ポインタ解析問題の自動生成

笹田 研悟1　　荒堀 喜貴1　　権藤 克彦1

1. はじめに

ポインタ解析[1]とは主に C 言語などにおいて変数が指している対象を求める解析の事である。ポインタ解析による情報はプログラムの最適化[2]などに役立つが、解析結果を完全に得る事は困難である為、計算量と精度のトレードオフを考慮した様々なポインタ解析のアルゴリズムが提案されている。

ポインタ解析のアルゴリズムを実装した解析器を評価する為の方法としてポインタ解析問題を用意し、解析器に解かせることで解析器の精度がどの程度かを評価する方法がある。ここでいうポインタ解析問題とは、解析対象のプログラムを問題として、プログラム中のある変数が指定したプログラムポイントにおいて何を指すかをあらかじめ求めたポインタ解析結果を答案とする問題と答案のセットの事である。ポインタ解析問題の答案を正確に生成するには手動でポインタ解析を行う必要があるが、手動で質の良い問題と答案の生成は難しい。

本研究は、解析対象のプログラムからポインタ解析問題を自動生成する手法を提案する。完全なポインタ解析の結果を得るアルゴリズムの生成は困難である為、本研究では不正確なポインタ解析と正確な答案により近いポインタ解析の二つの精度の解析結果を利用して対象とする解析器の精度の評価を行う。より正確なポインタ解析結果を得る為、本研究では動的記号実行[3]を用いたパス依存ポインタ解析により解析を行う。

2. 提案手法の概要

提案手法によるツールは引数として解析するプログラムファイルと解析対象とするプログラム中の変数、プログラムポイントを指定し、プログラムポイントでの変数のポインタ解析結果を二つの精度で出力する。ツールは次の二つのポインタ解析手法によって答案を出力する。

一つ目は Andersen によるフロー非依存・文脈非依存のポインタ解析[1]を行う。Andersen のアルゴリズムはポインタ解析の基本的なアルゴリズムで多くの解析アルゴリズムの元となっている。

二つ目は動的記号実行によるパス依存のポインタ解析を行う。これは静的なポインタ解析と比較してより正確なポインタ解析を得る事を目的としている。手法の詳細は次章にて説明を行う。

3. 動的記号実行によるポインタ解析

動的記号実行とはテストケース自動生成の研究などに用いられる、プログラムの実行と記号実行を組み合わせた手法の事である[3]。

動的記号実行は任意の値を入力の初期値として初期実行を行い、分岐条件などの実行中にパスを通過する為に必要な入力が取るべき値に関する制約式の収集を行う。そして集めた制約式から一部の制約式を反転させた制約式の集合をソルバに解かせる事で、別のパスを通過する為の具体的な入力を取得する事が出来る。このようにして取得した入力による再実行と動的記号実行による入力の取得を繰り返していく事で実行パスを網羅していく。

本研究のポインタ解析は図 1 の様にしてプログラムポイントに到達する実行パスをできる限り実行しながら対象とする変数が指している対象を求める事で、静的なポインタ解析結果に比べより正確な解析を行う事が出来ると考えている。

動的記号実行をポインタ解析に利用する際の改善点としてテストケース生成などに利用する場合は全ての実行パスを網羅するのに対して、本研究ではプログラムポイントに到達する実行パスのみを網羅するだけで良いという点が挙げられる。そこで、本研究では解析前にプログラムポイントに到達しないコード、及びポインタ解析に影響を及ぼさないコードに関して実行しないように処理をしておく他、動的記号実行による次の実行の為の入力の選択を行う事で効率的に探索出来るようにしている。

1 東京工業大学大学院 情報理工学院

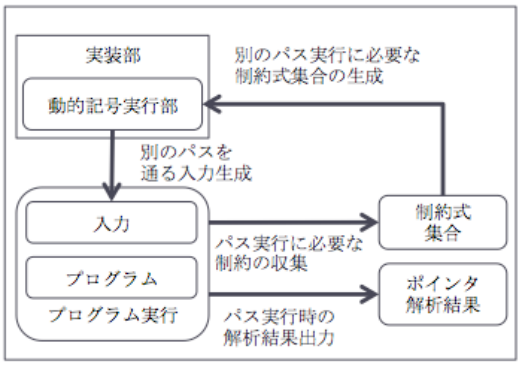

図 1 動的記号実行によるポインタ解析の概要

4. 実装

実装の対象言語は C 言語とする事が望ましいが、C 言語に対応する動的記号実行を既存のコンパイラから生成する事は技術的・時間的に困難である為、本研究では C 言語のサブセットである XC[4]を対象として XC のコンパイラ XCC[4]を元に実装を行う。XC は C 言語の文法を簡略化したものだが、基本的なポインタ演算や多重ポインタなどのポインタの文法は含まれており、C 言語に拡張した場合のポインタ解析結果への影響は少ないと考えている。

動的記号実行で利用するソルバは SMT ソルバ Z3[5]を用いる。選択した理由の一つとして SMT ソルバは SAT ソルバと比較して配列などの処理が追加されており、C 言語などと組み合わせる場合の労力が比較的少なく済む事ができる事が挙げられる。

5. 評価実験計画

本研究の評価手法として、本研究のツールが出力する答案の質の評価と、出力した問題を用いて別のポインタ解析器の評価を行う事を予定している。解析対象のプログラムは XC で作成したファイルを用意するほか、評価対象の解析器は自作、またはオープンソースの文脈依存ポインタ解析器を使用する。文脈依存解析はプログラムの各手続き呼び出しを区別して扱う解析の事で、ポインタ解析の中では高い精度で結果を出力する事が出来る。

評価項目としては現段階で次の三項目を想定している。まず生成した問題と答案によって評価対象の解析器の誤検出を特定できるかを調査する。これは実行時には到達不可能なパスがある場合などが考えられる。次に答案の質の評価としてパス依存解析のカバレッジを計測する。評価するカバレッジの種類に関して、パス依存解析は全ての実行パスを通る必要が無い為、新しいカバレッジの指標を用意する事も考慮する。最後に本研究のツールによる答案生成時間について計測を行う。

6. 関連研究

動的記号実行を利用した研究として Cadar らによるバグを引き起こす入力の検出ツール EXE の研究がある[3]。同様に、Paul らはパッチのテストをする為のツールとして KATCH を開発している[6]、KATCH は本研究のように動的記号実行の際に枝切りや入力選択による効率的なパス探索手法を実装しているが、KATCH がパッチに到達するパスを早く発見する事を目的としているのに対し、本研究ではプログラムポイントに到達するパスだけを効率よく網羅する事を目的としている点が異なる。

7. まとめ

本研究はポインタ解析器の評価に用いる解析問題を自動生成することを提案し、答案の生成手法として動的記号実行によるパス依存ポインタ解析の実装を行なった。今後の計画として未実装箇所の実装を行った後、評価実験を行う事を予定している。

参 考 文 献

[1] Lars Ole Andersen, “Program Analysis and Specialization for C Programming Language”, Phd Thesis, University of Copenhagen,1994.

[2] Manuvir Das,Ben Liblit, Manuel Fähndrich,Jakob Rehof, “Estimating the impact of Scalable Pointer Analysis on Optimization”, In Proc.8th International Symposium on Static Analysis,pp. 260-278,2001

[3] Cristian Cadar, Vijay Ganesh, Peter M. Pawlowski, David L.Dill,Dawson R. Engler, ”EXE:automatically generating inputs of death”, In Proc. 13th ACM conference on Computer and communications security, pp. 322-335, 2006

[4] 権藤克彦, 福安 直樹,荒堀 喜貴,”ネイティブアセンブリコードを出力する教育用コンパイラ(XCC)と水平スライスが可能な可視化ツール(MieruCompiler)”, 電子情報通信学会論文誌 ,vol.J95-D, No.5, pp.1225-1241, 2012

[5] The Z3 Theorem Prover, https://github.com/Z3Prover/z3

[6] Paul Dan Marinescu, Cristian Cadar ,”KATCH: high-coverage testing of software patches”, In Proc. 9th Joint Meeting on Foundations of Software Engineering , pp. 235-245, 2013