xv6 kernel 上での CbC による interface の実装
CbC interface implementation in xv6 kernel

学籍番号：165723C 氏名：坂本昂弘 指導教員：河野真治

2019 年 10 月 23 日

1 xv6 の OS の信頼性保証

現代では OS の信頼性の保証と拡張性の実現が求められている. 信頼性をノーマルレベルの計算に対して保 証し，拡張性をメタレベルの計算で実現することを目標に Continuation based C (CbC) を用いて Gears OS を設計中である. 前段階としてシンプルであるが基本的な機能を揃えた OS である xv6 を CbC で書き換えようとしている.

CbC には CodeGear という処理の基本的な単位があり, これらを利用する事によって stack に値を積む事なく CodeGear 間を遷移することができる. stack が無い事によって OS 内部の明確化が実現できる. さらに CbC は 定理証明支援系 Agda に置き換えることができるように構築されている.Agda を利用する事により,CbC の関数などが正常な動きをしているのかどうかを判断することが可能となる. CbC の Interface は Gears OS のモジュール化の仕組みである. この Interface は,Java の Interface や Haskell の型クラスに対応し, 導入することで仕様と実装に分けて記述することが出来る. そのため,CbC の Interface を使うとモジュール化ができる.

これらのことを用い,CbC の Interface を xv6 に搭載する事により kernel の中の処理を明確にし, OS の信頼性を保証したい. また,xv6 の Interface 実装によりモジュール化が可能とし, 拡張性の実現を目指す.

2 xv6

xv6[1] とは MIT のオペレーティングコースの教育目的で 2006 年に開発されたオペレーティングシステムである. xv6 はオリジナルである v6 が非常に古い C 言語で書かれている為, ANSI-C に書き換えられ x86 に再実装された. xv6 は read や write などの systemcall, プロセス, 仮想メモリ, カーネルとユーザーの分離, 割り込み, ファイルシステムなど Unix の基本的な構造を持っている.

3 Continuation based C

xv6 kernel 上で interface を実装する際, 当研究室で開発されたプログラミング言語 Continuation based C (CbC) を用いる. CbC は基本的な処理単位を CodeGear として定義し，CodeGea 間で遷移するようにプログラムを記述する C 言語と互換性のあるプログラミング言語である. CodeGear は返り値を持たない為, 関数内で処理が終了すると呼び出し元の関数に戻ることがなく別の CodeGear へ遷移する. 以下の Code1 に CodeGear 遷移時のコード例を示す.

```
__code cg0(Integer a, Integer b){
 int a_v = a->value;
 int b_v = b->value;
 Integer c = {a_v + b_v};
  goto cg1(c);
}
__code cg1(Integer c){
  goto cg2(c);
}
```

Code 1: CodeGear の継続の例

また CbC における CodeGear 間の継続にはスタックが使用できず, 呼び出し元の環境などを持たない為軽量継続と呼ぶ. 現在 CbC は C コンパイラである GCC 及び LLVM をバックエンドとした clang 上で実装されている.

4 CbC と Agda の対応

Agda とは定理証明を支援する関数型のプログラミング言語である. 数学の証明に使われる自然演繹と型付き λ 計算については対応があり,Agda では型付き λ 計算を記述する事によって検証することができる.

CbC では継続を用いてプログラムを記述する. 同じく Agda にも継続の記述方式が存在する. これらの記述は対応している. 以上のことより CbC を Agda に書き換える事ができ,CbC で記述されたプログラムは Agda によって検証することが可能である.

5 context

context とは一連の実行が行われる際に使用される CodeGear と DataGear の集合である. 従来のスレッドやプロセスに対応する.Context は接続可能な CodeGear, Data Gear のリスト. Data Gear を確保するメモリ空間, 実行される Task への Code Gear 等を持っている. CodeGear が別の CodeGear に遷移する際, 必ず context を参照し enum

で定義された CodeGear の番号を指定し遷移する. ノーマルレベルで見た際の CodeGar,DataGer および context の関係を以下の図 1 に簡潔に示す.

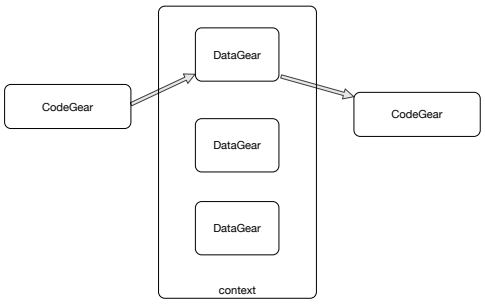

図 1: CodeGear,DataGear,contxt の関係図

6 CbC の Interface

先述した通り, CbC の Interface は Gears OS のモジュール化の仕組みである. Interface は呼び出しの引数になる Data Gear の集合であり，そこで呼び出される Code Gear のエントリである. 呼び出される Code Gear の引数となる Data Gear はここで 全て定義される. Interface を定義することで複数の実装を持つことができる. この Interface は,Java の Interface や Haskell の型クラスに対応し, 導入することで仕様と実装に分けて記述することが出来る.

以下の図 2 は, Queue の Interface と SingleLinkedQueue の実装例である.

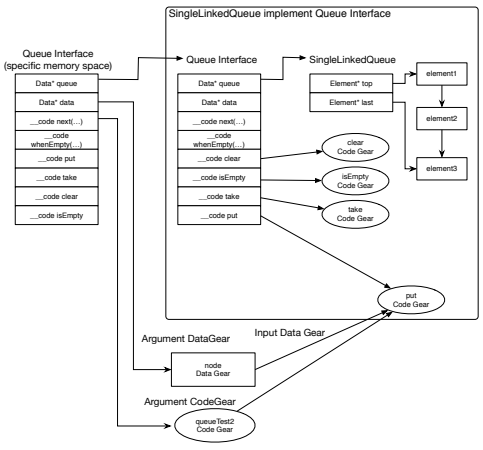

図 2: Queue の Interface と SingleLinkedQueue の実装例

7 xv6 の interface

先述した CbC の Interface を xv6 の Interface に実装する事によって仕様と実装を分けて記述できる.

先行研究 [2] では xv6 の kernel の一部である syscall の部分を CbC によって書き換えされている. それを元に xv6 の kernel 部のどの部分 Interface をし, 実装するべきかを考察した. 以下の図 3 では, syscall のさらに一部分である sys_read を Interface を用いて実装するとどうなるか考慮したものを簡潔に表した.

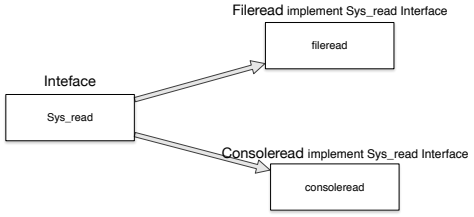

図 3: sys_read の Interface とその実装の考案

xv6 の kernel 部がどのような動きをし, 処理はどんな手順を踏むのかを考慮しながら xv6 に Interface を実装していく.

8 今後の課題

先行研究 [2] によって,xv6 の kernel の一部を CbC の特徴でもある継続を用いモジュール化された. しかし現在では, xv6 に interface はまだ搭載されていない. そのため今後, xv6 の kernel を interface を用いる事によってどのようにモジュール化できるか Interface とその実装を考察しながら CbC の interface を xv6 に実装していく必要がある. また,CbC の Intterface を利用しているため Agda を用いて検証することが可能であるため, 実装が正しい実装なのか検証することも求められる.

参考文献

[1] Xv6, a simple Unix-like teaching operating system. https://pdos.csail.mit.edu/6.828/2019/xv6.html. (2019 年 10 月 19 日閲覧).

[2] 宮城光希, 河野真治. 継続を基本とした言語による os のモジュール化. 琉球大学工学部情報工学科平成 31 年度学位論文 (修士), 2019.

[3] 伊波立樹, 河野真治. Gears os の並列処理. 琉球大学工学部情報工学科平成 30 年度学位論文 (修士), 2018.

[4] Hokama MASATAKA and Shinji KONO. Gearsos の hoare logic をベースにした検証手法. ソフトウェアサイエンス研究会, Jan 2019.

[5] Haogang Chen, Daniel Ziegler, Tej Chajed, Adam Chlipala, M. Frans Kaashoek, and Nickolai Zeldovich. Using crash hoare logic for certifying the fscq file sys-

tem. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pp. 18–37, New York, NY, USA, 2015. ACM.