神奈川大学理学部　情報科学科　馬谷研究室

第一級継続に対するリフレクション機能の提案

佐藤 智隆 201703355

1　はじめに

Scheme の特徴的な機能である継続について研究を行う．継続とは Scheme 以外のプログラミング言語にも備わっているが，それをプログラマが明示的に操作することはできなくなっている．一般的に call-with-current-continuation（以下 call/cc と略）という組込み関数を使うことで継続を生成する．

本研究では継続に対するリフレクション機能を提案する．本来ユーザプログラムは言語処理系が読み取ることで処理され動作する．リフレクション機能の追加をすることによって言語処理系を利用せずに，continuation 型のデータに対する処理をユーザプログラムの中で処理を済ますことができるようになる．提案するリフレクション機能を使った応用例の一つとして，実行途中の処理を一時的にファイルに記憶しておく．記憶しておくことで，実行途中に何らかのトラブルにより中断された場合にその記憶した時点に戻すこと（チェックポインティング）ができる．

2　背景

継続に関する先行研究としては，一度処理を止めた後に，その止めた場所から任意のタイミングで処理を再開することができる，いわゆるコルーチンの実装 [2] がある．その他の先行研究としては，call/cc 式に future 構文を利用することで同時にプログラムを動かす継続の拡張 [3] に関する研究もある．

call/cc は関数であり，生成した継続を関数に渡し，その関数を実行するものである．関数を引数として受け取るので（call/cc ＜関数＞）の形で用いられる．実際には以下のように使用される．

```
(define x 0)
(+ 1 (call/cc (lambda (c)
                (begin (set! x c) 2))))
```

call/cc で使いたい x を最初に宣言しておく．x に継続である「1 を足す」を代入している．よって式全体は 3 になる．

継続データにリフレクション機能を追加するために，実際継続で何が行われてるかを理解する必要がある．そのため，継続渡し形式のインタプリタの概要として if 式を用いて説明する [1]．if 式では最初にテストを評価する．そしてテスト式の結果が真か偽の値であるかを判定し，真の式か偽の式のどちらかを評価する新しい継続を構築する．継続データである if-test-cont は以下のようになる．

```
(if-test-cont (exp2 exp3 s-env s-cont)
 (if (expval->bool val)
  (value-of/k exp2 s-env s-cont)
  (value-of/k exp3 s-env s-cont)))
```

この定義を例で説明する．

```
(if (zero? x)
 x
 (- x 1))
```

ここでは最初にテストである x が 0 であるかどうか (zero? x) を評価する．そして，その評価結果が if-test-cont では val として受け取っている．val は bool 値として与えられていないので expval-＞ bool を利用して未定義として仮定している．テストの評価結果が真の場合は (value-of/k exp2 s-env s-cont) を評価

する．ここでの exp2 は x となる．偽の場合は (value-of/k exp3 s-env s-cont) を評価する．ここでの exp3 は (- x 1) となる．if 式以外についても同様となっている．

3 リフレクション機能の設計

ユーザプログラムから動かせる call/cc 関数を利用し継続に，以降の計算のデータをファイルに一時的に保存するというリフレクション機能を追加する．

```
(define out
 (open-output-file "newfile"))
(call/cc
 (lambda (cont val)
  (cases continuation cont
   (if-test-cont
    (exp2 exp3 s-env s-cont)
     (begin
      (if ((bool-val (bool) bool))
       (display exp2 out)
       (display exp3 out))
     (display s-env out)
     (display s-cont out)))
    (diff1-cont (...))
    (...))))
(close-output-port out)
```

始めに以降の計算を保存するためのファイル newfile を指定する．call/cc 内で継続の種類の判別をする．判別する式としてはゼロテスト式, let 式, 条件式, 差分式, 手続き呼出しがある．さらにすべての計算を終了した後に必要な継続 (end-cont) を加える．継続の種類を判別するには条件判断 cases 文を利用する．それにより，各式で利用したい継続データ s-cont（saved-cont の省略）をその式の継続データとなるようにしている．

1 つの例として if-test-cont の継続データについて説明する．if-test-cont は条件を評価し終えた後に何をするかという継続なので，その条件が真であったか，偽であったかを評価する．真であった場合は exp2 を評価することになるため，exp2 を評価するために exp2，s-env，s-cont の 3 つを作成したファイルに保存する．exp2 が (+ x 1) のように変数が利用された場合に，保存した先での計算ができるように変数の値が何であるかを保存する必要があるため環境である s-env も保存する必要がある．(- (if <条件> (call/cc (...)) <#f の場合>) 1) のような場合に if 式だけではなく，外側の式の情報もファイルに保存しておく必要がある．そのため式の情報が入っている s-cont も保存する必要がある．

4 まとめ

本研究では継続データにリフレクション機能を追加した．このリフレクションの一つとしてプログラムが中断されてしまっても，保存した時点から残りの計算をする疑似的なチェックポインティングが可能となった．今後の課題としては対応する式の種類を増やすこと，リフレクション機能の欠点であるパフォーマンスのオーバーヘッドを減らすことである．

参考文献

[1] Daniel P. Friedman, Mitchell Wand, Essentials of Programming Languages third edition, The MIT Press, 2008.

[2] 湯浅 太一，Scheme 入門，1991.

[3] 小宮 常康，湯淺 太一，Future ベースの並列 Scheme における継続の拡張，情報処理学会論文誌，35 巻 11 号，pp.2382–2391，1994.