Homework 4

SNU 4541.664A, 2009 봄

Due: 4/8 24:00

모든 프로그래밍 숙제는 OCaml로 작성합니다.

Exercise 1 "옵션설비된 실행기(instrumented interpreter)"

다음의 언어를 생각하자.

$$
\begin{array}{rcl}
C & \to & \texttt{skip} \\
& | & x \texttt{ := } E \mid \texttt{*}x \texttt{ := } E \\
& | & C \texttt{ ; } C \\
& | & \texttt{if } E\ C\ C \\
& | & \texttt{while } E\ C \\
E & \to & n \quad (n \in \mathbb{Z}) \\
& | & E \texttt{ + } E \mid \texttt{ - } E \\
& | & x \mid \texttt{*}x \mid \texttt{\&}x \\
& | & \texttt{read}
\end{array}
$$

프로그램은 명령 C이다. 명령(C)과 식(E)은 통상적인 의미를 가진다. 단, 입력식 `read`는 -5에서 +5까지의 정수중 하나를 뜻한다:

$$
\begin{array}{rcl}
M \in \mathit{Memory} & = & \mathit{Loc} \xrightarrow{\text{fin}} \mathit{Val} \\
l \in \mathit{Loc} & = & \mathit{Var} \\
v \in \mathit{Val} & = & \mathbb{Z} + \mathit{Loc}
\end{array}
$$

$$
\frac{}{M \vdash \texttt{read} \Rightarrow z} \ z \in [-5, +5]
$$

$$
\frac{}{M \vdash \texttt{*}x \Rightarrow M(M(x))}
$$

$$
\frac{}{M \vdash \texttt{\&}x \Rightarrow x}
$$

$$\frac{M \vdash E \Rightarrow v}{M \vdash *x := E \Rightarrow M\{M(x) \mapsto v\}}$$

다른 식들과 명령문의 의미는 통상적인 의미를 가진다.

위 언어의 다음과 같은 프로그램 실행기(interpreter)들을 구현하라. 구현할 프로그램 실행기는 통상적인 실행기와 달리, 프로그램의 모든 입력값에 대해서 모두 실행해 보면서 다음의 실행 정보를 기록하도록 한다:

- `tracingEval` 프로그램이 실행중에 가지는 기계상태의 모든 전이과정을 기록하는 실행기. 입력마다 전이과정은 기계상태들의 순서열(하나의 실)이 될 것이다. 가능한 입력값이 여럿 있으므로 이러한 실들의 다발이 모이게 된다.

- `collectingEval` 프로그램이 실행중에 가지는 기계상태 전이의 전후 관계에 대한 정보없이, 발생하는 기계상태들만을 모두 모은다.

- `pointCollectingEval` 프로그램이 실행중에 가지는 기계상태들을 프로그램의 각 명령문을 기준으로 따로따로 모은다. 이 때, 각 명령문이 실행되기 직전의 기계상태들을 모은다.

최소한 다음의 타입에 맞는 모듈을 완성하면 된다. 위 언어의 파서(`string` → `cmd`)는 제공된다.

```
module type K =
  sig
   exception Error of string
   type id
   type label
   type cmd = label * stmt
   and stmt = SKIP
            | ASSIGN of id * exp          (* x := exp *)
            | ASSIGNSTAR of id * exp      (* *x := exp *)
            | SEQ of cmd * cmd            (* c;c *)
            | IF of exp * cmd * cmd       (* if e c c *)
            | WHILE of exp * cmd          (* while e c *)
   and exp = NUM of int
           | ADD of exp * exp        (* e+e *)
           | MINUS of exp            (* -e *)
           | VAR of id               (* x *)
```

```
          | STAR of id              (* *x *)
          | AMPER of id             (* &x *)
          | READ

  type program = cmd
  type memory
  type traces
  type states
  type pointstates

  val emptyMemory: memory
  val tracingEval: cmd -> memory -> traces
  val collectingEval: cmd -> memory -> states
  val pointCollectingEval: cmd -> memory -> pointstates
 end
```

□