神奈川大学理学部　情報科学科　馬谷研究室

Scheme の為の抽象解釈による制御フロー解析の調査報告

茂木 洸樹 20183448

1　はじめに

本調査では Scheme を使用する．それを用いて抽象解釈による制御フロー解析について調査を進めていく．

本調査の目的は Scheme の為の抽象解釈による制御フロー解析がどのように行われているのかを調査し，行われている事象を明確にすることによって理解を深めることにある為，大まかに行うことは本題でもある抽象解釈と制御フロー解析における知識の確認と，それにおける抽象解釈による制御フロー解析について考えていく．その為に文献 [1, 2] を踏まえて調査を行っていく．

2　対象言語

文献 [2] のサイトに挙げられているプログラムを基準として調査をする．

上記のサイト内に紹介されている k-CFA.ss のプログラムが今回の目的調査に必要なプログラムとなり，言語は Shceme で構成されている．

以下のものは k-CFA.ss のパススタイルのラムダ計算機 (CPS) を実装したものである．

CPS のための具体的な意味論をいくつかの抽象的なインタプリタに分割した文法の (純粋な) CPS は便利なほど小さくなる傾向にある． [1]

```
exp ::= (make-ref <label> <var>)
    | (make-lam <label>
                (<var1> ... <varN>)
                <call>)
call ::= (make-call <label> <exp0>
                    <exp1> ... <expN>)
```

上記のラムダ計算機 (CPS) は各引数に対してラベルを与え，各引数に意味合いを持たせている．この場合、exp 内で書かれている<call>は

```
call ::= (make-call <label>
           <exp0> <exp1> ... <expN>)
```

であり，call 内に書かれている<expN>には exp に書かれている

```
exp ::= (make-ref <label> <var>)
```

が入ってくる．このラムダ計算機と k-CFA は同等である．

プログラムが書かれたファイル k-CFA.ss ではファイル名の通りに k-CFA が使用されている．

3　制御フロー解析

制御フロー解析とはプログラムの実行パスを解析するフロー解析の一種であり，この k-CFA の CFA は制御フロー分析のことを指し，バリューフローの問題を解決するアルゴリズムのクラスであり、CFA は控えめな過剰近似を計算するものである．CFA は、難解な技術に頼ることなく，誤検出の数を最小化しようとするものである．

フロー解析とは開発の早い段階から動的なテストでは検出が難しい問題を効率よく検出出来る．

しかし，指摘された問題はコードを実行した結果ではないので，実際はバグではない可能性もありえるという不正確性も持ち合わせている．

その不正確性を無くすためにも，抽象解析を行うことがよりプログラムでの解析の正確性を与えるこ

とになる.

4 k-CFA

ここでは k-CFA に関することを記述する．k-CFA の k とは何かしらの数が入ることを意味している．つまり，0CFA ということもあり得るということになる．k-CFA では複数のアドレスが存在するが，0CFA のような分析では，各変数に対して抽象的なアドレスが 1 つだけ存在する.

文献 [2] で紹介されているサイトにもある k-CFA は制御フローの問題に対する最初の一般的な解決策である.

k-CFA は、ますます精密になる制御フロー解析のよく知られた階層で，制御フロー問題の解を近似的に求める，精密な制御フロー解析の階層である.

制御フロー問題の解を近似するものであるという事になる.

これは，ますます正確でかつ，ますます遅くなる分析の階層である．それは k -CFA は元々，継続渡しスタイルに変換された Scheme の抽象解釈として指定されていたためである.

以下 Shceme で書かれた k-CFA である.

```
(define empty-set (set))
(define (map-set f s)
  (for/fold ([ns empty-set])
    ([e (in-set s)])
    (set-add ns (f e))))
...
```

文献 [2] の k-CFA で行われている実装に関してはチュートリアル/リファレンス実装である.

つまり，純粋に機能的であり，効率的なデータ構造よりも便利なデータ構造を使用し，収束を加速するためにいかなる形式の拡張も使用していないものとなる.

5 まとめ

ファイル `k-CFA.ss` では，限られた区間に対してのみの限定的な解析に向いてはいるものの，広範囲での解析を行うにあたってはその能力を十分に発揮することは無い.

そこで一般化出来る部分に分けることが `k-CFA.ss` では可能であるため，`k-CFA.ss` 内のプログラムを二つに分けて簡略化し，目的で使う部分のみの抽出を行うことが出来ればより k-CFA の抽象解析で行われる範囲よりもより広い範囲での抽象解析を行うことが可能となる.

参考文献

[1] Abstract interpreters for free (2010) `https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.167.5186`

[2] k-CFA: Determining types and/or control-flow in languages like Python, Java and Scheme `https://matt.might.net/articles/implementation-of-kcfa-and-0cfa/`