Homework 5
SNU 4910.210 Spring 2006

due: 6/6 24:00

Exercise 1 "불확정성의 언어 Mirage"

다음과 같은 언어 Mirage 를 생각합시다.

program	$\rightarrow$	c	
c	$\rightarrow$	$\mathtt{x} := e$	assignment
	$\mid$	$c\ ;\ c$	sequence
	$\mid$	c^*	non-deterministic 0 or more repetitions
	$\mid$	$c \cup c$	non-deterministic choice
	$\mid$	$\mathtt{x}\ \mathtt{eq}\ e\ ?\ c$	check
	$\mid$	$\mathtt{x}\ \mathtt{neq}\ e\ ?\ c$	check
e	$\rightarrow$	n	integer
	$\mid$	$e + e$	addition
	$\mid$	$e - e$	substraction
	$\mid$	$\mathtt{x}$	

각 명령어는 기계상태를 변환시켜줍니다. 기계는 프로그램 변수의 정수값을 가지고 있습니다. 프로그램 변수는 오직 하나만 있습니다.

- "$x := e$"는 현재 기계상태에서 e를 계산해서 x의 값으로 기계상태를 변환
- "$c_1\ ;\ c_2$"는 c_1 실행 후 결과 상태를 가지고 c_2를 실행
- "c^*"는 c를 0번 이상 임의의 횟수를 반복
- "$c_1 \cup c_2$"는 c_1이나 c_2중 하나를 선택해서 실행
- "$x\ \mathtt{eq}\ e\ ?\ c$"는 현재상태에서 x의 값이 e와 같은 지 확인. 같으면 c를 실행, 틀리면 건너뜀

- "x `neq` e ? c"는 현재상태에서 x의 값이 e와 같은 지 확인. 다르면 c를 실행, 같으면 건너뜀
- 변수가 가질 수 있는 값은 -5 에서 +5 까지의 정수. 그 이외의 값을 변수가 가지게 되면 현재상태로 실행 끝

Mirage 프로그램의 싹쓸이(exhaustive) 실행기 `exeval`

`exeval` : `pgm`− > `state`− > `statelist`

을 작성합시다. "싹쓸이"란 모든 가능한 경우를 실행하는 것을 말합니다.

```
type pgm = cmd
and cmd = ASSIGN of exp
        | SEQUENCE of cmd * cmd
        | REPEAT of cmd
        | CHOICE of cmd * cmd
        | EQ of exp * cmd
        | NEQ of exp * cmd
and exp = NUM of int
        | ADD of exp * exp
        | SUB of exp * exp
        | VAR
type state = int
```

예를들어,

```
x := 1; ((x eq 1? x := x+1) U (x neq 1? x := x-1))*
```

를 실행하면 결과 상태들은 `[1, 2]`.

또다른 예로,

```
x := 1; (x := x+1)*
```

를 실행하면 결과 상태는 `[1,2,3,4,5]`.

□

Exercise 2 "집합 모듈 함수"

집합 모듈을 만들어 주는 아래의 모듈 함수들(`SetFun`, `ProductSetFun`, `PowerSetFun`)을 정의해봅시다. "..." 부분을 메꾸면 됩니다. 만들어 지는 모든 집합 모듈들은 아래의 `SET` 시그너쳐를 만족해야 합니다.

```
signature SET =
      sig
       type t
       type 'a set
       val empty : t set
       val equal : t * t -> bool
       val member : t * t set -> bool
       val add : t * t set -> t set
       val cup : t set * t set -> t set
       val cap : t set * t set -> t set
       val minus : t set * t set -> t set
      end

functor SetFun(S: sig type t val equal: t * t -> bool end) =
  struct
   type t = S.t
   type 'a set = Empty | Set of 'a * 'a set
   ...
  end

functor ProductSetFun(A:SET, B:SET) =
  struct
    type t = A.t * B.t
    type 'a set = Empty | Set of 'a * 'a set
    ...
  end

functor PowerSetFun(A:SET) =
  struct
    type t = A.t A.set
    type 'a set = Empty | Set of 'a * 'a set
    ...
  end
```

그래서, 아래와 같이 다양한 집합모듈을 구성해서, 집합을 만들고 사용하는 프로그래밍이 가능해야 한다.

```
structure IntSet = SetFun(struct
                           type t = int
                           fun equal(x,y) = x=y
                          end)

val a = IntSet.add(1,IntSet.empty)    (* {1} *)
val b = IntSet.add(2,a)               (* {1,2} *)
val c = IntSet.cup(a,b)               (* {1}U{1,2} *)

structure StringSet = SetFun(struct
                              type t = string
                              fun equal(x,y) = x=y
                             end)
val d = StringSet.add(''a'', StringSet.empty)     (* {''a''} *)
val e = StringSet.minus(d,d)                      (* {} *)

structure A = ProductSetFun(IntSet, StringSet)

val f = A.add((1, ``a''), A.empty)  (* {<1,''a''>} *)
val g = A.add((2, ``c''), f)        (* {<2,''c''>, <1,''a''>} *)
val h = A.cup(f,g)                  (* {<2,''c''>, <1,''a''>} *)

structure B = PowerSetFun(IntSet)

val i = B.add(a, B.empty)        (* {{1}} *)
val j = B.add(b, i)              (* {{1, 2}, {1}} *)
val k = B.cap(i,j)               (* {{1}} *)
```

□

Exercise 3 "미로 올림픽 선수"

미로 올림픽에서는 누가누가 더 아름답고 어려운 미로를 종이위에 만드는지를 경연한다. 이번 숙제에서는 미로를 만드는 선수들을 자동으로 만들어 내는 "선수촌 모듈함수" `MazeGenGen` 을 제작하는 것이다. 미로 만들기 프로그램의 핵심을, 재사용 가능하고 일반화된 모듈(parameterized module, functor)로 구성하고 각 종목별 미로 만들기 선수들은 모듈함수 `MazeGenGen`를 적절한 인

자 모듈에 적용하여 자동으로 만들어 지도록 한다.

모든 미로를 다음과 같이 보자:

- 종이에 정4각형들이 빼곡히 채워져 있다(모눈종이). 각 정4각형을 하나의 방이다.
- 각 방들은 이웃한 방들과 사이의 벽들이 몇 개 터져 있기도 하고 막혀있기도 하다.
- 서로 다른 시작 방과 끝 방이 하나씩 정해져 있다.

미로 만들기 올림픽 종목에는 다음의 두 축의 조건들이 조합된 $3 \times 3 = 9$ 가지 종목이 있다:

- 평면의 조건
  - 4각형 평면으로 그려진 미로 (4선분이 떨어져 있슴)
  - 원통의 바깥면에 그려진 미로 (마주보는 2선분중 한쌍이 붙어 있슴)
  - 도넛스 형태로 모든 면이 연결된 평면에 그려진 미로 (마주보는 2선분 두쌍이 모두 붙어 있슴)
- 미로의 조건
  - 시작방과 끝방만 있고 다른 조건이 없는 미로
  - 시작방에서 끝방으로 가는 과정에 꼭 거쳐야 하는 방들이 하나 이상 있는 미로. 이 때 갔던 방을 반복해서 가야하면 안됨.
  - 방하나를 움직일 때 에너지 1이 소모되며, 몇개의 방들에서는 그 방에 할당된 정해진 에너지를 보충할 수 있다. 전체 미로 방의 갯수의 1/10 되는 초기 에너지로 미로를 찾아낼 수 있어야 한다.

너무나 쉬운 미로는 만들 수 있다: 예를들어 모든 방의 벽을 허물면 된다. 채점은 모듈함수를 얼마나 잘 구성했느냐는 것과, 만들어 내는 미로가 얼마나 어려운 지를 기준으로 한다. 미로의 난이도를 판별하는 채점 기준은 별도록 공지될 것이다. □

Exercise 4 "미로 올림픽 심판"

이번에는 위의 미로 올림픽에서 활동할 심판관들을 만드는 모듈함수 `MazeJudgeGen`을 구성한다. 심판관들은 종목의 조건에 맞는 미로가 만들어 졌는 지를 확인해 준다.

위의 문제에서 제작한 모듈함수와 비슷하게 미로 심판관의 핵심을, 재사용 가능하고 일반화된 모듈(parameterized module, functor)로 구성하고 각 종목별 미로 심판관들은 모듈함수 `MazeJudgeGen`를 적절한 인자 모듈에 적용하여 자동으로 만들어 지도록 한다. □