Open Source Operating System Annual Technical Conference (OS^{2}ATC 2017)

Symbolic Verification of Java Programs

Zhenbang Chen

(zbchen@nudt.edu.cn)

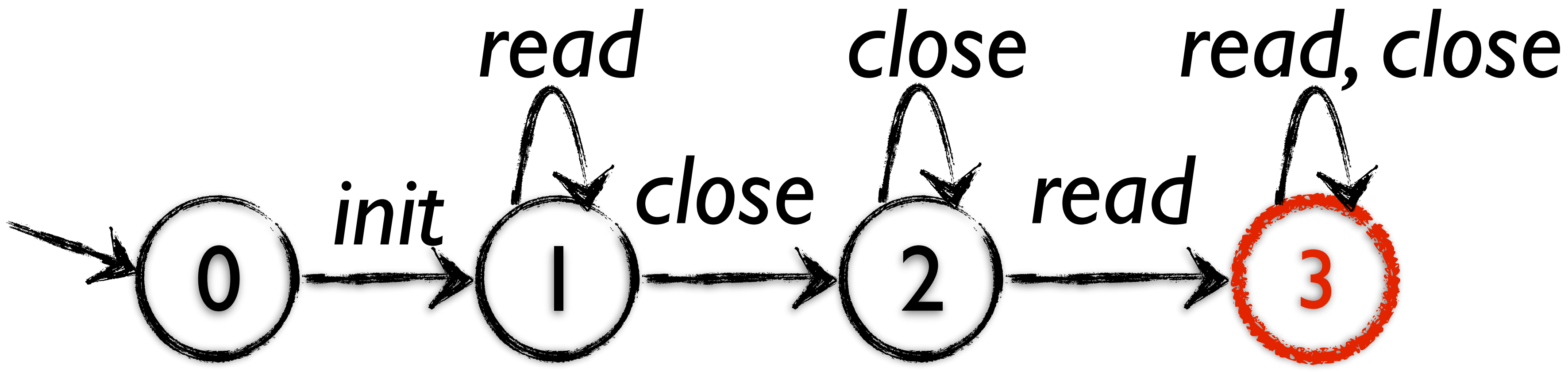
Joint work with Yufeng Zhang, Hengbiao Yu, Ji Wang, Zhendong Su, Wei Dong

College of Computer, National University of Defense Technology, China

Beijing, 2017.12.16

Regular Property Verification

- Regular properties/FSMs are widely used
  - Model-based testing
  - Typestate analysis, e.g., runtime verification
  - API protocol specification, e.g., OS kernel



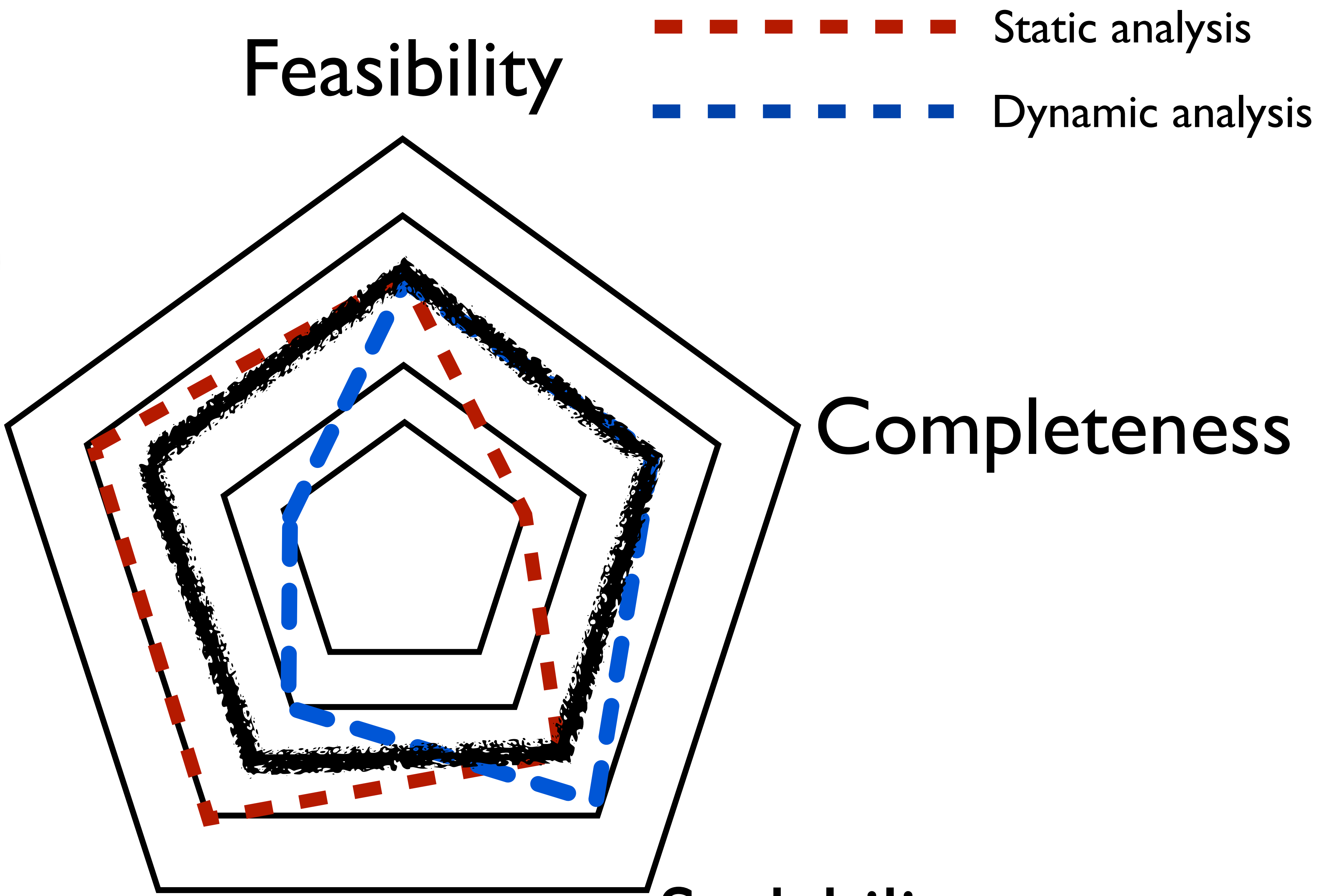
Regular Property Verification

- Regular properties/FSMs are widely used
  - Model-based testing
  - Typestate analysis, e.g., runtime verification
  - API protocol specification, e.g., OS kernel
- Verifying regular properties is challenging

Existing Work

- Static analysis
  - CheckStyle, PMD, Infer, Coverity, …
  - ESC/Java2, Bandera [ICSE'00], …
- Dynamic analysis
  - Dynamic verification: Java Path-Finder (JPF), …
  - Runtime verification: JavaMOP, …

Existing Work



Feasibility

Completeness

Scalability

Coverage

Soundness

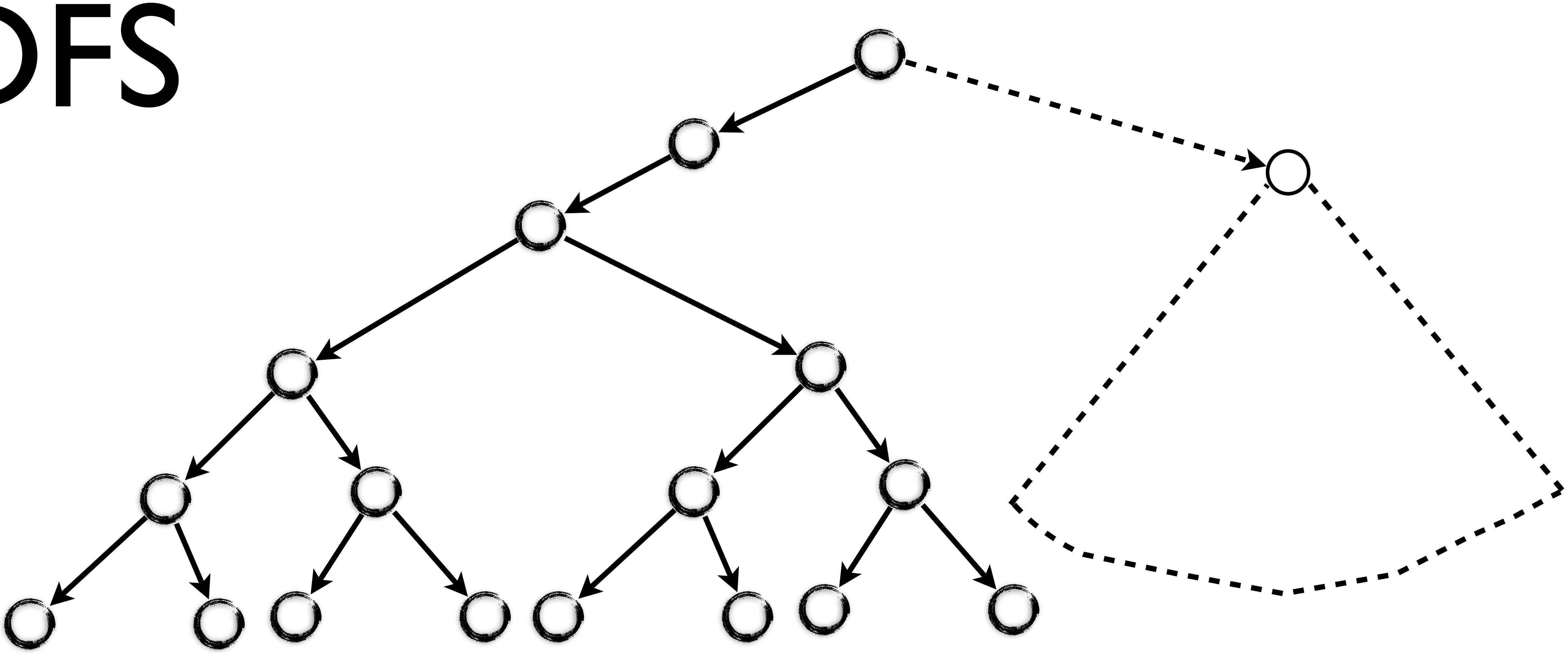
Static analysis

Dynamic analysis

Dynamic Symbolic Execution (PLDI'05)

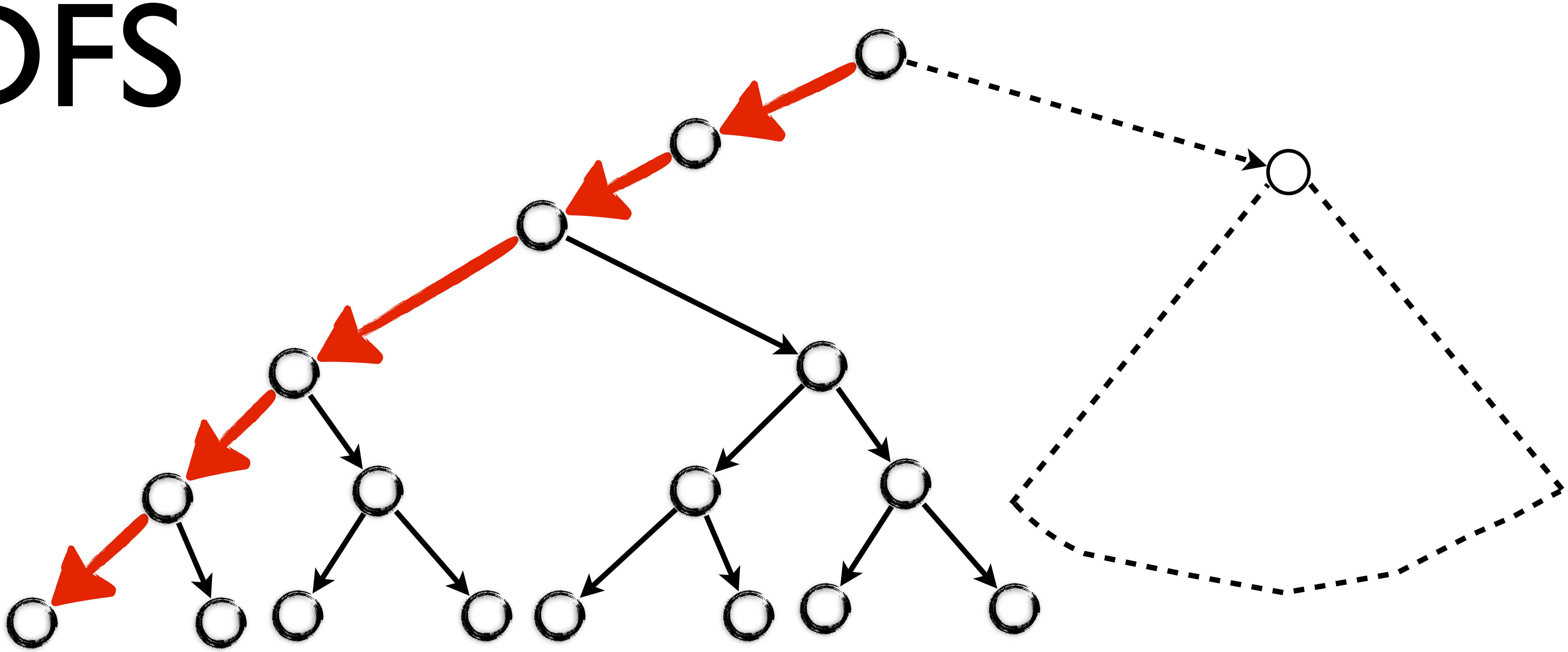
Dynamic Symbolic Execution

DFS



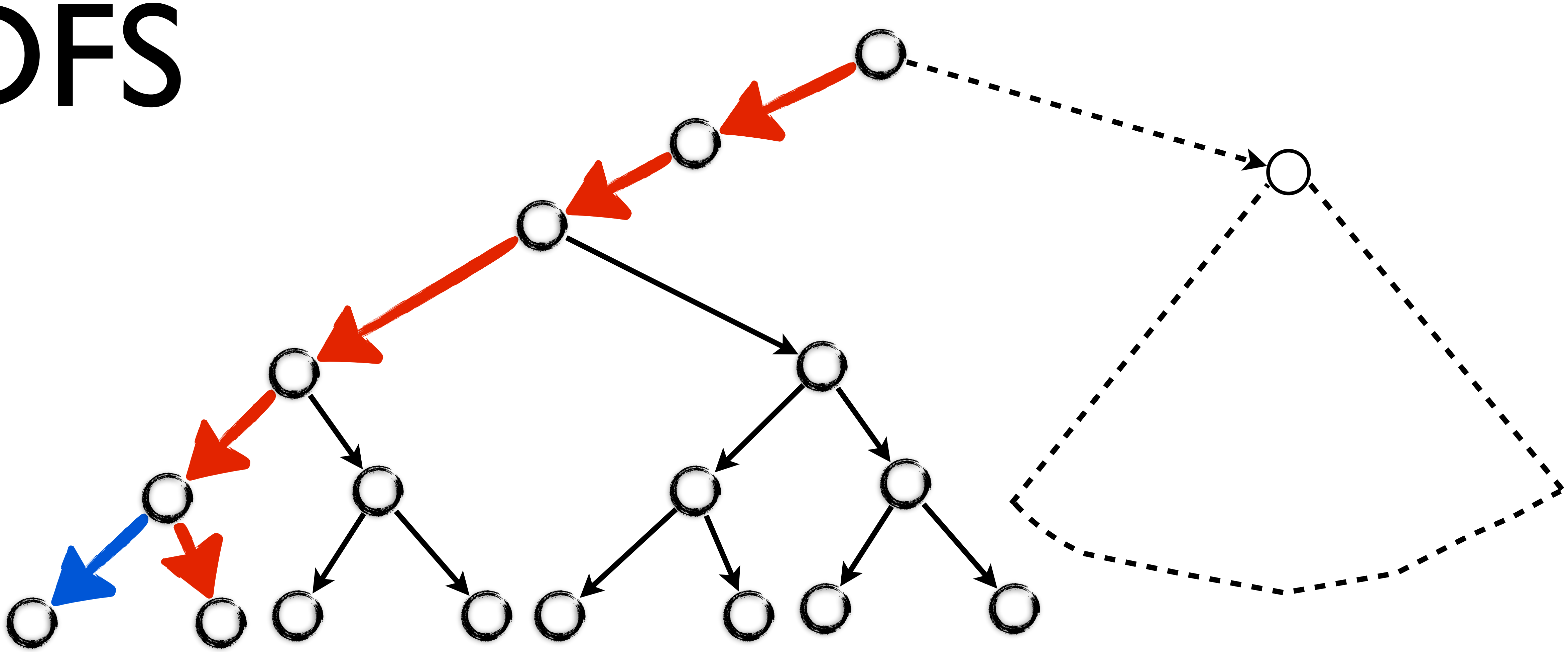
Dynamic Symbolic Execution

DFS



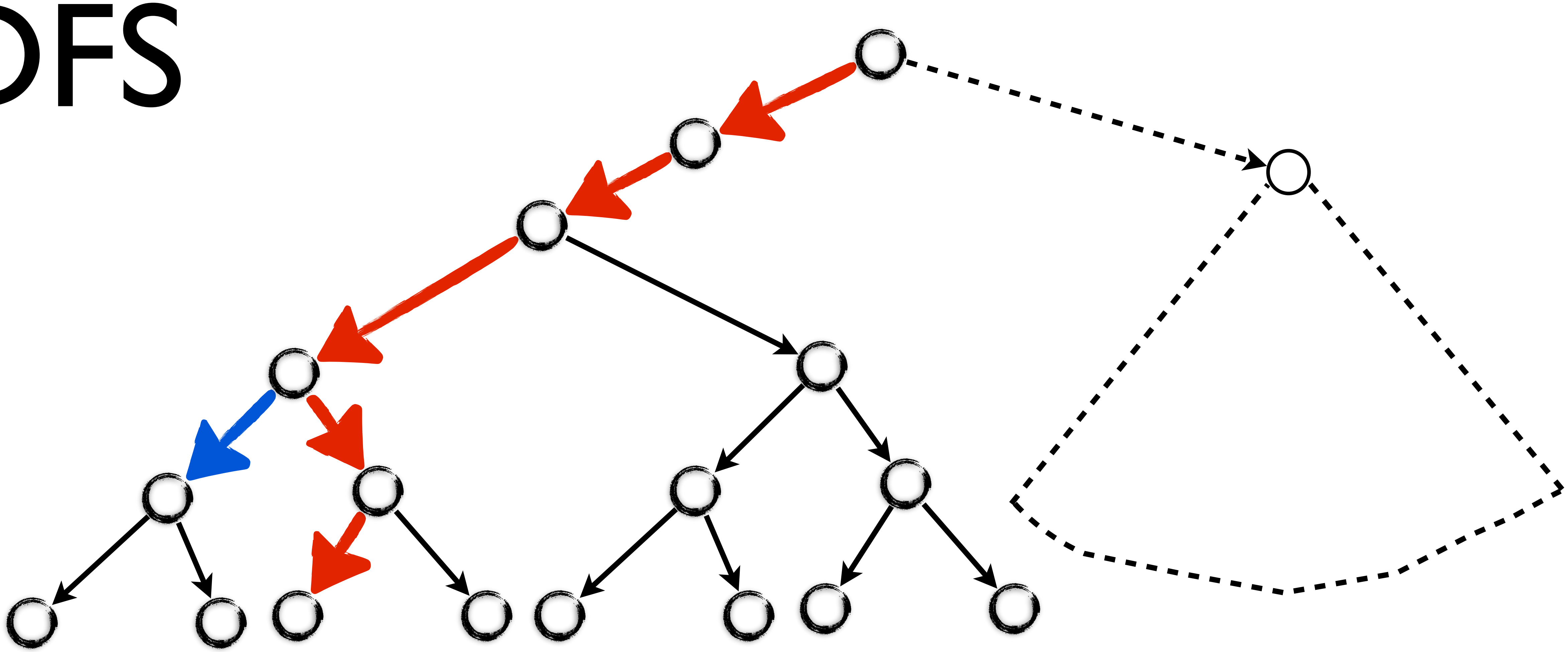
Dynamic Symbolic Execution

DFS



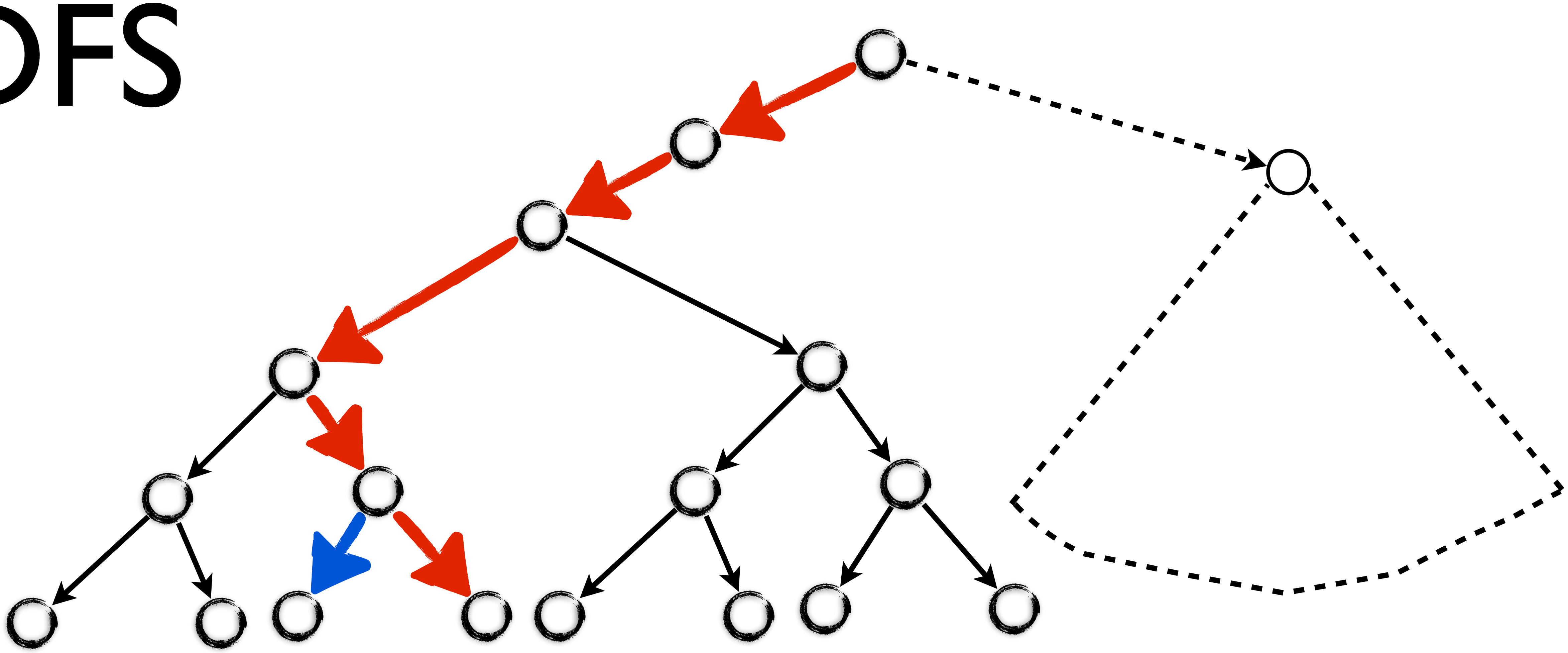
Dynamic Symbolic Execution

DFS



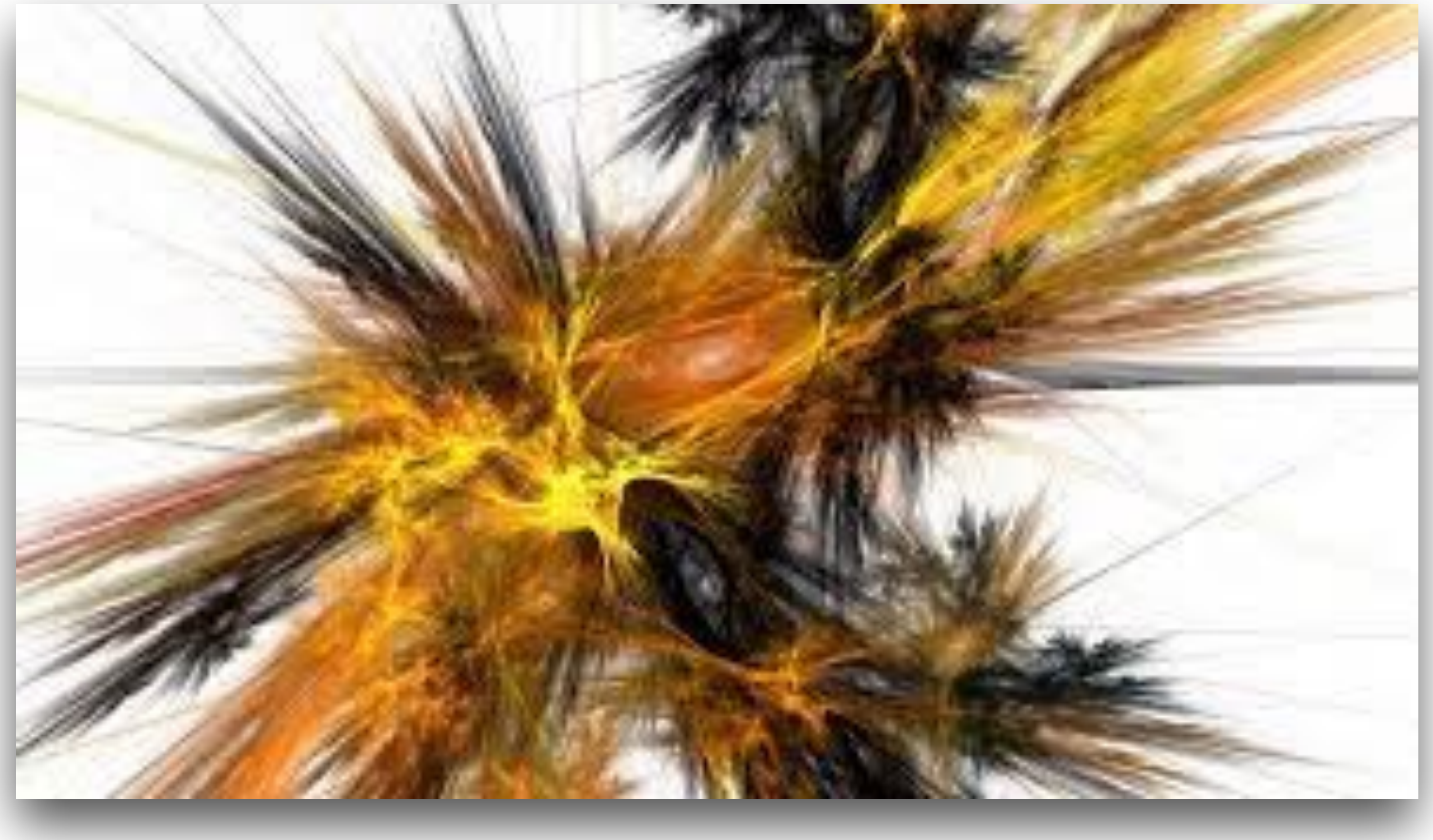
Dynamic Symbolic Execution

DFS



Challenge of Symbolic Execution

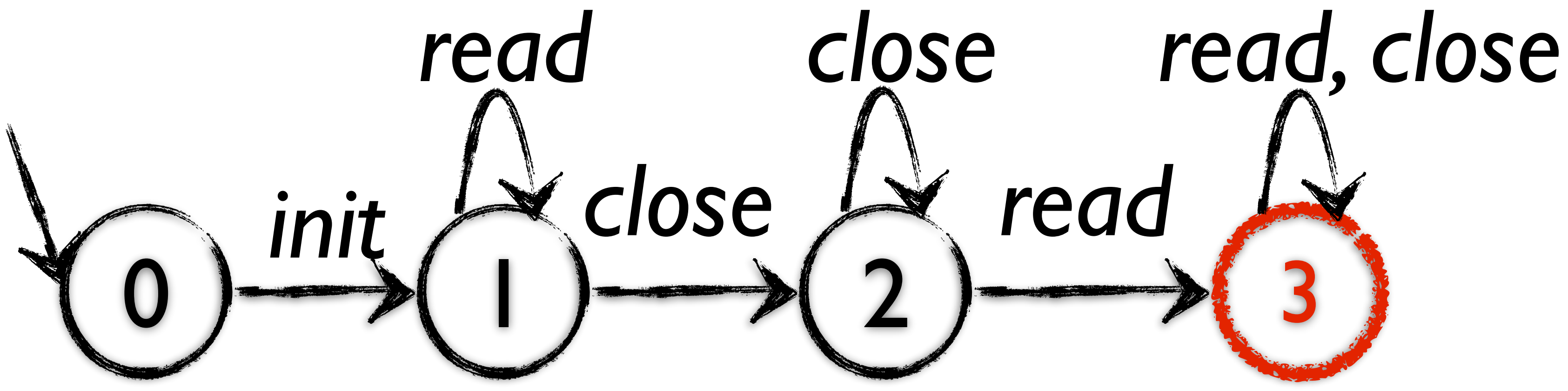
- Path explosion problem



How to boost completing path exploration and finding counterexample?

Observation and Insight

- Many irrelevant paths exist
- For relevant paths
  - The ones with specific sequences can violate the regular property
  - Many are equivalent w.r.t. verification



Observation and Insight

- Many irrelevant paths exist
- For relevant paths
  - The ones with specific sequences can violate the regular property
  - Many are equivalent w.r.t. verification

Prune irrelevant, uninteresting relevant and equivalent paths, and explore counter-example paths earlier

Key Idea

Verify a program satisfies a regular property *P*

P $\neg P$

Regular property-oriented path slicing
[ICSE'18]

+

Regular property guided DSE
[ICSE'15]

Pruning ***Guiding***

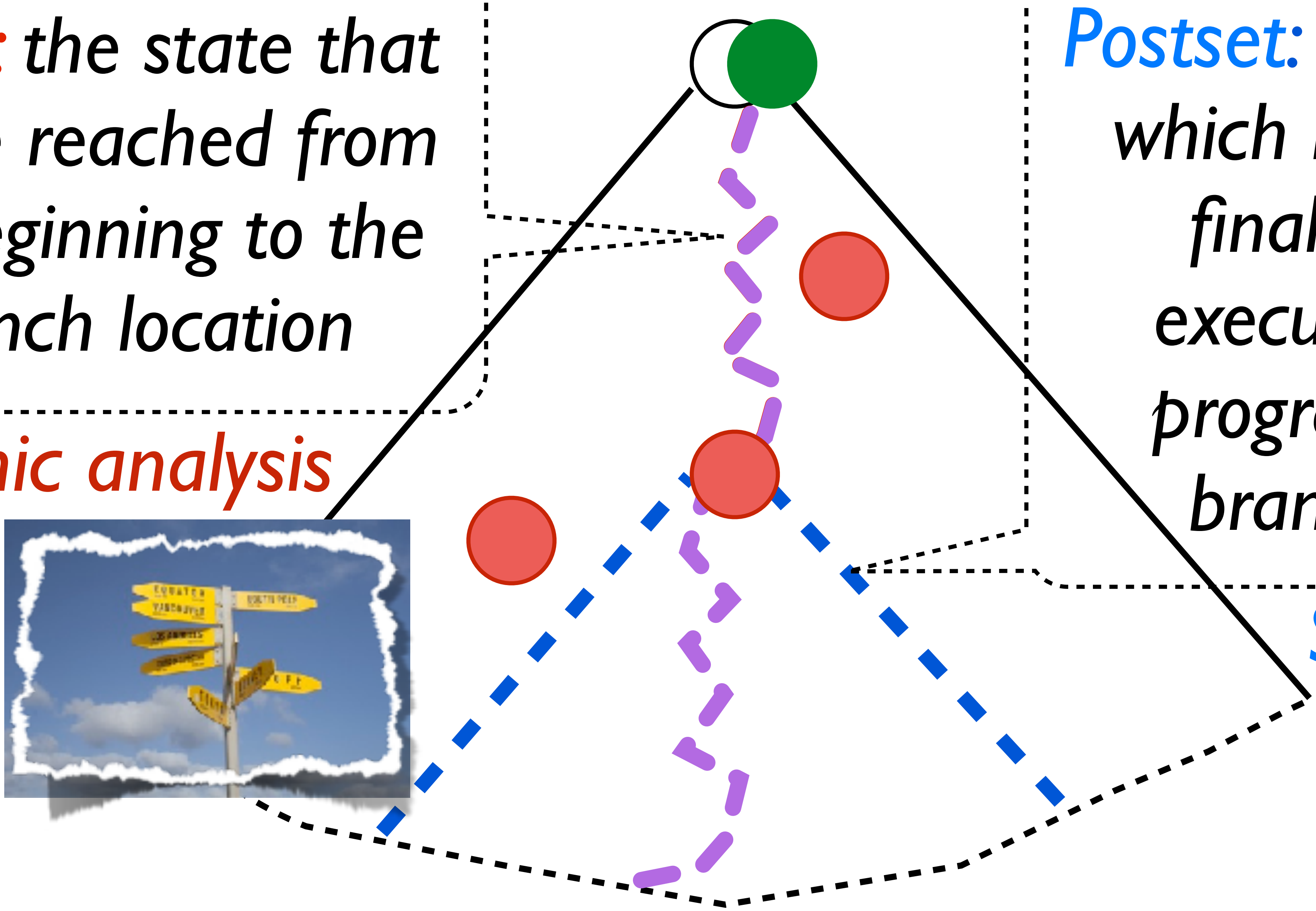
Prune redundant paths

Complement

Find counter-examples earlier

Key Idea-Guiding

history history ∩ future ≠ ∅ **future**



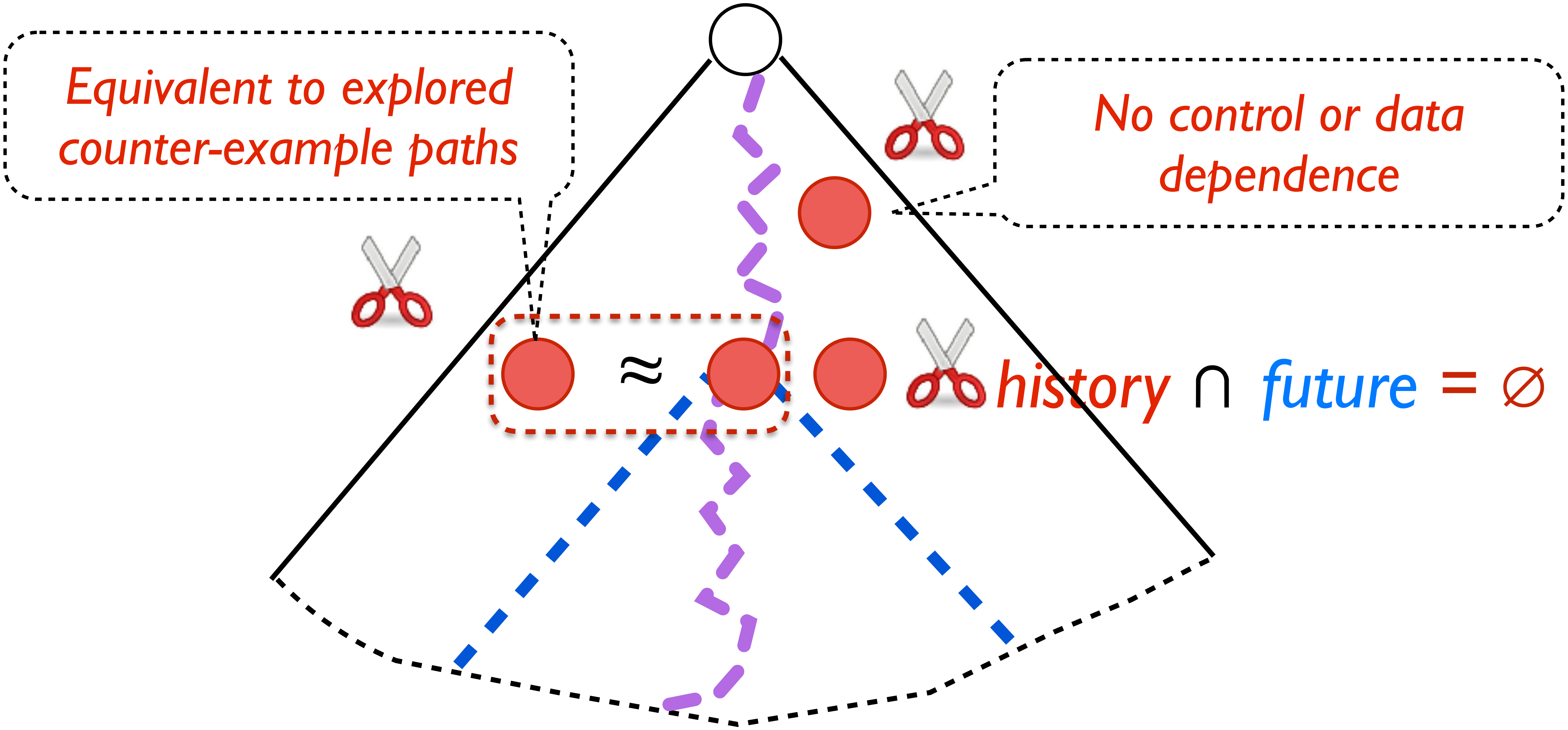
Preset: the state that can be reached from the beginning to the branch location

Dynamic analysis

Postset: the states from which it can reach a final state after executing the rest program after the branch location

Static analysis

Key Idea-Pruning

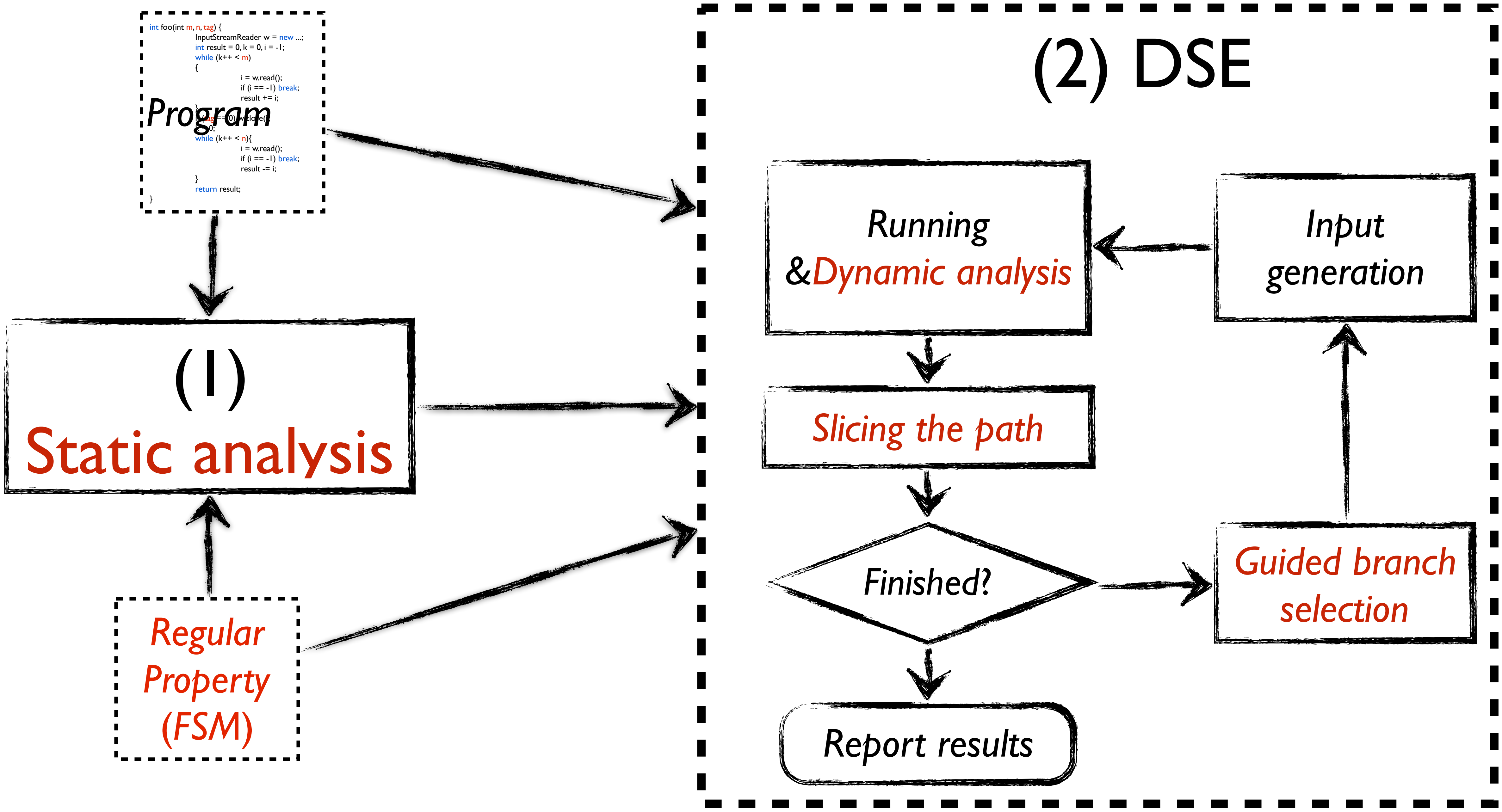


Equivalent to explored counter-example paths

No control or data dependence

Symbolic Verification of Regular Properties, ICSE 2018

Synergic Framework



```
int foo(int m, n, tag) {
        InputStreamReader w = new ...;
        int result = 0, k = 0, i = -1;
        while (k++ < m)
        {
                i = w.read();
                if (i == -1) break;
                result += i;
        }
        if (tag == 0) w.close();
        k = 0;
        while (k++ < n){
                i = w.read();
                if (i == -1) break;
                result -= i;
        }
        return result;
}
```

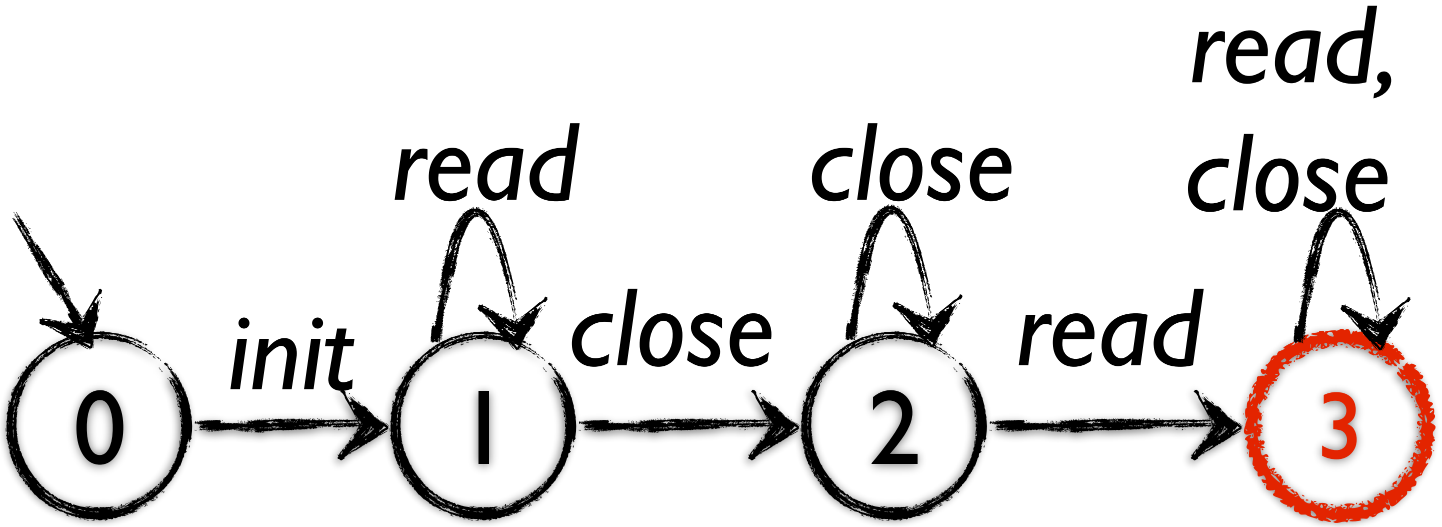
An Example

```
int foo(int m, int n, int[] a) {
   InputStreamReader w = new …;
   if (m > 50) m++;
   for (int i = 0; i < a.length - 1; i++) {
         if (a[i] > a[i+1]) {
            int temp = a[i];
            a[i+1] = a[i];
            a[i] = temp;
         }
   }
   if (a[i] == 100)
         w.close();
   while (n-- > 0){
         int j = w.read();
         if (j == -1) break;
         m += j;
   }
   return m;
}
```

Reader property

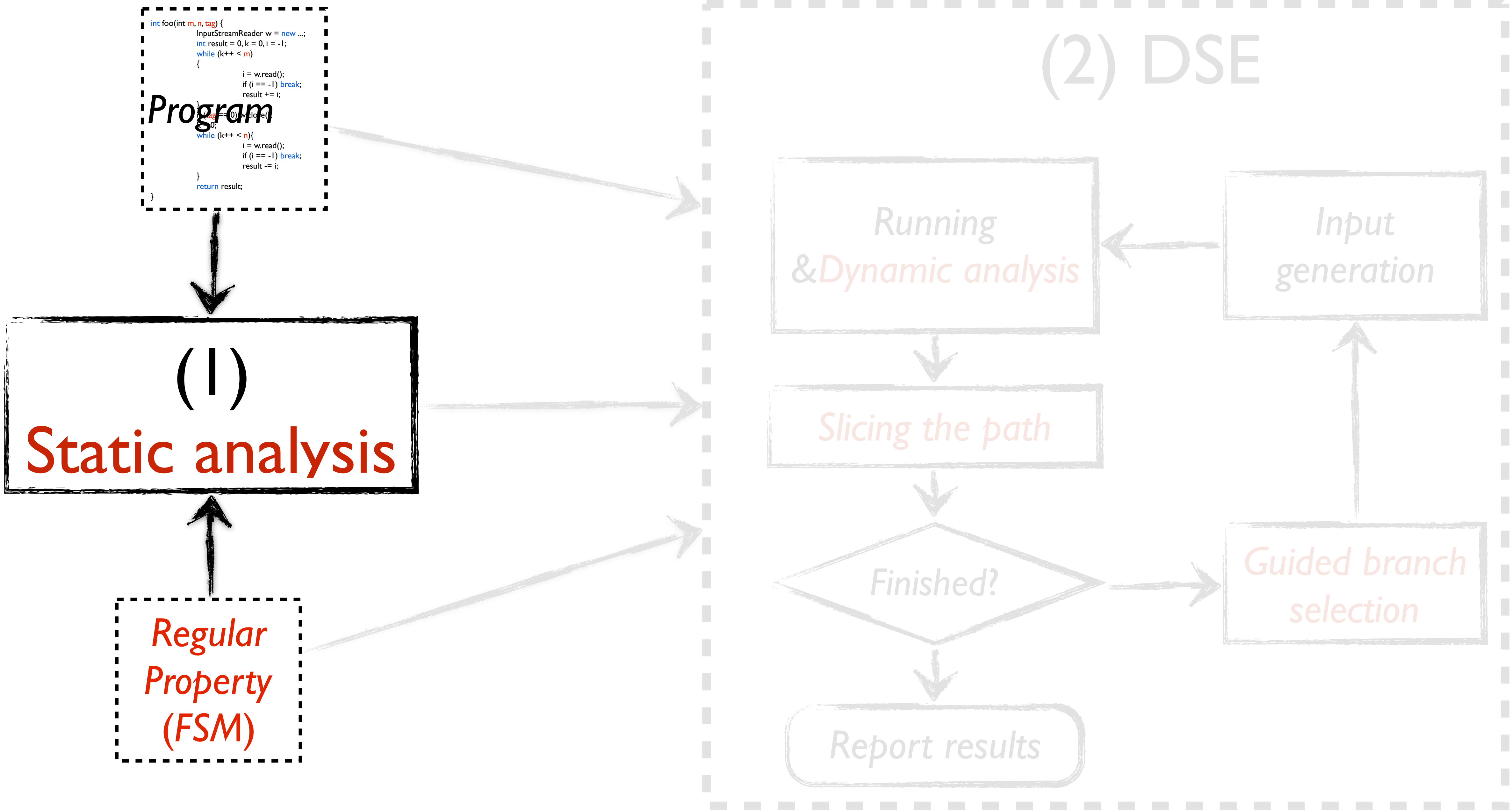
Cannot read after closed

The negation of the property



Read after closed

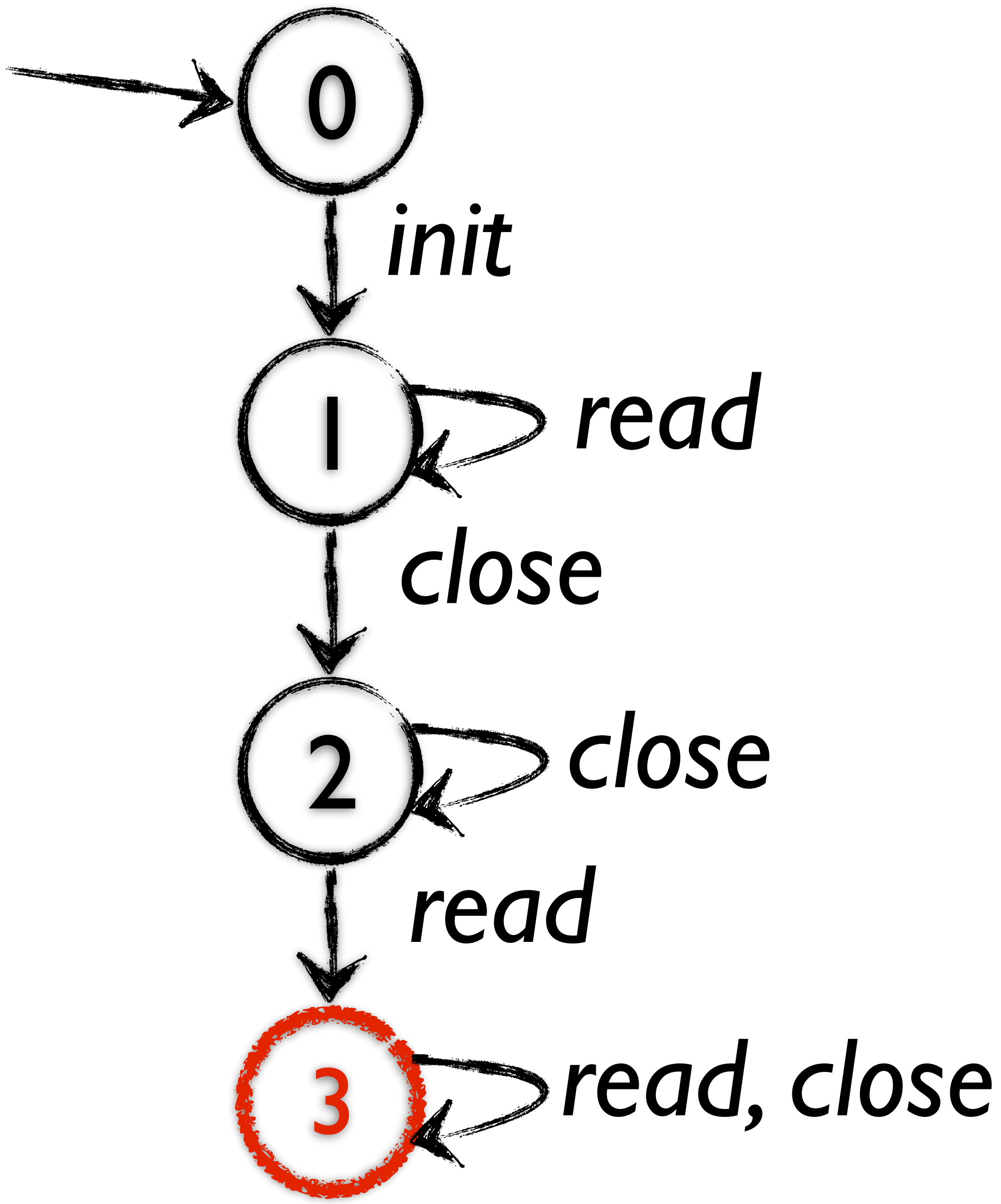
Synergic Framework



```
int foo(int m, n, tag) {
        InputStreamReader w = new ...;
        int result = 0, k = 0, i = -1;
        while (k++ < m)
        {
                i = w.read();
                if (i == -1) break;
                result += i;
        }
        if (tag == 0) w.close();
        k = 0;
        while (k++ < n){
                i = w.read();
                if (i == -1) break;
                result -= i;
        }
        return result;
}
```

Postset Calculation

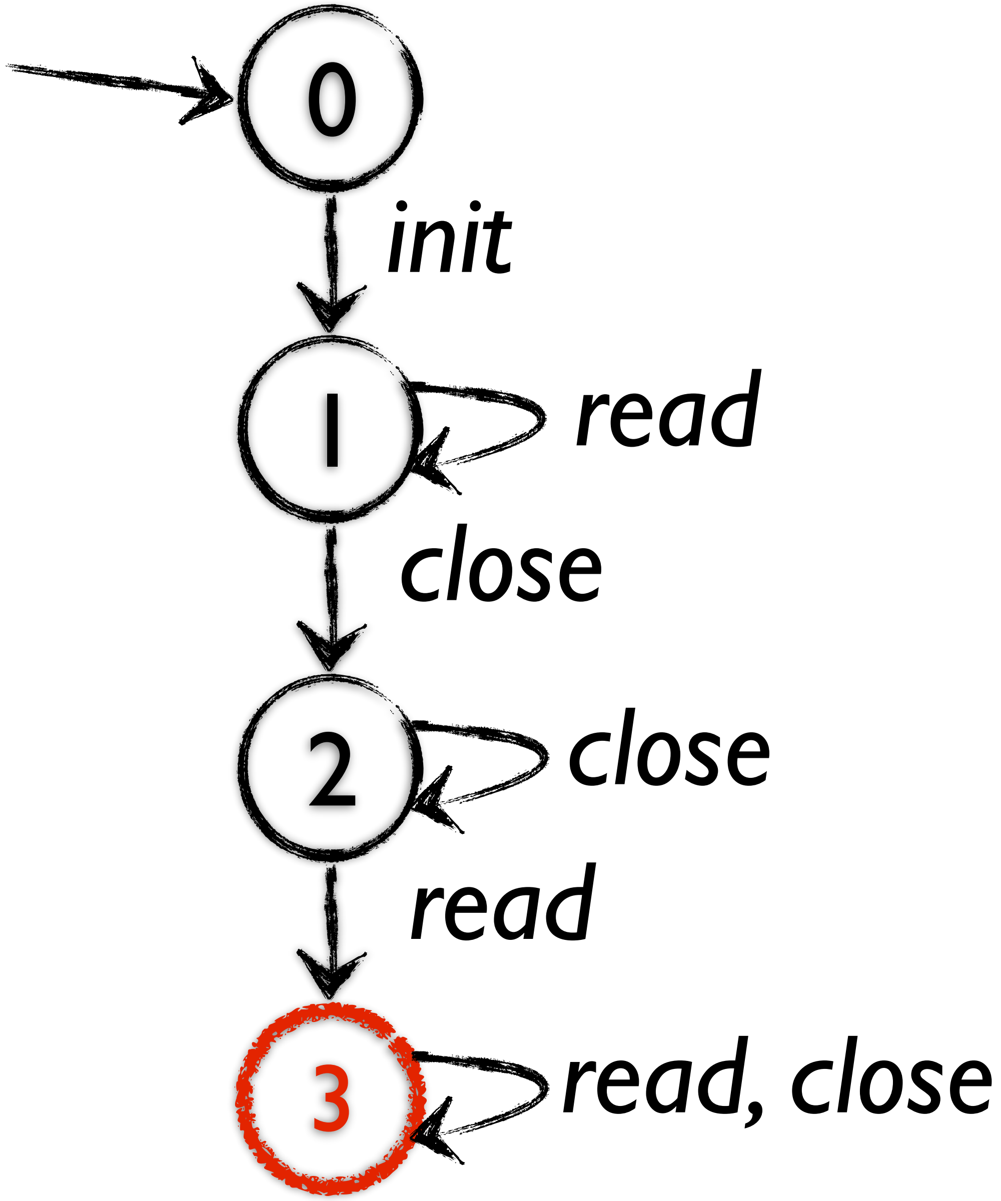
```
int foo(int m, int n, int[] a) {
  InputStreamReader w = new …;
  if (m > 50) m++;
  for (int i = 0; i < a.length-1; i++) {
      if (a[i] > a[i+1]) {
        int temp = a[i];
        a[i+1] = a[i];
        a[i] = temp;
      }
  }
  if (a[i] == 100)
      w.close();
  while (n-- > 0){
      int j = w.read();
      if (j == -1) break;
      m += j;
  }
  return m;
}
```



Backward data flow analysis [Clara, ICSE'10]

$O(|E| \times |D|^3)$

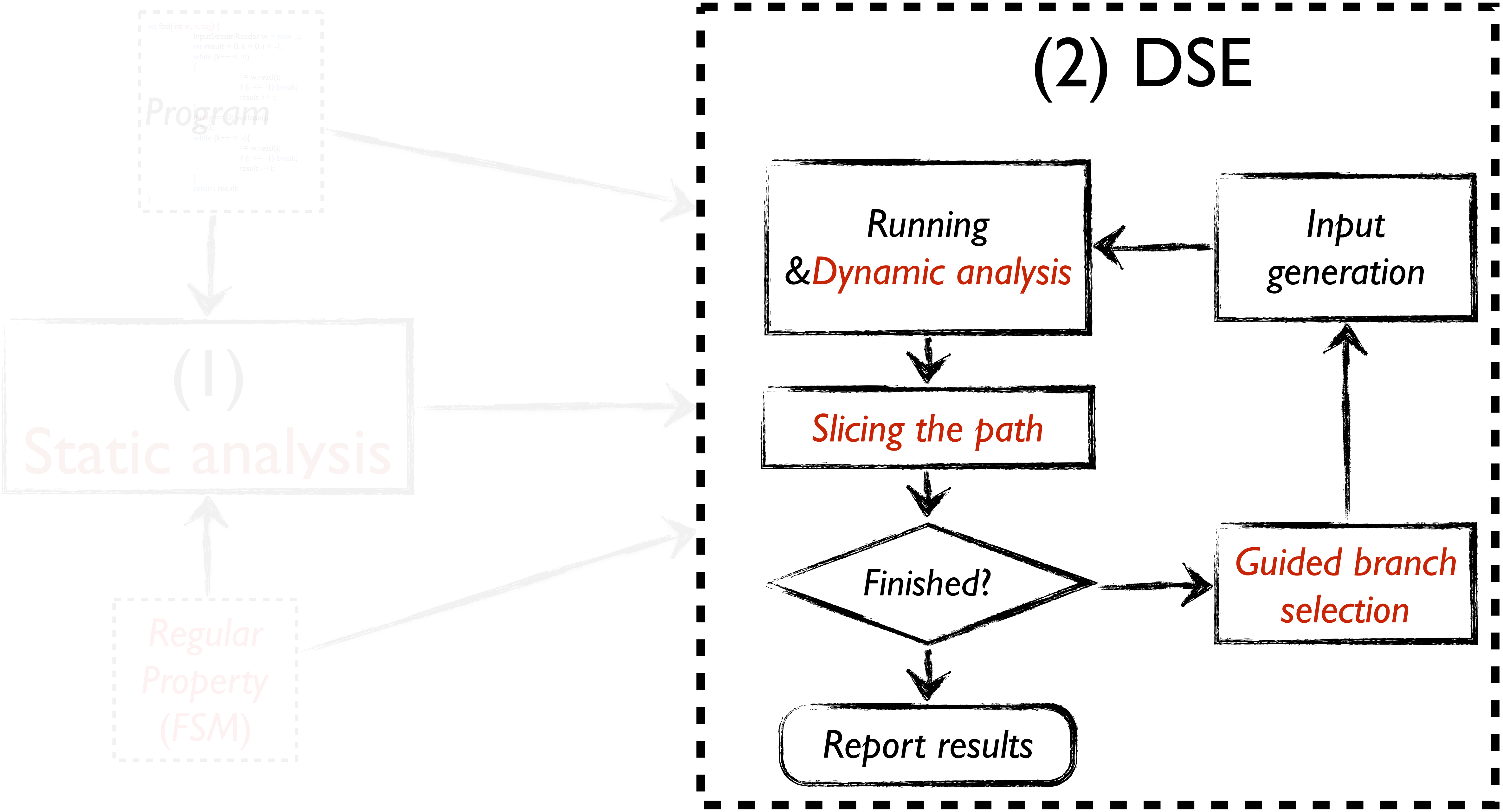
```
int foo(int m, int n, int[] a) {//{0}
  InputStreamReader w = new …;
  if (m > 50) m++; //{1, 2, 3}
  for (int i = 0; i < a.length-1; i++) {
      if (a[i] > a[i+1]) { //{1, 2, 3}
        int temp = a[i]; //{1, 2, 3}
        a[i+1] = a[i]; //{1, 2, 3}
        a[i] = temp; //{1, 2, 3}
      } //{1, 2, 3}
  } //{1, 2, 3}
  if (a[i] == 100) //{1, 2, 3}
      w.close(); //{2, 3}
  while (n-- > 0){ //{2, 3}
      int j = w.read(); //{2, 3}
      if (j == -1) break; //{2, 3}
      m += j; //{2, 3}
  } //{3}
  return m; //{3}
}
```



Backward data flow analysis [Clara, ICSE'10]

$O(|E| \times |D|^3)$

Synergic Framework

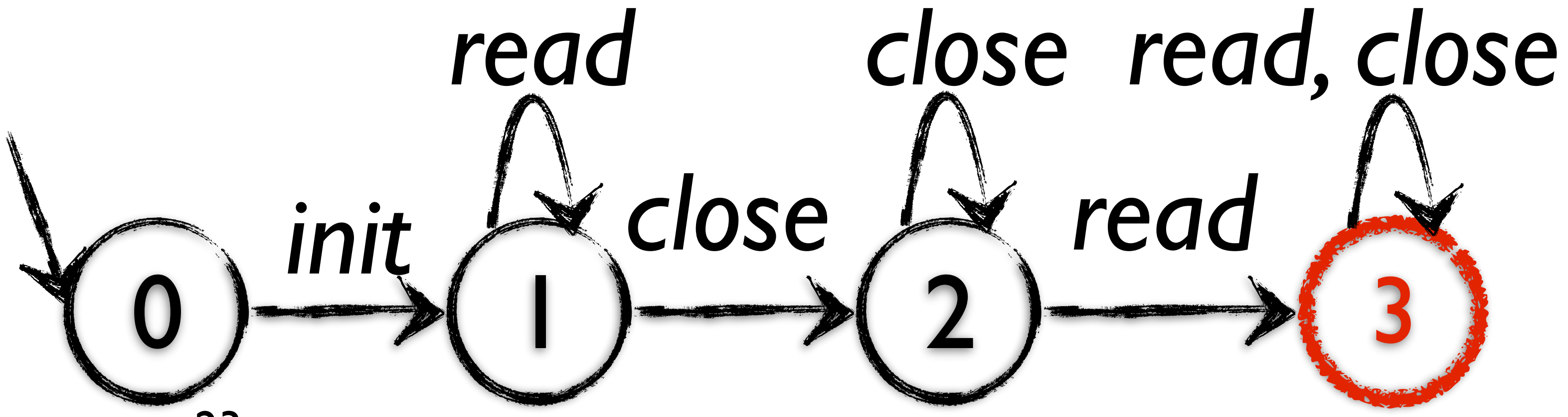


```
int foo(int m, n, tag) {
        InputStreamReader w = new ...;
        int result = 0, k = 0, i = -1;
        while (k++ < m)
        {
                i = w.read();
                if (i == -1) break;
                result += i;
        }
        if (tag == 0) w.close();
        k = 0;
        while (k++ < n){
                i = w.read();
                if (i == -1) break;
                result -= i;
        }
        return result;
}
```

1st Iteration

```
int foo(int m, int n, int[] a) {
   InputStreamReader w = new …;
   if (m > 50) m++;
   for (int i = 0; i < a.length - 1; i++) {
         if (a[i] > a[i+1]) {
            int temp = a[i];
            a[i+1] = a[i];
            a[i] = temp;
         }
   }
   if (a[i] == 100)
         w.close();
   while (n-- > 0){
         int j = w.read();
         if (j == -1) break;
         m += j;
   }
   return m;
}
```

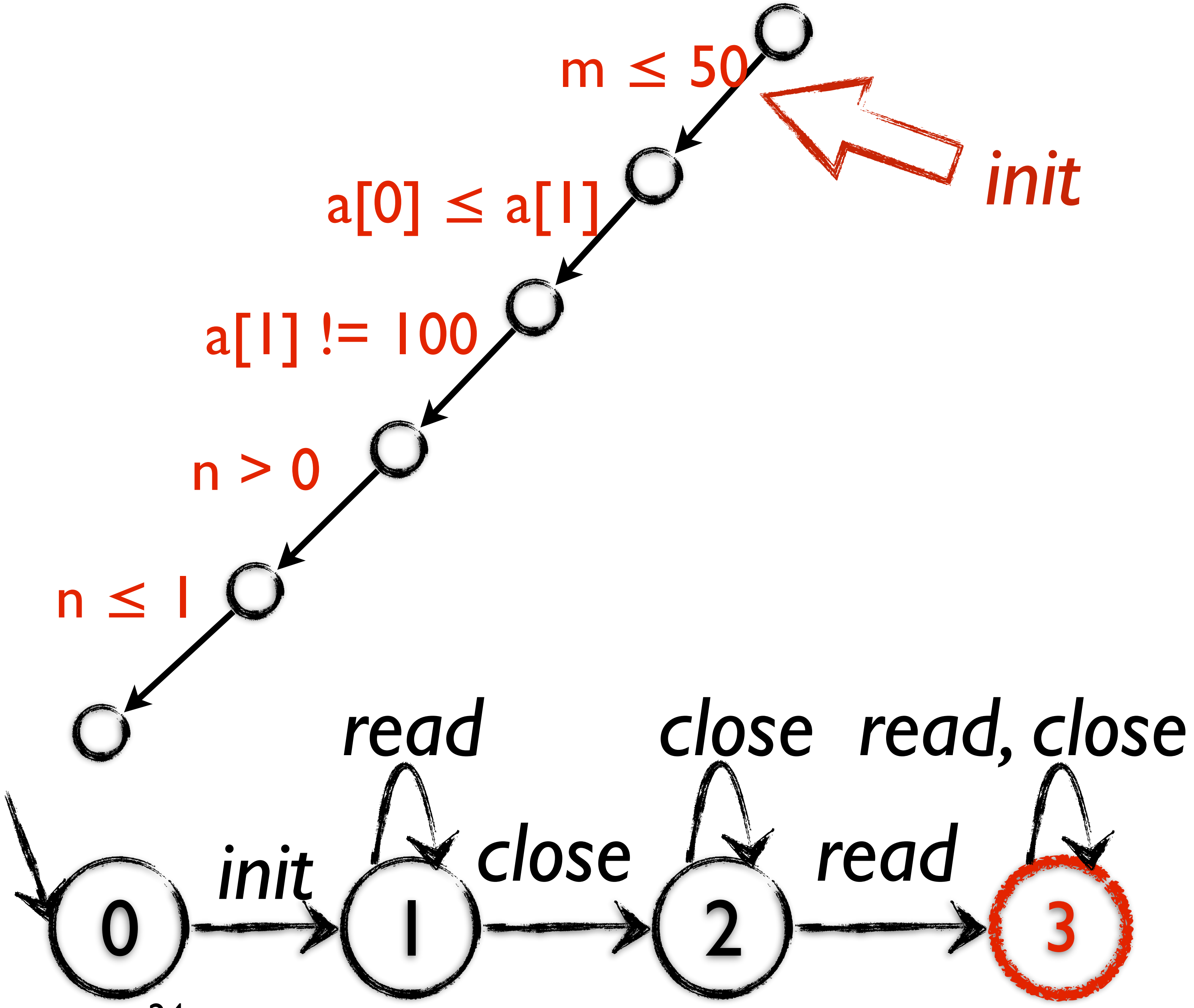
(m=1, n=1, a={0, 1})



1st Iteration

```
int foo(int m, int n, int[] a) {
   InputStreamReader w = new …;
   if (m > 50) m++;
   for (int i = 0; i < a.length - 1; i++) {
         if (a[i] > a[i+1]) {
            int temp = a[i];
            a[i+1] = a[i];
            a[i] = temp;
         }
   }
   if (a[i] == 100)
         w.close();
   while (n-- > 0){
         int j = w.read();
         if (j == -1) break;
         m += j;
   }
   return m;
}
```

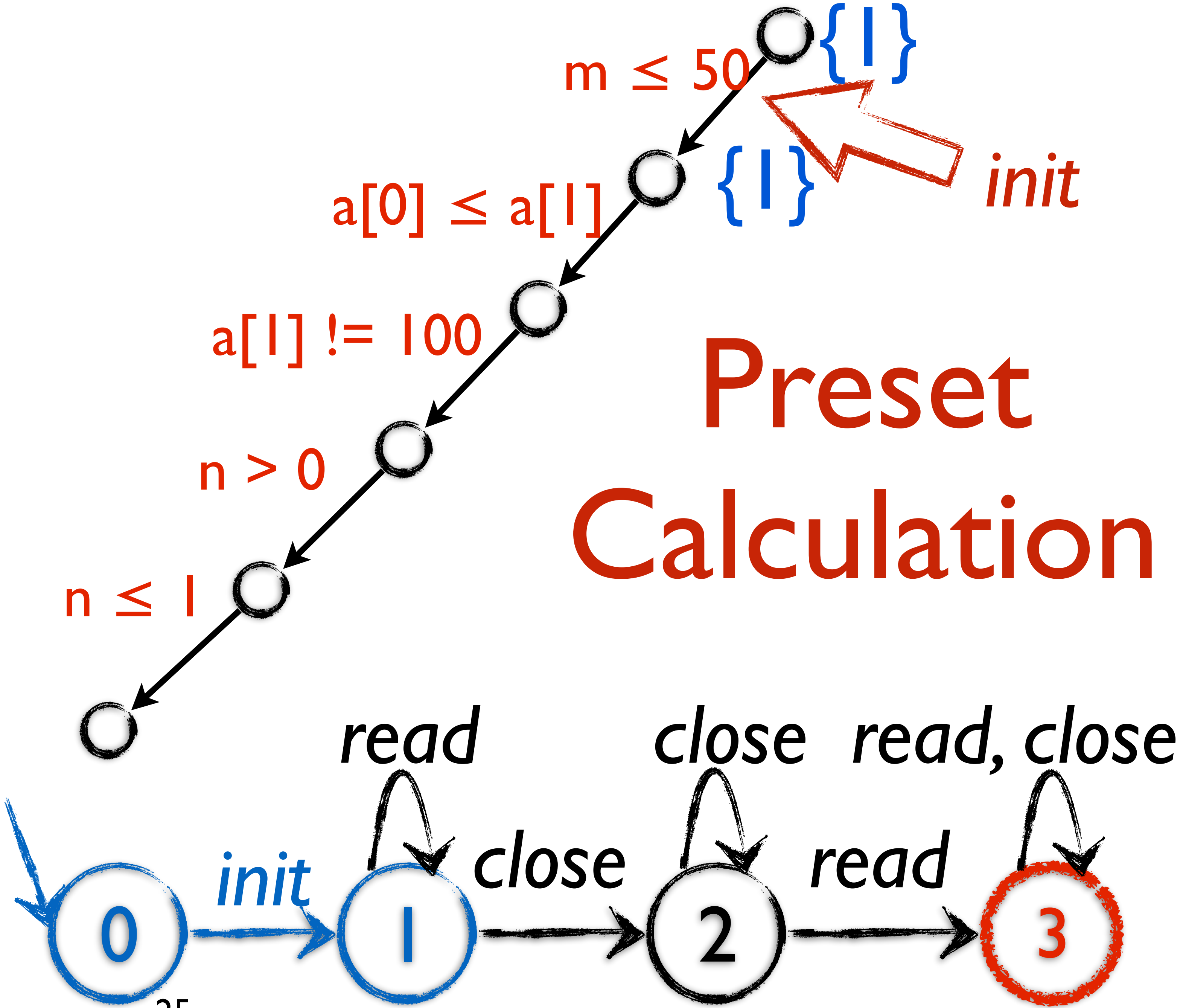
(m=1, n=1, a={0, 1})



1st Iteration

```
int foo(int m, int n, int[] a) {
   InputStreamReader w = new ...;
   if (m > 50) m++;
   for (int i = 0; i < a.length - 1; i++) {
        if (a[i] > a[i+1]) {
           int temp = a[i];
           a[i+1] = a[i];
           a[i] = temp;
        }
   }
   if (a[i] == 100)
        w.close();
   while (n-- > 0){
        int j = w.read();
        if (j == -1) break;
        m += j;
   }
   return m;
}
```

(m=1, n=1, a={0, 1})

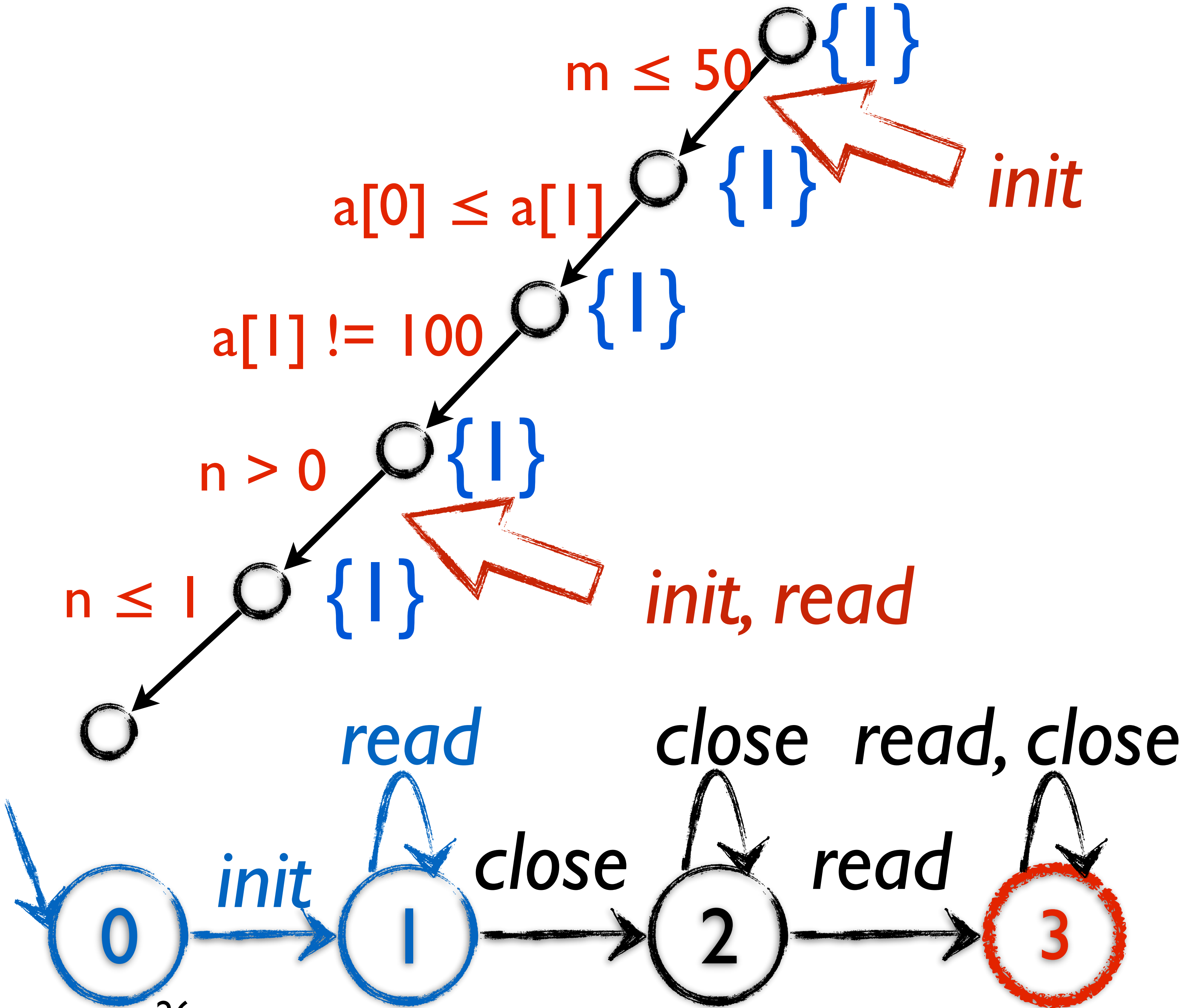


Preset Calculation

1st Iteration

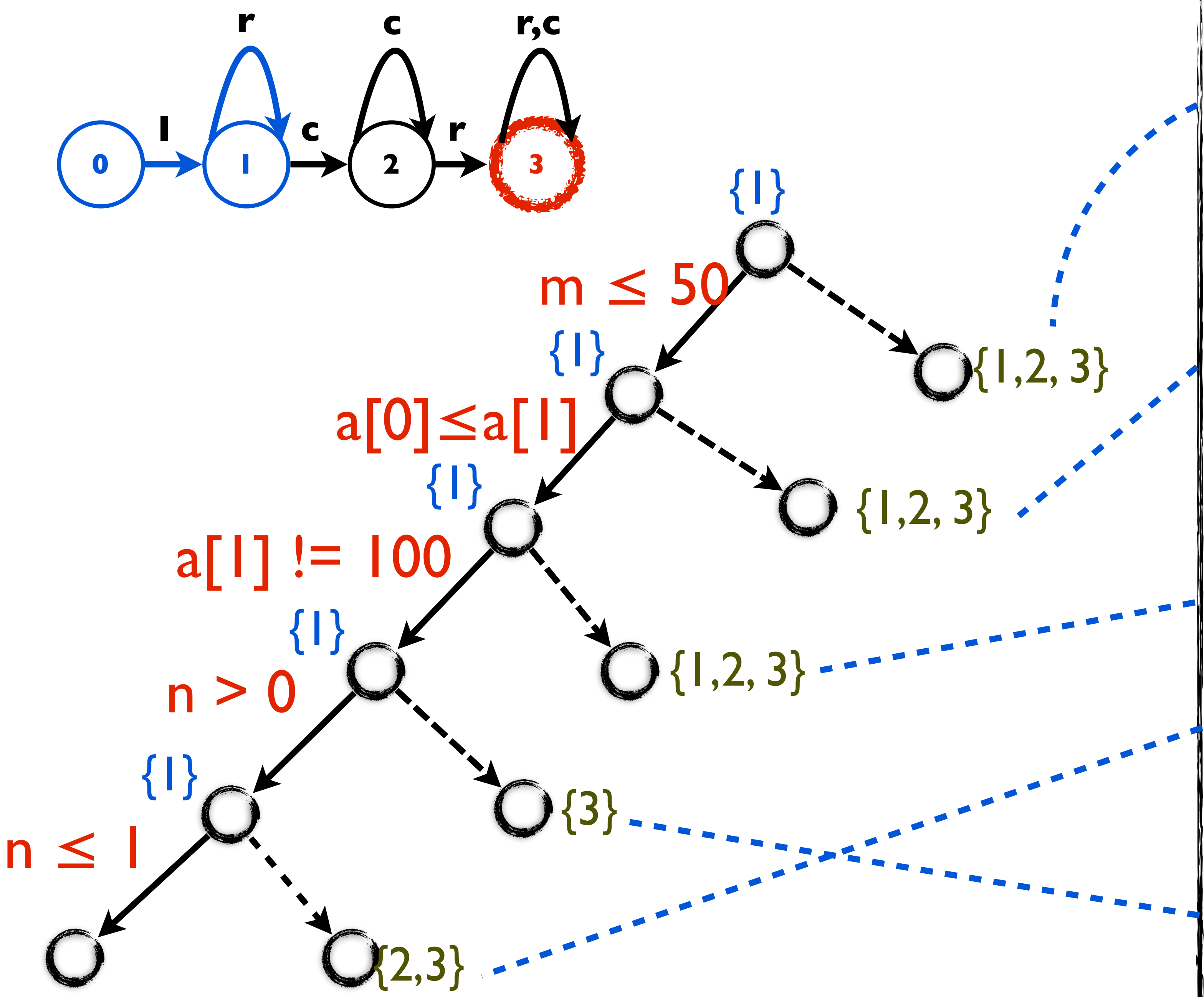
```
int foo(int m, int n, int[] a) {
   InputStreamReader w = new …;
   if (m > 50) m++;
   for (int i = 0; i < a.length - 1; i++) {
         if (a[i] > a[i+1]) {
            int temp = a[i];
            a[i+1] = a[i];
            a[i] = temp;
         }
   }
   if (a[i] == 100)
         w.close();
   while (n-- > 0){
         int j = w.read();
         if (j == -1) break;
         m += j;
   }
   return m;
}
```

(m=1, n=1, a={0, 1})



1st Iteration

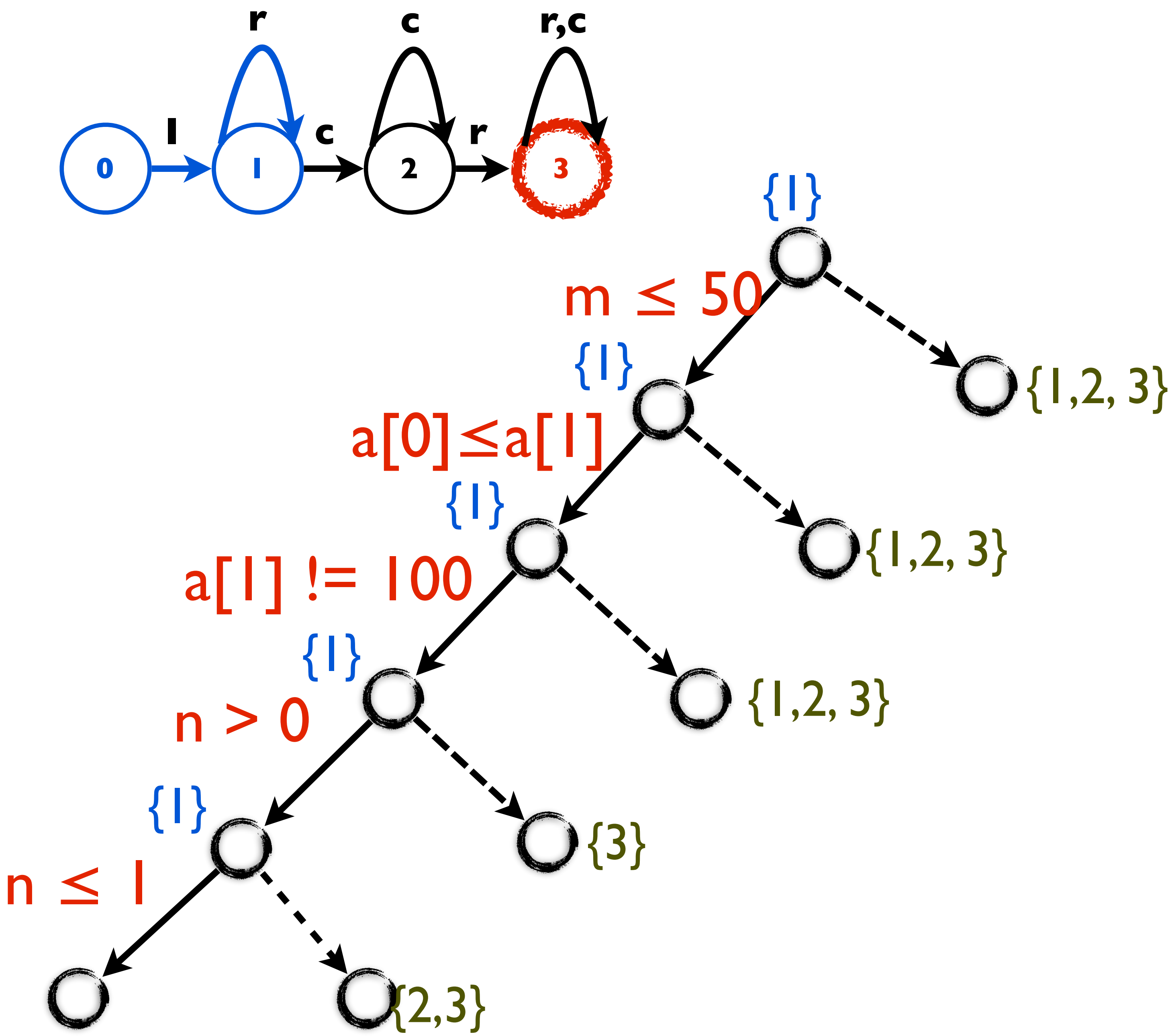
(m=1, n=1, a={0, 1})



```
int foo(int m, int n, int[] a) {//{0}
   InputStreamReader w = new …;
   if (m > 50) m++; //{1, 2, 3}
   for (int i = 0; i < a.length-1; i++) {
        if (a[i] > a[i+1]) { //{1, 2, 3}
          int temp = a[i]; //{1, 2, 3}
          a[i+1] = a[i]; //{1, 2, 3}
          a[i] = temp; //{1, 2, 3}
        } //{1, 2, 3}
   } //{1, 2, 3}
   if (a[i] == 100) //{1, 2, 3}
        w.close(); //{2, 3}
   while (n-- > 0){ //{2, 3}
        int j = w.read(); //{2, 3}
        if (j == -1) break; //{2, 3}
        m += j; //{2, 3}
   } //{3}
   return m; //{3}
}
```

1st Iteration

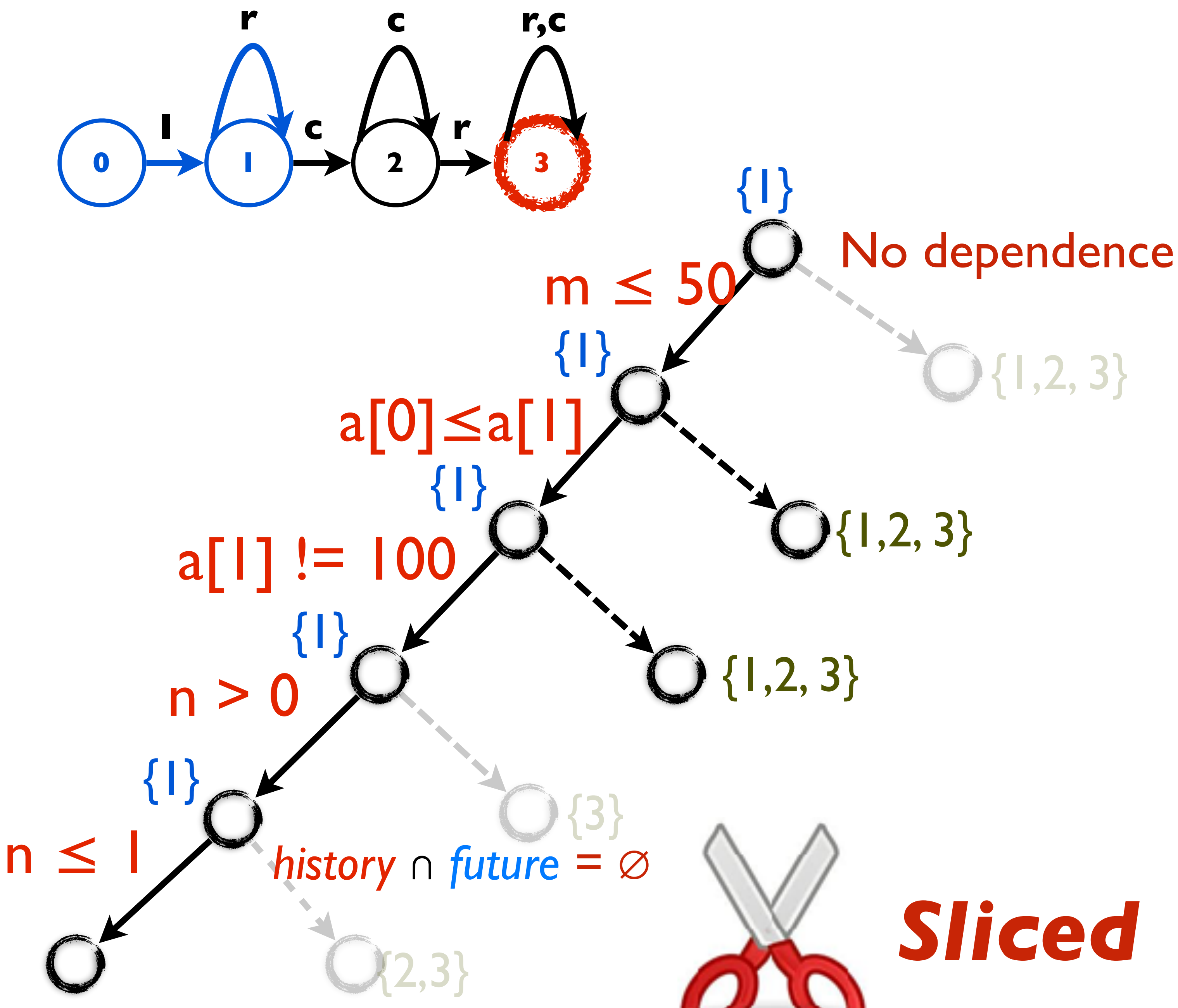
(m=1, n=1, a={0, 1})



```
int foo(int m, int n, int[] a) {//{0}
   InputStreamReader w = new …;
   if (m > 50) m++; //{1, 2, 3}
   for (int i = 0; i < a.length-1; i++) {
         if (a[i] > a[i+1]) { //{1, 2, 3}
            int temp = a[i]; //{1, 2, 3}
            a[i+1] = a[i]; //{1, 2, 3}
            a[i] = temp; //{1, 2, 3}
         } //{1, 2, 3}
   } //{1, 2, 3}
   if (a[i] == 100) //{1, 2, 3}
         w.close(); //{2, 3}
   while (n-- > 0){ //{2, 3}
         int j = w.read(); //{2, 3}
         if (j == -1) break; //{2, 3}
         m += j; //{2, 3}
   } //{3}
   return m; //{3}
}
```

1st Iteration

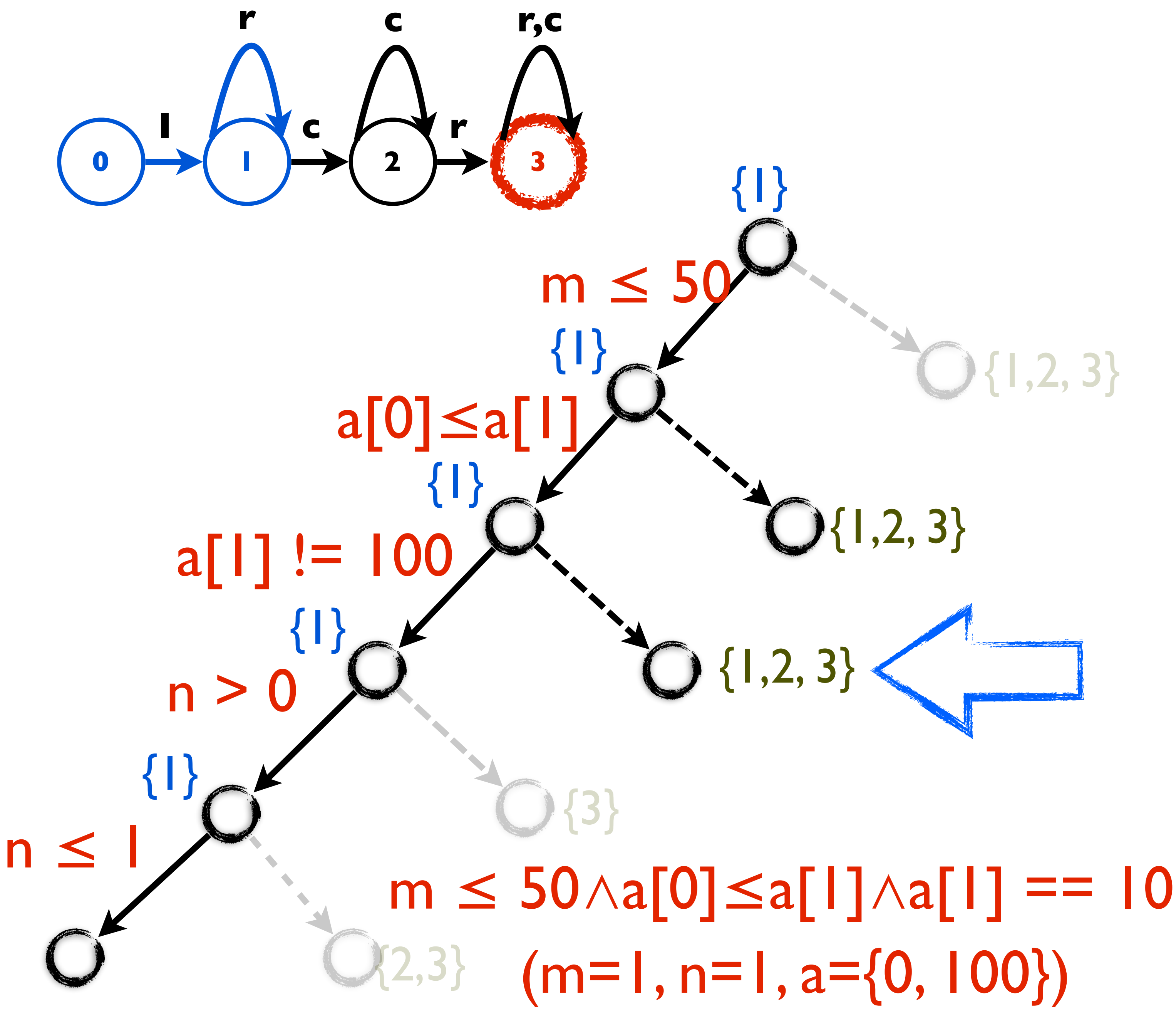
(m=1, n=1, a={0, 1})



```
int foo(int m, int n, int[] a) {//{0}
   InputStreamReader w = new …;
   if (m > 50) m++; //{1, 2, 3}
   for (int i = 0; i < a.length-1; i++) {
         if (a[i] > a[i+1]) { //{1, 2, 3}
            int temp = a[i]; //{1, 2, 3}
            a[i+1] = a[i]; //{1, 2, 3}
            a[i] = temp; //{1, 2, 3}
         } //{1, 2, 3}
   } //{1, 2, 3}
   if (a[i] == 100) //{1, 2, 3}
         w.close(); //{2, 3}
   while (n-- > 0){ //{2, 3}
         int j = w.read(); //{2, 3}
         if (j == -1) break; //{2, 3}
         m += j; //{2, 3}
   } //{3}
   return m; //{3}
}
```

1st Iteration

(m=1, n=1, a={0, 1})



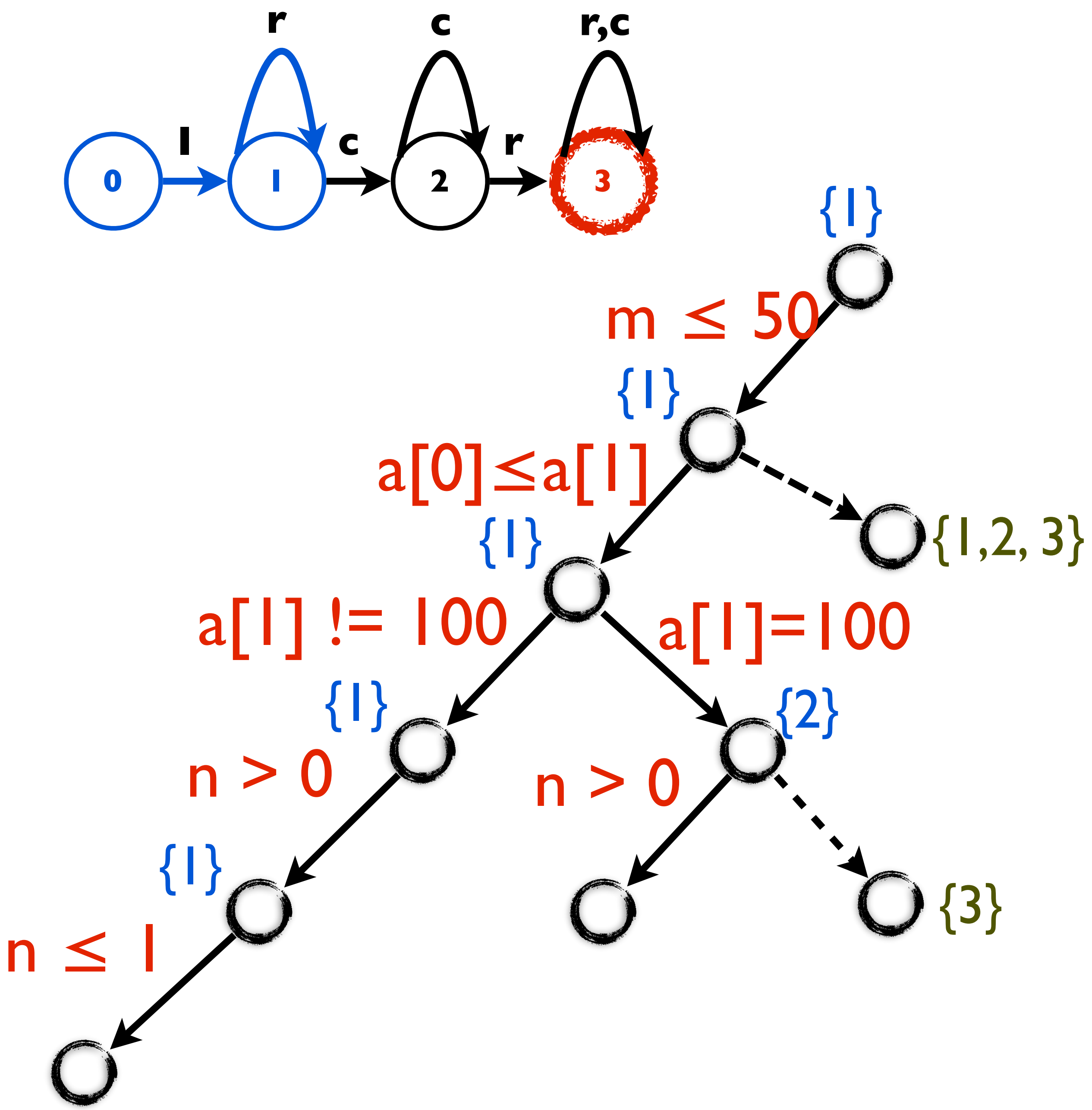
m ≤ 50∧a[0]≤a[1]∧a[1] == 100

(m=1, n=1, a={0, 100})

```
int foo(int m, int n, int[] a) {//{0}
   InputStreamReader w = new …;
   if (m > 50) m++; //{1, 2, 3}
   for (int i = 0; i < a.length-1; i++) {
       if (a[i] > a[i+1]) { //{1, 2, 3}
         int temp = a[i]; //{1, 2, 3}
         a[i+1] = a[i]; //{1, 2, 3}
         a[i] = temp; //{1, 2, 3}
       } //{1, 2, 3}
   } //{1, 2, 3}
   if (a[i] == 100) //{1, 2, 3}
       w.close(); //{2, 3}
   while (n-- > 0){ //{2, 3}
       int j = w.read(); //{2, 3}
       if (j == -1) break; //{2, 3}
       m += j; //{2, 3}
   } //{3}
   return m; //{3}
}
```

2nd Iteration

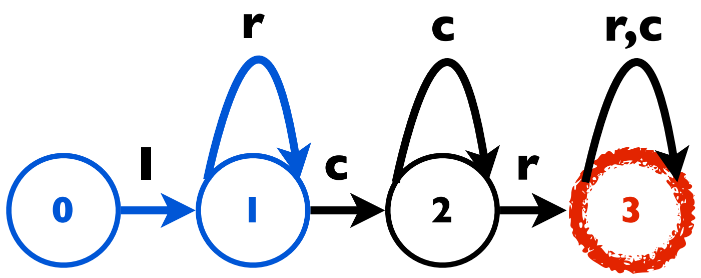
(m=1, n=1, a={0, 100})

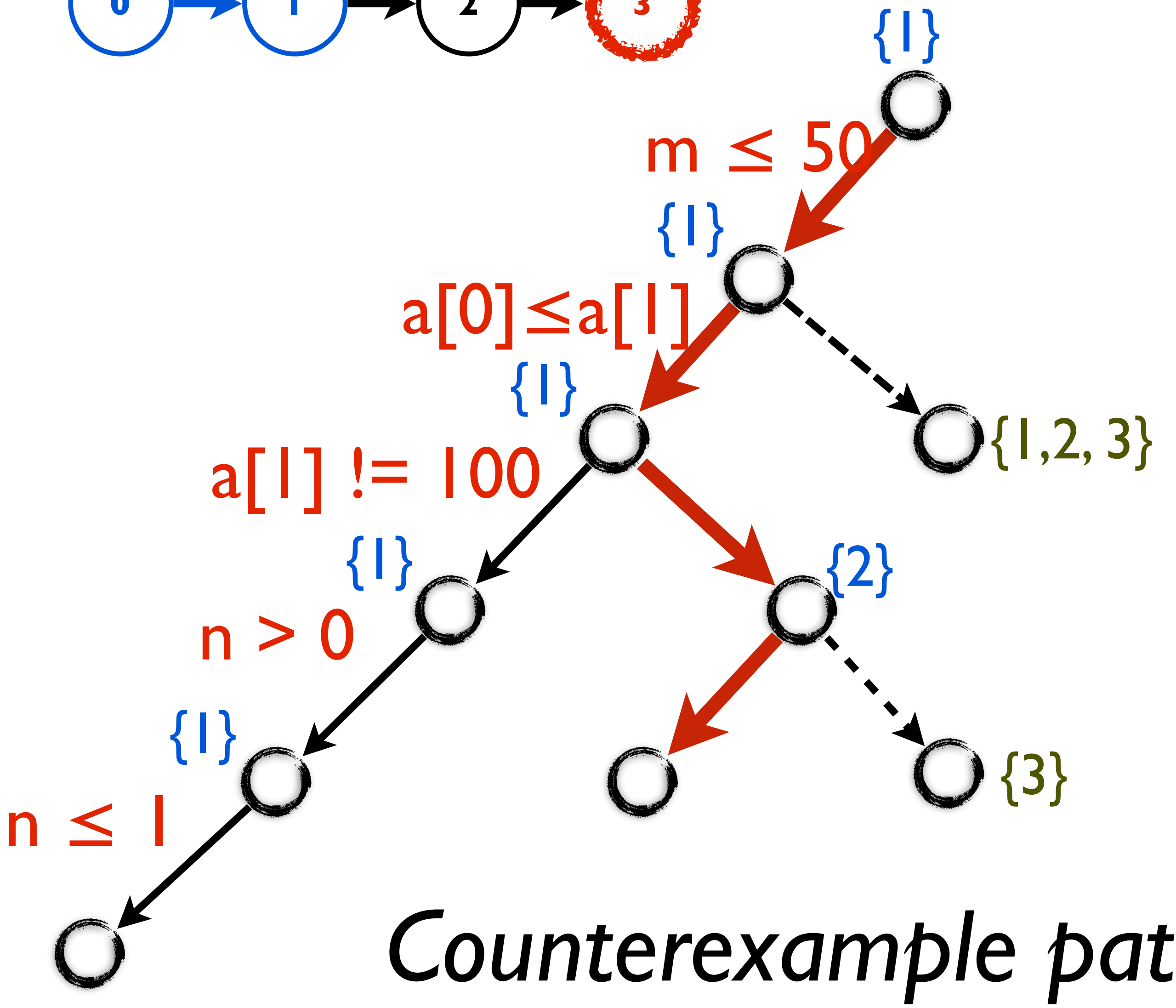


```
int foo(int m, int n, int[] a) {//{0}
   InputStreamReader w = new ...;
   if (m > 50) m++; //{1, 2, 3}
   for (int i = 0; i < a.length-1; i++) {
         if (a[i] > a[i+1]) { //{1, 2, 3}
            int temp = a[i]; //{1, 2, 3}
            a[i+1] = a[i]; //{1, 2, 3}
            a[i] = temp; //{1, 2, 3}
         } //{1, 2, 3}
   } //{1, 2, 3}
   if (a[i] == 100) //{1, 2, 3}
         w.close(); //{2, 3}
   while (n-- > 0){ //{2, 3}
         int j = w.read(); //{2, 3}
         if (j == -1) break; //{2, 3}
         m += j; //{2, 3}
   } //{3}
   return m; //{3}
}
```

2nd Iteration

(m=1, n=1, a={0, 100})



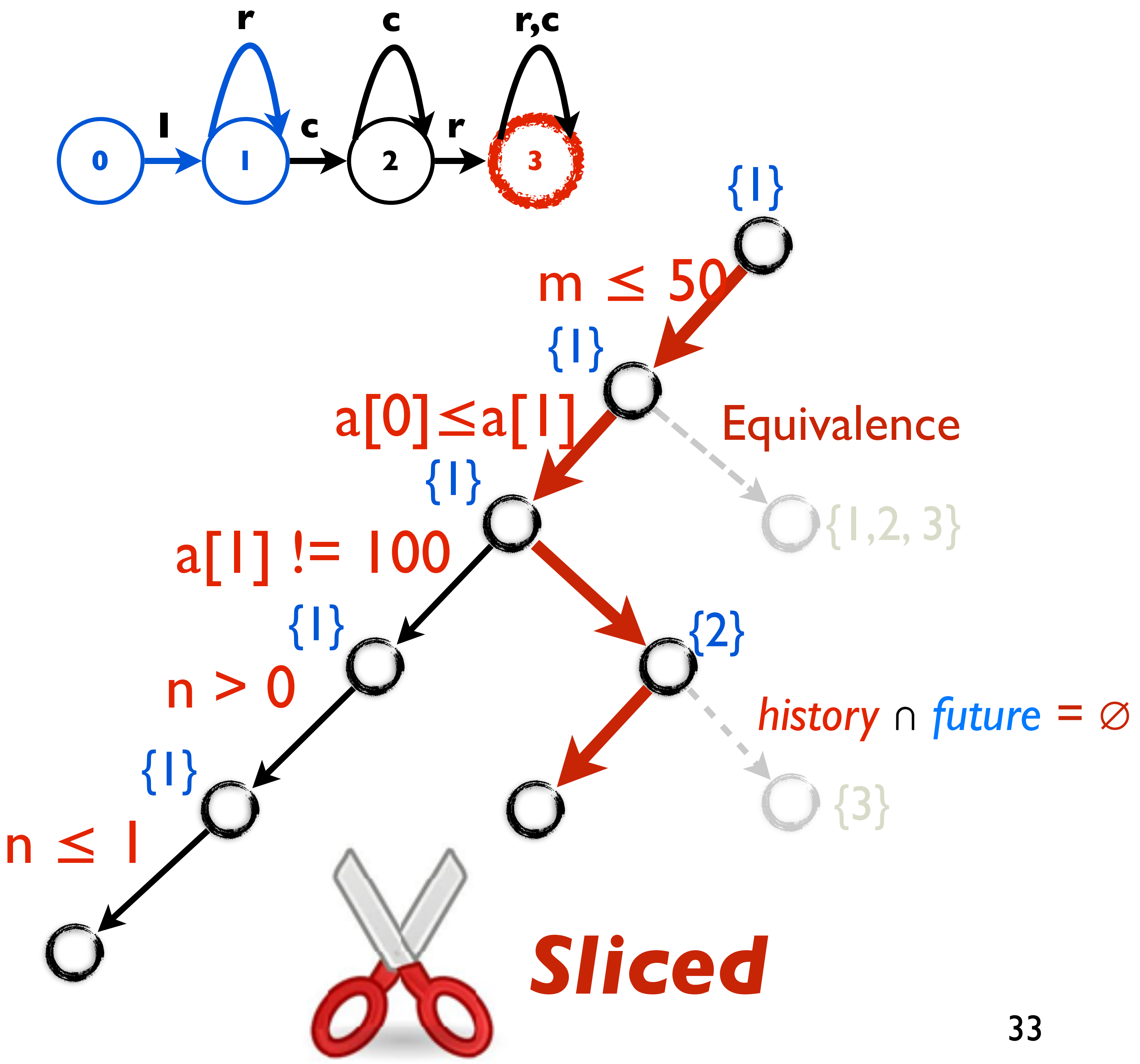


Counterexample path

```
int foo(int m, int n, int[] a) {//{0}
   InputStreamReader w = new …;
   if (m > 50) m++; //{1, 2, 3}
   for (int i = 0; i < a.length-1; i++) {
         if (a[i] > a[i+1]) { //{1, 2, 3}
            int temp = a[i]; //{1, 2, 3}
            a[i+1] = a[i]; //{1, 2, 3}
            a[i] = temp; //{1, 2, 3}
         } //{1, 2, 3}
   } //{1, 2, 3}
   if (a[i] == 100) //{1, 2, 3}
         w.close(); //{2, 3}
   while (n-- > 0){ //{2, 3}
         int j = w.read(); //{2, 3}
         if (j == -1) break; //{2, 3}
         m += j; //{2, 3}
   } //{3}
   return m; //{3}
}
```

2nd Iteration

(m=1, n=1, a={0, 100})



```
int foo(int m, int n, int[] a) {//{0}
   InputStreamReader w = new …;
   if (m > 50) m++; //{1, 2, 3}
   for (int i = 0; i < a.length-1; i++) {
         if (a[i] > a[i+1]) { //{1, 2, 3}
            int temp = a[i]; //{1, 2, 3}
            a[i+1] = a[i]; //{1, 2, 3}
            a[i] = temp; //{1, 2, 3}
         } //{1, 2, 3}
   } //{1, 2, 3}
   if (a[i] == 100) //{1, 2, 3}
         w.close(); //{2, 3}
   while (n-- > 0){ //{2, 3}
         int j = w.read(); //{2, 3}
         if (j == -1) break; //{2, 3}
         m += j; //{2, 3}
   } //{3}
   return m; //{3}
}
```

```
int foo(int m, int n, int[] a) {
  InputStreamReader w = new ...;
  if (m > 50) m++;
  for (int i = 0; i < a.length-1; i++) {
      if (a[i] > a[i+1]) {
        int temp = a[i];
        a[i+1] = a[i];
        a[i] = temp;
      }
  }
  if (a[i] == 100)
      w.close();
  while (n-- > 0){
      int j = w.read();
      if (j == -1) break;
      m += j;
  }
  return m;
}
```

no data or control dependence

equivalent to the counterexample path

Guiding to this branch in the 2nd iteration

not possible to violate the property

An Example

```
int foo(int m, int n, int[] a) {
   InputStreamReader w = new …;
   if (m > 50) m++;
   for (int i = 0; i < a.length - 1; i++) {
         if (a[i] > a[i+1]) {
            int temp = a[i];
            a[i+1] = a[i];
            a[i] = temp;
         }
   }
   if (a[i] == 100)
         w.close();
   while (n-- > 0){
         int j = w.read();
         if (j == -1) break;
         m += j;
   }
   return m;
}
```

Reader property

Cannot read after closed

Only 2 paths are needed to complete the path exploration

Method	Result
DFS	Unfolding two loops
Guiding	2nd path, Unfolding two loops
Path Slicing	Only one branch is sliced

Implementation & Experiment Setup

- Implement for Java based on RGSE (*FSE 2017*)
- 15 real world open source Java programs
    - 250K LOC in total

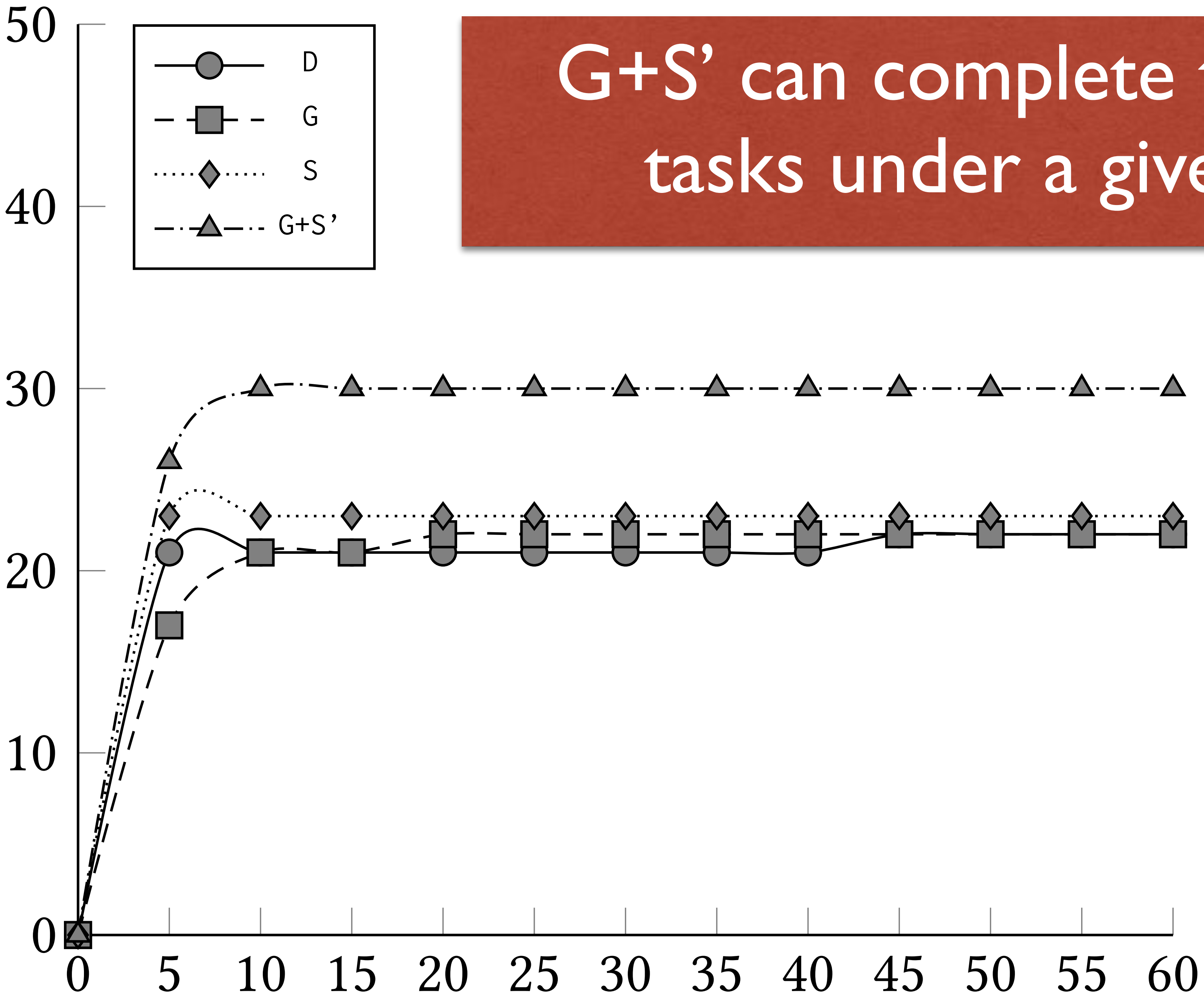
RGSE: A Regular Property Guided Symbolic Executor for Java, FSE 2017, Tool Demo

Program	**LOC**	**Brief Description**
rhino-a	19799	Javascript interpreter
soot-c	32358	Static analysis tool
jlex	4400	Lexical analyzer
bloat	45375	Java bytecode optimization
bmpdecoder	531	BMP file decoder
ftpclient	2436	FTP client in Java
pobs	5488	Java parser objects
jpat	3245	Java string parser
jericho	25657	Jericho HTML Parser
nano-xml	3317	Non-validating XML parser
htmlparser	21830	HTML parser in Java
xml	5138	XML parser in Java
fastjson	20223	JSON library from alibaba
jep	42868	Mathematics library
udl	26896	UDL language library
Total	**259642**	**15 open source programs**

Implementation & Experiment Setup

- Implement for Java based on RGSE (*FSE 2017*)
- 15 real world open source Java programs
  - 250K LOC in total
- Properties
  - JDK's single- and multi-objects typestate
  - User defined
- Verify each program/property in 90 minutes

Completed Verification Tasks

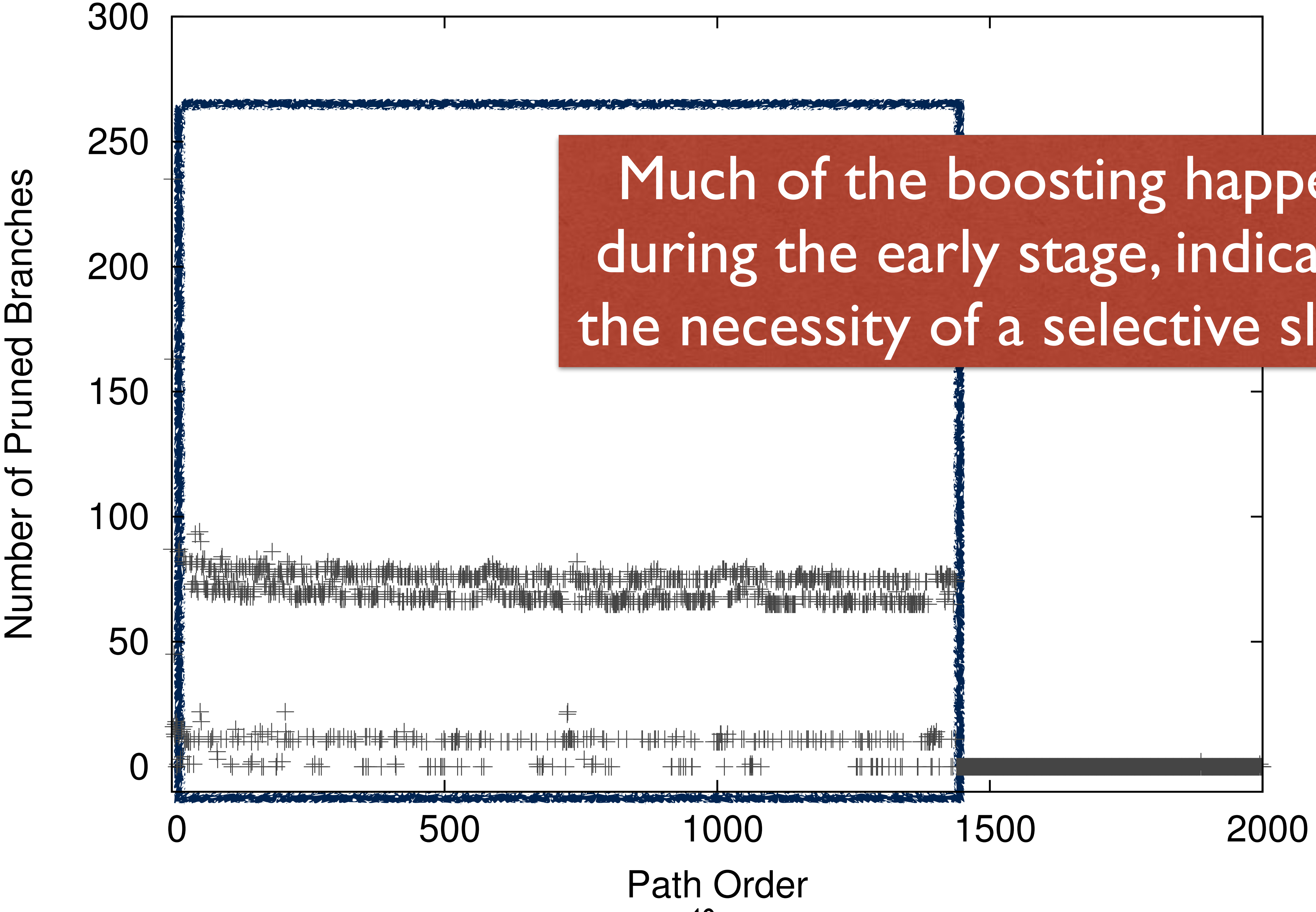


G+S' can complete the most number of tasks under a given time threshold

	#	Rate
G+S'	**30**	**76.9%**
DFS	22	56.4%
Guiding	22	56.4%
Slicing	23	58.9%

39 task in total

Branch Pruning



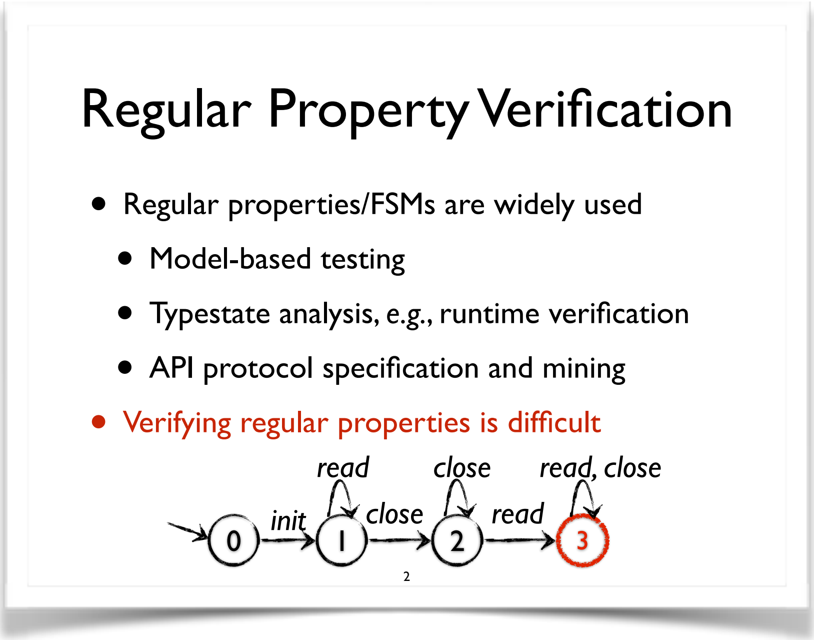
Much of the boosting happens during the early stage, indicating the necessity of a selective slicing

Found Bugs

Type	**#**
Array Index out of bound	8
Negative array size	3
Nullpointer	3
Division by zero	1
Dead loop	1
Runtime exception	1
Typestate error	2
In total	**19**

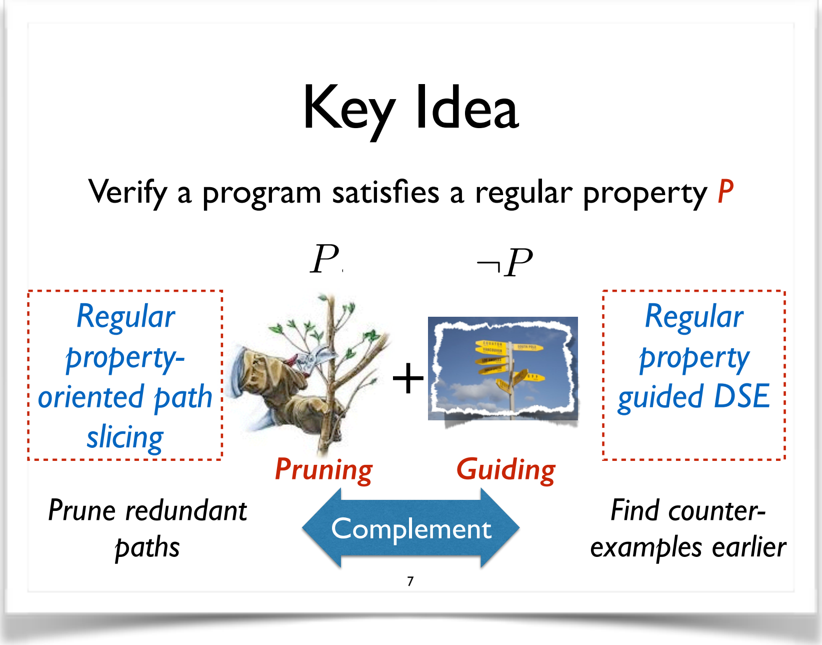
Bug Demonstration

Conclusion



Regular Property Verification

- Regular properties/FSMs are widely used
  - Model-based testing
  - Typestate analysis, e.g., runtime verification
  - API protocol specification and mining
- Verifying regular properties is difficult



Key Idea

Verify a program satisfies a regular property *P*

P $\neg P$

Regular property-oriented path slicing

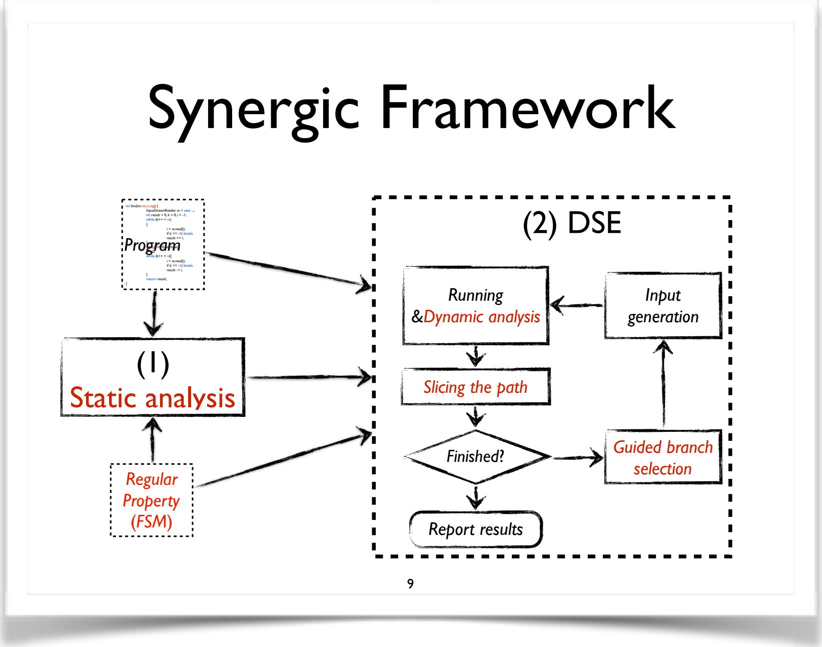
Pruning + *Guiding*

Regular property guided DSE

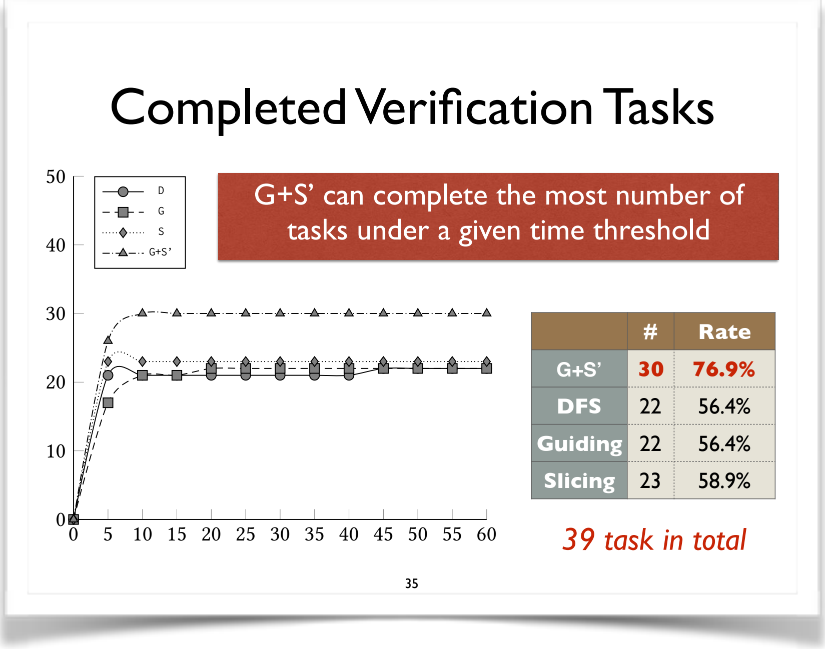
Prune redundant paths

Complement

Find counter-examples earlier



Synergic Framework



Completed Verification Tasks

G+S' can complete the most number of tasks under a given time threshold

	#	Rate
G+S'	30	76.9%
DFS	22	56.4%
Guiding	22	56.4%
Slicing	23	58.9%

39 task in total

Next Step

- Application in analyzing Linux drivers
  - In progress
- Reducing slicing overhead
- Improving usability and feasibility
- More applications, e.g., Android apps

Thank you
Any Questions?