Homework 4
COSE312, Spring 2020

Hakjoo Oh

Due: 5/26, 23:59

Problem 1 이번 숙제에서는 수업시간에 다룬 정적 부호 분석기(static sign analysis)를 구현해 봅시다. OCaml에서 `while` 언어는 다음과 같이 정의됩니다.

```
type var = string
type aexp =
  | Int of int
  | Var of var
  | Plus of aexp * aexp
  | Mult of aexp * aexp
  | Minus of aexp * aexp
type bexp =
  | True
  | False
  | Eq of aexp * aexp
  | Le of aexp * aexp
  | Neg of bexp
  | Conj of bexp * bexp
type cmd =
  | Assign of var * aexp
  | Skip
  | Seq of cmd * cmd
  | If of bexp * cmd * cmd
  | While of bexp * cmd
```

예를 들어, 프로그램

```
          x:=10; y:=2; while (x!=1) do (y:=y*x; x:=x-1)
```

은 아래와 같이 표현됩니다.

```
Seq (Assign ("x", Int 10),
  Seq (Assign("y", Int 2),
    While (Neg (Eq (Var "x", Int 1)),
       Seq(Assign("y", Mult(Var "y", Var "x")),
               Assign("x", Minus(Var "x", Int 1))))))
```

정수값, 부울값을 요약한 요약 도메인들을 다음과 같은 모듈들로 정의할 수 있습니다.

```
module ABool = struct
  type t = Bot | Top | TT | FF
  let alpha b = if b then TT else FF

  (* partial order *)
  let order : t -> t -> bool
  =fun b1 b2 -> true (* TODO *)

  (* least upper bound *)
  let lub : t -> t -> t
  =fun b1 b2 -> Top  (* TODO *)

  (* abstract negation *)
  let neg : t -> t
  =fun b -> Top (* TODO *)

  (* abstract conjunction *)
  let conj : t -> t -> t
  =fun b1 b2 -> Top (* TODO *)
end

module AValue = struct
  type t = Bot | Top | Neg | Zero | Pos | NonPos | NonZero | NonNeg
  let alpha : int -> t
  =fun n -> if n = 0 then Zero else if n > 0 then Pos else Neg

  (* partial order *)
  let order : t -> t -> bool
  =fun s1 s2 -> true (* TODO *)

  (* least upper bound *)
  let lub : t -> t -> t
  =fun s1 s2 -> Top (* TODO *)

  (* abstract addition *)
  let plus : t -> t -> t
  =fun a1 a2 -> Top (* TODO *)

  (* abstract multiplication *)
  let mult : t -> t -> t
  =fun a1 a2 -> Top (* TODO *)

  (* abstract subtraction *)
  let minus : t -> t -> t
  =fun a1 a2 -> Top (* TODO *)

  let eq : t -> t -> ABool.t
  =fun a1 a2 -> ABool.Top (* TODO *)

  let le : t -> t -> ABool.t
  =fun a1 a2 -> ABool.Top (* TODO *)
end
```

요약 도메인 AValue, ABool이 정의되면 요약 메모리 상태(abstract memory state)는 다음과 같이 일반적으로 정의됩니다.

```
module AState = struct
  module Map = Map.Make(struct type t = var let compare = compare end)
  type t = AValue.t Map.t (* var -> AValue.t map *)
  let bot = Map.empty
  let find x m = try Map.find x m with Not_found -> AValue.Bot
  let update x v m = Map.add x v m
  let update_join x v m = Map.add x (AValue.lub v (find x m)) m
  let order m1 m2 = Map.for_all (fun k v -> AValue.order v (find k m2)) m1
  let lub m1 m2 = Map.fold (fun k v m -> update_join k v m) m1 m2
end
```

요약 의미 함수(abstract semantic function) 또한 다음과 같이 일반적으로 정의됩니다.

```
let rec aeval : aexp -> AState.t -> AValue.t
=fun a s ->
  match a with
  | Int n -> AValue.alpha n
  | Var x -> AState.find x s
  | Plus (a1, a2) -> AValue.plus (aeval a1 s) (aeval a2 s)
  | Mult (a1, a2) -> AValue.mult (aeval a1 s) (aeval a2 s)
  | Minus (a1, a2) -> AValue.minus (aeval a1 s) (aeval a2 s)

let rec beval : bexp -> AState.t -> ABool.t
```

```
=fun b s ->
  match b with
  | True -> ABool.alpha true
  | False -> ABool.alpha false
  | Eq (a1, a2) -> AValue.eq (aeval a1 s) (aeval a2 s)
  | Le (a1, a2) -> AValue.le (aeval a1 s) (aeval a2 s)
  | Neg b -> ABool.neg (beval b s)
  | Conj (b1, b2) -> ABool.conj (beval b1 s) (beval b2 s)

let rec ceval : cmd -> AState.t -> AState.t
=fun c s ->
  match c with
  | Assign (x, a) -> AState.update x (aeval a s) s
  | Skip -> s
  | Seq (c1,c2) -> ceval c2 (ceval c1 s)
  | If (b, c1, c2) ->
      if beval b s = ABool.TT then (ceval c1 s)
      else if beval b s = ABool.FF then (ceval c2 s)
      else AState.lub (ceval c1 s) (ceval c2 s)
  | While (_, c) -> fix (ceval c) s

and fix f s0 = if AState.order (f s0) s0 then s0 else fix f (f s0)

let run : cmd -> AState.t
=fun c -> ceval c AState.bot
```

요약 도메인 `AValue`, `ABool` 구현을 완성하세요.