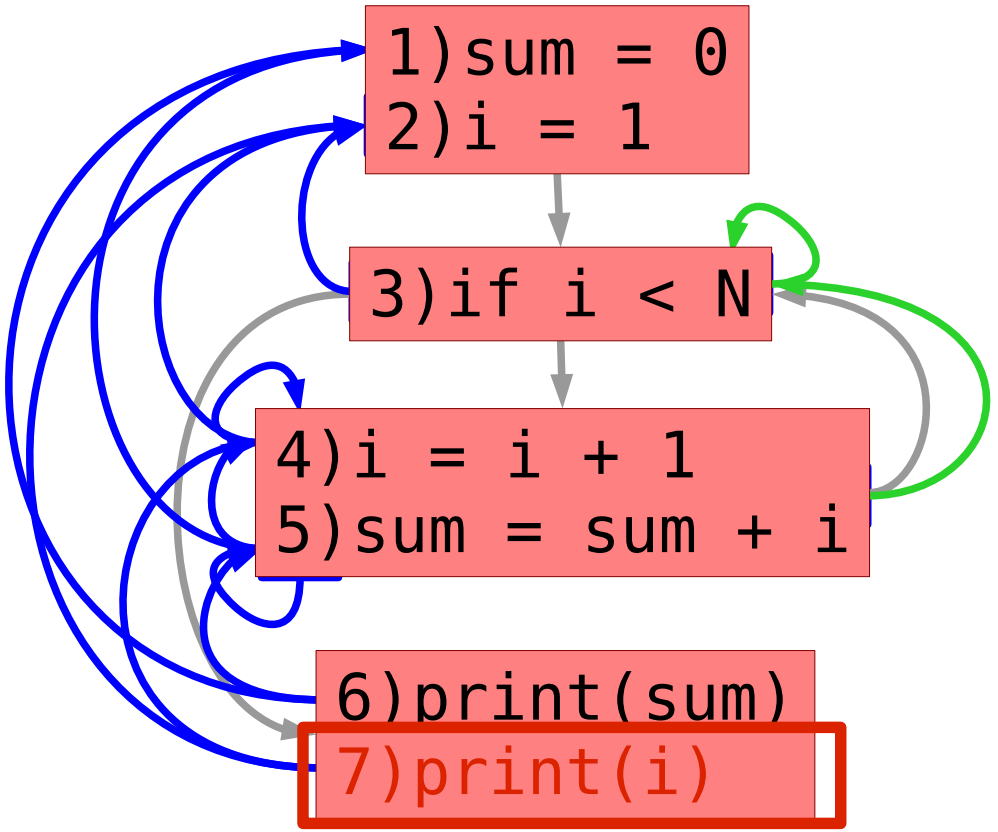
- Transitive closure of edges in the PDG
  - Start from v and just follow edges backward

PDG:



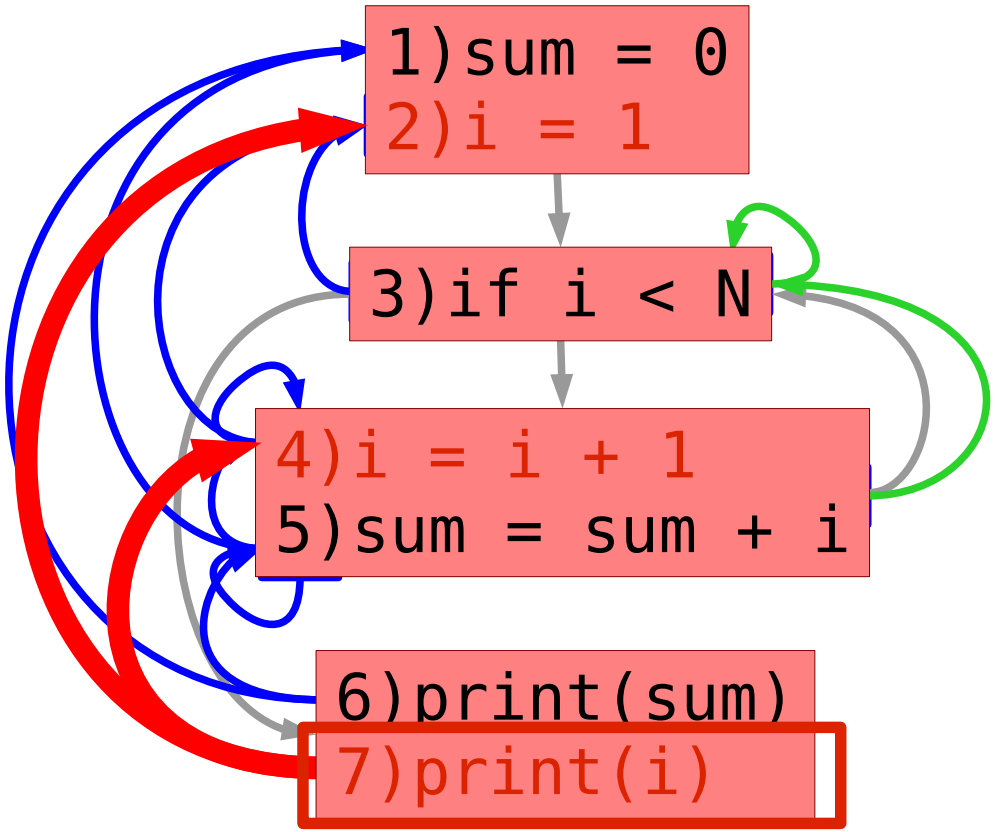
How to Slice?

- Transitive closure of edges in the PDG
  - Start from v and just follow edges backward



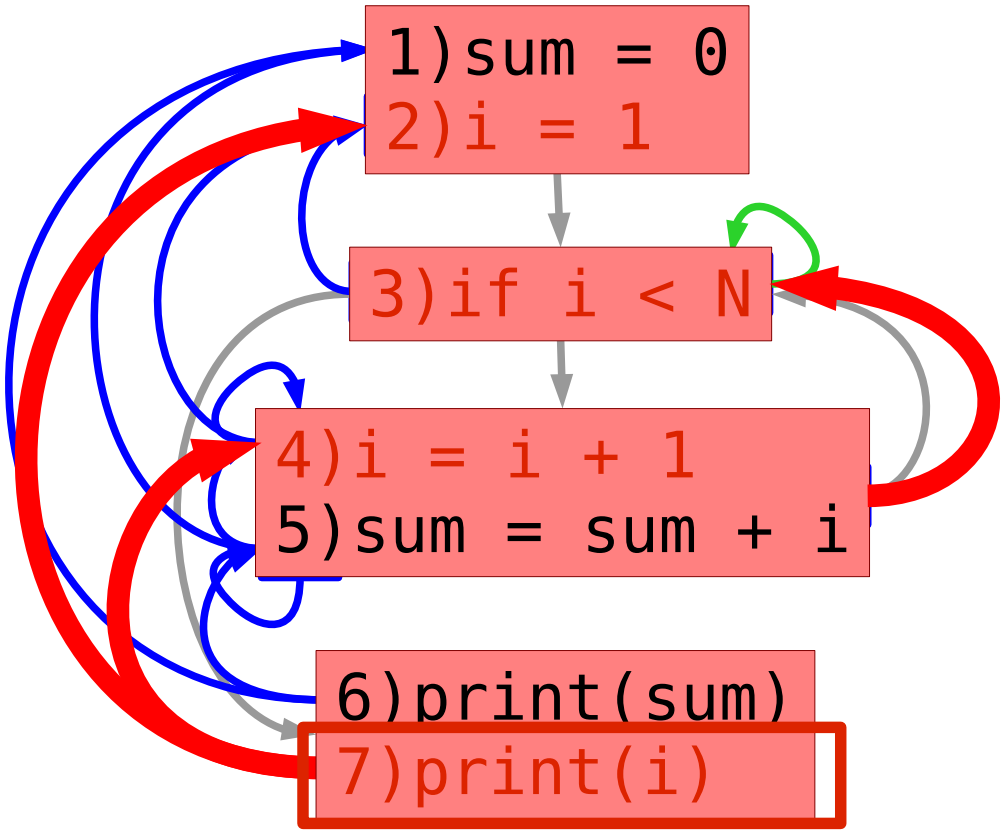
How to Slice?

- Transitive closure of edges in the PDG
  - Start from v and just follow edges backward



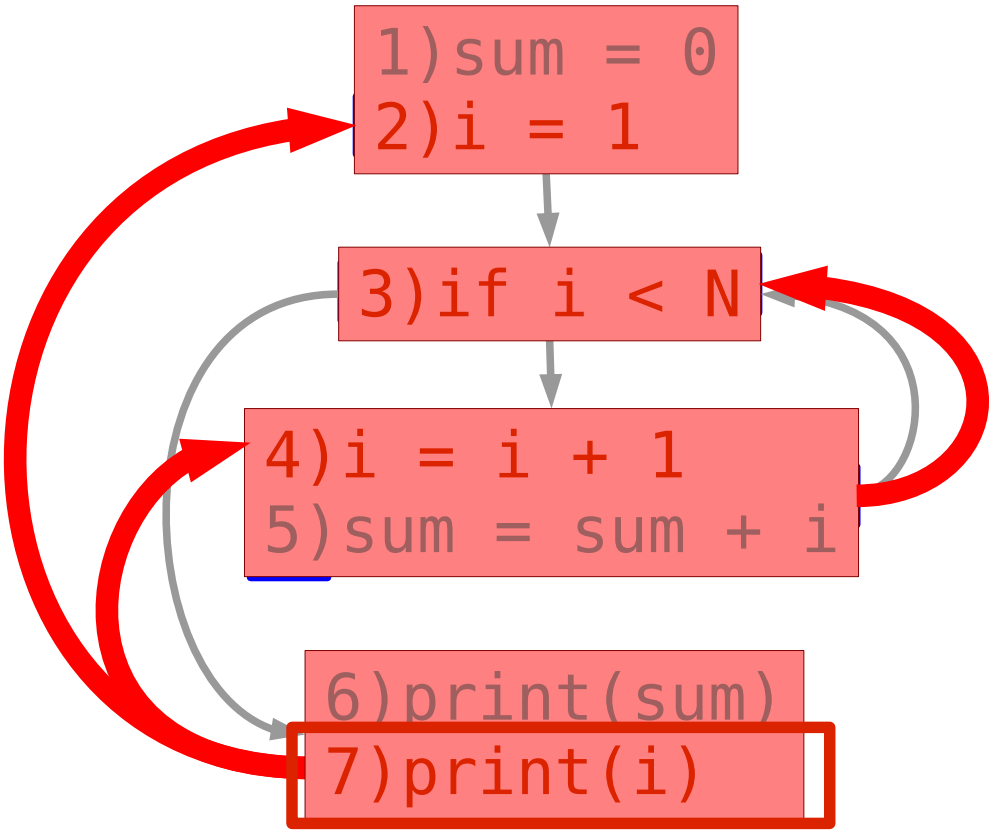
How to Slice?

- Transitive closure of edges in the PDG
  - Start from v and just follow edges backward



How to Slice?

- Transitive closure of edges in the PDG
  - Start from v and just follow edges backward



Very Configurable

- Static vs. Dynamic (PDG vs. DDG)
- Backward vs. Forward
- Executable vs. Nonexecutable
- Edges vs. Nodes

Very Configurable

- Static vs. Dynamic (PDG vs. DDG)
- Backward vs. Forward
- Executable vs. Nonexecutable
- Edges vs. Nodes

What do forward and backward *mean*?

Very Configurable

- Static vs. Dynamic (PDG vs. DDG)
- Backward vs. Forward
- Executable vs. Nonexecutable
- Edges vs. Nodes

What do forward and backward *mean*?

Why might a slice not be executable?

Very Configurable

- Static vs. Dynamic (PDG vs. DDG)
- Backward vs. Forward
- Executable vs. Nonexecutable
- Edges vs. Nodes

What do forward and backward *mean*?

Why might a slice not be executable?

What do nodes capture? Edges?

Strengths of Static Slicing

- Considers all possible executions
  - Necessary for conservative analyses
  - (“Might I leak secret information?”)

Strengths of Static Slicing

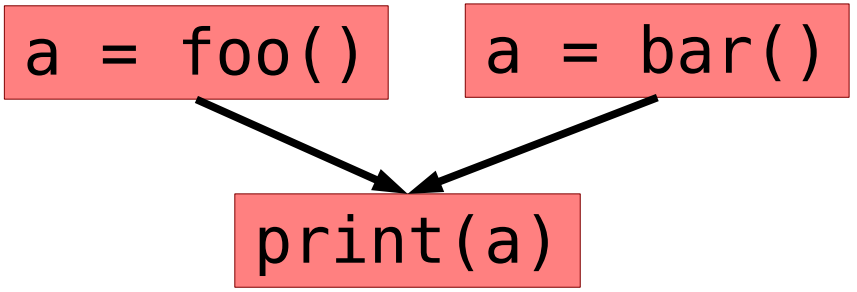
- Considers all possible executions
  - Necessary for conservative analyses
  - (“Might I leak secret information?”)
- Fast to compute

Strengths of Static Slicing

- Considers all possible executions
  - Necessary for conservative analyses
  - (“Might I leak secret information?”)
- Fast to compute
- Space efficient

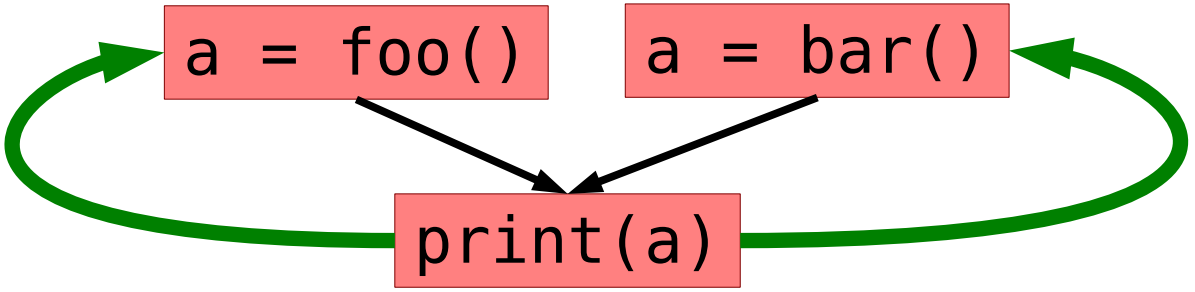
Issues with Static Slicing

- Multiple program paths



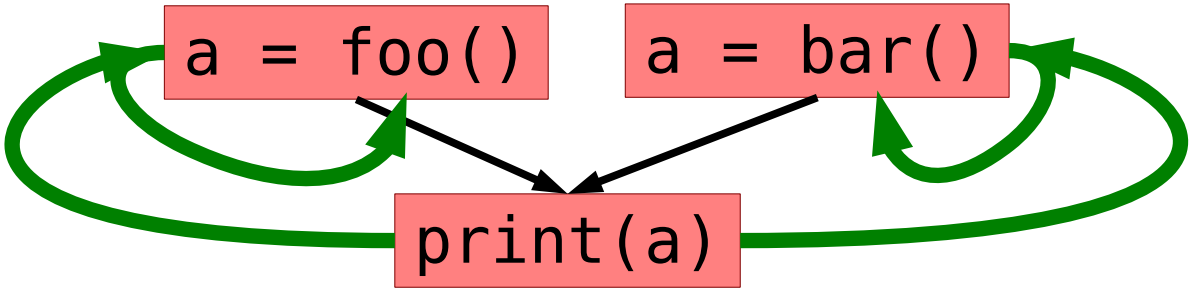
Issues with Static Slicing

- Multiple program paths



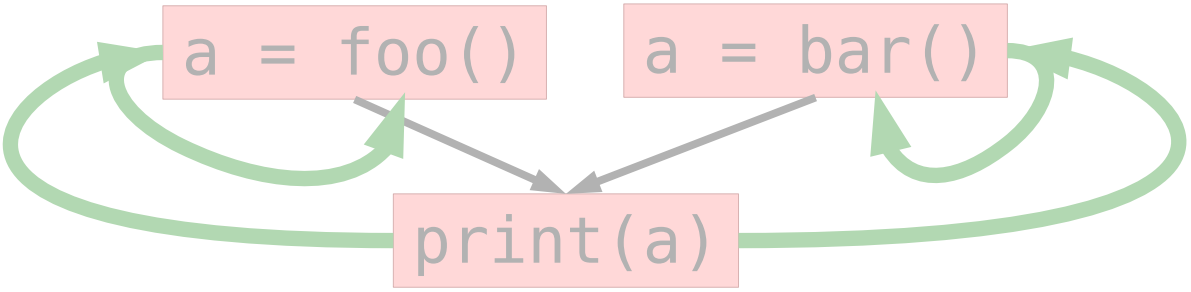
Issues with Static Slicing

- Multiple program paths

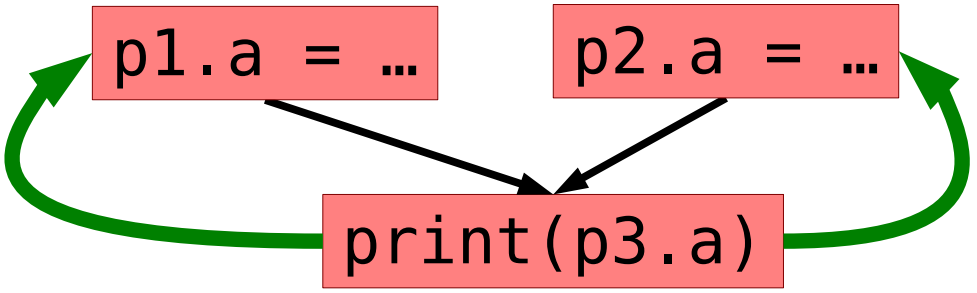


Issues with Static Slicing

- Multiple program paths

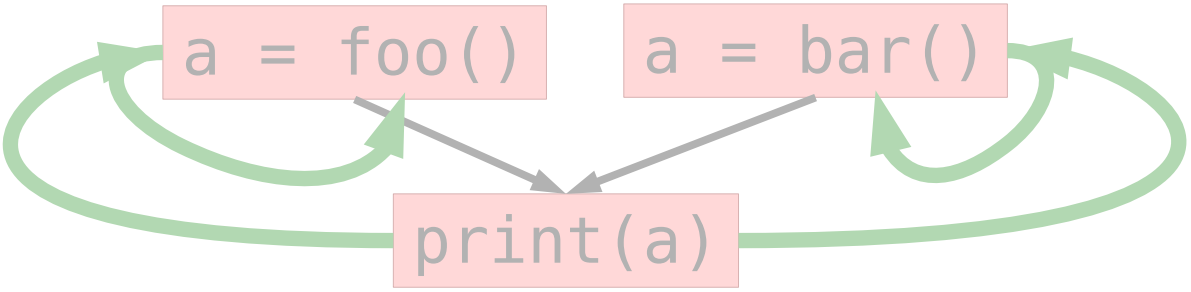


- Pointers – points-to graphs are imprecise

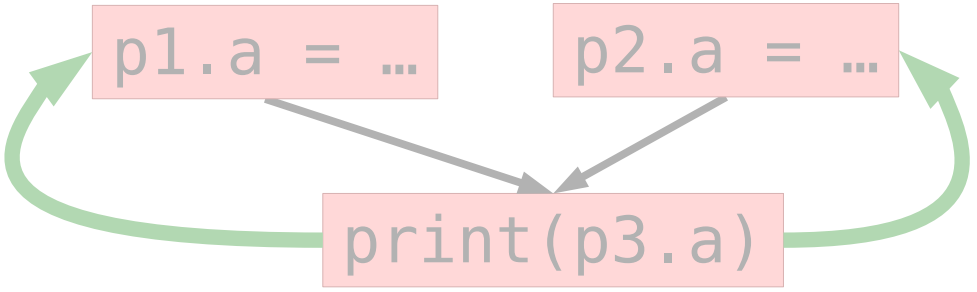


Issues with Static Slicing

- Multiple program paths



- Pointers – points-to graphs are imprecise



- Function pointers – must consider all possible call targets

Strengths of Dynamic Slicing

- Precisely considers a single execution (DDG)
  - “Did I …”

Strengths of Dynamic Slicing

- Precisely considers a single execution (DDG)
  - "Did I …"
- No imprecision from aliasing or multiple paths
  - Why?

Strengths of Dynamic Slicing

- Precisely considers a single execution (DDG)
  - "Did I …"
- No imprecision from aliasing or multiple paths
  - Why?
- Cover fewer static program statements

Issues with Dynamic Slicing

- Capturing a trace and computing a DDG is expensive
  - (GB sized trace files)

Issues with Dynamic Slicing

- Capturing a trace and computing a DDG is expensive
  - (GB sized trace files)
- Slow to compute
  - Churn a great deal of memory

Issues with Dynamic Slicing

- Capturing a trace and computing a DDG is expensive
  - (GB sized trace files)
- Slow to compute
  - Churn a great deal of memory
- Very many statement instances and dynamic dependences to examine

Issues with Dynamic Slicing

- Capturing a trace and computing a DDG is expensive
  - (GB sized trace files)
- Slow to compute
  - Churn a great deal of memory
- Very many statement instances and dynamic dependences to examine
- Misses alternative histories
  - What would have happened if … ?

Coping with Scale

Both types of slicing benefit from techniques that *prune* or *focus* slices on just what is *interesting*

Coping with Scale

Both types of slicing benefit from techniques that *prune* or *focus* slices on just what is *interesting*

- *Thin Slicing*- Focus on propagating v, ignoring data structures [PLDI07]

Coping with Scale

Both types of slicing benefit from techniques that *prune* or *focus* slices on just what is *interesting*

- *Thin Slicing*- Focus on propagating v, ignoring data structures [PLDI07]
- *Chopping*- Combine forward & backward info [ASE05]

Coping with Scale

Both types of slicing benefit from techniques that *prune* or *focus* slices on just what is *interesting*

- *Thin Slicing*- Focus on propagating v, ignoring data structures [PLDI07]
- *Chopping*- Combine forward & backward info [ASE05]
- *Confidence Analysis*- Instructions used to compute correct values less likely to be buggy [PLDI06]

Coping with Scale

Both types of slicing benefit from techniques that *prune* or *focus* slices on just what is *interesting*

- *Thin Slicing*- Focus on propagating v, ignoring data structures [PLDI07]
- *Chopping*- Combine forward & backward info [ASE05]
- *Confidence Analysis*- Instructions used to compute correct values less likely to be buggy [PLDI06]
- *Guided Browsers*- Zoom in on demand [ICSE06]

Coping with Scale

Both types of slicing benefit from techniques that *prune* or *focus* slices on just what is *interesting*

- *Thin Slicing*- Focus on propagating v, ignoring data structures [PLDI07]
- *Chopping*- Combine forward & backward info [ASE05]
- *Confidence Analysis*- Instructions used to compute correct values less likely to be buggy [PLDI06]
- *Guided Browsers*- Zoom in on demand [ICSE06]
- *Causal Models*- Compare executions to identify real causes [ICSE13]

Coping with Scale

Both types of slicing benefit from techniques that *prune* or *focus* slices on just what is *interesting*

- *Thin Slicing*- Focus on propagating v, ignoring data structures [PLDI07]
- *Chopping*- Combine forward & backward info [ASE05]
- *Confidence Analysis*- Instructions used to compute correct values less likely to be buggy [PLDI06]
- *Guided Browsers*- Zoom in on demand [ICSE06]
- *Causal Models*- Compare executions to identify real causes [ICSE13]
- Much more…