神奈川大学理学部　情報科学科　馬谷研究室

Lox 言語処理系における不要な pop 操作の除去

高森 隆 202103281

1　はじめに

Lox 言語の実行はスタックベースの仮想マシンを用いて行われる. スタックベースの仮想マシンは, オペランドにスタックをプッシュし, 演算結果を再びプッシュすることで計算を進める仕組みを採用している. しかし, スタック操作の効率が悪い場合, 計算のオーバーヘッドが大きくなることがある. 本研究では,Lox 仮想マシンのスタック操作における非効率性に着目し, 演算におけるスタックから値を取り出す pop() の使用回数を削減, また直接的なスタック参照に置き換えることで, スッタクポインタの移動を削減し, 計算効率を向上させることを目指す.

2　Lox 言語

Lox 言語とは, 教育目的のスクリプト言語で, 簡潔な構文と柔軟な設計を持つ点が特徴である. インタプリタは C 言語を用いた仮想マシンで動作する. 仮想マシンはスタックベースのアーキテクチャを採用しており, 計算やデータ操作がスタック上で行われている.

3　提案手法

Lox の仮想マシンでは, スタック上の値を取り出す際に pop() を使用し, 結果を push() を使用して, 再度スタックに戻すという計算方式を採用している. この手法では, スタックポインタが頻繁に移動するため, オーバーヘッドが発生してしまう. それなので,pop() を stackTop に書き換える, このようにすると,pop() の呼び出し回数を削減し, スタックポインタを動かさず, スタックトップの値を直接操作するようになり、スタック操作を効率化することができる.

4　実装

pop() を消去し,stackTop を用いて実装していく.

- 四則演算
  まず四則演算を行うための pop を除去し,pop だった部分を stackTop に変更する.

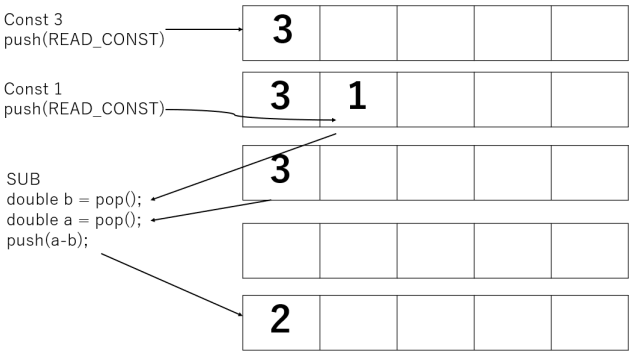

図 1　pop() を使用した計算

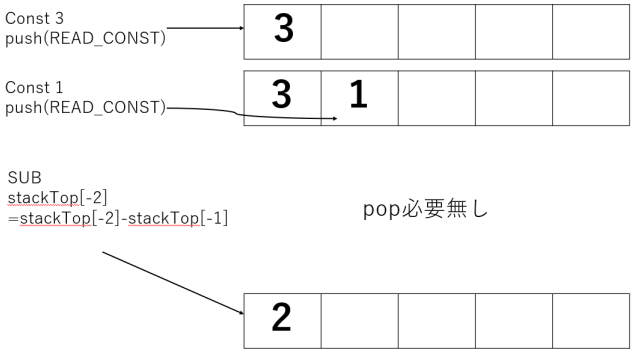

図 2　pop() を使用しない計算

変更前

```
#define BINARY_OP(valueType, op) \
do { \
if (!IS_NUMBER(peek(0)) ||
!IS_NUMBER(peek(1))) { \
    runtimeError
    ("Operands must be numbers."); \
    return INTERPRET_RUNTIME_ERROR; \
      } \
      double b = AS_NUMBER(pop()); \
      double a = AS_NUMBER(pop()); \
      push(valueType(a op b)); \
    } while (false)
```

変更後

```
#define BINARY_OP(valueType, op) \
 do { \
if (!IS_NUMBER(peek(0)) ||
!IS_NUMBER(peek(1))) { \
    runtimeError
    ("Operands must be numbers."); \
    return INTERPRET_RUNTIME_ERROR; \
  } \
 double b=
 AS_NUMBER(vm.stackTop[-1]);\
 double a=
 AS_NUMBER(vm.stackTop[-2]);\
 vm.stackTop--;\
 vm.stackTop[-1]=valueType(a op b);\
    } while (false)
```

- 正負逆転
  次に正負を逆転する際の pop を stackTop に変更する.

  変更前

```
case OP_NEGATE:
 if (!IS_NUMBER(peek(0))) {
 runtimeError
 ("Operand must be a number.");
    return INTERPRET_RUNTIME_ERROR;
        }
push(NUMBER_VAL(-AS_NUMBER(pop())));
     break;
```

  変更後

```
case OP_NEGATE:
if (!IS_NUMBER(peek(0))) {
runtimeError
("Operand must be a number.");
    return INTERPRET_RUNTIME_ERROR;
        }
vm.stackTop[-1] =
NUMBER_VAL
(-AS_NUMBER(vm.stackTop[-1]));
        break;
```

5 評価

実際に pop を使用したコードと pop を除去したコードを比較する. 比較するコードは四則演算のループ, 再帰的な階乗計算, ネストしたループの足し算の 3 つのコードを使用する.

- 変更前
  pop 数
  四則演算:9552 回　階乗:669 回　ネスト:11551 回
  push 数
  四則演算:9559 回　階乗:802 回　ネスト:11558 回
  時間
  合計:0.000638 秒
- 変更後
  pop() の回数
  四則演算:2048 回　階乗:231 回　ネスト:3247 回
  push() の回数
  四則演算:5557 回　階乗:552 回　ネスト:7406 回
  時間
  合計:0.000405 秒

pop を除去したため, 変更前と比べて pop() の回数が最大約 1/5 に減少,push() の回数は最大 4/5 に減少したため, 処理が軽減されたことがわかった.

6 まとめ

本研究では、Lox 仮想マシンにおける pop() の回数を削減することで, スタックポインタの移動を最小化し, 計算効率を向上させた.

評価の結果,pop() を除去する以前と比べ,pop() と push() の回数を大幅に削減し, 実行時間を短縮する効果が確認された. 今後の課題として, 他の命令に対する最適化や, 仮想マシン全体の効率向上に向けた研究が挙げられる.

参考文献

[1] Robert Nystrom, Crafting Interpreters, https://craftinginterpreters.com/.
https://members.tripod.com/e_luw/gakko/latex1/l_list.html.