Environments (Homework 2)

CSE 4102 Project 2, Spring Semester, 2018

Bryan Arnold

2/11/18

Section: 001

Instructor: Jeffrey A. Meunier

# Introduction

The purpose of this assignment was to be able to write interesting programs using ML. We need the ability to define names, or in other words, to bind values to names (i.e., variables). In the evaluator for the language, these names will be managed in a data structure called an environment. The environment is just a list of bindings, where a binding is a mapping of name to value. The assignment will implement the environment in two different ways.

# Output

First, I will show the output for each function definition in env1.sml to show the correct parameter and output values of each function:

```
- exception NameNotBound of string;
exception NameNotBound of string
-
- type Env = (string * int) list;
type Env = (string * int) list
-
- fun env_new () : Env = nil;
val env_new = fn : unit -> Env
-
- fun env_bind (e : Env) s v : Env = (s, v)::e;
val env_bind = fn : Env -> string -> int -> Env
-
```

```
- fun env_lookup ([] : Env) s = raise NameNotBound s
=   | env_lookup (e : Env) s = if #1(hd e) = s then #2(hd e) else env_lookup ((tl e) : Env) s;
val env_lookup = fn : Env -> string -> int
```

Now, I will do the test cases for env1.sml shown in the assignment sheet to show that each function works properly:

```
- val e1 = env_new ();
val e1 = [] : Env
-
- val e2 = env_bind e1 "x" 100;
val e2 = [("x",100)] : Env
-
- val e3 = env_bind e2 "y" 200;
val e3 = [("y",200),("x",100)] : Env
-
- env_lookup e3 "x";
val it = 100 : int
-
- env_lookup e3 "y";
val it = 200 : int
-
- env_lookup e3 "z";

uncaught exception NameNotBound
  raised at: stdIn:10.37-10.51
```

As you can see, each function is used, and each case is handled properly for env1.sml. Now, for env2.sml

First, the function definitions and their returns to show the typing is correct:

- type Env = string -> int;

type Env = string -> int

-

- fun env_new () : Env = fn(x:string) => 0;

val env_new = fn : unit -> Env

-

- fun env_bind (e : Env) (s:string) (v:int) : Env = env_new();

val env_bind = fn : Env -> string -> int -> Env

-

- fun env_lookup (e : Env) s : int = raise NameNotBound s;

val env_lookup = fn : Env -> string -> int

I was very confused how to go about env_bind and env_lookup, so I'm very certain they are wrong. Nonetheless, their return and input typing is correct. Now, I'll do the example for env2.sml provided in the assignment sheet:

- val e1 = env_new ();

val e1 = fn : Env

-

- val e2 = env_bind e1 "x" 100;

val e2 = fn : Env

-

- val e3 = env_bind e2 "y" 200;

val e3 = fn : Env

-

- env_lookup e3 "x";

uncaught exception NameNotBound

```
  raised at: stdIn:10.42-10.56
-
- env_lookup e3 "y";

uncaught exception NameNotBound
  raised at: stdIn:10.42-10.56
-
- env_lookup e3 "z";

uncaught exception NameNotBound
  raised at: stdIn:10.42-10.56
```

As you can see, due to my incomplete env_bind and env_lookup functions, env2.sml does not function correctly. It works for each case typing, but the results are wrong. I ran out of time for this assignment and couldn't finish it in time was the big issue, so I'll start much earlier in the future. So, my program is organized correctly and declared correctly, but the output is incorrect.

# Source Code

## Env1.sml:

```
(* Environments (Homework 2) *)
(* CSE 4102 Project 2, Spring Semester, 2018 *)
(* Bryan Arnold *)
(* 2/11/2018 *)
(* Section: 001 *)
(* Instructor: Jeffrey A. Meunier *)

(* This is the declaration of the NameNotBound exception. *)
```

```
(* This exception is raised if the given search for a string *)
(* in the list of environments isn't present. *)
(* Use this the env_lookup function to check for this. *)
exception NameNotBound of string;

(* Type declaration for Environment type. *)
(* This is a set of tuples in a list that *)
(* are each composed of a string and integer. *)
(* Need to use this to allow environments to be created. *)
type Env = (string * int) list;

(* Function to create a new environment data structure. *)
(* This creates a new environment data strcuture that is an empty *)
(* list.*)
(* Need to use this to allow environments to have new bindings on *)
(* them.*)
fun env_new () : Env = nil;

(* Function to assign a string and int tuple to a new environment. *)
(* This adds on a new tuple onto the environment list so that .*)
(* a string and int can be associated with each other and fetched *)
(* Use this to add on tuples to an existing environment *)
fun env_bind (e : Env) s v : Env = (s, v)::e;

(* Function to search and return an integer value given it's *)
(* corresponding string associated with it in a given environment. *)
(* If the string doesn't exist within the environment, raise *)
(* the NameNotBound exception indicating it isn't in the envrionment *)
```

```
(* Use this to find a value of a corresponding string in an *)
(* environment, or to check if one exists and display accordingly. *)
fun env_lookup ([] : Env) s = raise NameNotBound s
  | env_lookup (e : Env) s = if #1(hd e) = s then #2(hd e) else env_lookup ((tl e) : Env) s;
```

# Env2.sml:

```
(* Environments (Homework 2) *)
(* CSE 4102 Project 2, Spring Semester, 2018 *)
(* Bryan Arnold *)
(* 2/11/2018 *)
(* Section: 001 *)
(* Instructor: Jeffrey A. Meunier *)

(* This is the declaration of the NameNotBound exception. *)
(* This exception is raised if the given search for a string *)
(* in the list of environments isn't present. *)
(* Use this the env_lookup function to check for this. *)
exception NameNotBound of string;

(* Type declaration for Environment type. *)
(* This is a map of string to int *)
(* so each environment is composed of a string and integer. *)
(* Need to use this to allow environments to be created. *)
type Env = string -> int;

(* Function to create a new environment data structure. *)
(* This creates a new environment data strcuture that is empty.*)
(* Need to use this to allow environments to have new bindings on them. *)
```

```
fun env_new () : Env = fn(x:string) => 0;

(* Function to assign a string and int to a new environment. *)
(* This adds on a new mapping onto a environment so that .*)
(* a string and int can be associated with each other and fetched *)
(* Use this to add values to an individual existing environment *)
fun env_bind (e : Env) (s:string) (v:int) : Env = env_new();

(* Function to search and return an integer value given it's *)
(* corresponding string associated with it in a given environment. *)
(* If the string doesn't exist within the environment, raise *)
(* the NameNotBound exception indicating it isn't in the envrionment *)
(* Use this to find a value of a corresponding string in an *)
(* environment, or to check if one exists and display accordingly. *)
(* NOT FINISHED *)
fun env_lookup (e : Env) s : int = raise NameNotBound s;
```