神奈川大学理学部　情報科学科　馬谷研究室

マルチメソッド機能を備えた言語の制御フロー解析

長谷川 靖親 201703367

1　はじめに

一部のプログラム言語にはマルチメソッド機能という機能をそろえている言語がある．そのマルチメゾット機能とは，関数を呼び出すときに使用した引数の型によって実行時に呼び出す関数を選ぶという機能である．

この機能は同じ呼び出しに対し複数の定義をする必要があり，これらは複雑になることが多く，制御フローを解析するにあたり普通のプログラムより解析を行うのが難しい．

この研究では，マルチメゾット機能が使用された関数の流れを制約という形で読み取り，解析をすることにより制御フローの解析を行う．

2　制約生成の設計

解析言語には Racket を使用する．実装するにあたり Racket の実装されている全ての式すべてに制約作成を結びつけるのは多量の設定が必要なので，マルチメソッド機能の解析に必要になる物のみに絞りサブセット言語を作成し制約の作成を行った．そして制約は式単位で発行される．

制約作成に使用したサブセット言語の BNF をは以下の通りにした．

P ::= E E ...
E ::= (E E ...)
　　| (define X E)
　　| (define-generic (X X X ...))
　　| (define-instance
　　　　((X N N ...) X X...) E)
　　| (struct N (V ...))
　　| X | I

となり P はプログラム全体，E は式，N は構造体名，V は構造体のメンバー名，I は数値，X は記号のことを示す

制約作成時に作成される制約は 3 種類あり．

- $f \in f[1]$（関数の内包）これはある関数がを別の関数を関数呼び出しなどで使用した場合，それは呼び出した関数が呼び出された関数を内包しているという関係なのでこの制約が生成される．Racket では関数定義の define や関数呼び出し時に制約生成される．
- $e \subseteq x[1]$(包含関係) これはある変数や関数が代入など包含したりされたりされるとでこの制約が生成される．Racket では変数定義の define などで制約生成される．
- $f \in e \rightarrow e_1 \subseteq a_f^1 \wedge ... \wedge e_n \subseteq a_f^n \wedge e'_f \subseteq e_(e_1, ..., e_n)[1]$(関数呼び出し) これは関数が呼び出されたとき条件がそろっていれば，関数定義の時に設定した引数と呼び出したときに使った引数，そして返り値と呼び出した式自体が包含関係になっている，という制約である．

制約発行の例として

```
...
(define (inc i) (+ i 1))

(define (test)
 (begin
    (define f inc)
    (f n)))
...
```

というプログラムの部分からは $inc \in inc$, $inc \subseteq f$, $inc \in f \rightarrow n \subseteq i \wedge +i1 \subseteq f(n)$ のような制約が

生成される．

3 制約解消

プログラムから生成された上記の制約群から関数が辿る可能性を示すグラフが生成できる．制約から作成されるグラフは主にノード，ノードに付随するビット配列，各ビットに付随するペア，ノード間を結ぶ矢印から構成される．

ノードが示すものは関数や関数が定義されている変数であり，これは関数や変数が定義されたときに作られる．ノードに付随するビット配列が示すものは，そのノードがどの関数を内包しているかであり，関数の内包の制約により生成される．各ビットに付随するペアは，ビットが立ち内包された関数があらわになったときに成立するはずの包含関係がペアとして格納される．ノード間を結ぶ矢印が示すものはノード同士の包含関係であり，包含関係の制約生により生成される．

グラフの例として上記の制約からグラフを作成すると図 1 のようなグラフができる．

4 マルチメゾット機能

マルチメソッド機能を使用する際の記述は (define-generic (XXX ...)) と (define-instance ((XNN ...) XX...) E) の形があり，define-generic で関数がいくつ引数を受け取るか，define-instance でどの形の引数を受け取ったらどのような挙動をするかを定義する．なおマルチメゾット機能が使用された関数の引数は構造確か受け取らない．[2]

これらが行っているのは関数の定義なので，基本は (define XE) の関数の定義の場合の制約と，関数の引数名や数返り値などが記録される点では (define XE) の記法と同じだが，その他にもどのような引数をとったらどの返り値を返すというのも記録される．制約の処理は他の関数と同じように処理される．

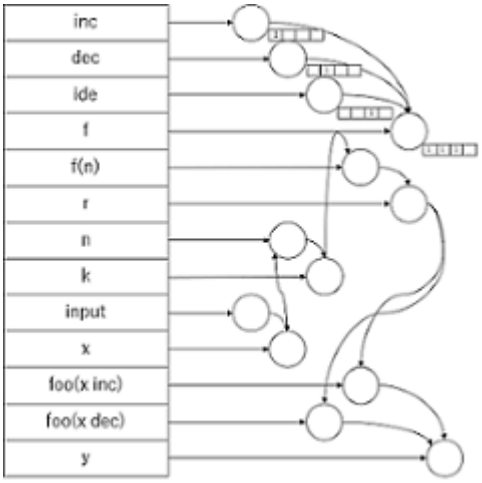

図 1 グラフの例

5 実装

これらのことを Racket で実装するにあたり制約の生成及び各処理に各関数を振り分け，表の状態記録にハッシュ表を使用した．ハッシュ表の構造は，キーにノード，中身にノード間の包含関係，ビット配列，ビットに付随するペアが格納されている．その他にも構造体のメンバーがリストで格納されて，マルチメゾット機能を使用した関数定義や通常の関数定義の引数名，返り値がハッシュ表で格納される．

6 まとめ

今回の研究ではマルチメソッド機能が使われている言語を解析するためのプログラムを作成した．今後の課題としては解析可能な関数をふやしより多くのプログラムを解析可能にすることである．

参考文献

[1] Static Program Analysis Anders Møller and Michael I. Schwartzbach, pp.105–111

[2] The Racket Reference. https://docs.racket-lang.org/reference/index.html