制御フローグラフのパタンマッチによるガベージコレクションのルート挿入漏れバグの検出

1190367　藤本太希　【 プログラミング言語研究室 】

1　はじめに

高速で信頼性の高いガベージコレクション (GC) を実装するには，オブジェクトへのポインタの集合である精確なルート集合を求めることが必要になる．その方法の一つに，オブジェクトへのポインタを持つ変数のアドレスをプログラマが明示的に GC に伝える方法がある．図 2 がその方法を実装した C 言語のプログラム例である．ADD がルート集合への登録，REMOVE がルート集合から取り除く処理である．関数 some_function の中で GC が起こる可能性がある場合，後に使われる可能性があるオブジェクトへのポインタを持つ変数 root は ADD でルート集合に加える必要がある．また，関数から戻る前には REMOVE で取り除く必要がある．この ADD と REMOVE の挿入は間違いやすく，バグの原因になりやすい．

本研究では，ADD と REMOVE が正しく挿入できているかを，制御フローグラフ上のパタンマッチにより検査する．パタンマッチには Coccinelle[1] を利用する．

2　パタンの設計

正しいプログラムの条件からバグの特徴を明らかにし，それらを検出するパタンを作成する．パタン作成の際，GC が起こりうる関数は実行コンテキストを表す特定の型の値を引数に取ることと，オブジェクトへのポインタを持つ変数は型で判定できることを前提とする．

正しいプログラムは,「オブジェクトへのポインタを持つ変数だけが，GC が起こりうる関数の実行時に一回だけルート集合に登録されている」という条件を満たす必要がある．これを満たさないプログラムは，次のいずれかの特徴を持つ．

- GC が起こりうる関数の呼出し前にオブジェクトへのポインタを持つ変数を一度も ADD していない
- ADD した変数がオブジェクトへのポインタを持つ変数の型のものではない
- ADD した後に REMOVE することなく ADD している
- GC が起こりうる関数の実行前に REMOVE している

これらの特徴を表すパタンを作成する．正しいプログラムの条件は他にもあるので，これらについても同様にバグの特徴に分解して，パタンを作成する．Coccinelle は作成したパタンを入力として受け取り，CTL(計算木論理) に変換して制御フローグラフ上でパタンマッチする．

3　評価

作成したパタンの正しさと有用性の確認をするために実験を行なった．対象の C プログラムは，JavaScript 仮装機械 (VM)[2] のソースコードである．このソースコードは，オブジェクトへのポインタを持つ変数を宣言と同時に初期化し，その直後に ADD を，スコープか関数の終わりに REMOVE を行うという方針で作られている．実験では，JavaScriptVM のソースコードから ADD と REMOVE を全て消したソースコードに対して，ADD と REMOVE を消した箇所を検出できるか確認した．

```
1 void foo(Context *ctx, Object *root) {
2   Object *obj;
3   ADD(&root);
4   obj = some_function(ctx, SIZE);
5   use(root);
6   ...
7   REMOVE(&root);
8 }
```

図 1　精確なルート集合を求める処理が入ったプログラム

実験の結果を図 2 に示す．42 箇所の ADD と，それに対応する REMOVE は検出しなかったが，その全てが本当は必要のない処理であった．誤検出は 1 箇所あった．この原因は，制御フローグラフ上の検査では全ての実行経路を検査するため，プログラム実行時には絶対に通らない経路も検査してしまうからである．さらに，そのソースコードで ADD や REMOVE がもれていたバグを，6 個検出することができた．

	検出せず	誤検出	バグ検出
合計	42	1	6

図 2　実験結果

4　おわりに

本研究では，ADD と REMOVE が正しく挿入できているか検査する Coccinelle のパタンを作成して評価した．その結果，作成したパタンの正しさと有用性が確認できた．

参考文献

[1] Julia L. Lawall, Julien Brunel, Nicolas Palix, Rene Rydhof Hansen, Henrik Stuart, Gilles Muller. WYSIWIB:A Declarative Approach to Finding API Protocols and Bugs in Linux Code. In *Proc. Dependable Systems and Networks 2009*, pp.43-52, 2009.

[2] T. Kataoka, T. Ugawa, and H. Iwasaki. A Framework for Constructing Javascript Virtual Machines with Customized Datatype Representations. In *Proc.* SAC '18, pp.1238-1247, 2018.