CoqにおけるMonads with Predicate Liftingsの実装と考察

須田 啓司 (千葉大学大学院 理学研究科)

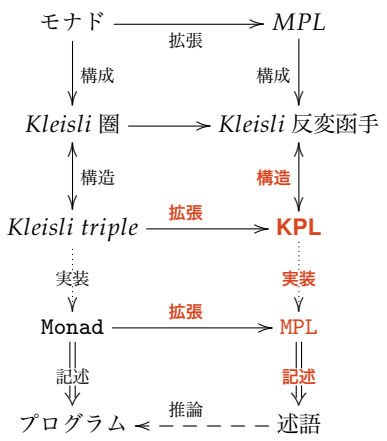
概要

MPL？

- **計算効果を伴うプログラム**についての**述語**を扱う枠組み
- 詳細な定義や議論は Jacobs による [1] 参照
- モナドと述語の関係から，MPL は**自然に表れる** [1]

ポイント

- 本研究では Jacobs による MPL を**一般化したもの**を利用
- Coq で実装するため，新たに **KPL** を定義
- **型クラス**として **Coq に於いて実装**
- MPL による**ホーアトリプルの記述**と推論の具体例



関連研究

- *Evaluation Logic*. A.M.Pitts. 1990.
  計算型付ラムダ計算を，その上の述語論理へ拡張
- *A Semantics for Evaluation Logic*. E.Moggi. 1993.
  モナドによる計算型付ラムダ計算の意味論を，添字付交わり半束などを用いて，Evaluation Logic の意味論へと拡張
- *Predicate Logic for Functors and Monads*. B.Jacobs. 2010
  添字付圏，Predicate Lifting を用いて，モナドの Kleisli 圏 (や代数) の述語がなす圏を構成

MPL&KPL

- MPL から構成される Kleisli 反変函手が**述語全体**を表す
- KPL は Kleisli 反変函手の構造を**直接**与えるもの
- MPL と KPL は，**互いに互いを構成可能**という意味で等価

MPL	KPL
$\mathcal{C}$ 上のモナド $\langle T, \eta, \mu\rangle$	$\mathcal{C}$ 上の Kleisli triple$\langle T, \eta, (-)^{\#}\rangle$
反変函手 $\Phi : \mathcal{C}^{\mathrm{op}} \to \mathcal{D}$	変換 $\Phi : \lvert\mathcal{C}\rvert \to \lvert\mathcal{D}\rvert$
自然変換 $\tau : \Phi \Rightarrow \Phi T$	射の族 $\{\Box_f : \Phi(Y) \to \Phi(Y) \mid f \in \mathcal{C}(X, TY)\}$
$id_{\Phi(X)} = \Phi(\eta_X) \circ \tau_X$ $\tau_{TX} \circ \tau_X = \Phi(\mu_X) \circ \tau_X$	$id_{\Phi(X)} = \Box_{\eta_X}$ $\Box_f \circ \Box_g = \Box_{g^{\#} \circ f}$

ホーアトリプル by MPL

ホーアトリプル $P\{f\}Q$ は **MPL を使って表せる**

- $\mathcal{D}$ を **PoSets** に制限 (or 具体化)
  - $\Phi(X)$ に半順序構造を付加する
  - $\Box_f$ は単調関数になる
- $P\{f\}Q :\Leftrightarrow P \leq \Box_f(Q)$
  - ホーアトリプルの一般化
  - “$P\{id_X\}P$” は自明
  - “$P\{f\}Q, Q\{g\}R \Rightarrow P\{f;g\}R$” は $\Box_f$ の単調性より従う
- 仕様構造 [2] との関連が理論的背景にある

まとめ

- **KPL を定義**し，これを用いて **Coq における MPL の実装**を型クラスとして与えた
  - Monad のサブクラス MPL として KPL を実装
- 仕様構造との関連から，**MPL による (一般化された) ホーアトリプルの記述**を与えた
  - MPL に半順序構造を付加した上での記述
- **MPL の推論への応用**を，Hoare State Monad と比較する形で行った
  - 特定のモナドを用いて記述されたプログラムに手を加える必要がない

課題と考察

- MPL そのものに対する考察
  - MPL にどのような性質を付加するか
  - tripos[4] の性質を MPL にどのようにして持ち込むか
- MPL の推論への応用例の増加
  - MPL の有用性の補強
  - MPL に必要な物を経験的に探る
- HasCASL[5] との比較
  - 計算効果を伴うプログラムの記述とその仕様の記述が可能
  - 特に Monad-independnt Hoare Logic[6] との関係
- 抽象的な推論のための規則の導出
  - ホーア論理との関係
  - **特定のインスタンスに依らない**議論
  - **統一的な推論環境**の構築
- 状態付き計算などの計算効果との関係
  - **特定の計算効果に特化した推論環境**の構築
  - 規則の導出をどのようにしておこなうか
    - 意味論などとの対応を基にした圏論的見地
    - 利用例などを基にした経験的見地

MPL on Coq

Coq における MPL の実装は**型クラス** MPL

- **KPL を型クラスとして実装したもの**
- Monad クラスの**サブクラス**
- 付加する性質は MPL のサブクラスとして記述

```
Class MPL (gpr: Set -> Type)‘(monad: Monad): Type :=
  {
    modal {X Y: Set}(f: X -> T Y): gpr Y -> gpr X
      where "f # P" := (modal f P);

    modal_ident:
      forall X (P: gpr X), P = ret#P;
    modal_compose:
      forall X Y Z (f: X -> T Y)(g: Y -> T Z) R,
        f#g#R = (f|>g)#R
  }.
```

Tree Relabeling

関数 relabel が “仕様” を満たすことを示す
Hoare State Monad によるもの [3] を改良

流れ

1. MPL に半順序構造を付加
   - 型クラス hasPord として性質を記述
2. (一般化された) ホーアトリプルを定義
   - 空文の公理や合成規則の証明が可能
3. relabel を State モナドを用いて記述
4. GHT を使って示したい定理を記述し証明

比較

- [3] では，推論のために **State モナドを拡張** relabel の定義も**書き直す**
- MPL を用いれば，**State モナドのまま**でよい relabel を**そのまま**推論の対象にする

```
Class hasPord ‘(mpl: MPL) :=
  {
    gpr_pord X :> PartialOrder (gpr X);

    modal_monotone:
      forall X Y (f: X -> T Y)(P Q: gpr Y),
        P <= Q -> f#P <= f#Q
  }.
```

```
Definition GHT P f Q := (P <= f#Q).
Notation "P {- f -} Q" := (GHT P f Q).

Lemma GHT_ret:
  forall {X: Set}(P: gpr X), P{- ret -}P.

Lemma GHT_compose:
  forall X Y Z P (f: X -> T Y) Q (g: Y -> T Z) R,
    P{- f -}Q -> Q{- g -}R -> P{- f|>g -}R.
```

```
Fixpoint relabel {A: Set}(t: Tree A): T (Tree nat) :=
  match t with
    | Leaf _ => x <- get;
                put ((S x):nes_nat) >> ret (Leaf x)
    | Node l r => l’ <- relabel l;
                  r’ <- relabel r;
                  ret (Node l’ r’)
  end.
```

```
Fixpoint flatten {A: Set}(t: Tree A): list A :=
  match t with
    | Leaf a => [a]
    | Node l r => flatten l ++ flatten r
  end.

Theorem relabelNoDup:
  forall (A: Set)(n: nes_nat),
    StateIs n {- relabel (A:=A) -}$(NoDup(.)flatten).
```

参考文献

[1] Bart Jacobs. Predicate Logic for Functors and Monads. Nov 2010.

[2] S. Abramsky, S. J. Gay, and R. Nagarajan. A specification structure for deadlock-freedom of synchronous processes. *TCS*, Vol. 222, pp. 1–53, 1999.

[3] Wouter Swierstra. A hoare logic for the state monad. In *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics*, TPHOLs ’09, pp. 440–451. Springer-Verlag, 2009.

[4] Eugenio Moggi. A Semantics for Evaluation Logic. *Fundamenta Informaticae*, Vol. 22, pp. 22–117, 1995.

[5] Lutz Schröder and Till Mossakowski. HasCASL: Integrated Higher-order Specification and Program Development. *Theoretical Computer Science*, Vol. 410, No. 12–13, pp. 1217–1260, 2009.

[6] Lutz Schröder and Till Mossakowski. Monad-independent Hoare Logic in HasCASL. In *FUNDAMENTAL ASPECTS OF SOFTWARE ENGINEERING, VOLUME 2621 OF LNCS*, pp. 261–277. Springer-Verlag, 2003.