1K-01

バグ発生の制約条件追加による コンコリックテストのバグ発見能力の向上

武元　憲将†　　高田　眞吾†　　名倉　正剛‡
慶應義塾大学†　　南山大学‡

1. はじめに

コンコリックテストは，シンボルを入力として実行を行うシンボリック実行と，具体的な入力値を用いた実行を組み合わせるテスト手法である．そして，カバレッジを効率よく高めることができるという利点のため，注目されている．しかし，カバレッジを100%にしたとしてもゼロによる割り算，配列への不正アクセス，辞書型への不正アクセスなどの一部のバグは見逃してしまう可能性がある．そこで，本研究では，Pythonを対象に，コンコリックテストを実施する際にこれらのバグを検出することのできるように制約条件を追加する手法を提案する．

2. コンコリックテスト

コンコリックテストは2005年にSenらによって提案された，具体的な入力値を用いた実行とシンボルを入力とするシンボリック実行を組み合わせたテスト手法である[1]．シンボリック実行は実行される命令やデータフローを解析することによりプログラム中の条件分岐を抽出し，実行パスの制約条件を，シンボルを用いて表現する手法である．これを利用することでコンコリックテストでは実行パスごとに対応する制約条件を満たす入力値による実行を行う．コンコリックテストのアルゴリズムは次の通りである．

(1) ランダムに入力値を生成する．
(2) 具体的な入力値を用いた実行を行う．同時に，その実行パスに対してシンボリック実行を行うことで実行したパスの制約条件を求める．
(3) (2)で求めたパスに対して特定の分岐点において別の枝に進むようなパスの制約条件を求める．
(4) (3)で求めた制約条件を満たすような入力値を求めて(2)に戻る．

3. 関連研究

ゼロによる割り算，配列への不正アクセス，辞書型への不正アクセスといったバグは，発生するかどうかが入力値に依存することがある．コンコリックテストでは条件分岐の真偽により実行パスの制約条件を表現し，実行パスごとに対応する入力値を生成して実行を行う．ところが，生成した入力値によってはこれらのバグが含まれる実行パスを実行したとしてもバグを見落としてしまう可能性がある．

そこで，この問題を解決するためにC言語用のコンコリックテストツールに対してint型のオーバーフローによるバグやゼロ割り算を発生させるための制約条件を追加するTSM[2]が提案された．また，C/C++用のコンコリックテストとファジングのハイブリッドテストツールに対して符号付きint型，符号無しint型のオーバーフローによるバグ，シフト演算によるバグ，配列への不正アクセスを発生させるための制約条件を追加するSAVIOR[3]が提案された．

4. 提案手法

本研究ではPythonを対象に，コンコリックテストを実施する際に入力値に依存するバグの発生を検出できるように，データフローの解析結果にしたがい制約条件を追加する手法を提案する．対象とするバグは，ゼロによる割り算に対するZeroDivisionError，配列への不正アクセスに対するIndexError，辞書型への不正アクセスに対するKeyErrorの各例外を発生させるようなバグである．実装はPython用のコンコリックテストツールpy-conbyte[1]をベースに行った．これらの例外を発生させるバグが残ったままリリースされることが多く，GitHub上でもこれらのバグに関するissueが多く公開されている．

従来のコンコリックテストのアルゴリズムでは2節の手順において(2)でシンボリック実行を行うことで条件分岐の真偽を用いて実行パスの制約条件を求め，(3)でそのうちの一つを否定することにより次の実行のための制約条件を求める．提案手法では(2)において条件分岐での真偽に加えて，3種類の例外が発生する可能性がある箇所を通る際には，例外を発生させる入力値であるかどうかを制約条件に追加する．これによ

Concolic testing with constraints for improving bug detection
†Kazusa TAKEMOTO, Shingo TAKADA, Keio University
‡Masataka NAGURA, Nanzan University

[1] https://github.com/spencerwuwu/py-conbyte

り，(3)で次の実行に使用する入力に対して制約条件の一つを否定することで，例外を発生しなかった実行の後に例外を発生するような制約条件が生成される．次に3種類の例外を発生するようなバグを検出するために，どのような条件を追加するかを説明する．

4.1. ZeroDivisionErrorについて

シンボリック実行の解析中に割り算を行う命令を検出した場合に，ゼロによる割り算が発生するかどうかを，その命令を実行する実行パスの制約条件に追加する．制約条件は，次に示す条件式によって表現する．ここで，除数のシンボルをaとする．

- ZeroDivisionErrorが発生しない場合:$a \neq 0$
- ZeroDivisionErrorが発生する場合:$a = 0$

上記は，ZeroDivisionErrorが発生しない場合，次は除数が0でないのでa≠0という制約条件を追加し，また，ZeroDivisionErrorが発生する場合，除数が0であるのでa=0という制約条件を追加するということを意味する．

4.2. IndexErrorについて

シンボリック実行の解析中に配列へのアクセスを行う命令を検出した場合に，配列への不正アクセスが発生するかどうかを，その命令を実行する実行パスの制約条件に追加する．制約条件は，次に示す条件式によって表現する．ここで，配列へアクセスする変数のシンボルをa，配列の長さの値bとする．

- IndexErrorが発生しない場合:$|a| < b$
- IndexErrorが発生する場合:$|a| \geq b$

4.3. KeyErrorについて

シンボリック実行の解析中に辞書型へ参照を行う命令を検出した場合，辞書型への不正アクセスが発生するかどうかを，その命令を実行する実行パスの制約条件に追加する．制約条件は，次に示す条件式によって表現する．ここで，辞書型へアクセスするキーの値のシンボルをkey，辞書型のキーの値を$key_1, key_2, \cdots, key_n$とする．

- KeyErrorが発生しない場合:
  $key = key_1 \vee key = key_2 \vee \cdots \vee key = key_n$
- KeyErrorが発生する場合:
  $key \neq key_1 \wedge key \neq key_2 \wedge \cdots \wedge key \neq key_n$

5. 評価

本研究では提案手法を用いたコンコリックテストのZeroDivisionError，IndexError，KeyErrorの検出能力と実行時間を，従来のコンコリックテストと比較することにより評価を行なった．評価実験には，GitHub上に公開されている，メインの開発言語がPythonである任意の7プロジェクトから，割り算や配列や辞書型をプログラム内で利用している8個の関数を選択した．それぞれに対して次のいずれかを適用し，合計10個のバグを含む関数を用意した．

- 元々例外処理が記述されていなかった関数はそのまま利用．
- 例外処理が記述されたコードの，例外処理部分のコードを削除して利用．
- 例外が発生しないコードに，例外が発生するようなコードを埋め込んで利用．

結果を表1に示す．従来の手法では10個のバグのうち9個を見逃していたが，提案手法では全てのバグを検出できた．このように，提案手法では3種類の例外を発生するバグを検出するための制約条件を追加することで，従来手法に比べてより多くのバグの検出が可能となった．一方で，実行時間は従来の手法に比べ，提案手法では最小1.5倍，最大27倍となった．解く条件数と実行するテストケースが増えたため，実行時間が増加した．

表 1　実験結果

関数名	例外の種類	検出可否		実行時間		
		従来	提案	従来 (sec)	提案 (sec)	提案/従来
division	ZeroDivisionError	×	○	0.003	0.081	27.0
getKana	KeyError	×	○	0.486	1.078	2.2
month_name	IndexError	×	○	0.007	0.087	12.4
k_napsack	IndexError	○	○	4.093	9.247	2.3
gcd	ZeroDivisionError	×	○	0.068	1.351	19.9
transform_rank	KeyError	×	○	0.007	0.098	14.0
toKana	KeyError	×	○	2.467	18.283	7.4
check_BMI	ZeroDivisionError	×	○	0.791	1.194	1.5
check_BMI	IndexError	×	○	0.771	2.612	3.4
check_BMI	KeyError	×	○	0.768	1.945	2.5

6. おわりに

本研究では，Python を対象にコンコリックテストを実施する際に，従来検出できなかったバグを検出することができるように制約条件を追加する手法を提案した．今後は対応するバグの種類を増やすことにより，よりコンコリックテストのバグ検出能力の向上を目指す．

謝辞　本研究の成果の一部は，科研費基盤研究(C)20K11758，2020年度南山大学パッヘ研究奨励金I-A-2の助成による．

参考文献

[1] K.Sen, et al. Cute: A concolic unit testing engine for C. ACM SIGSOFT SE Notes, 30(5), pp. 263-272, 2005.
[2] H.Liang, et al. A novel method makes concolic system more effective, CSCloud 2017, pp. 243–248, 2017.
[3] Y.Chen, et al. Savior: Towards bug-driven hybrid testing, SP 2020, pp. 1580-1596, 2020.