Symbolic Verification of Regular Properties

Zhenbang Chen
(zbchen@nudt.edu.cn)

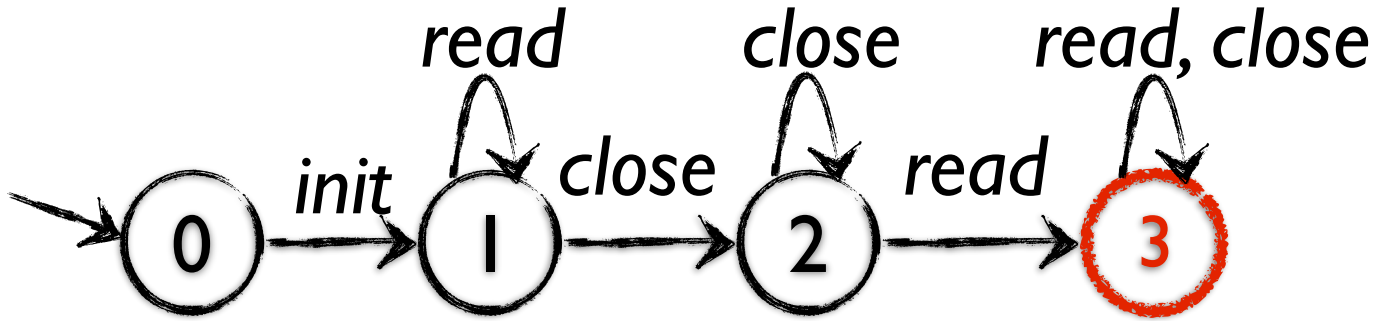
joint work with Hengbiao Yu, Ji Wang, Zhendong Su and Wei Dong

College of Computer, National University of Defense Technology, China
Department of Computer Science, University of California, Davis, USA

06/01/2018

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 challenging

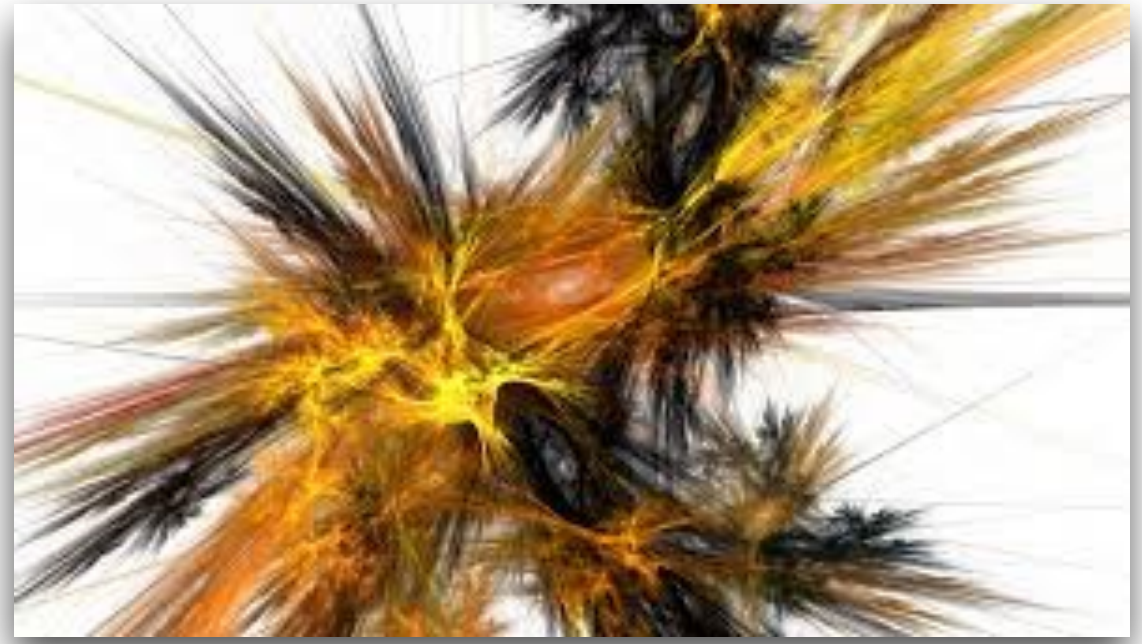


Existing Work

Dynamic Symbolic Execution (PLDI'05)

Challenge of Symbolic Execution

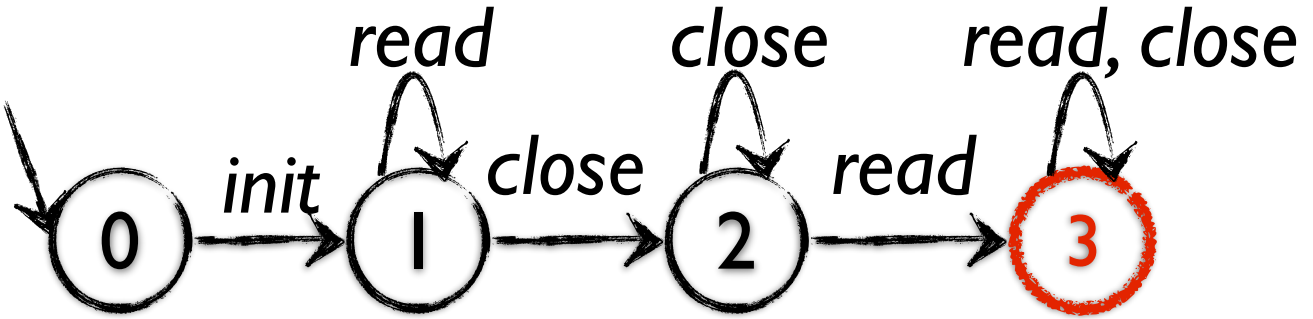
- Path explosion problem



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

Pruning

+

Guiding

Regular property guided DSE [ICSE'15]

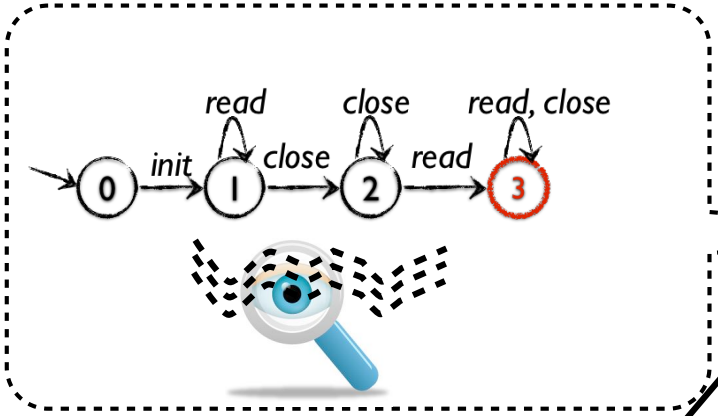
Prune redundant paths

Complement

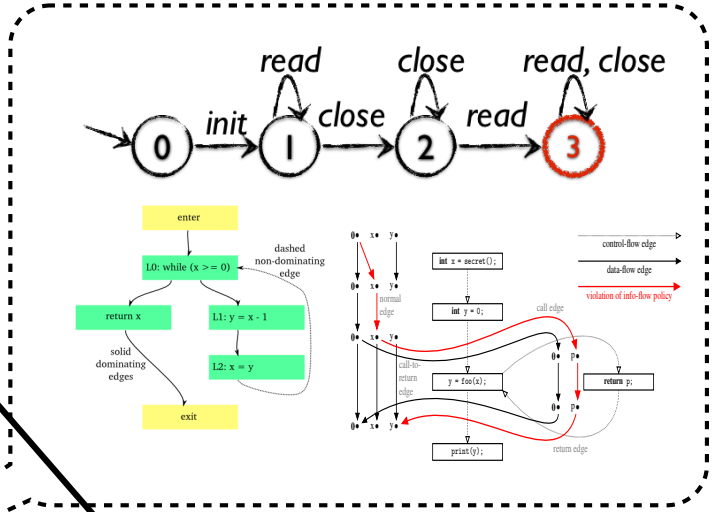
Find counter-examples earlier

Key Idea-Guiding w.r.t $\neg P$

history *history* ∩ *future* ≠ ∅ **future**

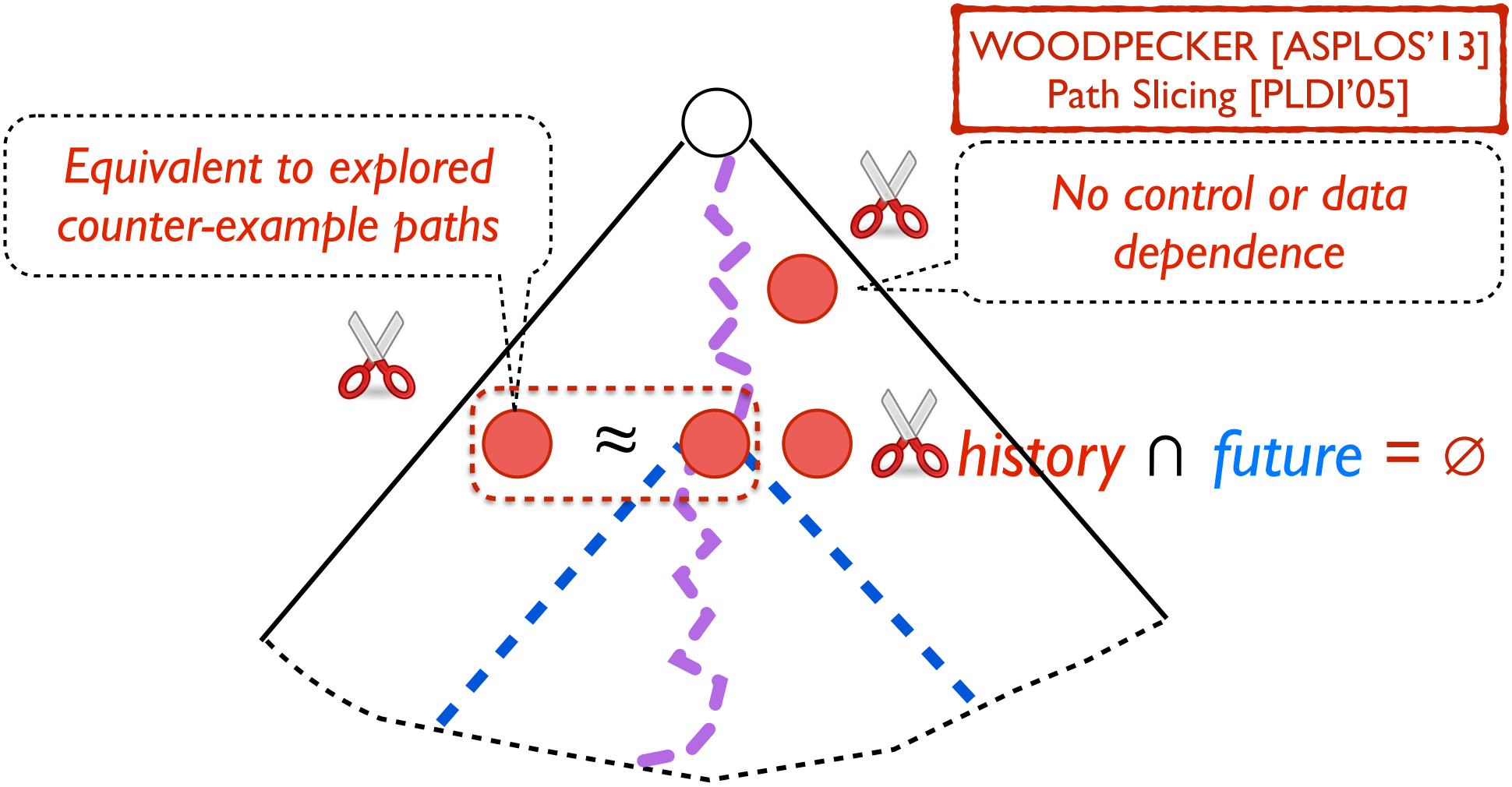


Dynamic analysis

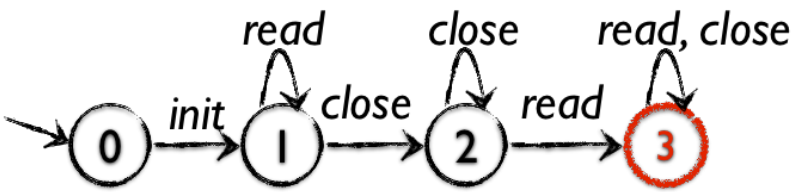


Static analysis

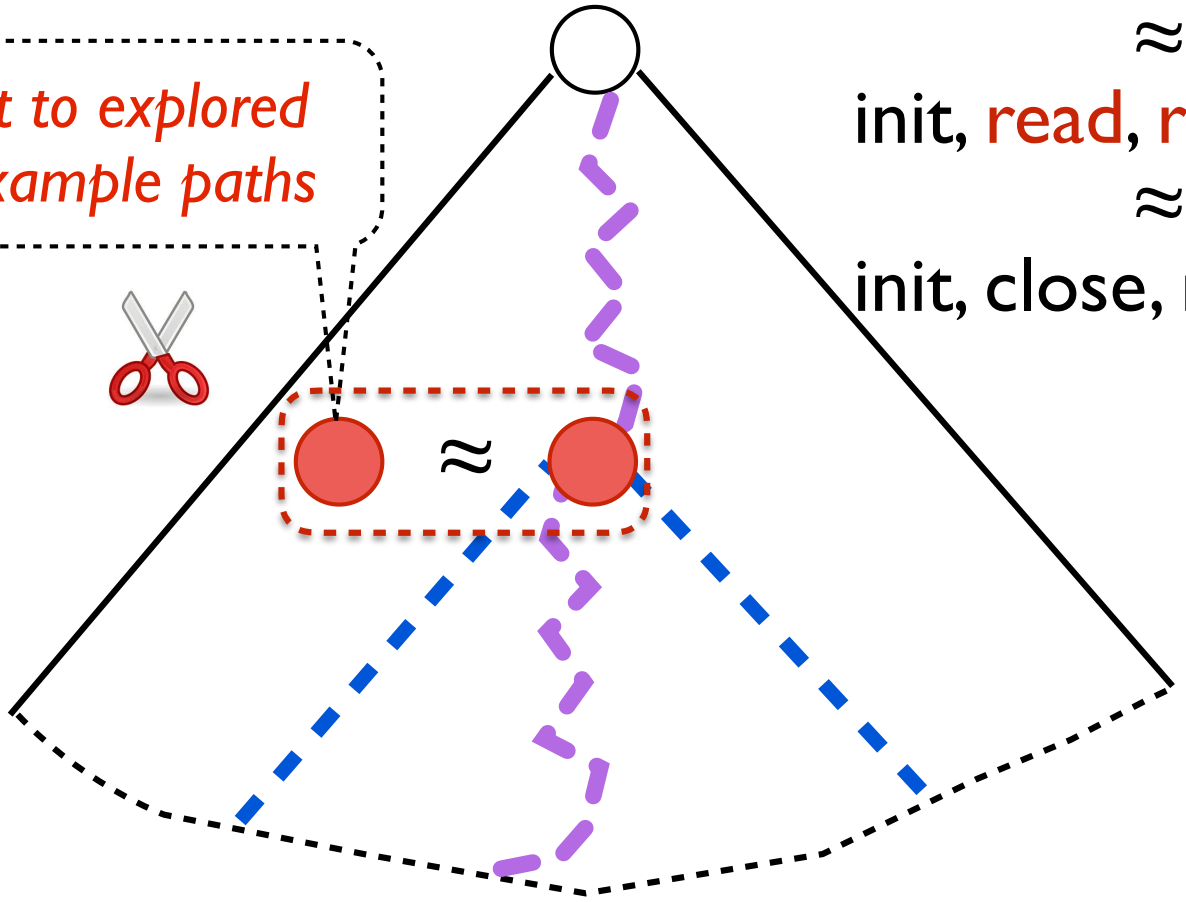
Key Idea-Pruning



Key Idea-Pruning



Equivalent to explored counter-example paths



init, close, read

≈

init, read, close, read

≈

init, read, read, close, read

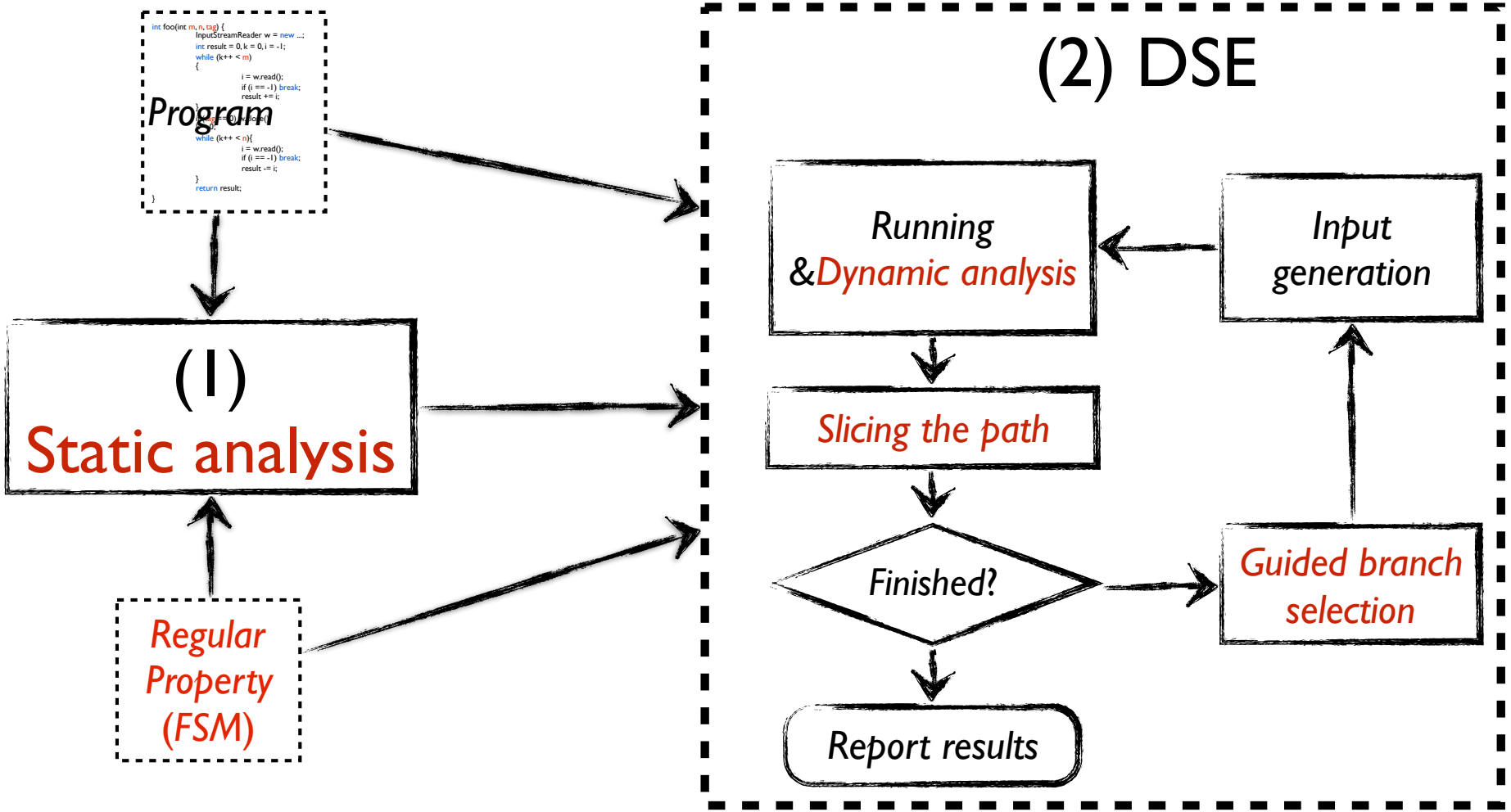
≈

init, close, read, read

Sneak Preview of Results

- For 39 verification tasks (1 *hour* for each)
  - 30 are completed by our method
  - DFS (22), pure guiding (22) and slicing (23)
- For the completed verification tasks
  - >8.4X, >8.6X, and >7X time speedups over DFS, pure guiding and path slicing

Synergic Framework



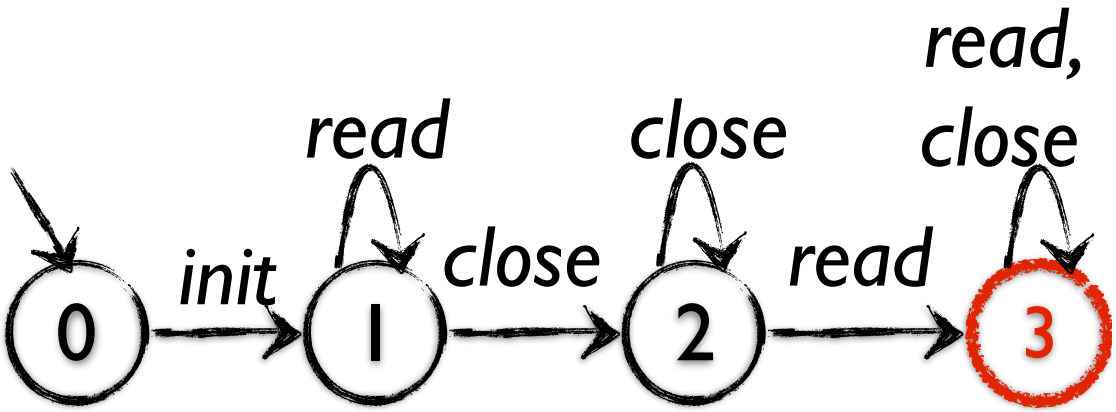
An Example

```
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

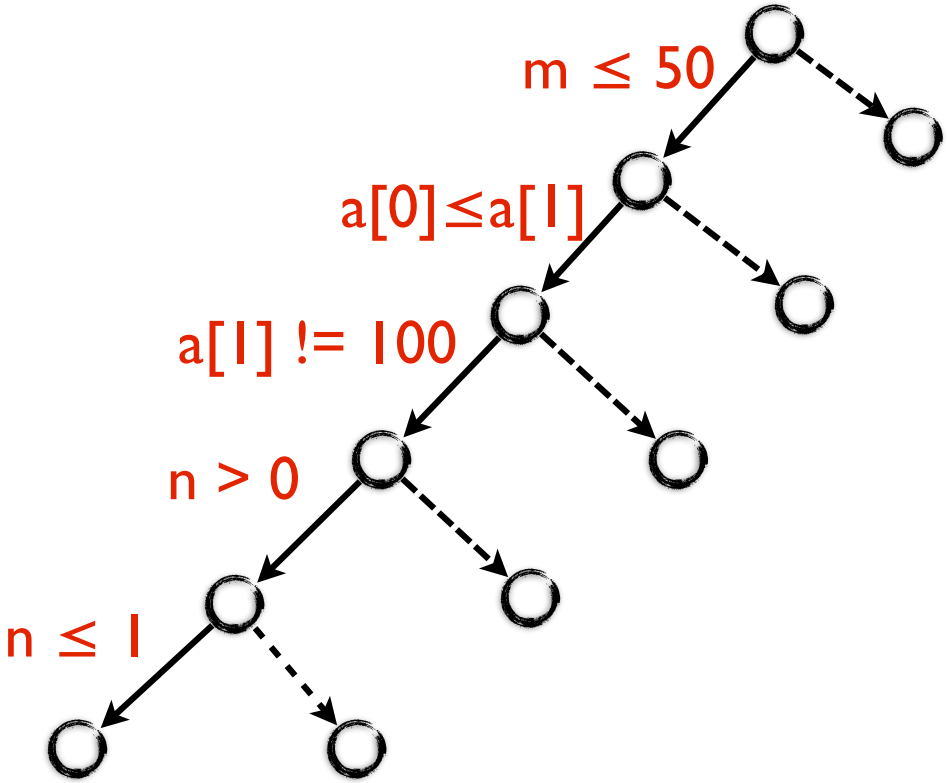


Read after closed

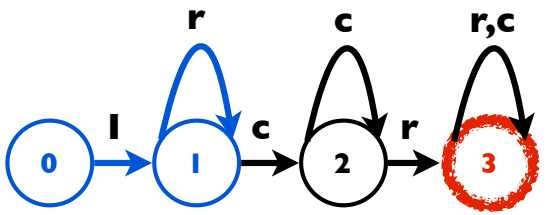
```
int foo(int m, int n, int[] a) {
  InputStreamReader w = new …;
  if (m > 50) m++;
  for (int i = 0; i < a.length-1; i++) {
      if (a[i] > a[i+1]) {
        int temp = a[i];
        a[i+1] = a[i];
        a[i] = temp;
      }
  }
  if (a[i] == 100)
      w.close();
  while (n-- > 0){
      int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

1st Iteration

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



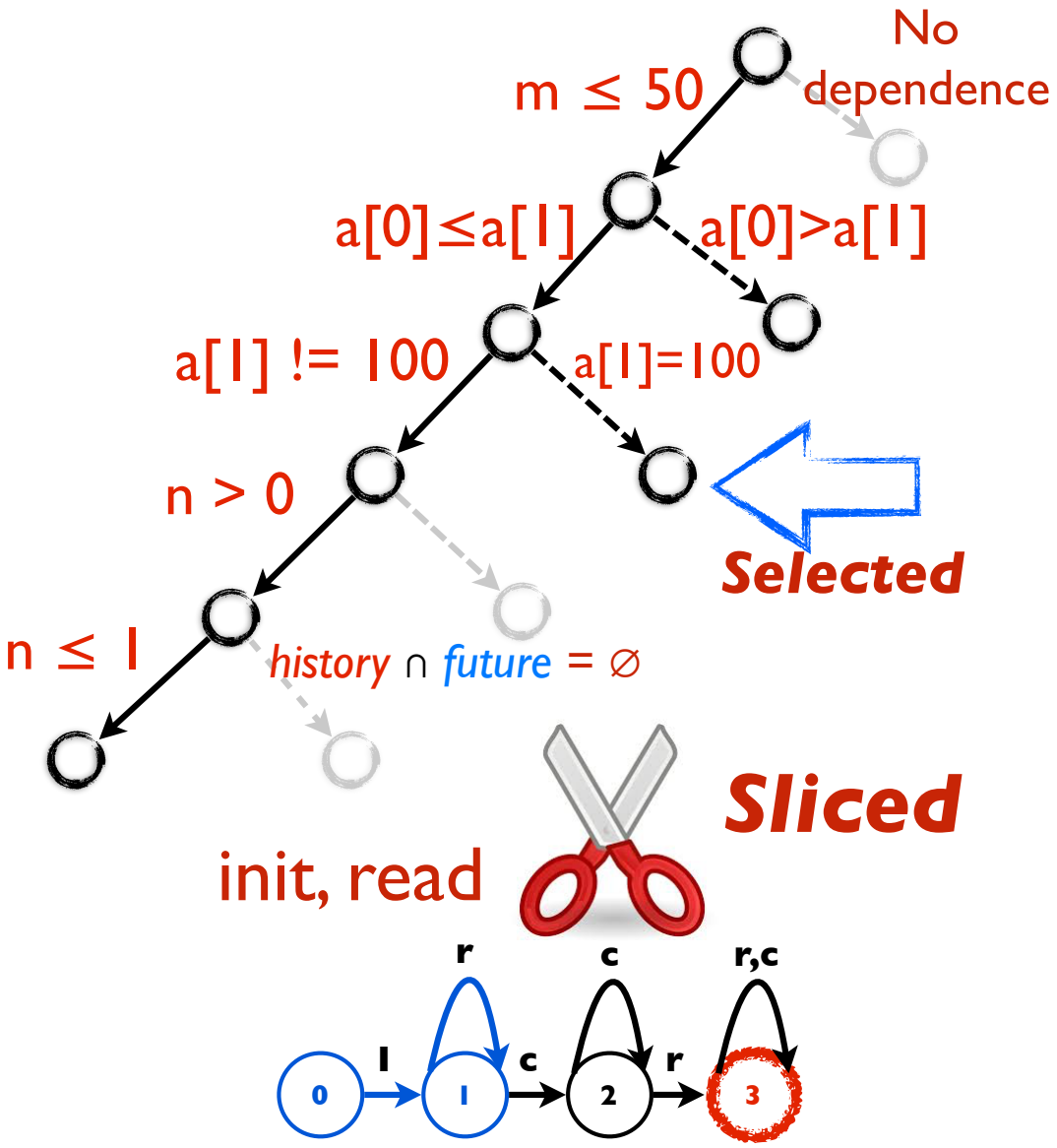
init, read



```
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;
}
```

1st Iteration

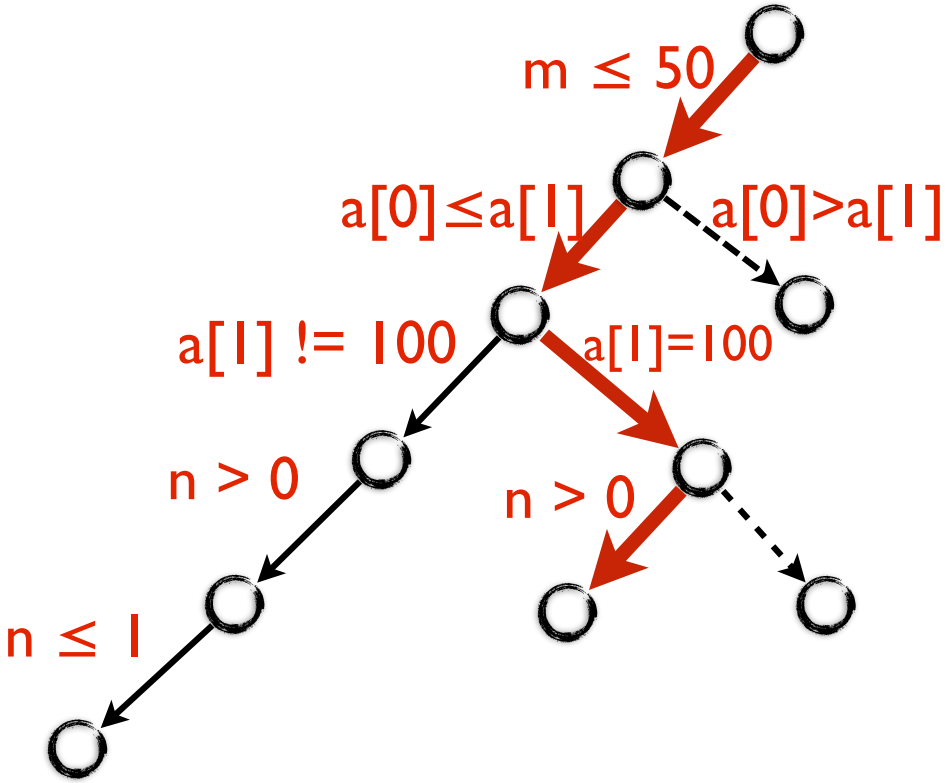
(m=1, n=1, a={0, 1})



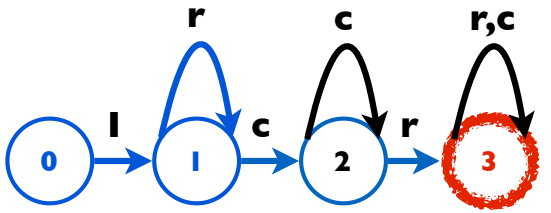
```
int foo(int m, int n, int[] a) {
  InputStreamReader w = new …;
  if (m > 50) m++;
  for (int i = 0; i < a.length-1; i++) {
      if (a[i] > a[i+1]) {
        int temp = a[i];
        a[i+1] = a[i];
        a[i] = temp;
      }
  }
  if (a[i] == 100)
      w.close();
  while (n-- > 0){
      int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

2nd Iteration

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



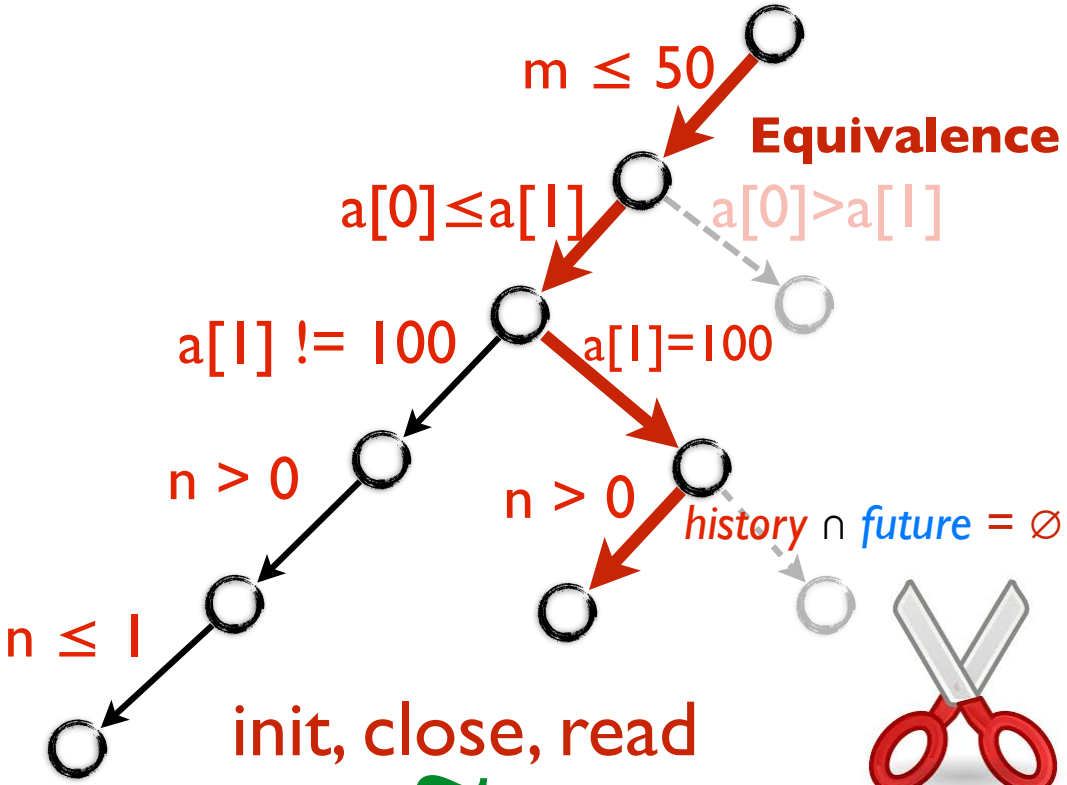
init, close, read



```
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;
}
```

2nd Iteration

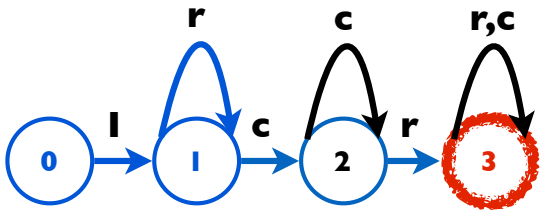
(m=1, n=1, a={0, 100})



init, close, read

≈

init, close, read+



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 verification

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
- 16 real world open source Java programs
  - 270K LOC in total

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

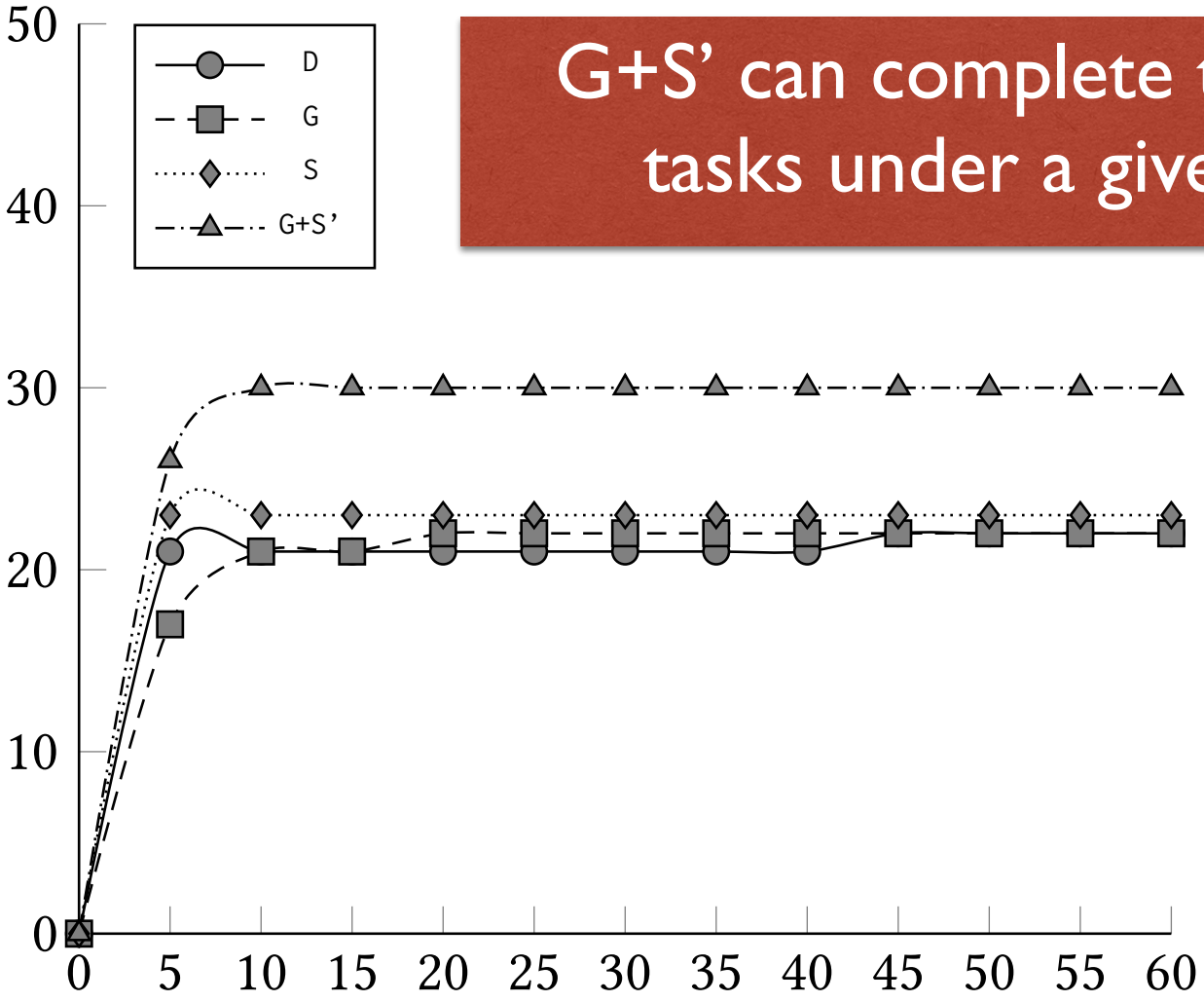
- Implement for Java based on RGSE
- 16 real world open source Java programs
  - 270K LOC in total
- Properties
  - JDK's single- and multi-objects typestate properties
  - User defined
- Verify each program/property in 1 hour

Results

Program (Property)	Type	Total Time(s)				First Violation Time(s)			
		D	G	S	S'	D	G	S	S'
soot-c (Writer)	0	28.43	374.23	95.14	398.98	NO	NO	NO	NO
	bug1	28.59	354.26	101.32	413.83	18.41	343.72	85.01	413.71
	bug2	27.61	358.01	97.06	389.1	19.39	346.6	81.62	389.09
	bug3	26.71	369.87	104.37	469.51	15.12	358.99	83.77	469.39
soot-c (Writer*)	0	27.2	177.91	91.86	214.85	NO	NO	NO	NO
	bug4	29.82	187.74	97.91	219.26	NO	NO	NO	NO
	bug5	27.9	187.41	98.15	218.9	15.71	176.27	79.4	216.47
	bug6	29.2	174.32	103	206.73	NO	NO	NO	NO
bloat (Iterator)	0	24.49	48.1	57.56	66.97	10.13	36.02	35.2	50.14
bloat (Iterator*)	0	27.1	71.75	54.23	90.54	NO	NO	NO	NO
	bug1	29.2	71.81	102.07	128.85	NO	NO	NO	NO
	bug2	25.85	70.02	60.74	92.19	25.85	70.02	42.86	92.19
	bug3	26.63	72.05	64.57	96.88	26.63	70.95	40.11	78.92
bmpdecoder (Reader)	0	8.65	16.97	21.71	16.71	NO	NO	NO	NO
	bug1	9.18	17.45	19.79	22.04	NO	NO	NO	NO
	bug2	9.15	17.92	21.01	18	7.93	12.54	20.48	17.96
	bug3	9.26	17.88	20.97	23.26	NO	NO	NO	NO
ftpclient (Socket)	0	12.44	37.08	37.83	49.78	NO	NO	NO	NO
	bug1	14.12	41.48	42.12	55.1	9.19	36.44	38.89	54.55
	bug2	13.57	37.66	37.47	50.55	11.73	33.96	37.2	50.55
	bug3	15.52	40.45	40.29	53.39	NO	NO	NO	NO
jlex (Reader)	0	TO	TO	TO	29.48	NA	NA	NA	NO
	bug1	TO	TO	TO	TO	12.75	23.35	400.71	63.97
	bug2	TO	TO	TO	TO	NA	14.58	NA	27.04
	bug3	TO	TO	TO	TO	NA	NA	NA	NA
jlex (Reader*)	0	TO	TO	TO	29.81	NA	NA	NA	NO
	bug4	TO	TO	TO	TO	NA	20.07	NA	52.24
	bug5	TO	TO	TO	TO	217.56	38.18	NA	109.39
	bug6	TO	TO	TO	TO	51.33	146.88	NA	NA
rhino-a (Enumeration)	0	TO	TO	TO	TO	NA	NA	NA	NA
jpat (UserDefined)	0	TO	TO	TO	46.94	NA	23.36	NA	43.99
nano-xml (UserDefined)	0	TO	TO	TO	19.18	NA	14.02	NA	19.16
pobs (UserDefined)	0	TO	TO	21.44	26.31	NA	14.96	20.79	23.07
jericho (UserDefined)	0	TO	TO	53.7	27.66	NA	19.6	53.33	27.66
fastjason (UserDefined)	0	TO	TO	TO	102.6	NA	NA	NA	102.52
jep (UserDefined)	0	2590.38	1090.05	TO	167.87	1439.06	29.72	NA	167.84
htmlparser (UserDefined)	0	TO	TO	TO	TO	27.62	50.95	NA	106.03
udl (UserDefined)	0	TO	TO	TO	TO	NA	2829.57	NA	NA
xmlparser (UserDefined)	0	TO	TO	TO	24.89	NA	18.25	NA	24.89

- Use the fewest iterations to complete path exploration
- For 24 tasks with counterexamples, slicing can boost the finding in 7 (29%) tasks
- #iterations using our slicing is two orders of magnitude less than that using path slicing
- Completed tasks by us but not by path slicing
  - Cannot finish after 24 hours except jep

Completed Verification Tasks



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

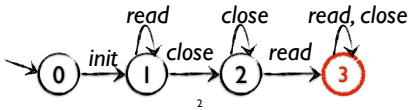
	#	Rate
G+S'	**30**	**76.9%**
DFS	22	56.4%
Guiding	22	56.4%
Slicing	23	58.9%

39 tasks in total

Conclusion

Regular Property Verification

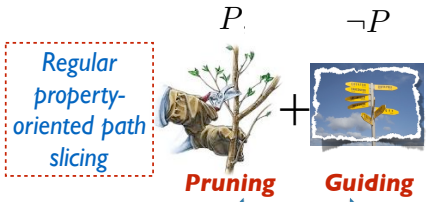
- Regular properties/FSMs are widely used
  - Model-based testing
  - Typestate analysis, e.g., runtime verification
  - API protocol specification and mining
- Verifying regular properties is difficult



2

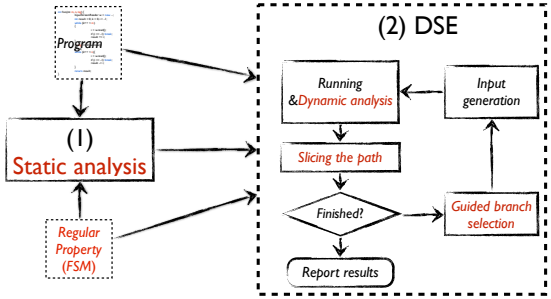
Key Idea

Verify a program satisfies a regular property *P*



7

Synergic Framework



9

Sneak Preview of Results

- For 39 verification tasks (1 *hour* for each)
  - 30 are completed by our method
  - DFS (22), pure guiding (22) and slicing (23)
- For the completed verification tasks
  - >8.4X, >8.6X, and >7X time speedups over DFS, pure guiding and path slicing

11

Thank you
Any Questions?