修論発表

CbC インターフェースによる CbCXv6 の書き換え

並列信頼研
桃原 優

概要

- OS の信頼性を上げたい
- CbCという言語を使って xv6 を書き換える事で信頼性を保証
- インターフェースを使って OS の信頼性で重要なメモリ管理部分の書き換えを行った
- 将来的にはこのOSでコンテナやVMを実装できると考えている

OS の信頼性を上げたい

- OS自体が複雑で検証が困難。
  - 様々なバグの原因に
- OSの仕様を定義してそれを満たすことを証明して信頼性を保証

- 証明しやすい形の記述を作る

- CbCの goto で書く
  - 状態遷移ベースで記述できる(証明しやすい)
  - 入力に対しての出力を検証する

ノーマルレベルとメタレベル

- ノーマルレベル
  - CbCによる基本的な記述

- メタレベル
  - メモリやCPU自体の操作
  - ほとんどの OS では system call で操作する部分
  - メタレベルから(ノーマルレベルの)プログラムの正しさの証明をする

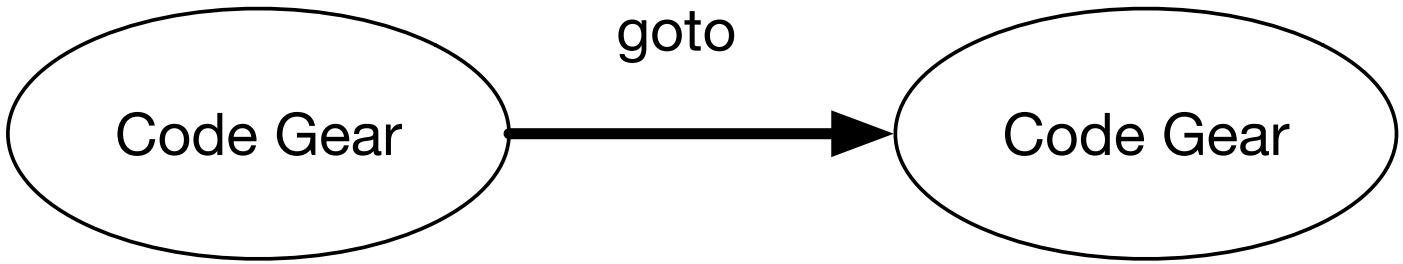
処理とデータの単位

- ノーマルレベルとメタレベルそれぞれに処理の単位とデータの単位がある
- ノーマルレベル
  - Code Gear(処理)
  - Data Gear(データ)
- メタレベル
  - Meta Code Gear(処理)
  - Meta Data Gear(データ)

ノーマルレベル

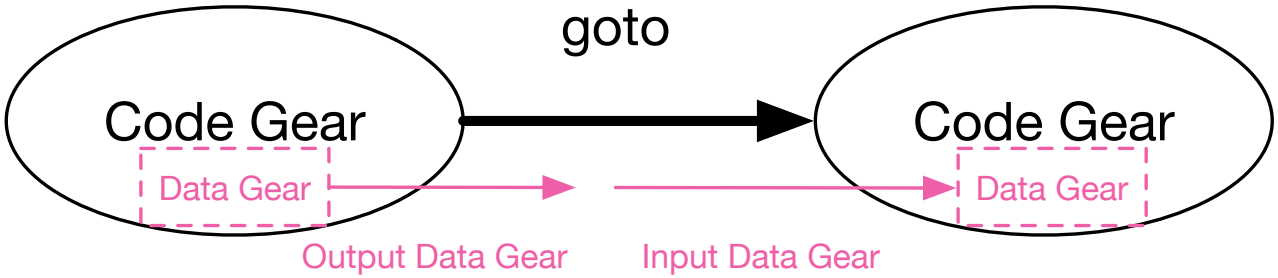
Code Gear

- 基本的な処理の単位
- Code Gear の処理の間を goto によって遷移していく



Data Gear

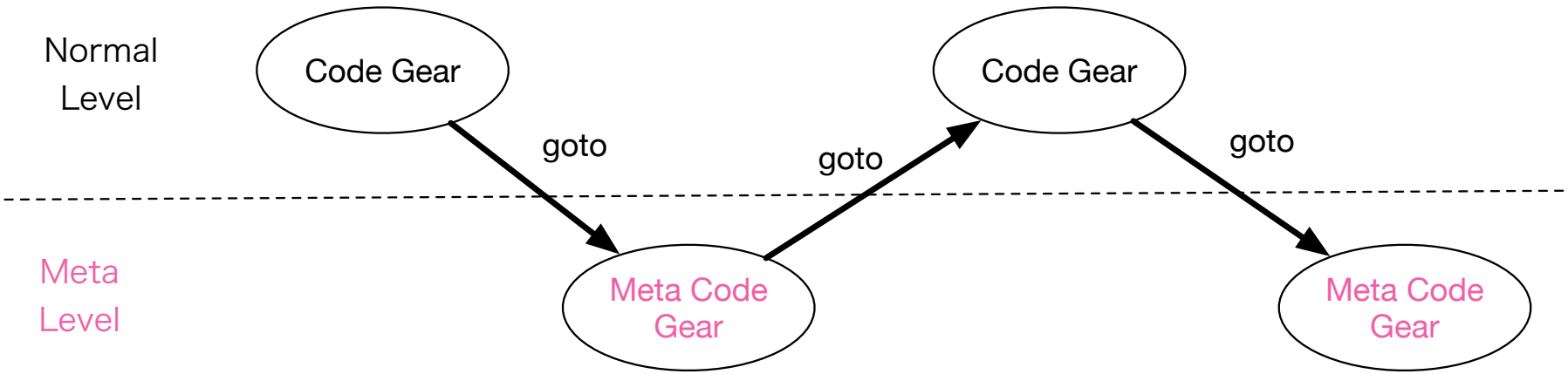
- Code Gear からアクセスできるデータ(引数等)
- input Data Gear に対する Output Data Gear で検証する



メタレベル

Meta Code Gear

- メタレベルで見ると Code Gear の間にメタレベルの処理が挟まっている
- cmake で自動生成している



Meta Data Gear

- ノーマルレベルでの書き換えやアクセスを防ぐために存在
- Code Gear, Data Gear のリスト
- Data Gear を確保するメモリ空間
- Context

Context

- Meta Data Gear
- Contextには全てのData Gear と Code Gear が登録されている
- 1つのユーザープロセスに対応して1つのcontextが存在する
- Contextの切り替えによる実行環境の入れ替えでVMやコンテナの実装を目指している

xv6

- MIT の講義用教材として作られた
  - 1万行程の軽量なOS
- xv6 をCbCで書き換える

xv6の書き換え方針

- メタレベルからノーマルレベルを保証するOSを作りたい
- 段階的に書き換えていきたい
  - Paging から書き換える理由
    - OS の信頼性を保証する上で重要なメモリ管理部分

インターフェースの導入

- ノーマルレベルとメタレベルでデータの見え方が異なり、CbC 記述が煩雑になるためインターフェースを導入
- 入力の切り替えによる実装の入れ替えができる
- インターフェースを定義してそこから呼び出す

ポスター

- 実際のインターフェースの定義や実装
- CbC での 状態遷移ベースの記述
- Raspberry Pi での xv6 デモ