APPENDIX

A. THE VALUE RESTRICTION IN ML IMPLEMENTATIONS

There are three options of what to do when an unconstrained type cannot be generalised due to the value restriction. CakeML signals a type error, as do Moscow ML and OCaml when compiling. PolyML and SML/NJ create a new abstract type variable, `_a` or `?.X1`, which then ensures the function cannot be called, since there are no values of that type. At the REPL, Moscow ML and OCaml both create special, mutable type variables that are set on the first use and actually change the type of `id_mono`. MLton is included for completeness, but since it is a whole program compiler, this particular issue cannot arise.

A.1 CakeML

```
 -- CakeML starting up --
val id_mono = if true then (fn x => x) else (raise Bind);
<type error>
id_mono 1;
<type error>
```

A.2 PolyML

```
Poly/ML 5.5.2 Release
> val id_mono = if true then (fn x => x) else (raise Bind);
Warning-The type of (id_mono) contains a free type variable. Setting it to a unique
   monotype.
val id_mono = fn: _a -> _a
> id_mono 1;
Error-Type error in function application.
   Function: id_mono : _a -> _a
   Argument: 1 : int
   Reason:
      Can't unify int (*In Basis*) with
      _a (*Constructed from a free type variable.*)
      (Different type constructors)
Found near id_mono 1
Static Errors
```

A.3 SML/NJ

```
Standard ML of New Jersey v110.78 [built: Thu Aug 20 19:23:18 2015]
- val id_mono = if true then (fn x => x) else (raise Bind);
stdIn:1.6-1.58 Warning: type vars not generalized because of
   value restriction are instantiated to dummy types (X1,X2,...)
val id_mono = fn : ?.X1 -> ?.X1
- id_mono 1;
stdIn:2.1-2.10 Error: operator and operand don't agree [overload conflict]
  operator domain: ?.X1
  operand:         [int ty]
  in expression:
    id_mono 1
```

A.4 Moscow ML repl

```
Moscow ML version 2.10
Enter `quit();' to quit.
- val id_mono = if true then (fn x => x) else (raise Bind);
! Warning: Value polymorphism:
! Free type variable(s) at top level in value identifier id_mono
> val id_mono = fn : 'a -> 'a
- id_mono 1;
! Warning: the free type variable 'a has been instantiated to int
> val it = 1 : int
- id_mono;
> val it = fn : int -> int
- id_mono true;
```

```
! Toplevel input:
! id_mono true;
!         ^^^^
! Type clash: expression of type
!   bool
! cannot have type
!   int
```

A.5 Moscow ML compiled

```
dhcp297B:tmp sao$ cat test.sml
structure test = struct
  val id_mono = if true then (fn x => x) else (raise Bind)
end;
dhcp297B:tmp sao$ mosmlc test.sml
! Value polymorphism: Free type variable at top level in value identifier id_mono
```

A.6 OCaml repl

```
        OCaml version 4.02.3

# let id_mono = if true then (fun x -> x) else assert false;;
val id_mono : '_a -> '_a = <fun>
# id_mono 1;;
- : int = 1
# id_mono;;
- : int -> int = <fun>
# id_mono true;;
Error: This expression has type bool but an expression was expected of type
         int
```

A.7 OCaml compiled

```
dhcp297B:tmp sao$ cat test.ml
let id_mono = if true then (fun x -> x) else assert false
dhcp297B:tmp sao$ ocamlc test.ml
File "test.ml", line 1, characters 14-57:
Error: The type of this expression, '_a -> '_a,
       contains type variables that cannot be generalized
```

A.8 MLton

```
dhcp297B:tmp sao$ cat test.sml
structure S = struct
  val id_mono = if true then (fn x => x) else (raise Bind)
end;
val _ = S.id_mono 1;
val _ = S.id_mono true;
dhcp297B:tmp sao$ mlton test.sml
Warning: test.sml 2.3.
  Unable to locally determine type of variable: id_mono.
    type: ??? -> ???
    in: val id_mono = if true then fn x => x else raise Bind
Error: test.sml 5.9.
  Function applied to incorrect argument.
    expects: [int]
    but got: [bool]
    in: S.id_mono true
compilation aborted: parseAndElaborate reported errors
```