

単純な並列プログラムに対する GUIを備えた可逆デバッガ

結縁・中澤研究室 101830125 近藤諒一

単純な並列プログラム

- 以下のように定義する

$\begin{aligned} P &::= DQR \mid DQ \text{ par } \{Q\}(\{Q\})^+ R \\ D &::= (\text{var } X;)^* \\ R &::= (\text{remove } X;)^* \\ Q &::= (S;)^* S \\ S &::= \text{skip} \mid X = E \mid \text{if } C \text{ then } Q \text{ else } Q \text{ fi} \mid \text{while } C \text{ do } Q \text{ od} \\ E &::= X \mid n \mid E \text{ op } E \mid (E) \\ C &::= B \mid C \&\& C \mid \text{not } C \mid (C) \\ B &::= E == E \mid E > E \\ (X : \text{変数 } n, \text{ op} : \{+, \times, -\}) \end{aligned}$
--

$()^+$ は1回以上, $()^*$ は0回以上の繰り返しを意味する

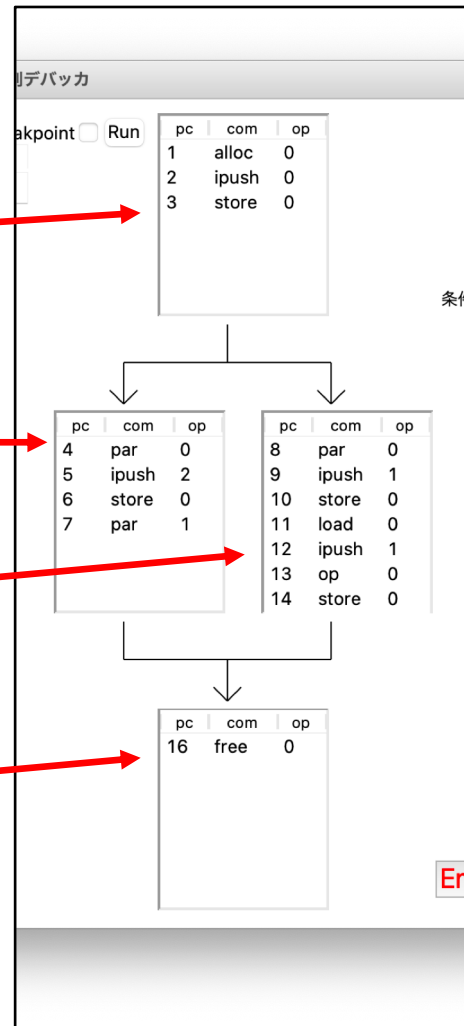
研究背景

- 並列プログラムのデバッグ
 - それぞれのプロセスが非同期的に実行を行う
 - リプレイをして同じ結果を得るのが困難
- 可逆デバッグ
 - 前向き実行時に履歴を保存する
 - 履歴から逆向き実行を行い，前向き実行の状態遷移を遡る方向に実行

抽象機械による前向き実行

プログラム

```
var x;  
x=0;  
par {  
  x=2;  
} {  
  x=1;  
  x=x+1;  
}  
remove x;
```

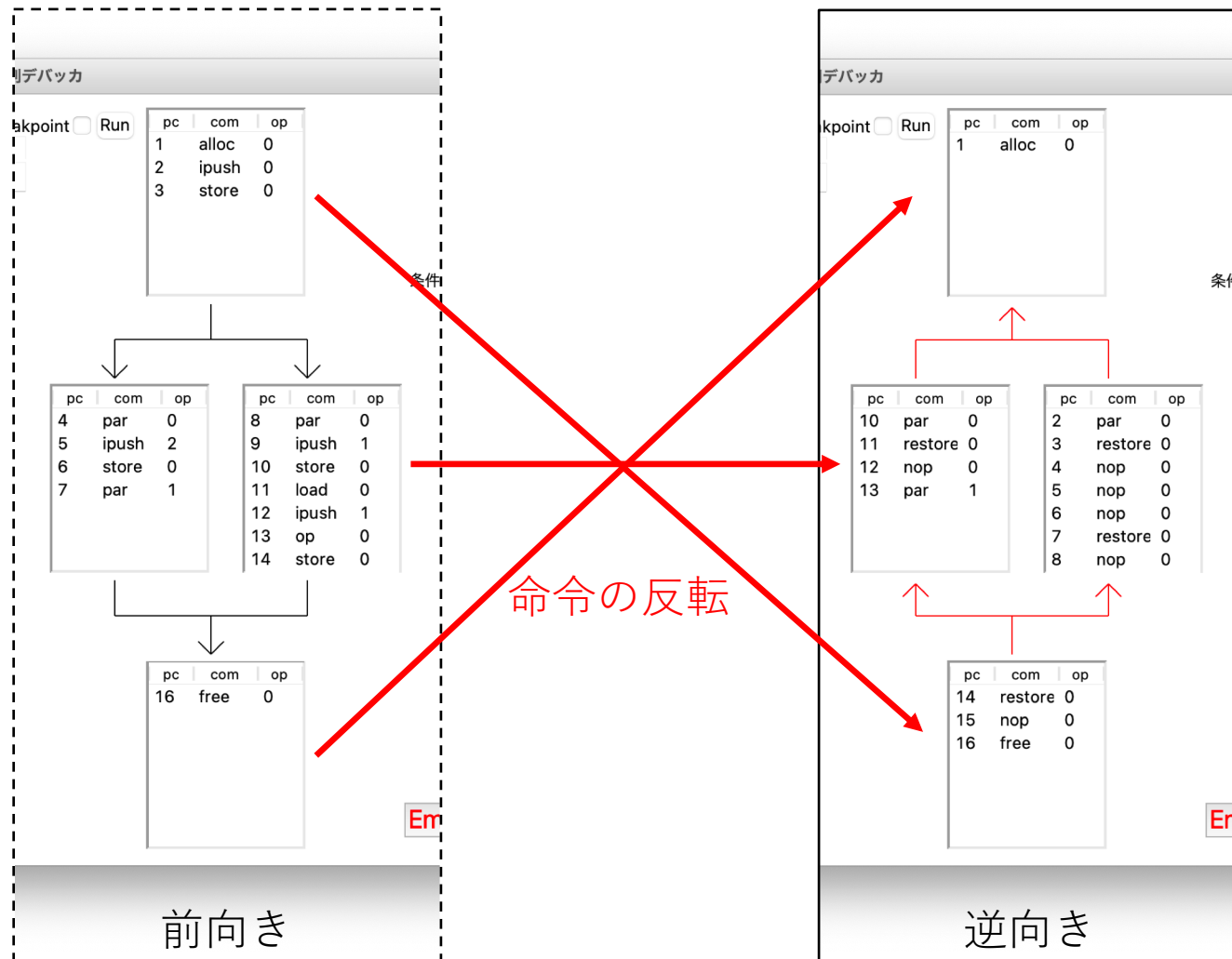


スタック

逆向き実行に
必要な情報

前向き実行の
際に保存

抽象機械による逆向き実行



スタック

逆向き実行に必要な情報

逆向き実行の際に使用

食事する哲学者問題(デッドロック)

哲学者は自分の都合でフォークを取る

ルール

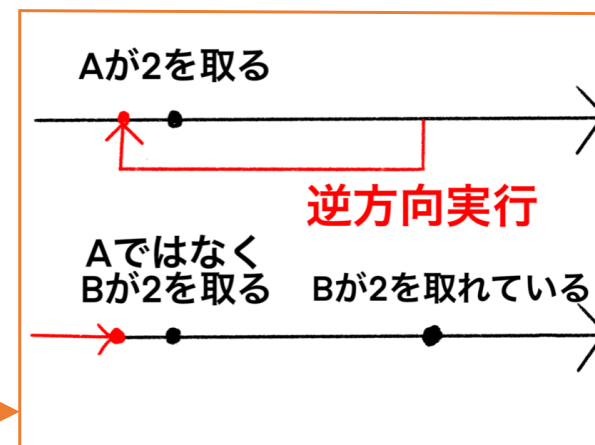
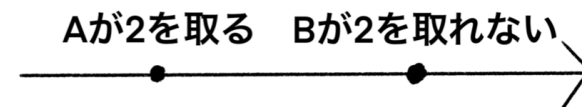
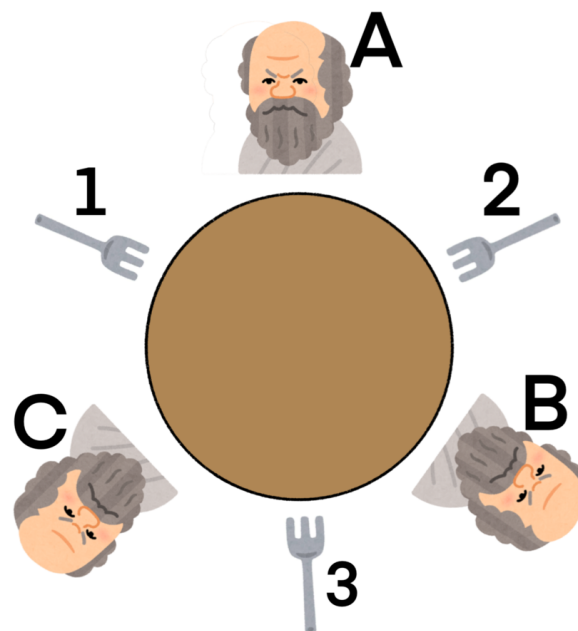
- ・ 食事には左右2本のフォークが必要
- ・ 同時に2本のフォークを取ることはできない
- ・ 自分の食事が終わるまでフォークを置かない

それぞれの哲学者が自分の右側のフォークを取った状態になり、それ以上動けなくなって飢え死にすることがある

もちろん、うまくいくこともある

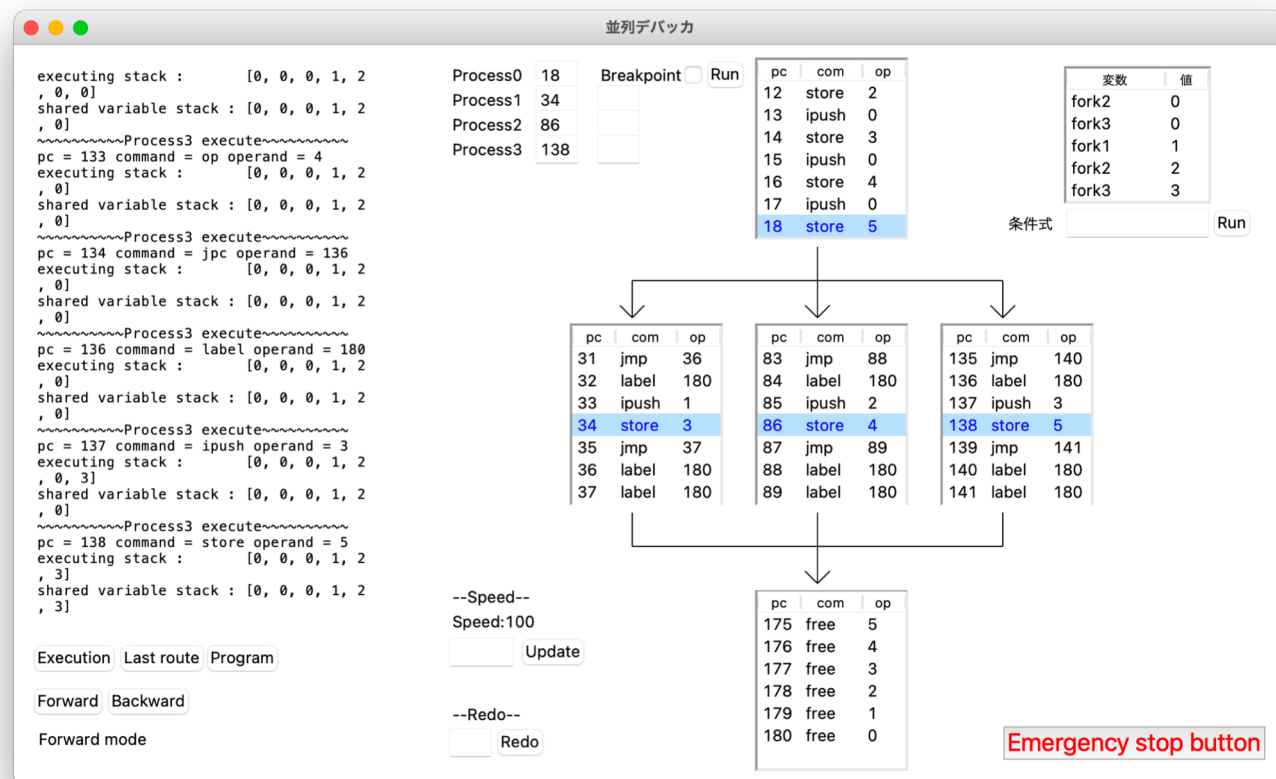
順方向実行だと、止まったらそこで終了して、もう一度やってみてもううまくいかないかも...

可逆実行なら、フォークを取る前に戻って問題点を分析できる



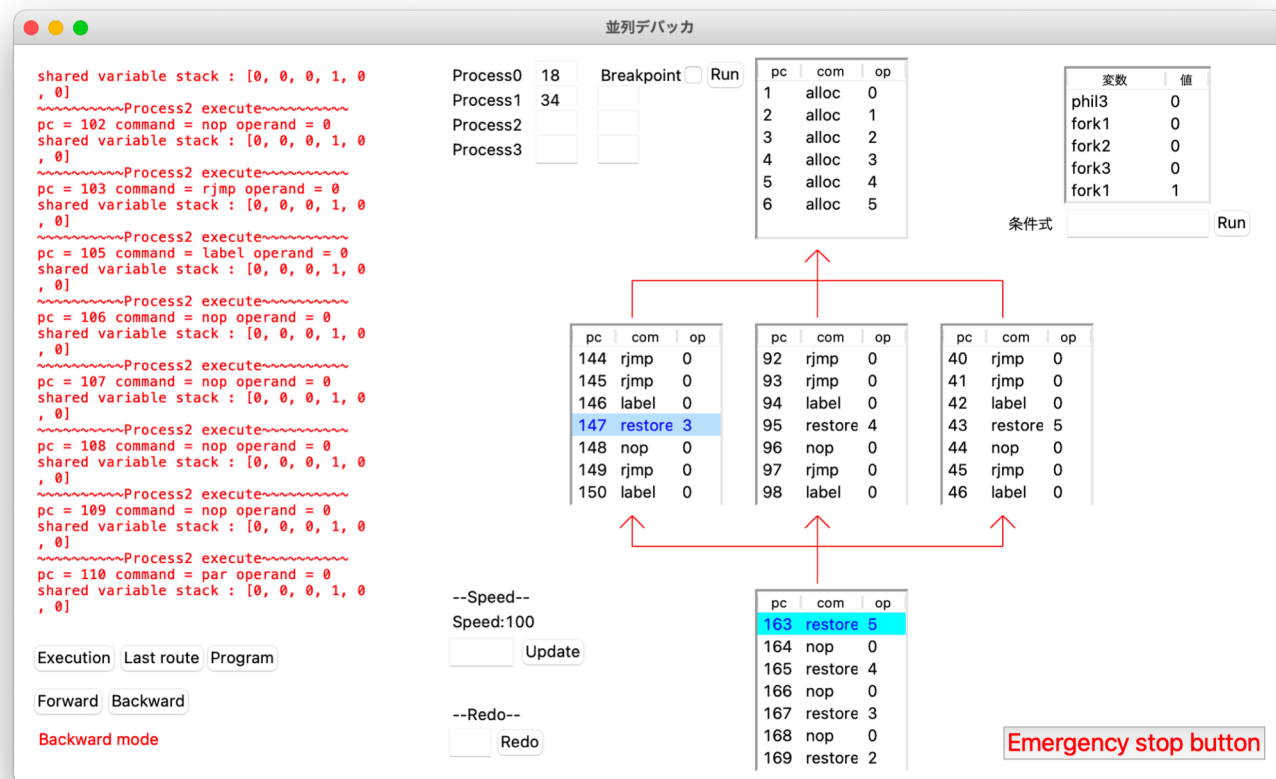
デッドロックの例

- fork1=1, fork2=2, fork3=3
- 哲学者たちがみんな片手にフォークを持っている状態
- 前向き実行を行ってもプログラムは終了しない



デッドロックの例

- 逆向き実行によってデッドロックが起こる前に遡る
- fork1=1
- ある1人の哲学者だけが片手にフォークを持っている状態
- プログラムを終了できるような手順が存在する



まとめ

- 研究成果
 - Pythonによる単純な並列プログラムに対する可逆デバッガーの実現
 - Pythonの標準ライブラリTkinterを利用してGUIを作成