A Tase Of ATS

Aditya Siram

June 24, 2019

Outline

ATS

- An ML with ADTs, pattern matching, tail calls
- Can be exactly as good the C equivalent
  - Control over memory
  - Performance
- And type safe.

ATS

- Compiles to *predictable* C
    - Recursion is well supported
- Compiles to *predictable* C
    - Allows C idioms
    - malloc/free, pointers, stack control
- No compiler optimizations except TCO
    - Almost no . . .
- Linear/refinement types, proof level language

ATS

- Extremely difficult
    - Syntax
    - Errors
- But I want to get into the more interesting features

```
implement main0(argc,argv) =
  let
    val a = fopen("test.txt","r")
    val b = fopen("test.txt","r")
    var f = lam@(s:string):void => println! s
  in (
    fwithline(a,f);
    fclose(a);
    fclose(b)
  )
  end
```

```
implement main0(argc,argv) =
  let
    val a = fopen("test.txt","r")


  in (




  )
  end
```

```
datavtype FileHandle = FileHandle of ()
```

```
fun fopen(path:string,mode:string): FileHandle =
  let
    extern castfn toFileHandle(p:ptr0):<> FileHandle
  in
    toFileHandle($extfcall(ptr0,"fopen",path,mode))
  end
```

```
fun fopen(path:string,mode:string): FileHandle =
  let

  in
                 ($extfcall(ptr0,"fopen",path,mode))
  end
```

```
fun fopen(path:string,mode:string): FileHandle =
  let
                   toFileHandle(p:ptr0):<> FileHandle
  in
    toFileHandle($extfcall(ptr0,"fopen",path,mode))
  end
```

```
fun fopen(path:string,mode:string): FileHandle =
  let
    extern castfn toFileHandle(p:ptr0):<> FileHandle
  in
    toFileHandle($extfcall(ptr0,"fopen",path,mode))
  end
```

```
implement main0(argc,argv) =
  let
    val a = fopen("test.txt","r")


  in (




  )
  end
```

```
implement main0(argc,argv) =
  let
    val a = fopen("test.txt","r")
    val b = fopen("test.txt","r")

  in (




  )
  end
```

```
implement main0(argc,argv) =
  let
    val a = fopen("test.txt","r")
    val b = fopen("test.txt","r")
    var f = lam@(s:string):void => println! s
  in (        |
              +--------- stack allocated closure!



  )
  end
```

```
implement main0(argc,argv) =
  let
    val a = fopen("test.txt","r")
    val b = fopen("test.txt","r")
    var f = lam@(s:string):void => println! s
  in (
    fwithline(a,f);



  )
  end
```

```
fun fwithline(
    fh: !FileHandle,
    f: &(string) -<clo1> void
    ):void =
  let









  in

  end
```

```
fun fwithline(
    fh: !FileHandle,
    f: &(string) -<clo1> void
    ):void =
  let







    val _ = $extfcall(int,"getline",        ,    ,          )
  in

  end
```

```
fun fwithline(
    fh: !FileHandle,
    f: &(string) -<clo1> void
    ):void =
  let
    var len = i2sz(0)
    val lenP = addr@len




    val _ = $extfcall(int,"getline",       ,lenP,         )
  in

  end
```

```
fun fwithline(
    fh: !FileHandle,
    f: &(string) -<clo1> void
    ):void =
  let
    var len = i2sz(0)
    val lenP = addr@len
    var buffer = the_null_ptr
    val bufferP = addr@buffer

    val _ = $extfcall(int,"getline",bufferP,lenP,         )
  in

  end
```

```
fun fwithline(
    fh: !FileHandle,
    f: &(string) -<clo1> void
    ):void =
  let
    var len = i2sz(0)
    val lenP = addr@len
    var buffer = the_null_ptr
    val bufferP = addr@buffer
                  toPtr{l:addr}(f: !FileHandle):<> ptr0
    val _ = $extfcall(int,"getline",bufferP,lenP,toPtr(fh))
  in

  end
```

```
fun fwithline(
    fh: !FileHandle,
    f: &(string) -<clo1> void
    ):void =
  let
    var len = i2sz(0)
    val lenP = addr@len
    var buffer = the_null_ptr
    val bufferP = addr@buffer
    extern castfn toPtr{l:addr}(f: !FileHandle):<> ptr0
    val _ = $extfcall(int,"getline",bufferP,lenP,toPtr(fh))
  in

  end
```

```
fun fwithline(
    fh: !FileHandle,
    f: &(string) -<clo1> void
    ):void =
  let



    var buffer = the_null_ptr






  in
    f (                     (buffer))
  end
```

```
fun fwithline(
    fh: !FileHandle,
    f: &(string) -<clo1> void
    ):void =
  let


    var buffer = the_null_ptr




  in
    f ($UN.castvwtp0{string}(buffer))
  end
```

```
implement main0(argc,argv) =
  let
    val a = fopen("test.txt","r")
    val b = fopen("test.txt","r")
    var f = lam@(s:string):void => println! s
  in (
    fwithline(a,f);



  )
  end
```

```
implement main0(argc,argv) =
  let
    val a = fopen("test.txt","r")
    val b = fopen("test.txt","r")
    var f = lam@(s:string):void => println! s
  in (
    fwithline(a,f);
    fclose(a);

  )
  end
```

```
fun fclose(f:FileHandle):void =
  let
    extern castfn fromFH(f:FileHandle):<> ptr0
  in
    $extfcall(void,"fclose",fromFH(f))
  end
```

```
implement main0(argc,argv) =
  let
    val a = fopen("test.txt","r")
    val b = fopen("test.txt","r")
    var f = lam@(s:string):void => println! s
  in (
    fwithline(a,f);
    fclose(a);
    fclose(b)
  )
  end
```

```
fun fwithline(
    fh: !FileHandle,

    ):void =

fun fclose(f: FileHandle):void =
```