I get by with a little help from my friends: crowdsourcing program repair

Kathryn (Katie) Stolee

Assistant Professor

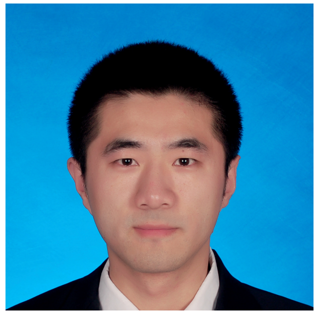
North Carolina State University

I get by with a little help from my ~~friends~~ students: crowdsourcing program repair

Kathryn (Katie) Stolee

Assistant Professor

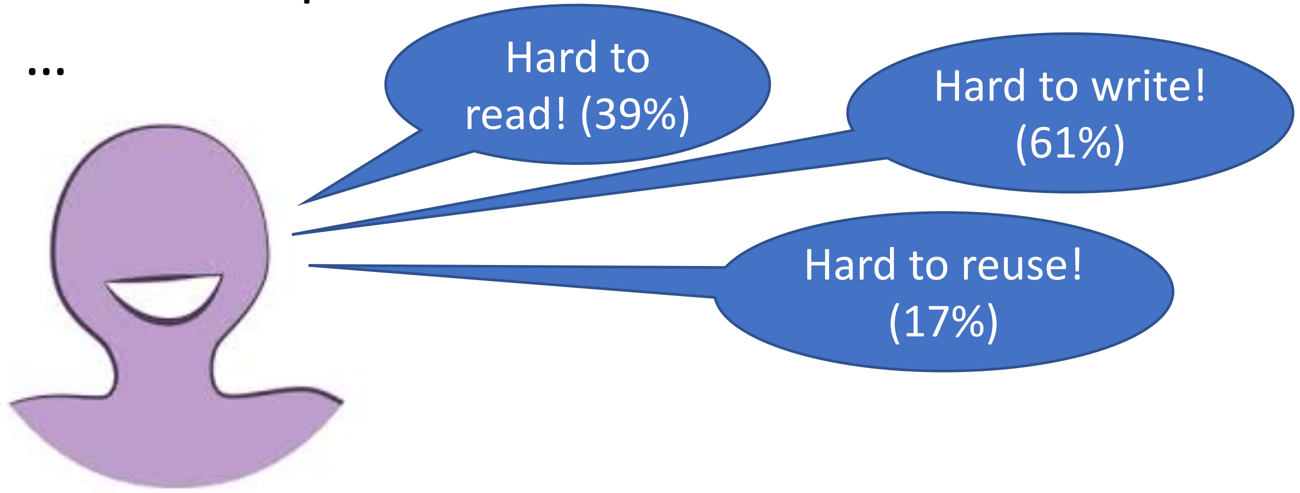
North Carolina State University

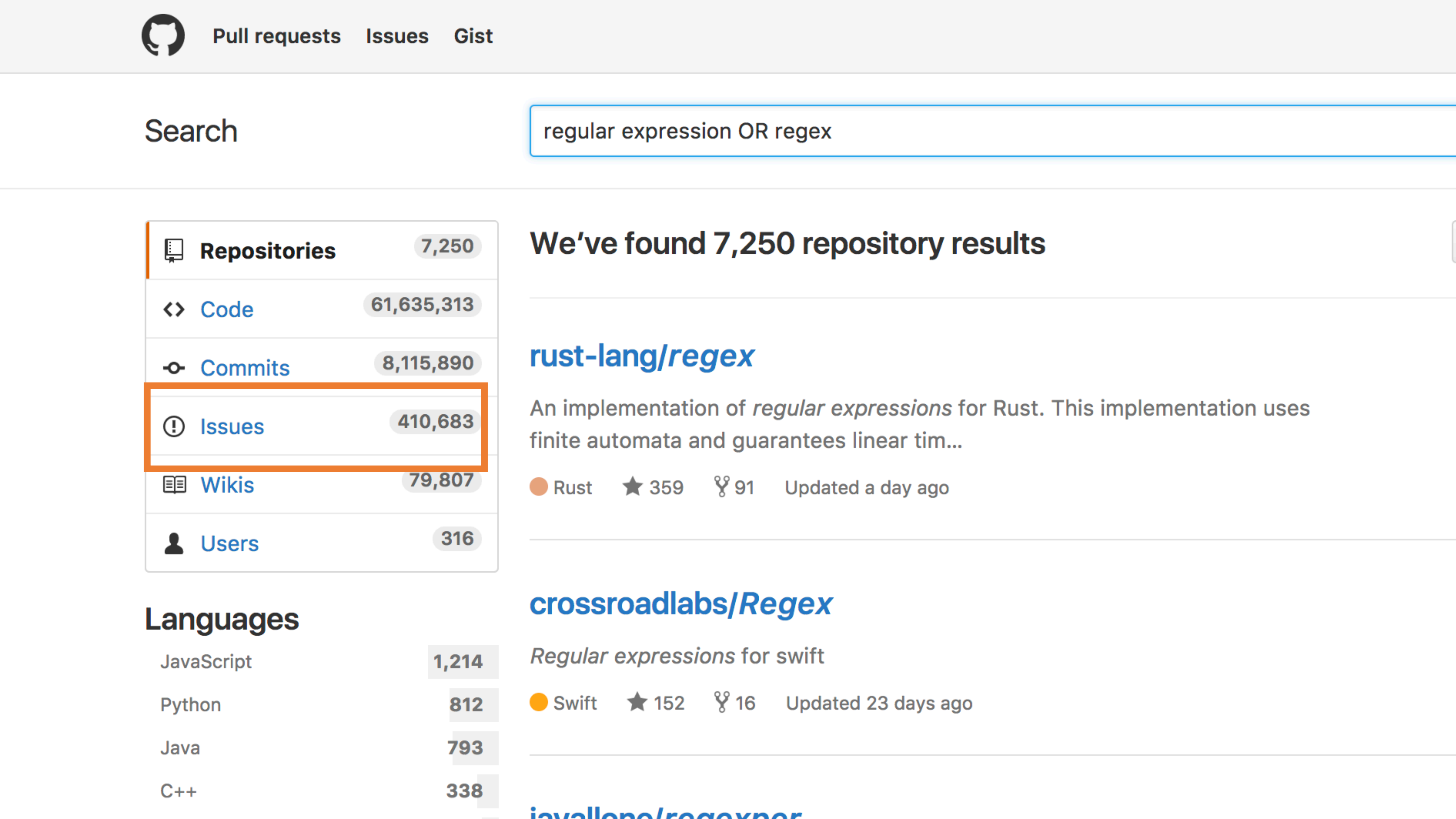


We’re going to talk about regular expressions.

Why regular expressions?

- Frequently used in code
- Largely unstudied in software engineering
- Variety of usage contexts
  - Data sanitization in JavaScript
  - Log analysis
  - Database queries
  - …
-





Pull requests Issues Gist

Search

regular expression OR regex

Repositories	7,250
Code	61,635,313
Commits	8,115,890
Issues	410,683
Wikis	79,807
Users	316

Languages

JavaScript	1,214
Python	812
Java	793
C++	338

We've found 7,250 repository results

rust-lang/*regex*

An implementation of *regular expressions* for Rust. This implementation uses finite automata and guarantees linear tim...

● Rust ★ 359 91 Updated a day ago

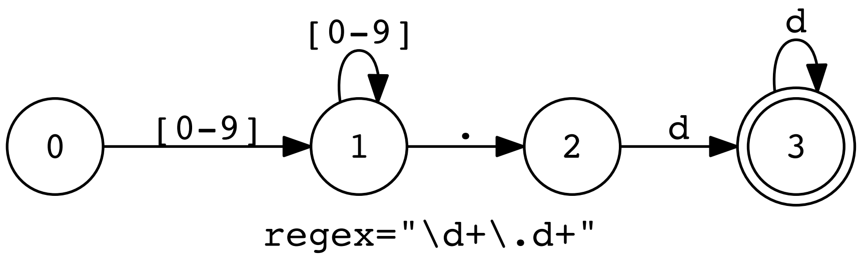
crossroadlabs/*Regex*

Regular expressions for swift

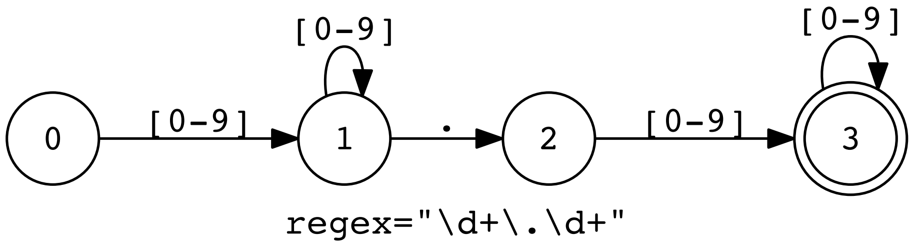
● Swift ★ 152 16 Updated 23 days ago

Example – Maven-NAR bug

Buggy



Fixed



https://github.com/maven-nar/nar-maven-plugin/issues/228

Let’s automate this.

Things we don’t know (and need to)

- What kinds of mistakes do devs make with regexes?
- How well are regexes tested?

Then: we can think about how to repair regexes!

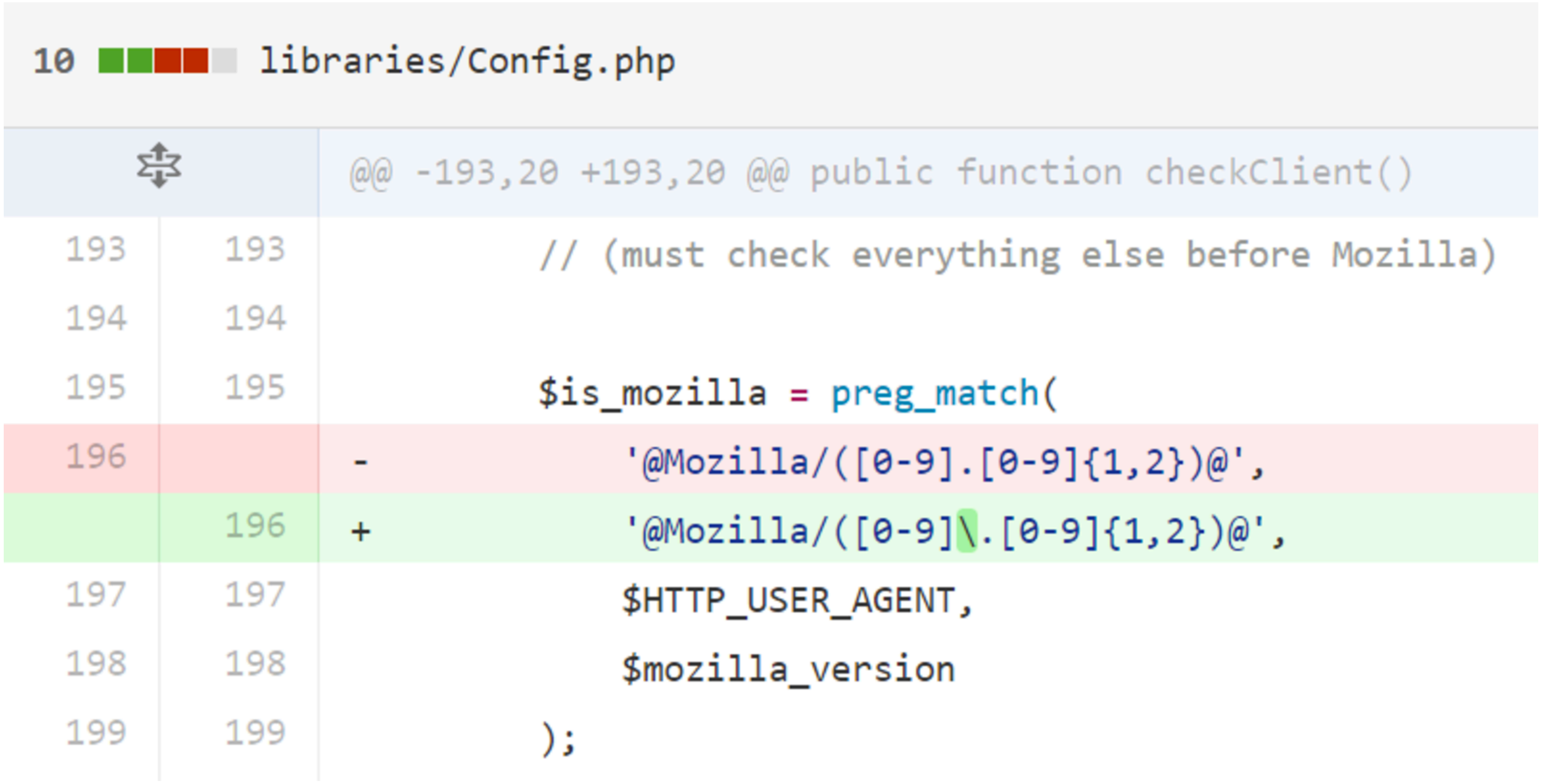
What kinds of mistakes do devs make with regexes?

- Explore GitHub issues
- Explore StackOverflow
- Ask developers
- Observe developers

GitHub lens: Types of errors in regexes

- 19 issues in GitHub
- Sampled from queries:
  - “regex in:title state:closed label:bug”
  - “regular expression in:title state:close label:bug”

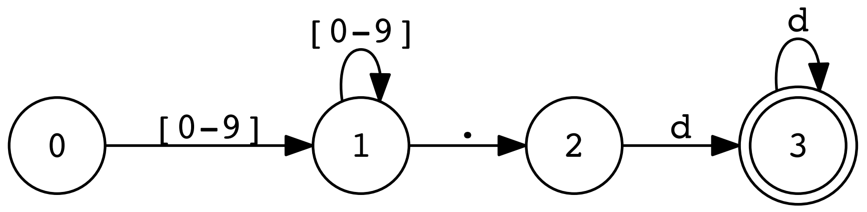
Missing token



```
10 🟩🟩🟥🟥⬜ libraries/Config.php

@@ -193,20 +193,20 @@ public function checkClient()
193 193          // (must check everything else before Mozilla)
194 194
195 195          $is_mozilla = preg_match(
196      -            '@Mozilla/([0-9].[0-9]{1,2})@',
    196  +            '@Mozilla/([0-9]\.[0-9]{1,2})@',
197 197              $HTTP_USER_AGENT,
198 198              $mozilla_version
199 199          );
```

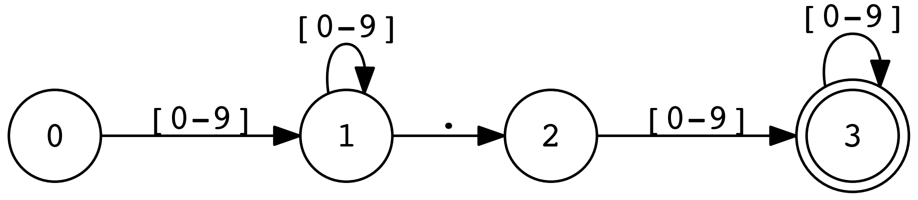
Example – Maven-NAR bug

Buggy



`regex="\d+\.d+"`

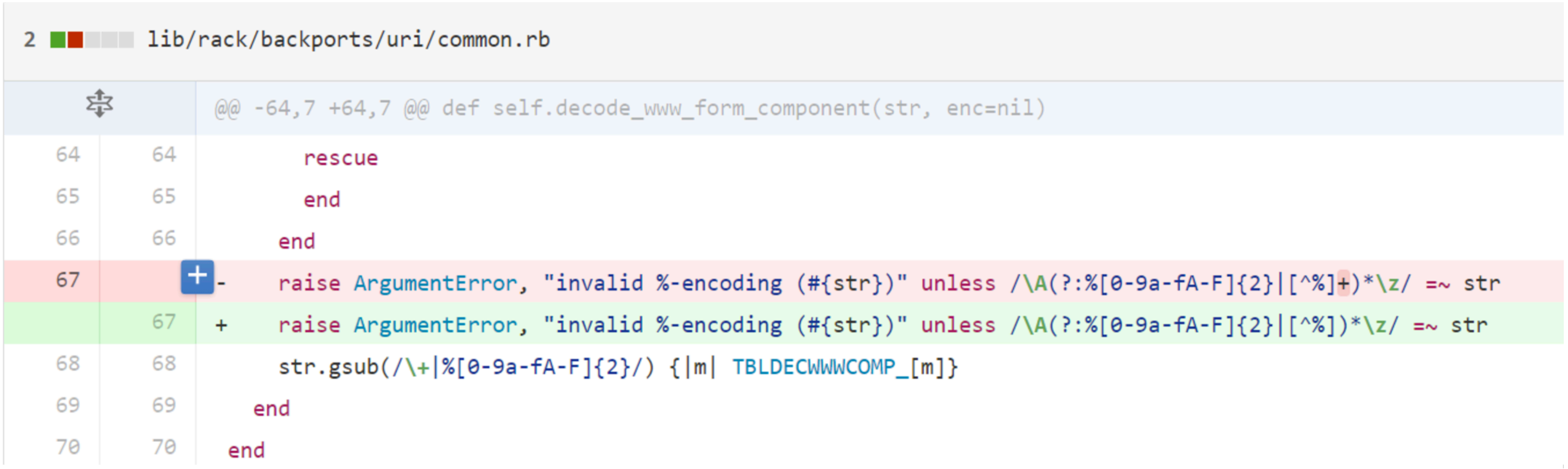
Fixed



`regex="\d+\.\d+"`

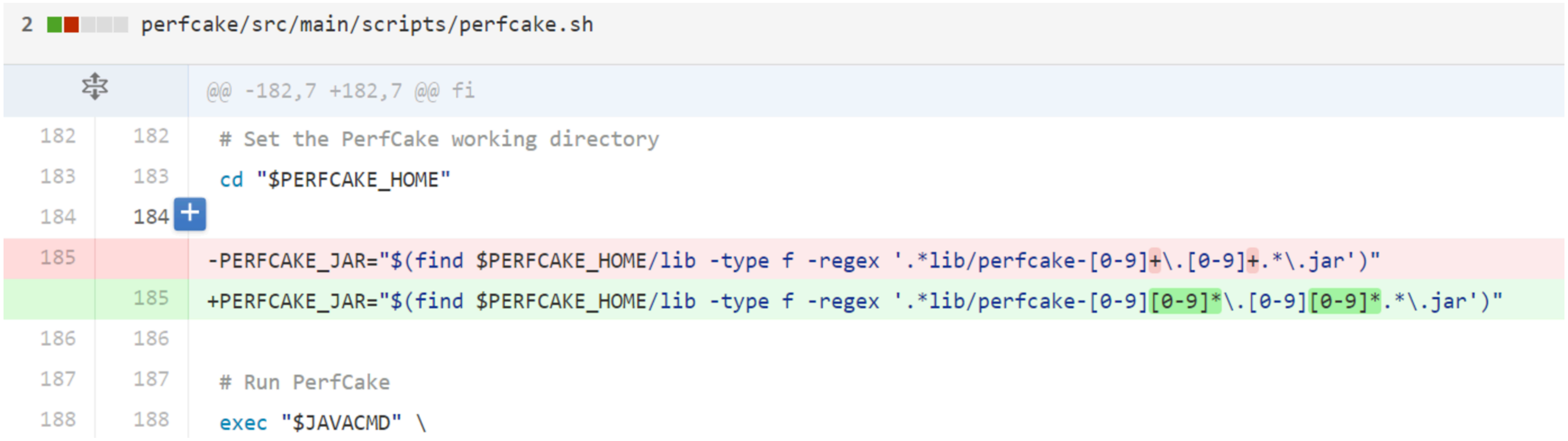
https://github.com/maven-nar/nar-maven-plugin/issues/228

Extra/Spurious token



```
2 lib/rack/backports/uri/common.rb
@@ -64,7 +64,7 @@ def self.decode_www_form_component(str, enc=nil)
        rescue
        end
      end
-     raise ArgumentError, "invalid %-encoding (#{str})" unless /\A(?:%[0-9a-fA-F]{2}|[^%]+)*\z/ =~ str
+     raise ArgumentError, "invalid %-encoding (#{str})" unless /\A(?:%[0-9a-fA-F]{2}|[^%])*\z/ =~ str
      str.gsub(/\+|%[0-9a-fA-F]{2}/) {|m| TBLDECWWWCOMP_[m]}
    end
  end
```

Wrong design decision – not a bug!

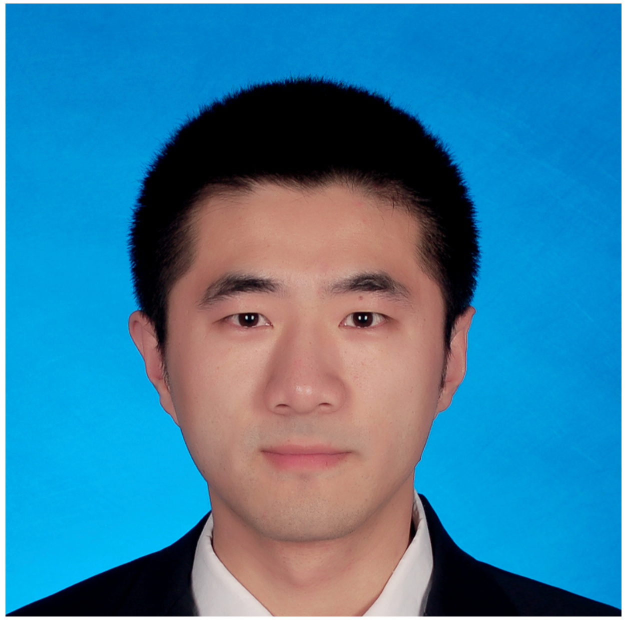


```
2 perfcake/src/main/scripts/perfcake.sh

@@ -182,7 +182,7 @@ fi
 # Set the PerfCake working directory
 cd "$PERFCAKE_HOME"

-PERFCAKE_JAR="$(find $PERFCAKE_HOME/lib -type f -regex '.*lib/perfcake-[0-9]+\.[0-9]+.*\.jar')"
+PERFCAKE_JAR="$(find $PERFCAKE_HOME/lib -type f -regex '.*lib/perfcake-[0-9][0-9]*\.[0-9][0-9]*.*\.jar')"

 # Run PerfCake
 exec "$JAVACMD" \
```

Summary

- Other categories:
  - Missing design element (a token missing = syntax error? a larger design element = semantic error?)
  - Wrong character class (e.g., \W instead of \S)

StackOverflow lens: Types of errors in regexes



I'm working on it!

Asking Developers

- 18 professional developers, small company
- “What pain points do you encounter while working with regular expressions”
  - 62%: Hard to compose
  - 39%: Hard to read
  - 17%: Hard to reuse across languages

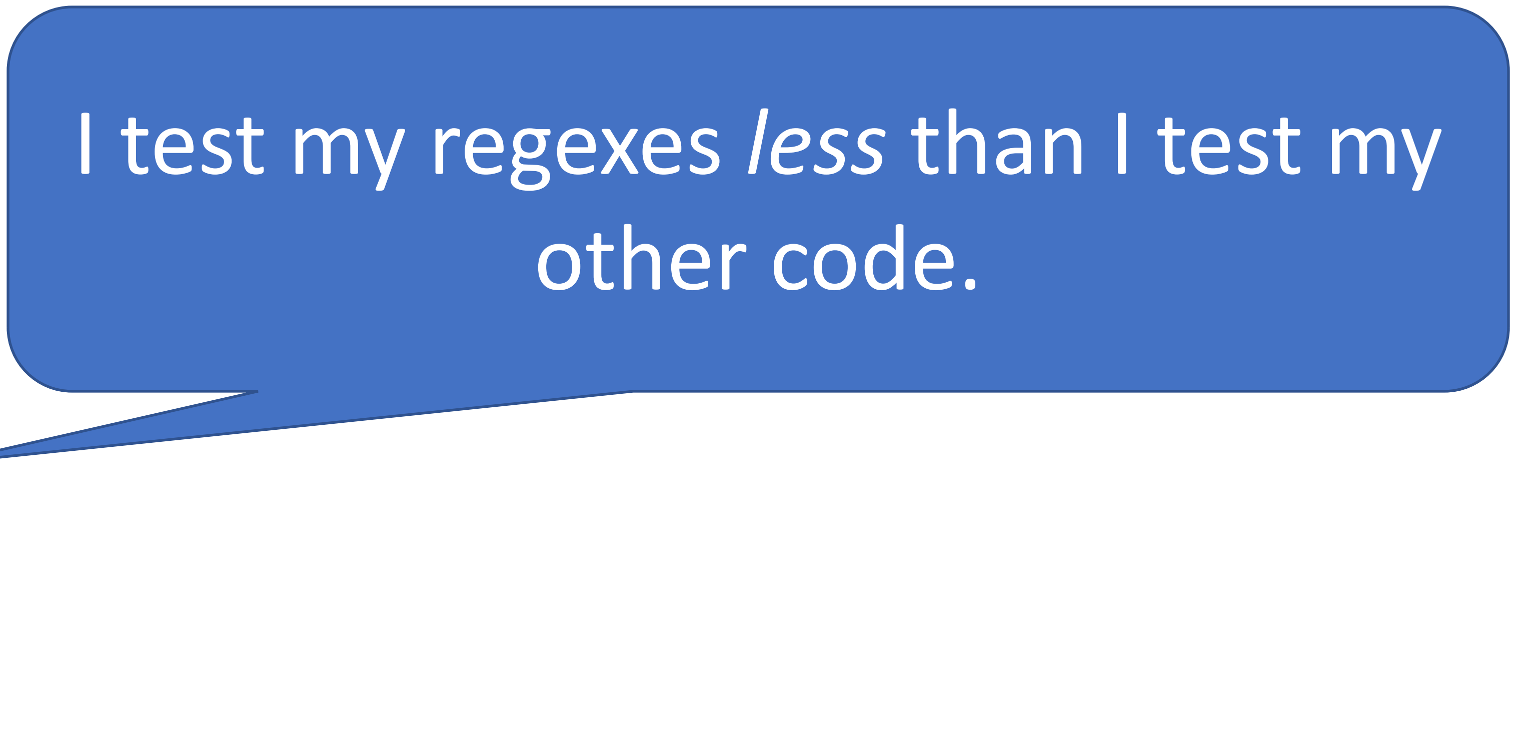
Not covered: how serious are regex-related errors?

Observing Developers

- 20 students in a lab, 1 hour time block
- Given: textual description of a regex and test cases
- Asked to write regexes that pass all the tests
- Screen capture

Looking for a student to analyze this data!

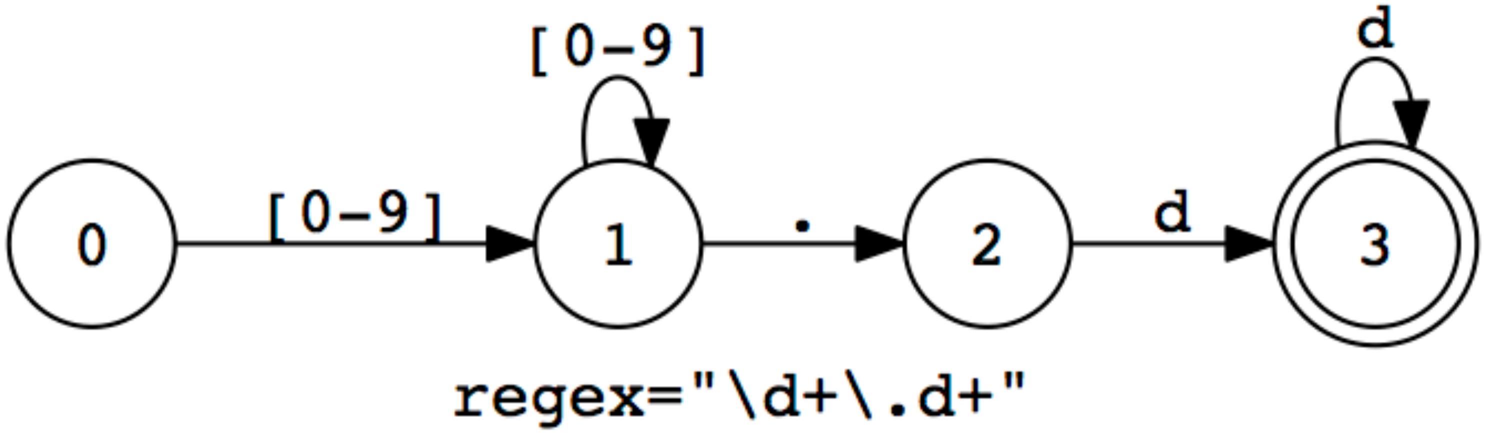
How well are regexes tested?



Average Developer

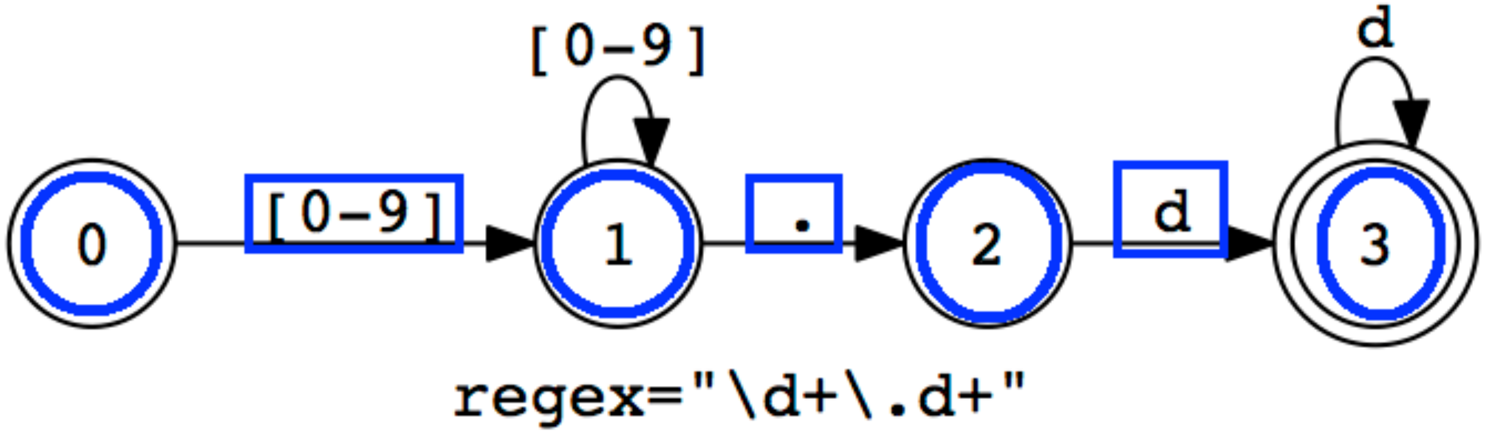
Can we validate this?

Input
0.d



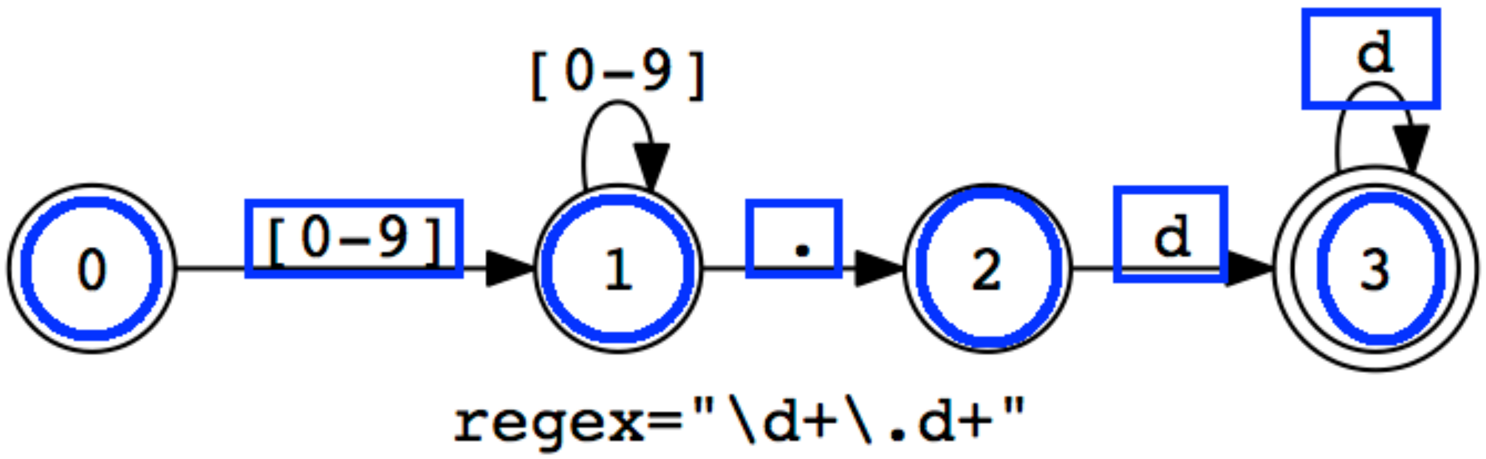
Idea

Input
0.d



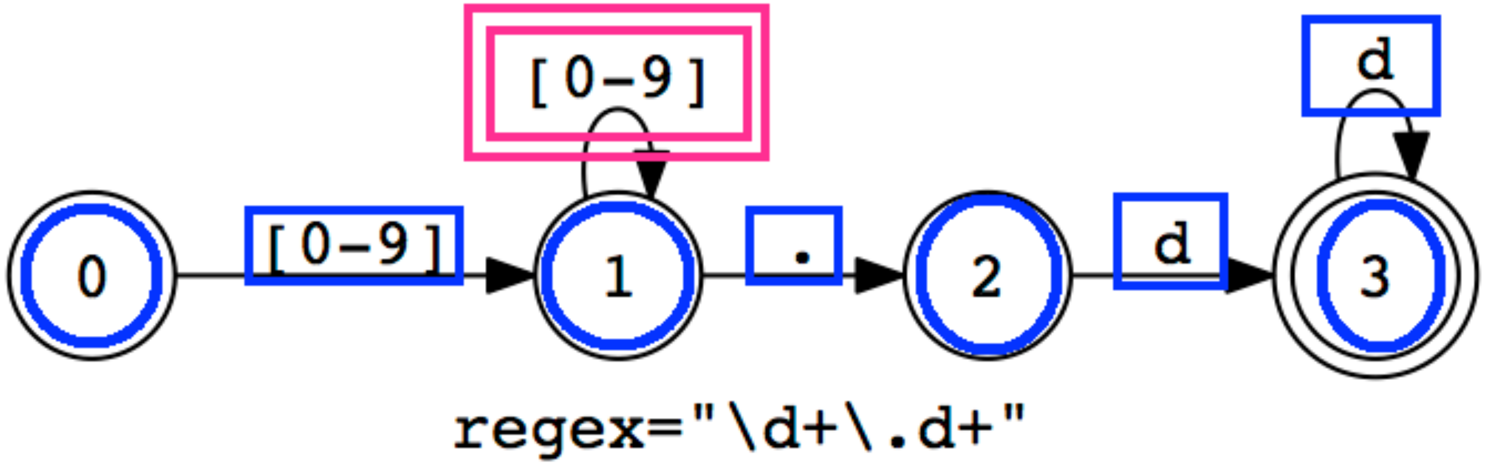
Idea

Input
0.d
0.dd

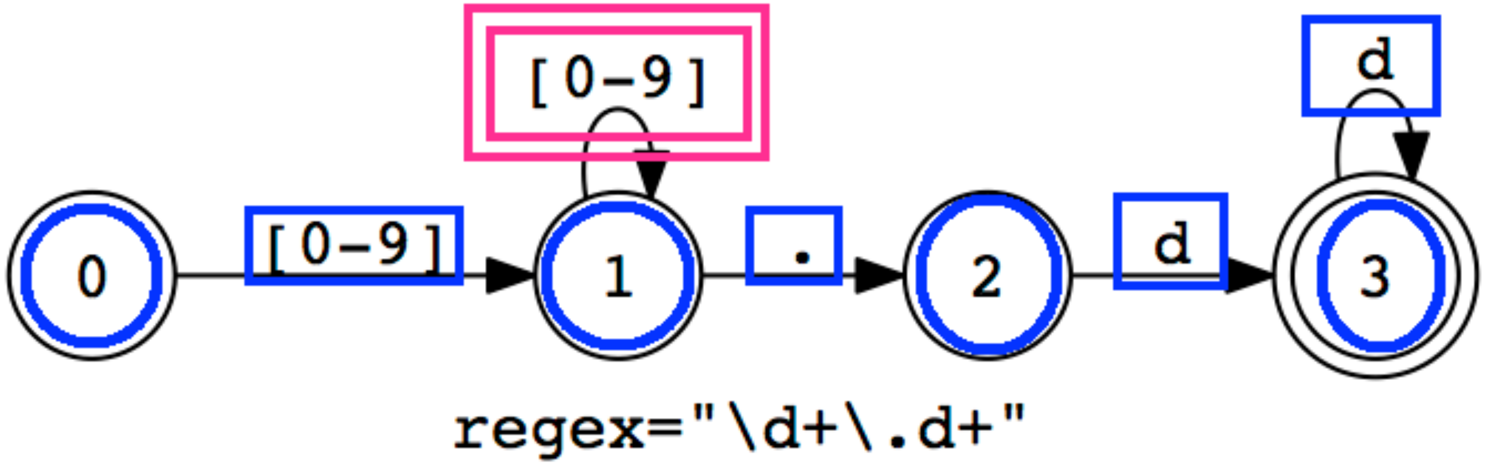


Idea

Input
0.d
0.dd



Idea



regex="\d+\.d+"

Input
0.d
0.dd
00.d

Can we validate this?

I'm working on it!

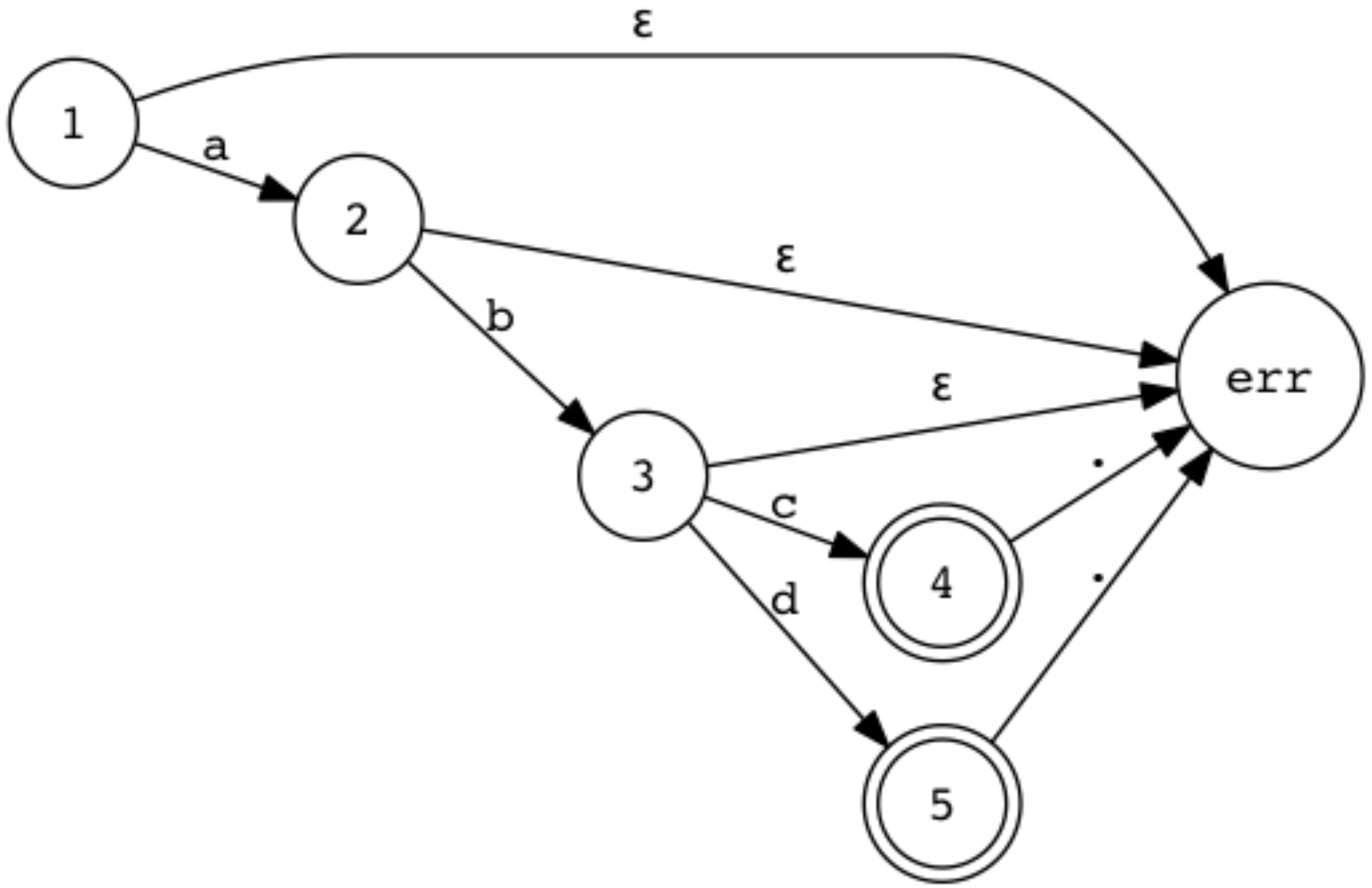
Next question:
Do regex faults lie along untested nodes/edges/edge-pairs/….?

Now for patching. Let’s assume:

- Regexes have bugs
- Regexes have passing and failing test cases

Next: Fault localization

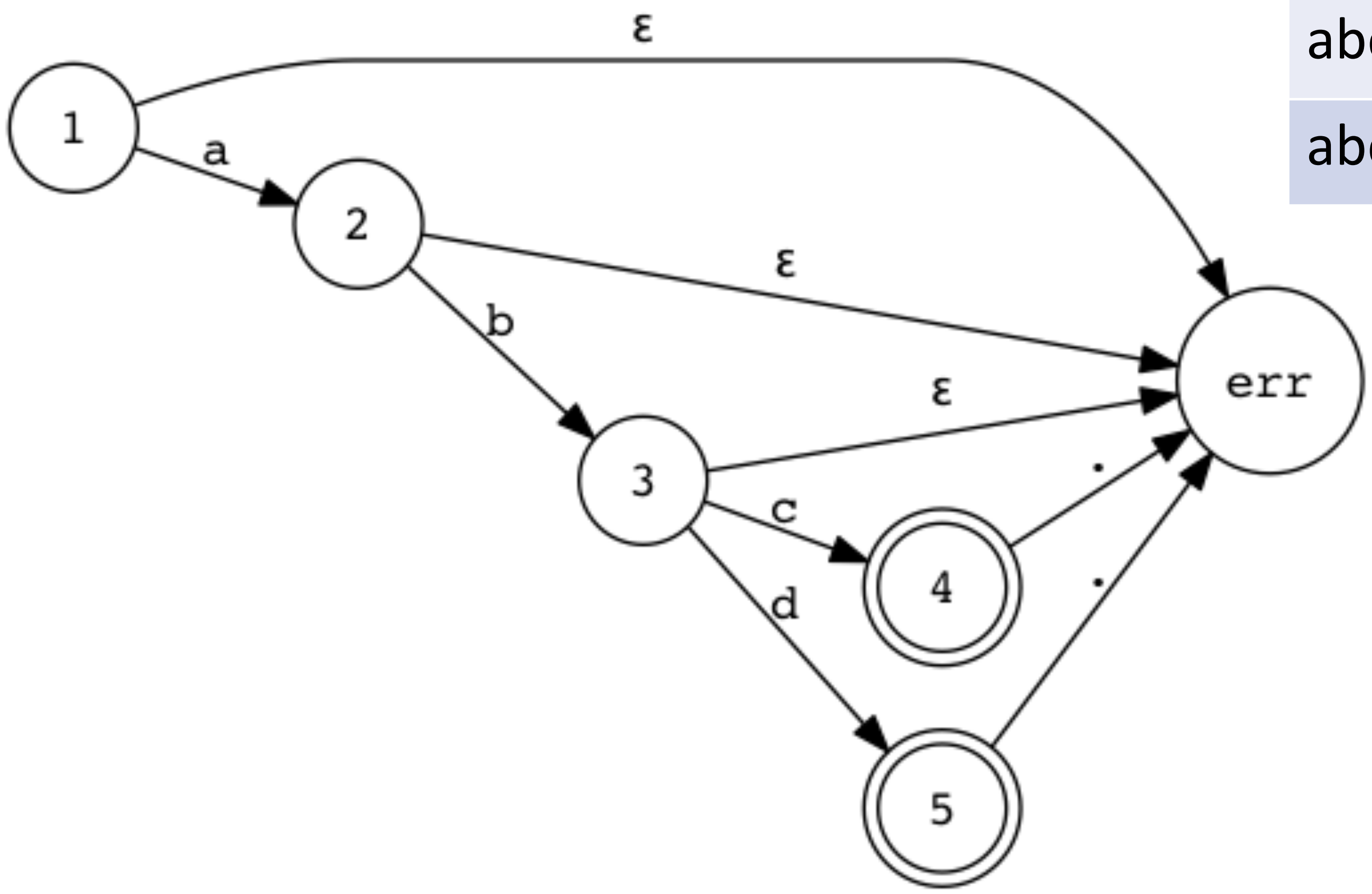
Spectrum-based + finite automata



`regex = (ab)(c|d)$`

Spectrum-based + finite automata

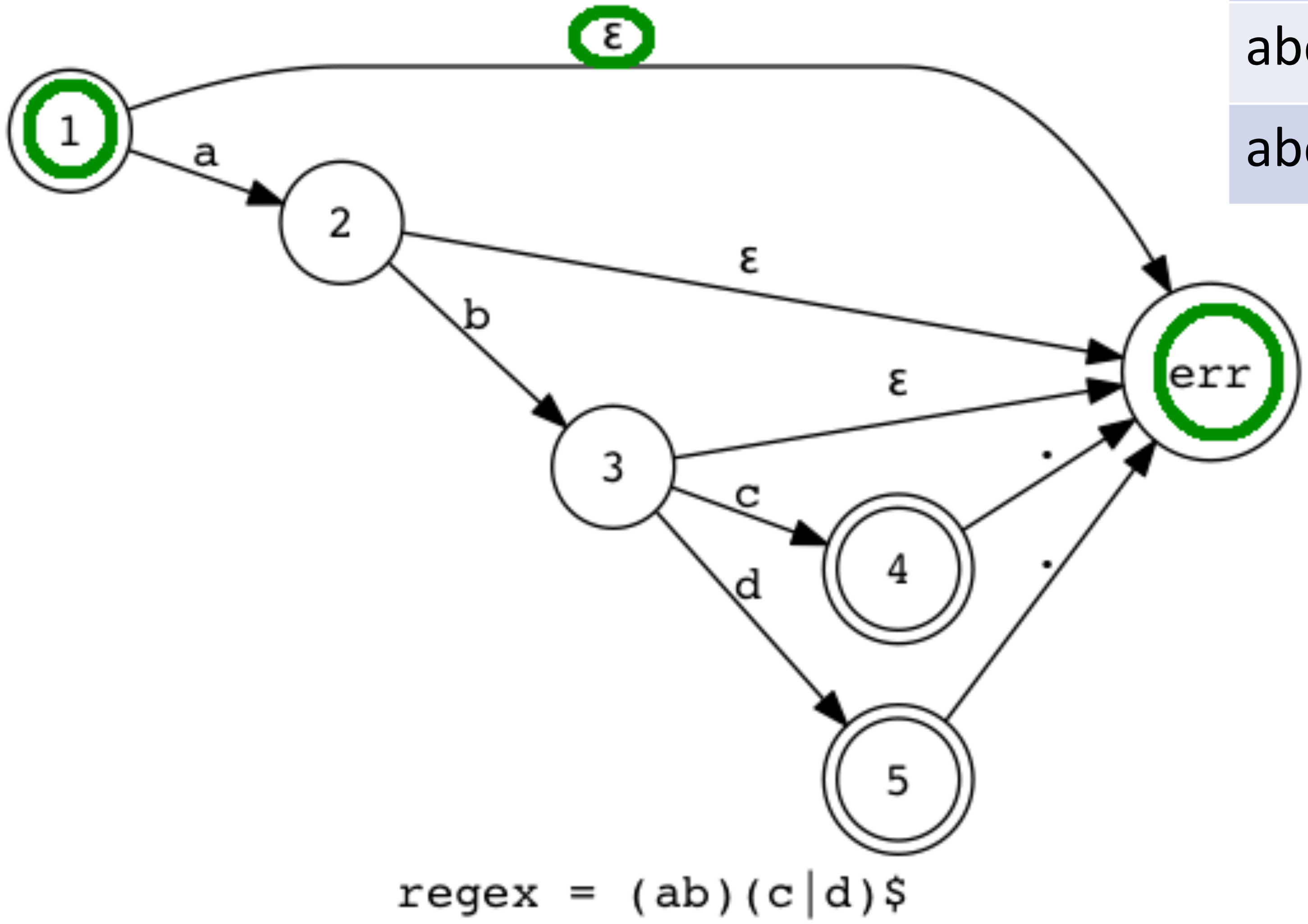
Input	Exp	Act
	err	err
abc	accept	accept
abcde	accept	err



regex = (ab)(c|d)$

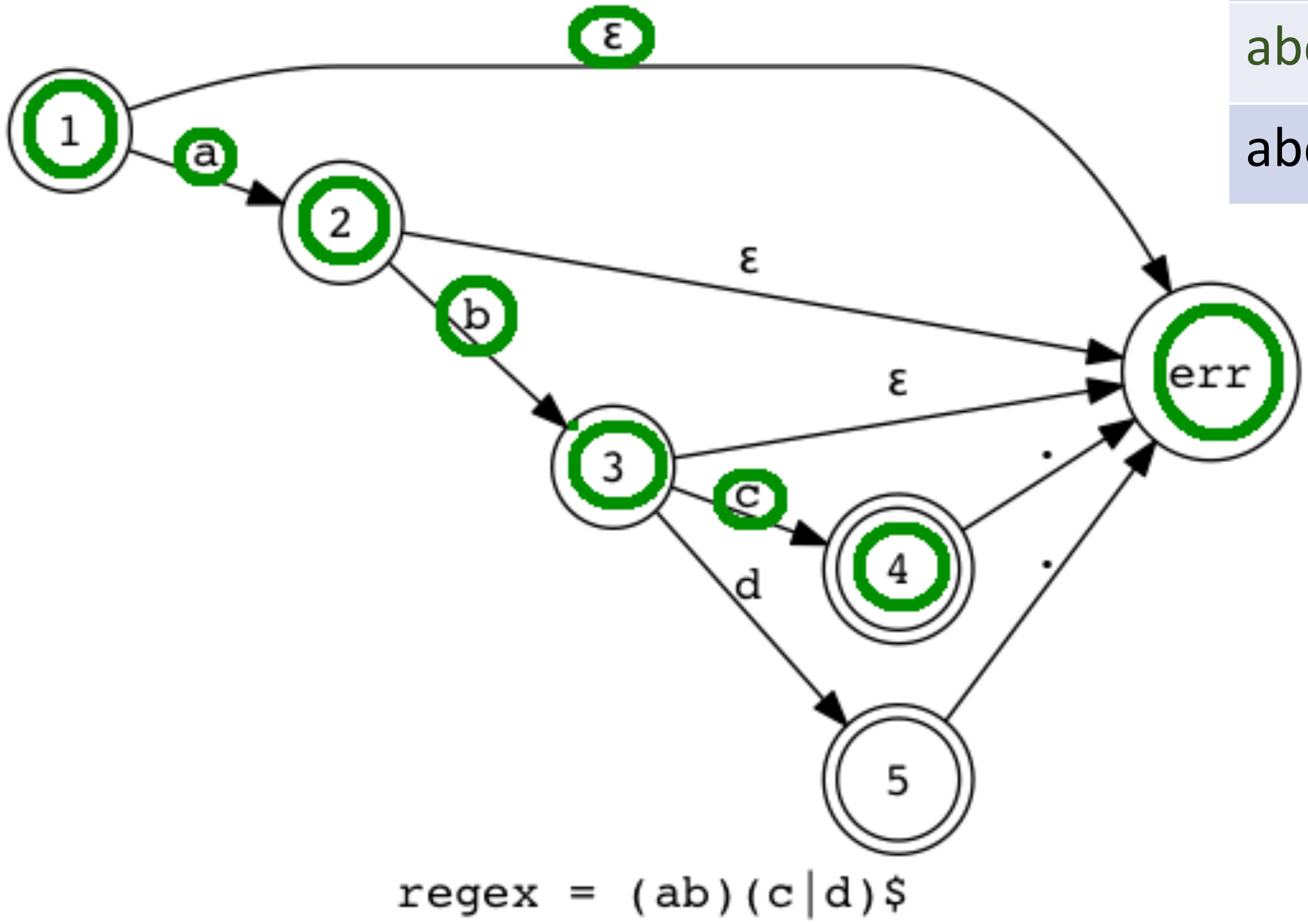
Spectrum-based + finite automata

Input	Exp	Act
	err	err
abc	accept	accept
abcde	accept	err



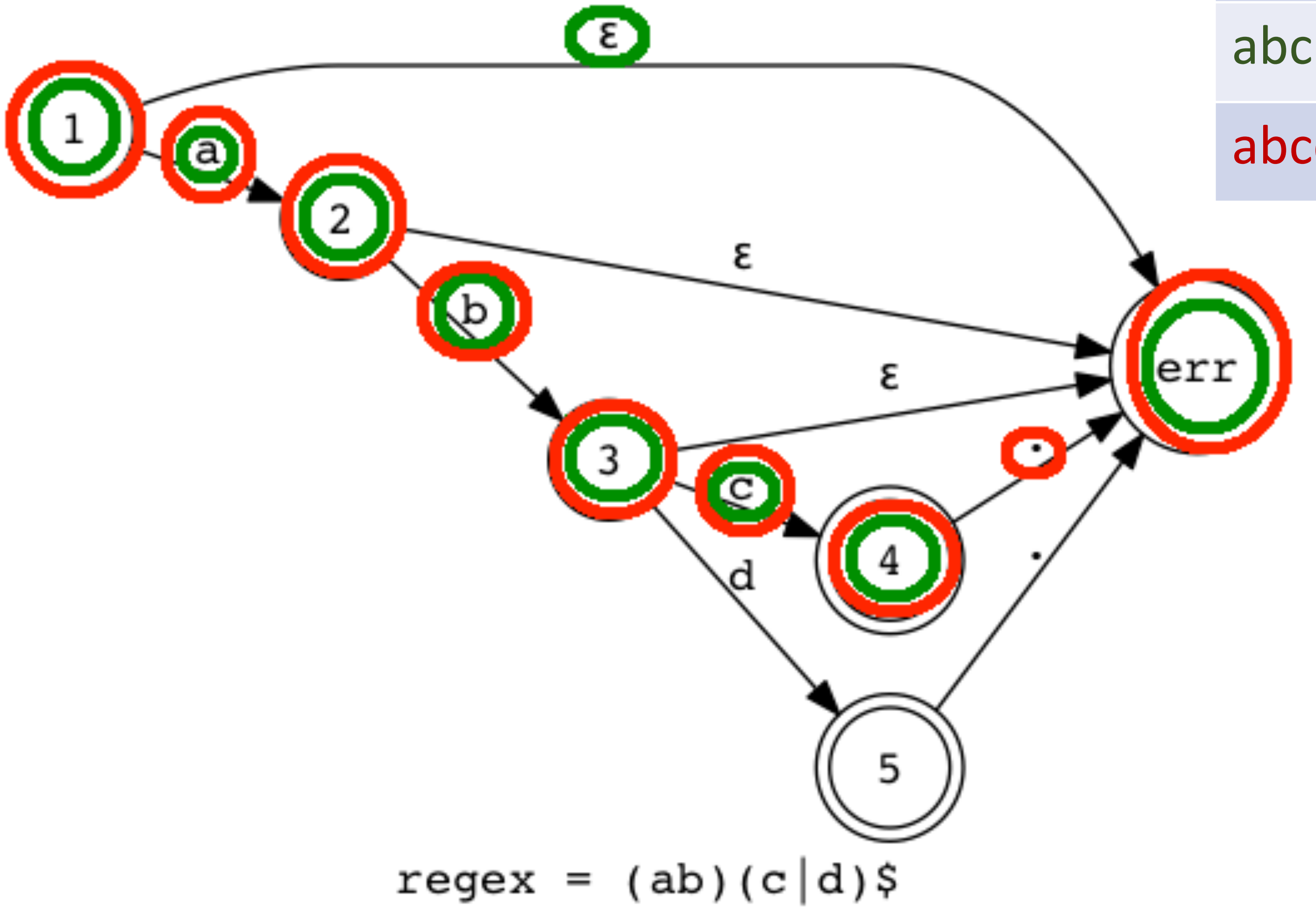
Spectrum-based + finite automata

Input	Exp	Act
	err	err
abc	accept	accept
abcde	accept	err



Spectrum-based + finite automata

Input	Exp	Act
	err	err
abc	accept	accept
abcde	accept	err



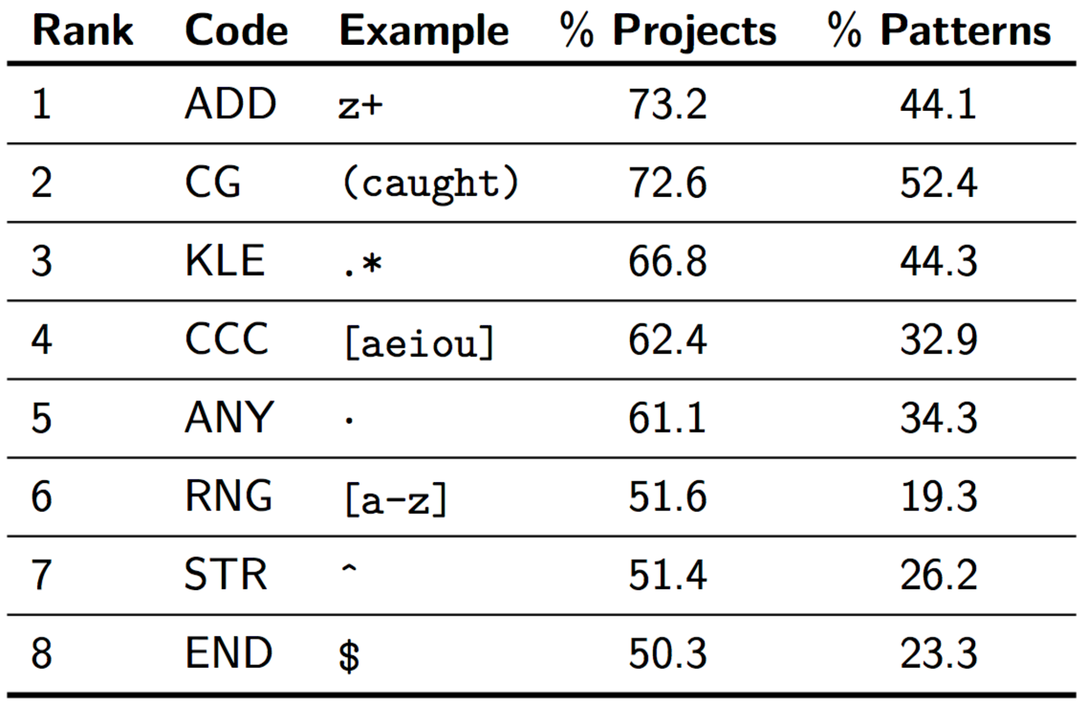
But… perhaps fault localization isn’t strictly necessary.

Patching Regexes

- SearchRepair Hypothesis: a fix exists in existing code
  - If true:
    - Explore the space of ***existing regexes*** from source code
    - Fault localization perhaps *un*necessary
  - If false:
    - Develop **mutation operators** for regexes and try a search-based approach
    - Fault localization necessary

Idea: develop **mutation operators** and use a search-based approach

- Add escape character
- Change repetition counts (e.g., + to *, {2,4} to {2,5})
- Change character classes…
- …



Rank	Code	Example	% Projects	% Patterns
1	ADD	`z+`	73.2	44.1
2	CG	`(caught)`	72.6	52.4
3	KLE	`.*`	66.8	44.3
4	CCC	`[aeiou]`	62.4	32.9
5	ANY	`.`	61.1	34.3
6	RNG	`[a-z]`	51.6	19.3
7	STR	`^`	51.4	26.2
8	END	`$`	50.3	23.3

Idea: generate clusters of **existing regexes**

- How to cluster?
  - by-hand inspection
  - cluster by syntactic similarity like Jaccard or longest substring
  - formal analytical subsumption, no sufficient tools at that moment
  - cluster by behavioral similarity using matching string overlap

Idea: generate clusters of **existing regexes**

- How to cluster?
  - ~~by-hand inspection~~
  - ~~cluster by syntactic similarity like Jaccard or longest substring~~
  - ~~formal analytical subsumption, no sufficient tools at that moment~~
  - cluster by behavioral similarity using matching string overlap

Idea: generate clusters of **existing regexes**

```
Pattern A matches 100/100 of A’s strings
Pattern B matches  90/100 of A’s strings
Pattern A matches  50/100 of B’s strings
Pattern B matches 100/100 of B’s strings
```

	A	B
A	1.0	0.9
B	0.5	1.0

	A	B	C	D
A	1.0	0.0	0.9	0.0
B	0.2	1.0	0.8	0.7
C	0.6	0.8	1.0	0.2
D	0.0	0.6	0.1	1.0

	A	B	C	D
A	1.0			
B	0.1	1.0		
C	0.75	0.8	1.0	
D	0.0	0.65	0.15	1.0

index	pattern	nProjects	index	pattern	nProjects
1	‘:+’	8	5	‘[:]’	6
2	‘(:)’	8	6	‘([^:]+):(.*)’	6
3	‘(:+)’	8	7	‘\s*:\s*’	4
4	‘(:)(:*)’	8	8	‘\:’	2

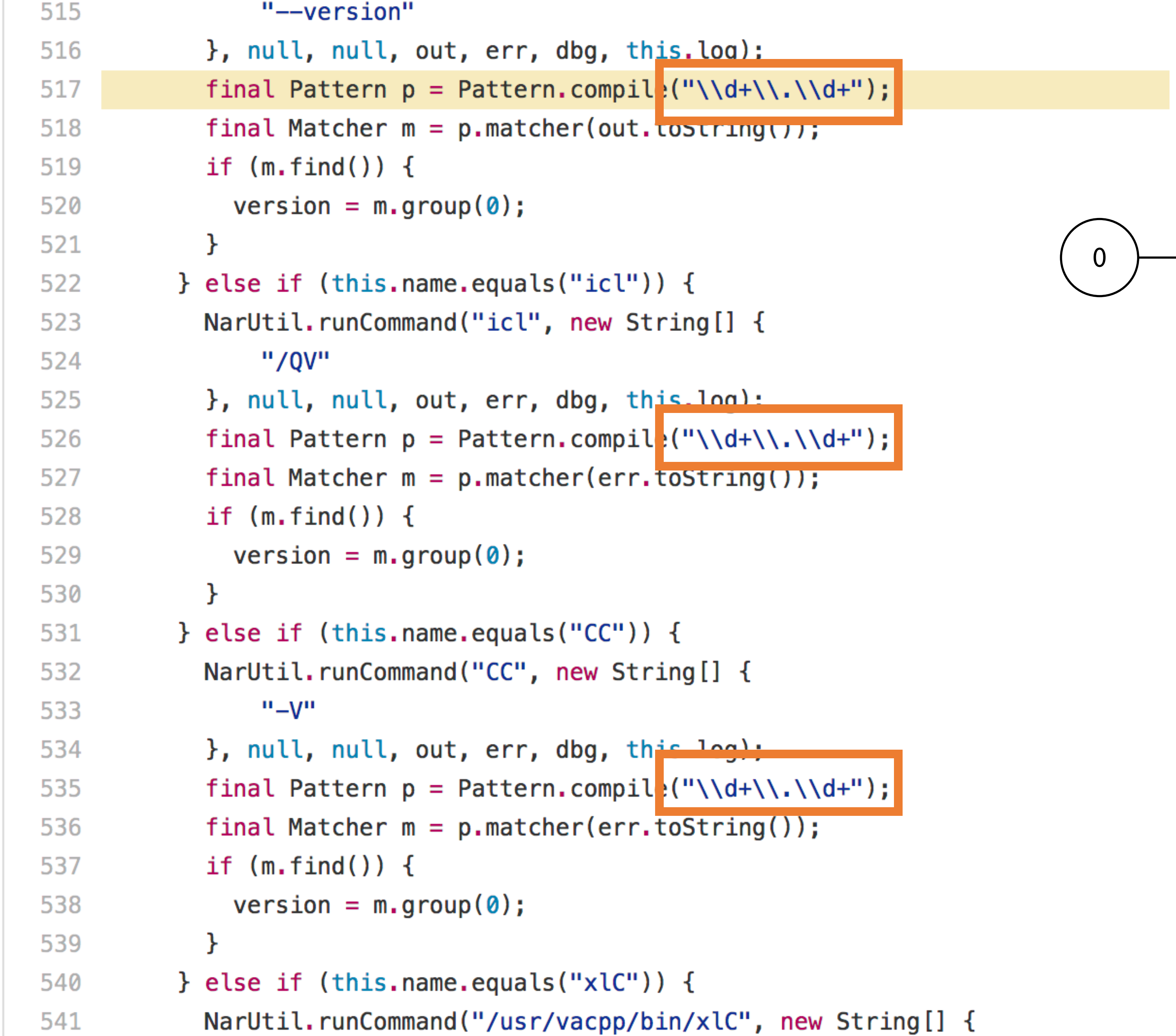
Categories of clusters (top 100)

Category	# Clusters	Example
Multiple matching alternatives	21	‘\W’ or ‘\d’
Anchored patterns	20	‘^\s’
Specific char must match	17	’:+’
Two or more chars	16	’@[a-z]+’
Code search	15	’^https?://’
Content of parens	10	‘\(.*\)’

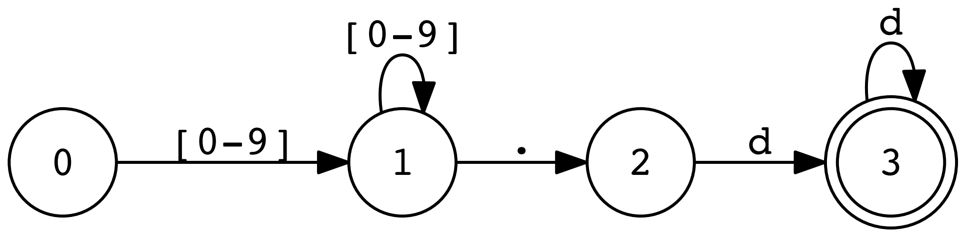
Clusters as fix space

- Given a buggy regex, find closest cluster.
- Guess and check regexes against test cases
- … basically SearchRepair for regexes

What if there are no test cases?



```
      "--version"
   }, null, null, out, err, dbg, this.log);
   final Pattern p = Pattern.compile("\\d+\\.\\d+");
   final Matcher m = p.matcher(out.toString());
   if (m.find()) {
     version = m.group(0);
   }
 } else if (this.name.equals("icl")) {
   NarUtil.runCommand("icl", new String[] {
       "/QV"
   }, null, null, out, err, dbg, this.log);
   final Pattern p = Pattern.compile("\\d+\\.\\d+");
   final Matcher m = p.matcher(err.toString());
   if (m.find()) {
     version = m.group(0);
   }
 } else if (this.name.equals("CC")) {
   NarUtil.runCommand("CC", new String[] {
       "-V"
   }, null, null, out, err, dbg, this.log);
   final Pattern p = Pattern.compile("\\d+\\.\\d+");
   final Matcher m = p.matcher(err.toString());
   if (m.find()) {
     version = m.group(0);
   }
 } else if (this.name.equals("xlC")) {
   NarUtil.runCommand("/usr/vacpp/bin/xlC", new String[] {
```



regex="\d+\.d+"

Challenges

- Different languages have different regex processing approaches
  - Flags often differ between languages
- Existing tools for building automata from regexes (e.g., brics) have incomplete coverage of regex language
- Not all regular expressions are regular
  - e.g., 0.5% of regexes from Python projects use the back-reference feature (non-regular)
- Still need to characterize the potential impact