Continuation based C による赤黒木の Hoare Logic を用いた検証
Verification of red-black tree implemented in Continuation based C using Hoare Logic

学籍番号 175706H 氏名 上地 悠斗
指導教員 : 河野 真治

要 旨

当研究室にて Continuation based C (以下 CbC) なる C 言語の下位言語に当たる言語を開発している。外間による先行研究にて Floyd-Hoare Logic(以下 Hoare Logic) を用いてその検証を行なった。本稿では、先行研究にて実施されなかった CbC における赤黒木の検証を Hoare Logic を用いて検証することを目指す。

We are developing a language called Continuation based C (CbC), which is a Subordinate language of the C. M.Eng Hokama verified it by using Floyd-Hoare Logic (Hoare Logic) in a previous study. In this paper, we aim to use Hoare Logic to validate the red-black tree in CbC, which was not performed in previous studies.

1 研究目的

OS やアプリケーションの信頼性を高めることは重要な課題である。信頼性を高める為には仕様を満たしたプログラムが実装されていることを検証する必要がある。具体的には「モデル検査」や「定理証明」などが検証手法として挙げられる。

研究室で CbC という言語を開発している。CbC とは、C 言語からループ制御構造とサブルーチンコールを取り除き、継続を導入した C 言語の下位言語である。この言語の信用性を検証したい。

仕様に合った実装を実施していることの検証手法として Hoare Logic が知られている。Hoare Logic は事前条件が成り立っているときにある計算 (以下コマンド) を実行した後に、に事後条件が成り立つことでコマンドの検証を行う。

CbC の実行を継続するという性質が Hoare Logic の事前条件と事後条件の定義から検証を行うことと非常に相性が良い。これらのことから、本稿では Hoare Logic を用いて CbC を検証することを目指す。

2 Continuation based C

前述した通り CbC とは C 言語からループ制御構造とサブルーチンコールを取り除き、継続を導入した C 言語の下位言語である。継続呼び出しは引数付き goto 文で表現される。また、CodeGear を処理の単位、DataGear をデータの単位として記述するプログラミング言語である。CbC のプログラミングでは DataGear を CodeGear で変更し、その変更を次の CodeGear に渡して処理を行う。

2.1 Code Gear / Data Gear

CbC では、検証しやすいプログラムの単位として DataGear と CodeGear という単位を用いるプログラミングスタイルを提案している。

CodeGear はプログラムの処理そのものであり、一般的なプログラム言語における関数と同じ役割である。DataGear は CodeGear で扱うデータの単位であり、処理に必要なデータである。CodeGear の入力となる DataGear を Input DataGear と呼び、出力は Output DataGear と呼ぶ。

CodeGear 間の移動は継続を用いて行われる。継続は関数呼び出しとは異なり、呼び出した後に元のコードに戻らず、次の CodeGear へ継続を行う。これは、関数型プログラミングでは末尾関数呼び出しを行うことに相当する。

2.2 Meta Code Gear / Meta Data Gear

プログラムの記述する際は、ノーマルレベルの計算の他に、メモリ管理、スレッド管理、資源管理等を記述しなければならない処理が存在する。これらの計算はノーマルレベルの計算と区別してメタ計算と呼ぶ。

メタ計算は OS の機能を通して処理することが多く、信頼性の高い記述が求められる。そのため、CbC ではメタ計算を分離するために Meta CodeGear、Meta DataGear を定義している。

Meta CodeGear は CbC 上でのメタ計算で、通常の CodeGear を実行する際に必要なメタ計算を分離するための単位である。CodeGear を実行する前後や DataGear の大枠として Meta Gear が存在している。

例として CodeGear が DataGear から値を取得する際に使われる、stub CodeGear について説明する。

CbC では CodeGear を実行する際、ノーマルレベルの計算からは見えないが必要な DataGear を Context と呼ばれる Meta DataGear を通して取得することになる。これはユーザーが直接データを扱える状態では信頼性が高いとは言えないと考えるからである。そのために、Meta CodeGear として Context から必要な DataGear を取り出し、CodeGear に接続する stub CodeGear という Meta CodeGear を定義している。

Meta DataGear は CbC 上のメタ計算で扱われる DataGear である。例えば stub CodeGear では Context と呼ばれる接続可能な CodeGear、DataGear のリストや、DataGear のメモリ空間等を持った Meta DataGear を扱っている。

3 Hoare Logic

Hoare Logic とは C.A.R Hoare、R.W Floyd が考案したプログラムの検証の手法である。これは、「プログラムの事前条件 (P) が成立しているとき、コマンド (C) 実行して停止すると事後条件 (Q) が成り立つ」というもので、CbC の実行を継続するという性質に非常に相性が良い。Hoare Logic を表記すると以下のようになる。P C Q この 3 つ組は Hoare Triple と呼ばれる。

Hoare Triple の事後条件を受け取り異なる条件を返す別の Hoare Triple を繋げることでプログラムを記述していく。

Hoare Logic の検証では、「条件がすべて正しく接続されている」かつ「コマンドが停止する」ことが必要である。これらを満たし、事前条件から事後条件を導けることを検証することで Hoare Logic の健全性を示すことができる。

4 agda

agda とは、Agda は依存型をもつ純粋関数型の言語である。定理証明支援器でもある。

5 検証手法

手法は模索中であり、先行研究と同じ手法を取ろうとしている。本章では先行研究で述べられている検証手法について説明する。

5.1 CbC 記法で書く agda

CbC プログラムの検証をするに当たり、agda コードも CbC 記法で記述を行う。つまり継続渡しを用いて記述する必要がある。以下が例となるコードである。前述した加算を行うコードと比較すると、不定の型 (t) により継続を行なっている部分が見える。これが Agda で表現された CodeGear となる。

5.2 agda による Meta Gears

通常の Meta Gears はノーマルレベルの CodeGear、DataGear では扱えないメタレベルの計算を扱う単位である。Meta DataGear はメタ計算で使われる DataGear で、実行するメタ計算によって異なる。今回はその Meta Gears を agda による検証の為に用いる。検証での Meta Gears は DataGear が持つ同値関係や、大小関係などの関係を表す DataGear がそれに当たると考えられる。

5.3 agda における Meta DataGear

Agda 上で Meta DataGear を持つことでデータ構造自体が関係を持つデータを作ることができる。以下が While Program での制約条件をまとめたものになる。

Listing 1: Agda における Meta DataGear

```
data whileTestState  : Set where
  s1 : whileTestState
  s2 : whileTestState
  sf : whileTestState


whileTestStateP : whileTestState → Envc →  Set
whileTestStateP s1 env = (vari env ≡ 0) ∧ (varn env ≡ c10 env)
whileTestStateP s2 env = (varn env + vari env ≡ c10 env)
whileTestStateP sf env = (vari env ≡ c10 env)
```

whileTestState で Meta DataGear を識別するためのデータを分け、whileTestStateP でそれぞれの Meta DataGear を返している。ここでは = の後ろの (vari env ≡ 0) (varn env ≡c10 env)/ などのデータを Meta DataGear として扱う。

5.4 agda における Meta CodeGear

Meta CodeGear は 通常の CodeGear では扱えないメタレベルの計算を扱う CodeGear である。Agda での Meta CodeGear は Meta DataGear を引数に取りそれらの関係を返す CodeGear である。メタ計算で検証を行う際の Meta CodeGear は Agda で記述した CodeGear の検証そのものである。例として ソースコード 5.3 を示す。

Listing 2: Agda における Meta CodeGear

```
whileTestPwP : {l : Level} {t : Set l} → (c10 : ℕ) →
          ((env : Envc ) → (mdg : (vari env ≡ 0) ∧ (varn env ≡ c10 env)) →
       t) → t
whileTestPwP c10 next = next env record { pi1 = refl ; pi2 = refl } where
  env : Envc
  env = whileTestP c10 ( λ env → env )
```

6 今後の課題

7 類似技術

7.1 coq

References

[1] CbC の論文

[2] 外間先輩の先行研究

[3] Hoare Logic の論文

[4] Hoare Logic のスライド

[5] agda のサイト

[6] Aaron Stump の本

[7] attton さんの論文

[8] Haskell

[9] Coq