定理証明支援系 Coq のプログラムからの D 言語プログラムの抽出

1111200　山田 伊織　　中野研究室

1　背景

Coq [1] とは、定理証明支援系と呼ばれる、数学の定理証明を機械的に検査するシステムの一つである。数学の証明だけでなく、Coq に含まれるプログラミング言語を用いてプログラムを作成し、その性質を証明することで、動作の保証されたプログラムの作成も可能である。例として、C 言語のコンパイラ CompCert [3] は Coq によって開発されており、最適化を行っても正しくコンパイルすることが保証されている。Coq を用いてプログラムの動作を保証することは、プログラムを実行せず、また無限の入力に対して行えるため、信頼性が強く求められるプログラムの作成において特に有用である。

Coq が提供するプログラム抽出機能を用いれば、Coq で記述したプログラムを Coq 外の言語で利用することができる。プログラム抽出機能とは、Coq のプログラムから他言語のコードを生成する機能であり、標準では OCaml, Haskell, Scheme の 3 言語に対応している。抽出機能はプラグインによるものであるため、本体に手を加えることなく抽出する言語を追加することができる。実際、ユーザが作成した他言語への抽出プログラムが複数公開されている。Coq により信頼性が保証されたプログラムを他言語に抽出することで、抽出先の言語においても信頼性の高いプログラムが得られる。

一方、プログラム抽出は機械的に行われるため、抽出先の言語の機能を十分に活用できず、直接その言語で書かれたプログラムに比べ実行効率の低いプログラムになってしまう。しかし、抽出したプログラムに手を加えて高速化すると、書き換えたプログラムだけでなく、そのプログラムを利用するプログラムも動作が保証されなくなる。場合によっては、書き換えたプログラムそのものはエラーを起こさず、それを利用するプログラムでエラーが発生することで、原因の発見が困難になることもある。書き換え後のプログラムが保証がされないのは、保証済のプログラムは正しく動作することを前提としており、抽出後にチェックを行わないためである。

2　目的と方針

本研究では、書き換えても信頼性が失われないコードを抽出するよう Coq を拡張することを目的とする。

そのための方針として、既存の抽出機能に対しては拡張は行わず、D 言語 [4] への抽出機能を追加する。D 言語は C 言語をベースとした汎用プログラミング言語で、現在 Coq からの抽出機能は存在しない。D 言語は契約と呼ばれる実行時チェック機能を標準でサポートしいるため、これを用いて動作を保証することで目的を達成する。

3　契約

契約 [2] とは関数の入出力を実行時にチェックする機能である。例えば、以下の D 言語プログラムにおける関数 f,g は、引数 x が 0 以上の数であるという条件が、in 節によって課せられている。同様に、返り値 (r) の値が引数 x 以上であるという条件が、out 節によって課せられている。なお、body 節は関数本体を表す。

```
int f(int x)
  in{assert(x >= 0);}
  out(r){assert(r >= x);}
  body{ return x * 2; }
int g(int x)
  in{assert(x >= 0);}
  out(r){assert(r >= x);}
  body{ return x - 1; }
```

このとき、引数の 2 倍を返す関数 f は（in 節の条件が守られているとき）問題なく実行できるが、引数から 1 を引く関数 g は out 節の条件に違反するため実行時にエラーになる。このように契約を用いることで、入力や出力が持つべき条件を記述し、条件に反する場合には動作を止めることができる。

Coq でプログラムを保証する際に関数に求められる条件を契約として抽出することで、抽出後のコードの書き換えが可能になる。書き換え後のプログラムが契約に反する動作をした場合は契約がエラーにするため、動作が保証される。

4　D 言語のその他特徴

4.1　幅広い用途

D 言語は豊富な機能と標準ライブラリを持つため、ゲームから大規模データ処理まで幅広い用途に活用可能されている。また、D 言語の公式コンパイラ DMD [5] 自体も D 言語で作成されている。

このため、D 言語への抽出機能を Coq に追加することで、Coq の活用範囲を広げることが期待できる。

4.2　テンプレート

テンプレートとは、特定のパターンを持ったコードを簡潔に記述するための機能である。関数と異なり型を引数としてとることができるため、配列の処理のような型を選ばない関数を記述する際に多く用いられる。

Coq と D 言語の構文は大きく異なるため、テンプレートを用いてその差異を吸収することにより、抽出されたコードの可読性が向上、書き換えが容易になることが期待できる。

4.3 高効率

D 言語はコンパイルすることでネイティブコードを生成できるため、多くの場合 Python のようなインタプリタ言語や Java のような VM を用いる言語に対して高速に動作する。

また、インラインアセンブラなど、より高速なプログラムを作成するための機能も備えている。Coq から抽出したプログラムを書き換える際、これらの機能を用いることにより大きな効果が期待できる。

5 プログラムの抽出例

次のような Coq プログラムを考える。

```
Program Fixpoint drop A (l:list A)
  (n:{n:nat | n <= length l}):
  {r:list A | length r = length l - n} :=
match n with
  | O => l
  | S m => match l with
    | nil => l
    | _::l' => drop l' m
  end
end.
```

関数 drop は型変数 A、A 型のリスト l、自然数 n を引数として受け取り、l の先頭から n 個の要素を削除したリストを返す。ここで、n は l の要素数以下であり、返すリストの長さは l の長さより n だけ小さいという条件が付けられている。なお、drop に現れる match はパターンマッチと呼ばれる構文で、場合分けと変数への束縛を同時に行う。今回の場合では、n が 0 であるか m+1（m は任意の自然数）であるかで場合分けされている。

このような関数を D 言語に抽出した場合、以下のようなコードを出力することが期待される。

```
A[] drop(A)(A[] l, int n)
  in{assert(n >= 0 && n <= l.length);}
  out(r){assert(r.length == l.length - n);}
  body{
    switch(n){
      case 0: return l;
      default: int m = n-1;
      switch(l.length){
        case 0: return l;
        default: A[] l2 = l[1..$];
        return drop(l2,m);
      }
    }
  }
```

これは関数を出力するテンプレートになっている。D 言語では型を引数として受け取ることはできないため、テンプレートによって型変数 A を扱う。2 行目および 3 行目は契約を表す。in 節では n が l の長さ以下であることに加え、n が 0 以上であることも条件にしている。これは、D 言語では自然数を表す型がないためであり、この条件を追加することで D 言語が持つ int 型で Coq の nat 型を表すことができる。また、match に相当する構文は switch による場合分けと変数の初期化を別々に行っている。

抽出後の drop は Coq の定義に比べ記述量が増えているが、必要であれば抽出後に以下のように簡単に書き換えることができる。

```
A[] drop(A)(A[] l, int n)
  in{assert(n >= 0 && n <= l.length);}
  out(r){assert(r.length == l.length-n);}
  body{ return l[n..$]; }
```

書き換え前は body 節の中は 4 行になっていたが、書き換え後は 1 行にまとまり、処理内容が把握しやすくなっている。また、書き換え前は要素を 1 つずつ削除していたのに対し、書き換え後は n 個を同時に削除しているため、効率が良い。契約により、書き換え後の drop は正しい長さのリストを返すかエラーになることが保証されている。このように、契約を用いることにより、信頼性を失わずに速度や可読性の向上が可能となる。

6 現状と今後

これまでに、Coq のプログラム抽出機能および D 言語の仕様について調査を行うとともに、Coq と D 言語のプログラムの対応案を考えた。

今後は、対応案を確定させた後、D 言語の抽出機能を実装する。

参考文献

[1] Coq https://coq.inria.fr
[2] Bertrand Meyer. 1988. Object-Oriented Software Construction (1st ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
[3] CompCert http://compcert.inria.fr
[4] D 言語 http://dlang.org/
[5] DMD https://github.com/D-Programming-Language/dmd