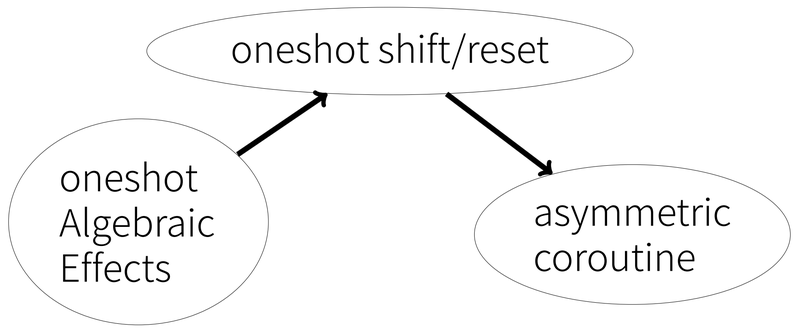
ワンショットの限定継続に着目した代数的効果から非対称コルーチンへの変換

河原 悟 亀山 幸義 (筑波大学)

研究内容

代数的効果(algebraic effects)と呼ばれる、近年関心を寄せている言語機能がある。代数的効果はエフェクトの発生以降のコンテキストを継続として持ち、shift/resetをエミュレートすることができる。代数的効果を持つMulticore OCamlは、継続の実行を高々1回に制限した代数的効果を持つ。一方、継続を一度だけ実行できる限定継続oneshot shift/resetは非対称コルーチンに変換できることが知られている。
本研究では、ワンショットの代数的効果からワンショットのshift/resetへ変換し、そしてそのshift/resetから非対称コルーチンに変換することを検討する。そしてワンショットの代数的効果から非対称なコルーチンへの直接の変換を考える。ワンショットの代数的効果から非対称なコルーチンへ変換をおこなうことにより、Luaのようなコルーチンを持つ体系でワンショットの代数的効果を用いたプログラミングが期待できる。



代数的効果

Moggiらのcomputational effectsを基に、2001年にPlotkinらにより提唱された [4]。コントロールの操作ができ、例外だけでなく、限定継続を扱うことでshift/resetをエミュレートすることができる。代数的効果のあるプログラム言語とその実装に、Eff、KokaやLinksなどがある。
Multicore OCamlは、代数的効果を持つOCamlの方言である。並列プログラミングやパフォーマンスのために、ハンドラにより取り出される限定継続の実行は、明示的なクローンをおこなわない限りは高々1回に制限されている [1]。

```
module State(S : sig type t end) = struct
  type t = S.t

  effect Put : t -> unit
  effect Get : t

  let get () = perform Get

  let run init f =
    init |> match f () with
    | x -> (fun s -> (s, x))
    | effect (Put s') k ->
      (fun s -> continue k () s')
    | effect Get k ->
      (fun s -> continue k s s)
end

effect Log : int -> unit
```

```
let log m = perform @@ Log m

try begin
  let module IS =
    State(struct type t = int end)
  in
  let incr () =
    IS.(perform @@ Put (get() + 1))
  in
  IS.run 0 (fun () ->
    incr ();
    log @@ IS.get ();
    incr ();
    log @@ IS.get ())
end
with effect (Log msg) k ->
  print_int msg; continue k ()
```

非対称コルーチン

非対称コルーチンは de Moura と Ierusalimschyにより提唱された、“呼び出す”、“呼び出される”関係を持つコルーチンである [3]。対称コルーチンやワンショットの継続と同等の表現力があることが知られており、非対称コルーチンはLua や JavaScript などの言語へ組み込まれているだけでなく、Unityなどのゲームエンジンの機能としても採用されている。

```
chr = {hp = 0; mp = 0}

chr.ctrl =
  coroutine.create(function(hp, mp)
    chr.hp = hp
    chr.mp = mp

    while true do
      local fld, v = coroutine.yield()
       chr[fld] = chr[fld] + v
    end
  end)

coroutine.resume(chr.ctrl, 100, 100)
coroutine.resume(chr.ctrl, "hp", -10)
coroutine.resume(chr.ctrl, "mp", -20)
```

ワンショット代数的効果からワンショットshift/resetへの変換

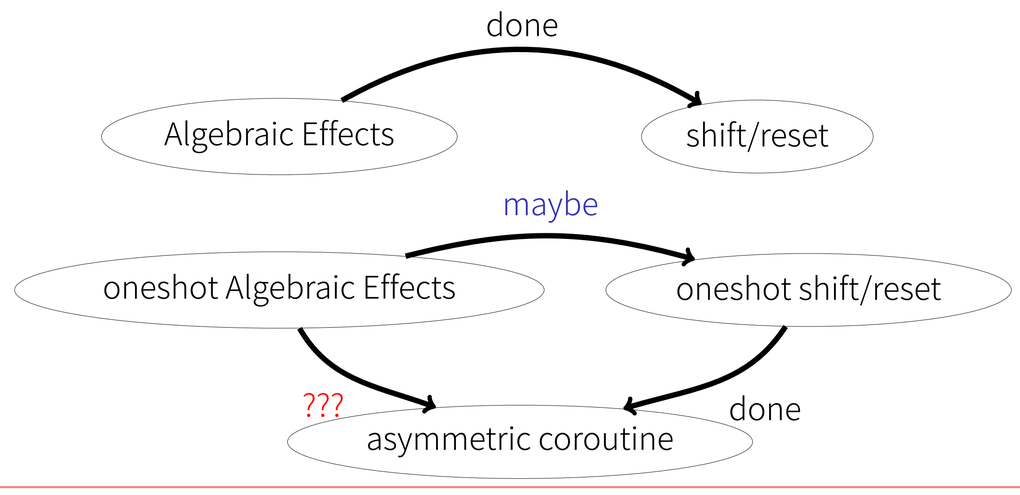
Kiselyovらによるshift/resetライブラリを用いたEffをOCamlへ埋め込む方法 [2] に基づいて実装をおこなった。本研究は、これをワンショットに限定した変換にする。

```
module Translate(D : DELIMCC) sig
  type ('a, 'b) free
  val newi : unit -> 'a D.prompt
  val op : ('a, 'b) free D.prompt -> 'a -> 'b
  val handler : ('g, 'g) free D.prompt ->
    ('g -> 'o) -> ('g * ('g -> 'o) -> 'o) -> 'g thunk -> 'o
  val handle : ('a thunk -> 'b) -> 'a thunk -> 'b
end = struct ..... end

let ans =
  let module Eff = Translate(DelimccOne) in
  let readerh inst = Eff.handler inst
      (fun v -> (fun _ -> v))
      (fun (x, k) -> (fun s ->
           (* runtime error: cannot call `k` more than once *)
           (* k 0 0; let z = x + s in k z s *)
           let z = x + s in k z s))
  in
  let readi = Eff.newi () in
  let hr = Eff.handle (readerh readi) @@ fun () ->
      let x = Eff.op readi 1 in
      let y = Eff.op readi x in y
  in hr 10
```

ワンショット代数的効果から非対称コルーチンへの変換

ワンショットの shift/reset は非対称コルーチンへ変換できることが知られている [5]。この変換を踏まえて、ワンショットの代数的効果から非対称コルーチンへの変換を考える。まず、ワンショットの代数的効果からワンショットのshift/reset へ変換する。そして得られたshift/resetを非対称コルーチンへ変換する。この結果を観察し、ワンショットの代数的効果から非対称コルーチンへの直接の変換を検討する。shift/reset への変換を経由しないことで、シンプルな変換として表現することが期待できる。
Kiselyovらによる代数的効果からshift/resetへの変換では、対象となるshift/resetはプロンプトのある体系である。一方でUsuiらの変換では、プロンプトをもたないshift/resetが用いられている。Usuiらの変換に基づき、プロンプトを持つワンショットのshift/resetをターゲットとした非対称コルーチンへの変換を考える。



参考文献

[1] Stephen Dolan, Leo White, K Sivaramakrishnan, Jeremy Yallop, and Anil Madhavapeddy. Effective concurrency through algebraic effects. In *OCaml Workshop*, p. 13, 2015.

[2] Oleg Kiselyov and KC Sivaramakrishnan. Eff directly in ocaml. In *ML Workshop*, 2016.

[3] Ana Lúcia De Moura and Roberto Ierusalimschy. Revisiting coroutines. *ACM Trans. Program. Lang. Syst.*, Vol. 31, No. 2, pp. 6:1–6:31, February 2009.

[4] Gordon Plotkin and John Power. Adequacy for algebraic effects. In *International Conference on Foundations of Software Science and Computation Structures*, pp. 1–24. Springer, 2001.

[5] Chiharu Usui. One-shot delimited continuations as coroutines, 2017.