OCaml プログラムの Coq への変換とプログラムの正しさの証明

毎田詠人, 中村薫, 才川隆文, Jacques Garrigue　名古屋大学

Coqgen project とは

- OCaml の型システムの健全性を Coq で確認することを目標とするプロジェクト.
- 具体的には OCaml から Coq へのコンパイラの作成, およびコンパイル後のプログラムに関する証明を行っている.
- 以下, Coqgen project で得られた成果の一部をまとめる.

モナド

- OCaml の副作用を Coq でも表現することができるようにモナドを利用する.
- メモリが変化する型を Env, 例外を Exn で表し, これらをまとめて次のようにモナドを定義する.

```
Definition M T := Env → Env * (T + Exn).
```

Type translation

OCaml の型はデータ型として定義されていて, Coq に翻訳される.

```
Fixpoint coq_type (T : ml_type) : Type := ...
```

今までの Coqgen

- core ML（λ計算 + 多相型, 再帰関数, データ型）
- 参照型, 例外
- 変換の例

```
let rec sum n = if n ≤ 0 then 0 else n + sum (n - 1);;
```

```
Fixpoint sum (h : nat) (n : coq_type ml_int) : M (coq_type ml_int) :=
  if h is h.+1 then
    do v <- ml_le h ml_int n 0%int63;
    if v then Ret 0%int63 else
      do v <- sum h (PrimInt63.sub n 1%int63); Ret (PrimInt63.add n v)
  else FailGas.
```

for loop

- OCaml のプログラム

```
let fact_for n = let v = ref 1 in
  for i = 1 to n do
    v := !v * i
  done; !v;;
```

- Coqgen のライブラリ

```
Fixpoint forloop (h : nat) (n_1 n_2 : int) (b : int → M unit) : M unit :=
  if h is h.+1 then
    if (n_2 < n_1)%sint63 then Ret tt
    else (do _ <- b n_1; forloop h (n_1 + 1)%sint63 n_2 b)
  else FailGas.
```

- コンパイル結果

```
Definition fact_for (h : nat) (n : coq_type ml_int) :
  M (coq_type ml_int) :=
  do v <- newref ml_int 1%int63;
  do _ <-
  (do u <- Ret 1%int63;
   do v_1 <- Ret n;
   forloop h u v_1
     (fun i ⇒
        do v_1 <- (do v_1 <- getref ml_int v; Ret (PrimInt63.mul v_1 i));
        setref ml_int v v_1));
  getref ml_int v.
```

本研究はTezos財団及びJSPS科研費JP22K11902の助成を受けたものです.

factorialについての証明

- 比較するもの

```
Definition fact_rec_int n : int := N2int (fact_rec (int2N n)).
```

・N2intはnat → intの変換, int2Nはint → natの変換
・fact_recは Coq のライブラリに元々存在する, 再帰的に定義された factorial

- 直感的な主張

「fact_forとfact_rec_intは同じ入力に対して同じ結果を返す.」
ただし, fact_forにはガスがあり, fact_rec_intはガスがないのでそのままの比較はできない.

- 正しい主張のための述語

```
Definition le_gasW {T} (f g : M T) := ∀ env, envok env →
  gasok (RunW (f env)) → RunW (f env) = RunW (g env).
```

・envokはgetrefやsetrefが成功するために必要な条件
・RunWは環境を捨て, 結果だけ見るという射影関数
・gasokはガスが足りていればtrue, なくなればfalse

- 正しい主張

```
Theorem fact_ok h n : int2N n < expn 2 61 →
  le_gasW (fact_for h n) (Ret (fact_rec_int n)).
```

証明のポイント

- n(整数)では帰納法が使えない
  → nをN2int(int2N n)に書き換え, int2N n(自然数)についての数学的帰納法を使った
- forloopは途中で切ることができるため, 1からn+1のループを1からnとnからn+1のループの二つに分けた帰納法の仮定(1からnのループ)を使えるようにした
- intに関する補題をいくつか作り, 証明して使った
  その際nが最大値でも最小値でもないという条件を必要とし, その証明に時間を要した

改善点

- 証明は100行ほどになり, 長い.
  原因はintや環境に対する条件の証明
  →補題を整理し, モナド自体にあらかじめ条件を入れて証明をシンプルにしたい.
- プログラムを全て開いて計算するようにして証明を進めたため, 一目で何をやっているか分からない.
  →なるべく等式推論で証明を進めたい.
- forloopでガスを使っているため証明が長くなった.
  →forloopはガスなしでも定義できるのではないか

ガスを使わないforloop

回数を先に計算することでガスを回避できる

```
Definition forloop (n_1 n_2 : int) (b : int → M unit) : M unit :=
  if (n_2 < n_1)%sint63 then Ret tt else
  iter (uint2N (n_2 - n_1)%sint63)   (* 差分を符号無しで自然数に *)
       (fun (m : M int) ⇒ do i <- m; do _ <- b i; Ret (Uint63.succ i))
       (Ret n_1) >> Ret tt.
```

Coqgen に関する情報

https://www.math.nagoya-u.ac.jp/~garrigue/cocti/coqgen/

Monae の利用

- Monae は monadic equational reasoning のための Coq のライブラリ
- モナドの等式理論を定義し, 変形によってプログラムの等価性を証明
- 各モナドのモデル(実装)を与えることで, 推論の正しさを保証する
- 様々なモナドやmonad transformerが定義されている

Coqgenのモナドを Monaeに入れる

- Array Monad という変更可能な配列のモナドをヒントに
  Typed Store Monad の等式理論を定義
- 未定義や型エラーのために Failure Monad も継承
- Coqgen の実装がそのモデルであることを証明
- 条件のない等式にすることで envok や mem_env が不要

```
HB.mixin Record isMonadTypedStore (M : UU0 → UU0)
    of Monad M := {
  cnew : ∀ {T}, coq_type M T → M (loc T) ;          (* newref *)
  cget : ∀ {T}, loc T → M (coq_type M T) ;          (* getref *)
  cput : ∀ {T}, loc T → coq_type M T → M unit ;     (* setref *)
  crun : ∀ {A : UU0}, M A → option A ;              (* 空のストアで実行 *)
  cnewget : ∀ T (s : coq_type M T) A (k : loc T → coq_type M T → M A),
    (do r <- cnew s; do x <- cget r; k r x) = (do r <- cnew s; k r s) ;
  cnewput : ∀ T (s t : coq_type M T) A (k : loc T → M A),
    (do r <- cnew s; do _ <- cput r t; k r) = (do r <- cnew t; k r) ; ...
```

述語cchkを導入することでcnewとcgetやcputの可換性が証明できる.
cchk rはrがストアで定義され, 正しい型を持っているときに成功する.

```
Definition cchk T (r : loc T) : M unit := do _ <- cget r; skip.
Lemma cchknewget T T' (r : loc T) s (A : UU0) k :
  (do _ <- cchk r; do r' <- cnew T' s; do u <- cget r; k r' u) =
  (do u <- cget r; do r' <- cnew T' s; k r' u) :> M A. ...
```

factorialについての証明・改

```
Theorem fact_for_ok : n < int2N Sint63.max_int →
  crun (fact_for (N2int n)) = Some (N2int (fact_rec n)).
Proof.
rewrite /fact_for.
under eq_bind do rewrite !bindA !bindretf.
set fn := N2int (fact_rec n).
transitivity (crun (cnew ml_int fn >> Ret fn : M _));
  last by rewrite crunret // crunnew0.
congr crun.
have {1}→ : (1 = N2int 1)%int63 by [].
rewrite -/(fact_rec 0).      (* u の初期値1をfact_rec 0とみなす *)
have → : (1 = succ (N2int 0))%int63 by [].
pose m := n.
have → : 0 = n - m by rewrite subnn.
have : m ≤ n by [].
elim: m ⇒ [|m IH] mn.                (* n-m はループを回した回数 *)
  rewrite subn0.
  under eq_bind do rewrite forloop0 (ltsb_succ,bindretf) // -cgetret.
  by rewrite cnewget.                (* ベースケースをcnewgetで証明 *)
rewrite -N2int_succ subnSK //.
under eq_bind do rewrite forloopS !(lesb_subr,bindA) //.
rewrite cnewget.          (* cnewget や cnewput で計算を進める *)
under eq_bind do rewrite bindretf.
rewrite cnewput -IH (ltnW,subnS) // -N2int_mul.
by rewrite mulnC -(@prednK (n-m)) // lt0n subn_eq0 -ltnNge.
Qed.
```

Monaeに関する情報: https://github.com/affeldt-aist/monae