Problem

- Generating high-covering test inputs manually is tedious and cumbersome.
- Various test-generation approaches such as Dynamic Symbolic Execution (DSE) have been proposed.
- Existing DSE-based approaches can effectively generate high-covering test inputs for various programs but cannot deal with programs that use complex regular-expression (RegEx) operations such as `Regex.IsMatch(s1, regex1)`.

Challenges

- Challenges for DSE-based approaches in testing programs with complex RegEx operations.
- Exploration space is often quite large.
- Collected symbolic constraints may be too complex to be solved.
- Strong need to reduce the exploration space when exploring complex RegEx operations.

Reggae Approach

1. Transform the used RegEx into an automaton, which checks whether an input string matches the RegEx.
2. Use a DSE-based approach to explore feasible paths in the program under test and the automaton (instead of the standard `Regex.IsMatch(s1, regex1)` method).

Example

(1) InputsChecker checks whether cardNum is a valid credit card number and email is a valid email address.

A DSE-based approach cannot generate test inputs to cover statements from Line 5.

```csharp
01: public bool InputsChecker(string cardNum, string email) {
02:     if (!Regex.IsMatch(email)) {
03:         return false;
04:     }
05:     if (!Regex.IsMatch(cardNum)) {
06:         return false;
07:     }
08:     return true;
09: }
```

(2) Replace the invocations of standard RegEx.IsMatch methods with our synthesized IsMatch methods.

Our own Regex class:

```csharp
01: public static bool IsMatch(string text) {
02:     // Code to simulate IsMatch method
03: }
```

Preliminary Results

A. Empirical Study on Open Source Application

- Subject: an email-extraction component in a job board and recruiting system named FlashRecruit.
- Pex, a DSE-based approach, generates 324 test inputs and achieves 80% branch coverage.
- These 324 test inputs cannot cover Statements 11 and 18.
- Reggae replaces `Regex.IsMatch` in Statements 10 and 17 with our own synthesized IsMatch code.
- Pex generates 320 test inputs and achieves 96% branch coverage.

B. Empirical Study on Synthesized Validators

- Subjects: synthesized validators based on complex RegExs collected from popular RegEx library, RegExLib.com. Each validator takes a string as input and uses a RegEx to validate the format of the input string.
- Pex with/without Reggae used to generate test inputs for these validators.
- Reggae helps Pex to achieve 79% branch coverage, increased from 29% without the assistance of Reggae.
- Reggae helps Pex to generate test inputs that effectively improve capability of detecting faults in RegExs.

Among 56 mutated RegExs, Reggae helps improve the percentage of killed mutants from 9% to 61%.

This work is supported in part by NSF grants CCF-0725190 and CCF-0845272, and ARO grants W911NF-08-1-0443 and W911NF-08-1-0105.