5K-05

確率的プログラムにおけるバグ検出

ヴォダイチン　　于海波
九州産業大学　情報科学研究科

1. はじめに

確率的プログラミング(PP)を用いることで、統計的モデルの取り扱い、機械学習などの研究開発がより円滑にできるため、近年 AI の発展により、非常に重要になってきた。しかし、現在は既存の開発環境を利用しており、確率的プログラムの特徴に対応していないという問題がある。本研究室では、実際の確率的プログラムのバグを調査分析し、確率的プログラミング言語独特なバグパターンを抽出し、そのバグパターンを自動的に検出するツールの開発研究を行なっている。

本研究では、確率的プログラムから抽出された確率的プログラムのバグパターンに基づいて、自動的にバグを検出するツールの開発を目的としており、バグ検出方法の調査結果及び現時点で考えているバグ検出ツールの提案について述べる。

2. バグ検出に関する研究調査結果

バグ検出にはいろいろな方法があり、大まかに静的検出方法と動的検出方法に分かれている。また、伝統的なプログラムのバグ検出するツールも複数存在している。これらについての調査結果を以下にまとめる。

2.1.静的検出方法

静的解析は、ソフトウェアの解析手法の一種で、コードを実行せずに行なう検証のことである。静的解析の利点として、コーディング時という開発の早い段階でエラーを検出できるため、バグ修正コストを抑制できる。また、プロジェクトの共通のコーディングスタイルを推進できるため、コードの可読性や保守性が向上するなどの利点がある。ソースコードに対する静的解析は、ソフトウェアの品質を確保する上で、もっとも効果を得やすく、自動化しやすい工程である。機能安全やセキュリティなどに対するコンプライアンスでも、ツールによる静的解析が重要な役割を担っている。

2.2. 動的検出方法

バグ動的検出とは、プログラムの実行中にバグを検出することを指す。これは、プログラムを実行しながら、実行時に発生するエラーや異常を検知することで、プログラムのバグを発見しようとする手法である。動的検出によって、プログラムの中で実行される処理を追跡し、問題が発生した場所を特定することができる。また、動的検出によっては、プログラムを実行中に自動的に修正することもできる場合がある。一方で、動的検出は静的検出に比べて、処理が複雑であるため、実行速度が低下することがある。また、実行時にしかバグを検出できないため、全てのバグを検出することはできない。そのため、静的検出と動的検出を組み合わせて使うことが一般的である。

2.3.伝統的なバグ検出ツールに関する調査

伝統的なバグ検出ツールは複数存在しているが、ここではよく利用されている FindBugs[1]という Java プログラムのバグを検出するオープンソースのツールについて紹介する。

FindBugs は、Java のバイトコードを解析することで、常に実行されることがないかもしれないコードや、不正確な API 呼び出し、潜在的なメモリリークなどのバグを検出することができる。

FindBugs は、プログラムのコードを解析するために、構文解析ツールを使用する。構文解析ツールは、プログラムのコードを構文木というデータ構造に変換することで、プログラムのコードを理解しやすくするものである。構文木を解析することで、FindBugs は、プログラムのコード内に存在する可能性のあるバグを検出することができる。

FindBugs は、構文木を解析するだけではなく、Java の API を使用して、プログラムを実行し、実行時に発生するエラーや異常を検知することで、さらにバグを検出することができる。

また、FindBugs は、既存のバグパターンデータベースを使用することで、より多くのバグを検出することができる。このデータベースは、以前に検出されたバグのパターンを保存しており、新しいプログラムを解析する際に、これらのパターンと比較することで、同じようなバグが存在するかどうかを判断することができる。

3. PyMC3 確率的プログラミング開発環境のバグ分析結果

本研究室では、九州大学の趙研究室と共同研究を行い、PyMC3 確率的プログラミング開発環境の

バグを分析し、バグパターンを抽出する研究を行った[2]。当研究では、PyMC3 で発行された issues のなかで、defects タグが付いたもの総計 271 個のバグ報告を調査し、最終的に 20 個の確率的プログラミング言語特有のバグを抽出することができた。この 20 個のバグからさらに 8 個のバグパターンを抽出した。当該研究のバグ分類方法について、文献[3]が提案された確率的プログラムのバグ分類方法を利用した。

4. バグ検出ツールの提案

本研究ではまず第 3 章で説明した抽出された PyMC3 確率的プログラミング開発環境のバグパターンに対するバグ検出ツールを開発し、その後他の確率的プログラムのバグパターンも検出できるようにバグパターンを追加して行く。

図 4.1 では FindBugs のバグ検出方法を参考し、現時点で考えている Python 言語で書かれた確率的プログラムのバグ検出ツールの流れを示している。

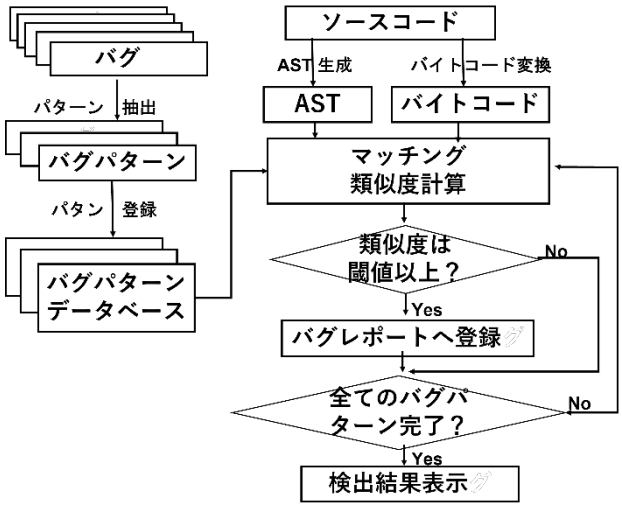

図 4.1 バグ検出ツールの流れ

バグ検出ツールの流れについて、以下で説明する。

4.1. バグの収集、バグパターンの抽出と登録

まず、確率的プログラミング言語を利用したプロジェクトのバグを Github や Stackoverflow から収集し、バグパターンを抽出する必要がある。本研究室のほかのメンバがこの部分の研究をおこなっており、本研究はそちらで抽出されたバグパターンを利用し、バグ検出ツールのバグパターンデータとして登録する。

4.2. ソースコードから AST 及びバイトコードの生成

Python の「ast」モジュール[4]を利用して、ソースコードの AST を生成することができる。また、「dis」モジュール[5]を利用して、ソースコードからバイトコードへ変換することができる。

4.3. バグの検出

バグ検出では、まず既に登録されたバグパターンをインポートし、バグを検出したいソースコードのバイトコード或いは AST に対して、インポートされたパタンとの類似度を計算する。類似度がある設定された閾値を超えた場合、警告を鳴らし、バグレポートに登録する。全ての検出したいバグパターンの検出作業が終了までこのバグ検出作業が繰り返される。

4.4 バグ抽出結果の表示

全ての検出したいバグパターンのバグの検出作業が終了したら、バグ検出結果としてバグレポートを表示する。設定された閾値以上の可能性の高いバグから順番で抽出したバグの情報を表示する。

5. まとめ

本研究では、確率的プログラムのバグ検出ツールの開発を目的として、バグ検出方法の調査結果及び現時点で考えているバグ検出ツールの提案について述べた。今後の課題としては、具体的なバグパターンを検出する方法を提案し、検出ツールを実装し、提案されたバグ検出方法の有効性を検証する。また、よく利用されている確率的プログラミング言語が複数あり、これらの言語を利用して開発されたシステムのバグパターンを続けて調査し、抽出された新しいバグパターンに対するバグ検出方法を提案し、バグ検出ツールを続けて開発する予定である。

参考文献

[1] Findbugs https://findbugs.sourceforge.net/
[2] Shoma Hamada, Haibo Yu, Vo Dai Trinh, Yuri Nishimura, Jianjun Zhao, Bug Patterns in Probabilistic Programming Systems, Workshop of Fault Prediction, Prevention, Detection, and Reliability Enhancement in QRS2022, 2022.
[3] Testing probabilistic programming systems. http://misailo.cs.illinois.edu/papers/probfuzzfse18.pdf. (Accessed on 02/02/2021).
[4] PythonAST, https://docs.python.org/3/library/ast.html
[5] Python Bytecode, https://docs.python.org/3/library/dis.html