

Efficient Pattern-based Static Analysis Approach via Regular-Expression Rules

Xiaowen Zhang*

Dept. of Comp. Sci. and Engr.
Southern University of Sci. and Tech.
Shenzhen, China

11510746@mail.sustech.edu.cn

Ying Zhou*

Dept. of Comp. Sci. and Engr.
Southern University of Sci. and Tech.
Shenzhen, China

11610701@mail.sustech.edu.cn

Shin Hwei Tan[†]

Dept. of Comp. Sci. and Engr.
Southern University of Sci. and Tech.
Shenzhen, China

stan6@sustech.edu.cn

Proceedings of the 30th International Conference on Software Analysis, Evolution and Reengineering (SANER '23), pp132-615

発表者：松尾 和輝

背景 | 静的解析ツール

- プログラムを実行せずにバグを検出するツール
 - e.g. SpotBugs
 - プログラム解析技術を使用しているバグパターンで検出
 - 以下の課題があるためあまり使われていない
 - 動作が遅い[1][2]
 - 全てのコードを解析してしまう[2][6][7]
 - AST/CFGにコードをパースする必要がある[6][8]
 - 偽陽性率が高い[1][13][14]
 - **正規表現**を用いることで課題を解決できるのでは？

[1] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why don't software developers use static analysis tools to find bugs?" in 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013, pp. 672–681.
[2] M. Christakis and C. Bird, "What developers want and need from program analysis: an empirical study," in Proceedings of the 31st IEEE/ACM international conference on automated software engineering, 2016, pp. 332–343.
[6] T. Clem and P. Thomson, "Static analysis at github: An experience report," Queue, vol. 19, no. 4, p. 42–67, aug 2021. [Online]. Available: <https://doi.org/10.1145/3487019.3487022>
[7] B. Dagenais and L. Hendren, "Enabling static analysis for partial java programs," in Proceedings of the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications, 2008, pp. 313–328.
[8] D. A. Tomassi, "Bugs in the wild: examining the effectiveness of static analyzers at finding real-world bugs," in Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018, pp. 980–982.
[13] H. Shen, J. Fang, and J. Zhao, "EFindBugs: Effective error ranking for findbugs," in 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation. IEEE, 2011, pp. 299–308.
[14] S. Heckman and L. Williams, "A systematic literature review of actionable alert identification techniques for automated static code analysis," Information and Software Technology, vol. 53, no. 4, pp. 363–387, 2011.

課題解決方法と本研究の目的

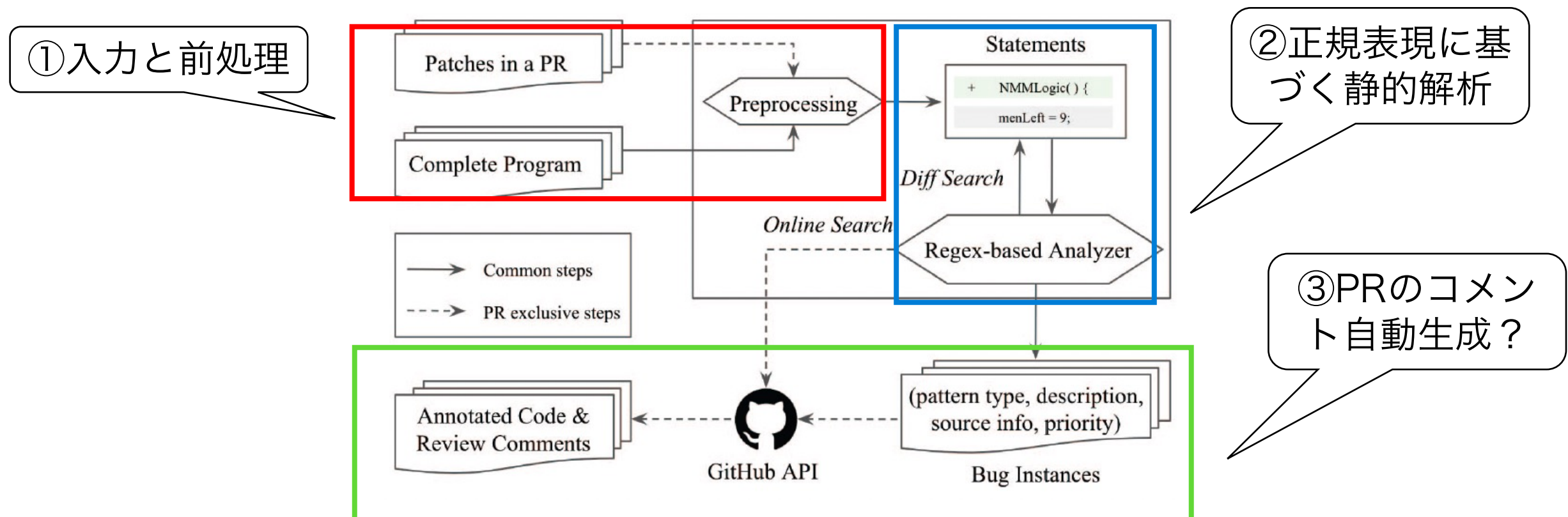
- 正規表現を用いることで . . .
 - 動作が遅い
 - コンパイルのオーバーヘッドを削減可
 - 全てのコードを解析してしまう
 - 解析に必要な情報はメソッドorステートメントレベルのみ (RQ1)
 - AST/CFGにコードをパースする必要がある
 - 文字列のまま解析可
 - 偽陽性率が高い
 - 偽陽性を減らすためのプログラム解析技術のうち、データ型情報が最も重要 (RQ2)

本研究の目的

正規表現を用いた静的解析ツールCodegexの提案

Codegex

・ ワークフロー



①入力と前処理

- 入力 |
 - PR : コードの追加行
 - Complete Program : プロジェクト内のすべてのファイルを追加するコード変更
- 前処理 | Javaプログラムの“;”, “{”, “}”, を元にコードを分割

②正規表現に基づく静的解析

SpotBugをベースとして87種類のパターンに再構築

- (1) シンタックス誘導型マッチング
 - ・ クラス/メソッドのシグネチャ情報をエンコードするキーワードを使用
- (2) 明示型駆動型マッチング
 - ・ 型チェックを要するパターンに対して暗黙のデータ型をマッチング
- (3) 語境界でのマッチング
 - ・ 分析の最適化
- (4) diff検索による解析範囲の拡大
 - ・ ステートメントではなく、すべてのコード変更点を検索、オンライン検索によるリポジトリ内の全ファイルの検索
- (5) 演算子の優先順位の符号化
 - ・ 算術演算やビット演算の解析精度を高める
- (6) アンチパターン
 - ・ 偽陽性のフィルタリング規則

Codegex

③PRのコメント自動生成？

- Codegexが警告を出したPRの各コードスニペットに対して、注釈付きコードでレビューコメントを出力
- 以下のテンプレートを利用

I detect that this code is problematic. According to the *cat*, *sd* (*pat*). *ld*

- バグパターンpatに違反するコードは、(1)バグカテゴリcat（例：STYLE）、(2)短い説明sd、(3)長い説明ldを持つ
- 例



Codegex 18 days ago • edited by bridgekeeper bot



I detect that this code is problematic. According to the [Bad practice \(BAD_PRACTICE\)](#), [Nm: Method names should start with a lower case letter \(NM_METHOD_NAMING_CONVENTION\)](#).

Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.

評価

- RQ3 : SpotBugsと比較して, Codegexが生成する警告の質はどの程度か?
- RQ4 : SpotBugsと比較して, Codegexはどの程度高速か?
- RQ5 : Codegexが生成するコメントの品質はどの程度か?
- RQ6 : Codegexがコードレビューを行う際の効率は?
 - 1PRあたり0.039秒

対象プロジェクトと分析方法

- 以下の条件を満たす48のJavaプロジェクトを対象
 - コンパイルにMavenを使用している
 - スター数上位100件に属するプロジェクトである
 - コンパイルが可能である
- ファイルをSpotBugとCodegexのそれぞれで解析(RQ3, RQ4)
 - 検出されるバグ
 - 解析にかかる時間
 - などに違いがあるか調査

RQ3 | SpotBugsと比較して、Codegexが生成する警告の質はどの程度か？

- 検出されるバグに違いがあるかを調査

Pattern	TP ^R		FN ^R		O	Accuracy		Recall	
	S	C	S	C		S	C	S	C
ES_COMPARING_STRINGS_WITH_EQ	16	5	0	11	5	100.0	31.25	100.0	31.25
SA_SELF_COMPUTATION	0	1	1	0	0	0.0	100.0	0.0	100.0
EQ_COMPARING_CLASS_NAMES	0	1	1	0	0	0.0	100.0	0.0	100.0
DMI_RANDOM_USED_ONLY_ONCE	176	215	39	0	176	81.86	100.0	81.86	100.0
SA_SELF_COMPARISON	0	1	1	0	0	0.0	100.0	0.0	100.0
VA_FORMAT_STRING_USES_NEWLINE	60	66	6	0	60	90.91	100.0	90.91	100.0
NM_CLASS_NAMING_CONVENTION	6	5	0	1	5	100.0	83.33	100.0	83.33
DM_STRING_CTOR	5	1	0	4	1	100.0	20.0	100.0	20.0
UI_INHERITANCE_UNSAFE_GETRESOURCE	8	9	1	0	8	88.89	100.0	88.89	100.0
DMI_USELESS_SUBSTRING	0	1	1	0	0	0.0	100.0	0.0	100.0
DM_BOXED_PRIMITIVE_FOR_COMPARE	2	1	0	1	1	100.0	50.0	100.0	50.0
DM_BOXED_PRIMITIVE_FOR_PARSING	13	5	0	8	5	100.0	38.46	100.0	38.46
IIO_INEFFICIENT_LAST_INDEX_OF	26	25	0	1	25	100.0	96.15	100.0	96.15
DMI_HARDCODED_ABSOLUTE_FILENAME	46	16	1	31	15	97.87	34.04	97.87	34.04
IIO_INEFFICIENT_INDEX_OF	363	354	3	12	351	99.18	96.72	99.18	96.72
Others (20 unique patterns)	108	108	0	0	108	100.0	100.0	100.0	100.0
Total	829	814	54	69	760	93.88	92.19	93.88	92.19

RQ3の回答
CodegexはSpotBugsと同等の精度を達成

RQ4 | SpotBugsと比較して、Codegexはどの程度高速か？

- 検出にかかる時間に違いがあるか調査

Project	KSLOC	SpotBugs			Codegex	Speedup		
		IC	SC	A(S)		S1	S2	S3
community	3.72	160.80	6.55	5.13	0.58	283.91	19.98	8.78
Angular2AndJavaEE	1.42	239.20	92.80	5.44	0.25	963.40	386.87	21.42
biojava	117.78	59.22	13.70	42.44	19.30	5.27	2.91	2.20
nacos-spring	1.80	584.01	12.84	15.36	0.31	1925.08	90.57	49.33
spring-boot	0.48	59.09	3.59	3.60	0.07	904.59	103.78	51.92
spring-boot-java	5.30	70.00	9.80	5.00	0.92	81.80	16.14	5.46
spring-context	3.06	14.42	1.94	3.86	0.38	48.06	15.25	10.15
iot-dc3	14.67	295.20	28.01	45.70	2.97	114.65	24.79	15.37
nacos-spring-project	7.40	104.20	6.94	8.31	0.90	124.99	16.94	9.23
spring-boot-graalvm	0.05	59.06	3.21	3.37	0.01	10927.37	1151.37	589.86
cms-admin-end	8.23	74.80	5.19	5.64	1.47	54.91	7.39	3.85
spring4.x-project	0.58	348.80	2.08	22.42	0.11	3495.71	230.69	211.12
Average	42.73	425.70	23.07	12.67	2.18	1979.28	237.06	55.72

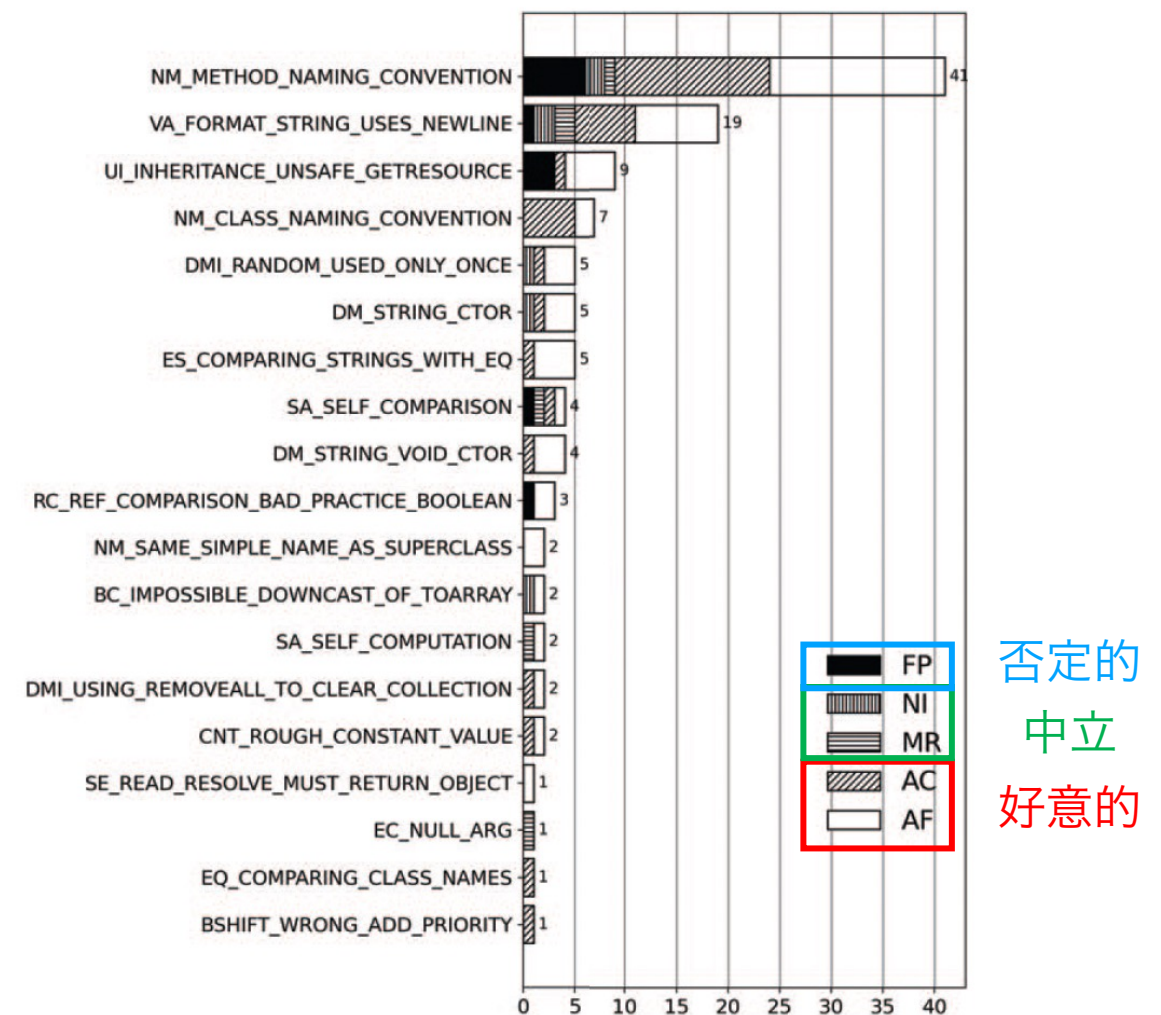
RQ4の回答

Codegexは平均解析時間2.18s（SpotBugsは平均12.67s）と高速

RQ5 | Codegexが生成するコメントの品質はどの程度か？

- 自動生成されたコメントへのフィードバックが好意的か計測
 - PRの開発者から合計116件のフィードバックを収集

得られたフィードバックのほとんどが好意的



RQ5の回答

開発者からいただいた116件のフィードバックのうち、78.45%が好意的なフィードバックである

まとめ

- 正規表現ベースの静的解析ツールCodegexの提案
 - 87のパターンを作成
 - PRのコメントを自動生成
- Codegexの性能評価
 - SpotBugと同程度の精度を有しながら、より高速に動作することを確認
 - Codegexが生成するコメントは有効である

所感

- 読みにくかった. . .
 - 知識が足りてないだけかも？
- 作成した正規表現や, SpotBugsのバグパターンなどの具体例があればわかりやすかったかも？
 - やっぱりSpotBugsの知識が必要そう
- コメント自動生成の部分がいまいちよくわからない
 - 詳しく書いてほしい