神奈川大学理学部　情報科学科　馬谷研究室

RPython を用いた Scheme サブセット処理系の構築

脇坂 海輝 201803405

1　はじめに

プログラムを高速化する方法はいくつか存在するが，その中でも大幅に高速化が見込める方法として，最適化処理の優れているコンパイラの作成が挙げられる．しかし，コンパイラを一から作成することは，膨大な時間を要する．そこで，プログラムを高速化する別の方法として，RPython を用いたインタプリタ作成に焦点を当てた．

最大の目的は，全ての言語機能を再現し，速度の上昇を目指すことである．しかし，すべての言語機能を再現することは現実的ではないため，小さなサブセットの構築から考える．

本研究では，RPython を用いて Scheme サブセット処理系の構築を目的とする．Scheme サブセットを選択した理由は，遷移規則に基づいた操作的意味論が厳密に定義されているため，そのインタプリタを RPython でも容易に記述できると考えたためである．さらに，実際に簡単な Scheme プログラムを実行し，Python と RPython の性能比較を比較する．

2　背景

RPython について説明する．RPython は Python のサブセットである．Just-in-Time（JIT）コンパイルという，実行時にコードのチャンクを実行時に逐一コンパイルするシステムを使って高速化される．RPython は Python にいくつかの制限を加えた言語であるため，Python で記述が可能なコードであっても，RPython では記述が可能ではないコードが存在する．特に，RPython の組み込み関数は非常に制限されている．

例として，引数で指定した集合の総和を計算する sum() は RPython ではサポートされていない．さらに，min() のようなサポートされている関数は，Python で同じ機能を提供しない．

```
print(min(1, 2)) #RPython
print(min([1, 2, 3, 4, 5])) #Python
```

どちらも最小値である 1 が出力される．しかし，RPython の min() 関数は 2 つの値しか比較できないが，Python ではリスト中の最小値を見つけるのに使うことができる．

3　設計

Scheme サブセット処理系の構築を行うために，Scheme の操作的意味論である遷移規則 [1] を以下に示す．

```
P ::= (store (sf...) E)

(store ((x1 v1) ... (x v)
   (x2 v2) ...) E[(set! x v')]) ->
(store ((x1 v1) ... (x v')
   (x2 v2) ...) E[unspecified])   [MSet]

P[(begin v e1 e2 ...)] ->
P[(begin e1 e2 ...)]              [MSeq]
```

文脈 P はホールを持つ式全体を表す．MSet は，左辺のような文法の形をとる場合，関連する値をストア内の指定された識別子を指定された置換文字列で置き換える規則である．MSeq は，少なくとも 2 つの部分式があるときだけ遷移可能で，begin の最初の部分式でのみ評価が可能である．

4 評価

MSet，MSeq の遷移規則で動作する RPython を実装する．
RPython コードの一部を以下に示す．

```
while True:
    sf.append(P[i])
    if P[i] == '(':
        b += 1
    if P[i] == ')':
        b -= 1
    if b == 0:
        break
    i += 1
```

式全体を示す文脈 P は (store (sf...) E) の形をとるため，メインループへの引数として P，sf，E をそれぞれのリストとしてを渡す．上記のコードを含むループを行うことで，sf，E のリストを作成する．

```
def beginC(E, pc):
i = pc
begin1 = []
while True:
    begin1.append(E[i])
    if E[i] == ')':
        return begin1
    i += 1
```

E 内の評価において，begin の識別子を発見した場合，MSeq により最初の部分式を評価する．関数 beginC はリスト E と，リスト E における最初の部分式の (を指す pc を引数とし，最初の部分式のみのリスト begin1 を返す．

```
while True:
  if sf[i] == x:
      sf[i+2] = v  #(x v)
      break
  i += 1
```

set!の識別子を発見した場合 Mset により，set!内の識別子と値を x と v に代入する．上記のループにより x と sf 内の識別子が一致した場合，x に対応する v を sf 内で置き換える．置き換えたリスト sf と，E から最初の分部式を取り除いたリストを組み合わせることで新たな式を得ることができる．また，新たな式全体，sf，E を引数とすることで再帰可能となる．

```
(store ((x 1)(y 2))
       (begin (set! x 3)(- x y)))
```

例として上記の式を入力として与えた場合，以下の結果が出力される．

```
(store ((x 3)(y 2))(begin(- x y)))
```

Python と RPython の実行時間に関しては，Python で 0.040s，RPython で 0.002s という結果となった．

5 まとめ

本研究では，RPython を用いて Scheme サブセット処理系の構築を行った．性能比較から，簡単なコードではあるものの高速化を達成できた．RPython での実行は他の処理系と比較しても，より高速になる可能性を秘めている．

参考文献

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

[2] Mingshen Sunand Baidu X-Lab, “RPython By Example”, http://mesapy.org/rpython-by-example/.