神奈川大学理学部　情報科学科　馬谷研究室

Redex 上の Scheme インタプリタにおける 抽象解釈のための継続のヒープ化

山本 怜 201803722

1　はじめに

本研究は, Racket 言語用ライブラリである Redex を使って書かれた Scheme インタプリタの継続データのヒープ化をすることを目的とする.

Redex とは，プログラミング言語，型システム，プログラム論理，プログラム解析の構想，設計，構築，テストをサポートするスクリプト言語および関連ツールのセットである．この機能を用いることで，言語，型，簡約関係を簡潔な記法で定義および実行できるようになる．Redex を用いて Scheme 言語用の抽象解釈器を構築する場合，以下の 3 つを実現する必要がある.

(1) Scheme 言語のサブセット [1] を Redex に実装し，実行できるようにすること.
(2) 実装したものの継続データをヒープ化すること.
(3) ヒープ上の継続データを有限化すること.

本研究では，(1) の研究 [2] を基に，これらのうち (2) に取り組む．(3) は後の研究 [3] で取り組む.

2　研究背景

本研究では，Scheme サブセットを Redex における define-language 形式で定義した SCM という言語名と，簡約関係 r についてヒープ化する.

ヒープとは，値（オブジェクト）を動的に割り当てるためのメモリ領域であり，アドレスから値へのマッピングのことである．スタック領域が処理したデータを下から順番に積み上げていくのに対し， ヒープ領域には順序がない． これにより任意のデータの確保，廃棄の効率が良くなる.

本研究では，Scheme 言語の サブセットを Redex 上で言語定義したもの [2] を前提とする.

3　言語定義

まず，継続データを作ることを目的とし，SCMs を定義する.

```
(define-extended-language SCMs SCM
  (F (v ... [] e ...) | (if0 [] e e))
  (C (F ⋯ ))
  (s (e C) | v)
```

F は，フレームのことで，評価コンテキストが内部にネストされていないものを表す．C は F が 0 個以上の集合であることを表す．また，継続と評価コンテキストが相互に変換可能であることも示している.

次に，変数を直接値にマッピングする代わりに，変数からアドレスへのマッピングとアドレスから値へのマッピングの二段階に分割するように修正した SCMt を定義する.

```
(define-extended-language SCMt SCM
  (ρ ((x A) ...))
  (Σ ((A v) ...))
  (t (s Σ) v))
```

ρ は，変数名からアドレスのマッピングを表し，Σ は，アドレスから値のマッピングを表す.

次に，継続データのヒープ化を表現するために，SCMt の拡張言語である SCMt* を定義する.

```
(define-extended-language SCMt* SCMt
```

```
(C () | (F A))
(Σ ((A U) ...))
(U v | C))
```

SCMt* での C は，SCMs を上書きしたもので，コンテキストクロージャごとの遷移規則を表す．SCMs の C を書き換えることによって，継続は空か，継続の残りの部分へのポインタを持つ単一フレームになる．このようにしてフレーム単位に分割した継続を，さらにヒープ上に配置することで継続のヒープ化が完了するが，その詳細については省略する．

4 簡約

まず，Scheme 言語 [1] で表された簡約規則の [MApp] を，Redex 上で表現する．

$$(\mathbf{store}((x_1\ v_1)\ \cdots)\ E[((\mathbf{lambda}\ (x_2\ \cdots)\ e)\ v_2 \cdots)]) \\ \rightarrow (\mathbf{store}((x_1\ v_1)\ \cdots (x_2\ v_2)\ \cdots) \\ E[\{x_2\ \cdots \mapsto x_2'\ \cdots\} e])$$

subst [4] メタ関数を用いて，x_2 を x_2' に置き換える簡約規則を作る．fresh 構文と side-condition を用いて，$x_2'\ \cdots$ がそれぞれ新しい非終端記号に置き換わることを表現する．また，$x_2\ \cdots$ の項の長さと，$x_2'\ \cdots$ の項の長さがそれぞれ等しいことを表現する．この SCM 言語での簡約規則を rSs とする．

次に，ヒープで束縛の割り当てと値の参照をするために，SCMs 言語 を用いて，簡約規則 rSs を Σ が入った文中で動作するようにした，拡張した簡約規則を rSt とする．これにより，[MApp] を，ヒープを経由したマッピングとして簡約できるようになる．

簡約規則 rSt は，変数束縛のためのアドレスを割り当てるヘルパーメタ関数 alloc に依存する．そこで，形式的なパラメータ名を抽出するためのメタ関数である formal を SCMt 言語で使えるように定義する．

次に，それぞれの形式パラメータ名に対して新しいアドレスを生成する alloc というメタ関数を定義する．アドレスは，束縛される変数名と，その特定の束縛のための固有のシンボルからなる 2 要素のリストである．これにより，具象マシンと近似マシンを関連付けることができる．

最後に，継続のヒープ化を表現するために，メタ関数 alloc を拡張した alloc* というメタ関数を定義する．継続的な遷移が行われようとしている項が与えられると，2 つの要素のリストが生成される．1 つ目の要素は割り当ての対象となるフレームで，2 つ目は固有のシンボルとなる．

5 まとめ

Scheme サブセットは Redex 上で束縛と継続を割り当てるヒープベースの機械として表現された．ただし，プログラムが終了しないかもしれない場合を考慮しなければいけないという課題がある．詳細はこれに続く研究 [3] で記述する．

参考文献

[1] Jacob Matthews, Robert Bruce Findler, An Operational Semantics for Scheme, Journal of Functional Programming, 18(1), 47–86, 2007.

[2] 武田雄一郎，Scheme サブセットの操作的意味論の Redex 上での実装，2021 年度神奈川大学理学部情報科学科卒研要旨集，2022.

[3] 森田翔大，Redex 上の Scheme インタプリタにおけるヒープの有限化による抽象解釈の実現，2021 年度神奈川大学理学部情報科学科卒研要旨集，2022.

[4] David Van Horn, An Introduction to Redex with Abstracting Abstract Machines (v0.6). https://dvanhorn.github.io/redex-aam-tutorial/