神奈川大学理学部　情報科学科　馬谷研究室

CPython 上で動作する命令型言語の RPython 上での実行

浅野 雄大 201803406

1　はじめに

Python はインタプリタ型のプログラミング言語として知られているが，その中でも RPython という Python のサブセットがある．RPython は，JIT(Just-in-Time) コンパイルという手法を用いて，Python のサブセットから高速な実行コードを生成する．しかし，RPython は Python にいくつかの制限を加えたサブセットであるため，CPython で記述が可能なコードであっても，RPython では記述が可能でないコードが存在する．インタプリタを書く上で，CPython と RPython の表現能力に差がある．RPython へ移植することで，RPython の記述力を調査する本研究では，2 節に背景について，3 節では，CPython 上で動作する命令型言語を紹介する．4 節では，研究の目的である命令型言語を RPython 上で実装し，5 節でまとめとして本研究の総括を行う．

2　背景

RPython が Python で記述できない例について説明する．文献 [3] の研究でも触れられているが，RPython の組み込み関数は，他の関数より制限されており，引数を指定して合計値を返す関数 sum や，リストの最小値を返す関数 min は Python と RPython とで返す実行結果が違う．例として文献 [2] の資料から以下のコードを抜粋した．

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

どちらも最小値 1 が出力される．ただし，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が，RPython の min() 関数は 2 つの値しか比較できない．文献 [3] の資料では，RPython を用いた小さい Scheme のサブセット処理系の構築を研究テーマとしている．1 から作ったため，含まれている言語機能はかなり少ない．Scheme の小さなサブセットでは RPython の表現力を完全に引き出されていないため，Scheme サブセットよりも大きいサイズのインタプリタを紹介する．

3　Inutoba インタプリタ

Inutoba インタプリタ [1] とは，インタプリタの基本的な動作を理解するために作られた教育用命令型言語である．一から理解できるように，if 文や while 文といった基本的な制御文も使われている．以下に while 文を使った Inutoba コードの簡単な例を示す．

```
i = 1
k = 0
while i <= 5
    k += f(i)
    i += 1
end
func f(x)
    k = x + 1
    m = 5
    return k + m
end
print "k = ", k
exit
```

この例では while 文で i が 5 回繰り返され，func 文で i の回数によって返される数が決まる．Inutoba インタプリタでは，end を for 文のブロックの末尾に必要で，exit は実行の終了を表す．

Inutoba インタプリタは，実行の際にプログラマが書いたソースコードからいったん中間コードを生成し，その中間コードを実行する仕組みになってい

る．動作の流れをつかむために文献 [1] の資料から Inutoba インタプリタのメインルーチンの一部を以下に示す．

```
...
print('__file__ = ', __file__)
try:
    toInterCode()
    getLocalVarSize()
    posChk()
    setStartEndAddr()
    returnBreakChk()
    ret = synChk()
    if not ret:
        sys.exit()
    execute()
except Exception as e:
    print(e)
    sys.exit()
...
```

すべてのメインルーチンは収まりきらないので一部を抜粋したが，プログラマは，この Inutoba のメインルーチンに自分で作ったコード (例の while 文など) を読み込ませると実行できる．

4 RPython への移植

この CPython 上で動かす Inutoba ソースコードを RPython で実装する．実際に CPython 上で動かす Inutoba ソースコードのメインルーチン (の一部) から RPython を移植したソースコードを下記に示す．

```
...
if not checkic(LParen):
        return False
    if len(argList) == 0:
        if not checkic(RParen):
            return False
...
```

関数内の一部で行われるコードが 0，ブール値 False を返す場合，通常 Python では false と見なされるため，RPython では true と見なされる条件式を if not にする Inutoba ソースコードのメインルーチン内の print 文は，括弧を外し「”」に変換する．

```
print ('__file__ = ', __file__) # CPython
print "__file__ = ",__file__    # Rpython
```

メインルーチン内にある print 文は RPython が認識できない文字列や記号，コメントは SyntaxError になってしまうため，エンコーディングするか，英数字で書き直す．RPytohn に移植して実装する際，コードの文末にターゲット関数を書く．コマンドライン引数は，target 関数で返す entryPoint 関数の引数に渡される．

```
...
def entry_point(argv):
      toInterCord()
      return 0
def target(*args): return entry_point
if __name__ == "__main__":
    import sys; entry_point(sys.argv)
    main()
```

__name__を使ったブロックは RPython では実行されないため，他の CPython インタプリタと互換性を維持できる．

5 まとめ

本研究では，CPython 上で動作する命令型言語の RPython 上での実装を行った．RPython を移植することで，Inutoba インタプリタの言語機能が制限されたことを確認した．

参考文献

[1] 吉田 節，Python でインタプリタを作る，株式会社インプレス R&D，2021.

[2] RPython-by-Example
https://mssun.github.io/rpython-by-example/index.html

[3] 脇坂 海輝，RPython を用いた Scheme サブセット処理系の構築，2021 年度 2021 年度神奈川大学理学部情報科学科卒研要旨集.