A (Not So Gentle) Introduction To Systems Programming In ATS

Aditya Siram

September 28, 2017

Outline

ATS

- Is an ML (not standard)
  - ADTS, pattern-matching etc.
- FP fully supported (TCO)
- Exactly the same performance/memory predictability
  - Decompiles to C
  - No optimizations (except TCO)
  - GCC does the rest
- Exactly the same control of C
  - Pointer arithmetic
  - malloc/free
  - stack allocation
- Completely verified at compile time
  - type system has zero overhead

ATS

- Top 4/5 on The Benchmarks Game
- Now taken down
  - No idea why!

ATS

- Linear logic to manage resources
  - Prove it exists, consume proof, repeat
  - file handles, sockets, anything

ATS

```
fun bar
    ...
    =
  let
     val (awesome_proof | fd) = open_file("some_file.txt")
          ^^^^^^^^^^^^^^
     val contents = read_file (awesome_proof | fd)
                               ^^^^^^^^^^^^^^
     ...
  in
     ...
  end
```

ATS

- But especially memory
  - Prove pointer is initialized, dereference, repeat
  - Type checked pointer arithmetic

ATS

- Refinement types

```
fun foo
    {                              }
    (i : int   ) ...
```

ATS

- Refinement types

```
fun foo
    {                              }
    (i : int n) ...
```

ATS

- Refinement types

```
fun foo
    {n:int                          }
    (i : int n) ...
```

ATS

- Refinement types

```
fun foo
    {n:int | n > 0          }
    (i : int n) ...
```

ATS

- Refinement types

```
fun foo
    {n:int | n > 0 && n < 10}
    (i : int n) ...
```

ATS

- Very Difficult
- Intersects
  - refinement types
  - linear logic
  - proofs
  - C
- Research!
  - Funded by the NSF
- No easy story, or newcomer "onboarding"
- Tiny community
- Sparse docs

Swap

- Easiest way to get started is C interop
- A generic swap in C
  - Yes, I realize 'size_t' is bad!

```
void swap (void* p1, void* p2, size_t size) {
  char* buffer = (char*)malloc(sizeof(char)*size);
  memcpy(buffer, p1, size);
  memcpy(p1, p2, size);
  memcpy(p2, buffer, size);
  free(buffer);
}
```

Swap

- A slightly non-standard swap

```
%{
  #include <stdio.h>
  #include <stdlib.h>
  void swap(void *i, void *j, size_t size) {
    ...
  }
%}
```

Swap

- A slightly non-standard swap

```
%{
  #include <stdio.h>
  #include <stdlib.h>
  void swap(void *i, void *j, size_t size) {
    ...
  }
%}
extern fun swap (i:ptr, j:ptr, s:size_t): void = "ext#swap"
```

Swap

- A slightly non-standard swap

```
%{
  #include <stdio.h>
  #include <stdlib.h>
  void swap(void *i, void *j, size_t size) {
    ...
  }
%}
extern fun swap (i:ptr, j:ptr, s:size_t) : void = "ext#swap"
extern fun malloc(s:size_t):ptr = "ext#malloc"
```

Swap

- Runner

```
implement main0 () =
  let
     val i = malloc(sizeof<int>)
     val j = malloc(sizeof<double>)
     val _ = swap(i,j,sizeof<double>)
  in
     ()
  end
```

Swap

- Runner

```
implement main0 () =
  let
     val i = malloc(sizeof<int>) // all good


  in

  end
```

Swap

- Runner

```
implement main0 () =
  let
     val i = malloc(sizeof<int>)
     val j = malloc(sizeof<double>) // uh oh!

  in

  end
```

Swap

- Runner

```
implement main0 () =
  let
     val i = malloc(sizeof<int>)
     val j = malloc(sizeof<double>)
     val _ = swap(i,j,sizeof<double>) // oh noes!
  in

  end
```

Swap

- Runner

```
implement main0 () =
  let
     val i = malloc(sizeof<int>)
     val j = malloc(sizeof<double>)
     val _ = swap(i,j,sizeof<double>)
  in
     () // free as in leak
  end
```

Swap

- Can totally mimic C
- Including the bugs
- Gradual migration

Swap

- Safe swap

```
extern fun swap (i:ptr, j:ptr, s:size_t) : void = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap                                    : void = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap                                 :          = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
```

```
= "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
```

```
= "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr |                }
                                = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
```

```
= "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
                                  = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (                  i : ptr l1                              ):
                              = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (                  i : ptr l1, j : ptr l2                  ):
                          = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (                    i : ptr l1, j : ptr l2, s: sizeof_t a):
                                  = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (                  | i : ptr l1, j : ptr l2, s: sizeof_t a):
                                  = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (a @ l1           | i : ptr l1, j : ptr l2, s: sizeof_t a):
                          = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (a @ l1 , a @ l2 | i : ptr l1, j : ptr l2, s: sizeof_t a):
                                  = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (a @ l1 , a @ l2 | i : ptr l1, j : ptr l2, s: sizeof_t a):
    (                          ) = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (a @ l1 , a @ l2 | i : ptr l1, j : ptr l2, s: sizeof_t a):
    (                    void) = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (a @ l1 , a @ l2 | i : ptr l1, j : ptr l2, s: sizeof_t a):
    (                | void) = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (a @ l1 , a @ l2 | i : ptr l1, j : ptr l2, s: sizeof_t a):
    (a @ l1          | void) = "ext#swap"
```

Swap

- Safe swap

```
extern fun swap
  {a : t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (a @ l1 , a @ l2 | i : ptr l1, j : ptr l2, s: sizeof_t a):
    (a @ l1, a @ l2 | void) = "ext#swap"
```

Swap

- Safe swap

```
extern fun malloc(s:size_t):ptr = "ext#malloc"
```

Swap

- Safe swap

```
extern fun malloc
```

```
= "ext#malloc"
```

Swap

- Safe swap

```
extern fun malloc
       {a:t@ype}


   = "ext#malloc"
```

Swap

- Safe swap

```
extern fun malloc
       {a:t@ype}
       (s:sizeof_t a):

   = "ext#malloc"
```

Swap

- Safe swap

```
extern fun malloc
       {a:t@ype}
       (s:sizeof_t a):
                              (          ptr l)
   = "ext#malloc"
```

Swap

- Safe swap

```
extern fun malloc
       {a:t@ype}
       (s:sizeof_t a):
                              (a? @ l | ptr l)
  = "ext#malloc"
```

Swap

- Safe swap

```
extern fun malloc
        {a:t@ype}
        (s:sizeof_t a):
        [                    ] (a? @ l | ptr l)
    = "ext#malloc"
```

Swap

- Safe swap

```
extern fun malloc
       {a:t@ype}
       (s:sizeof_t a):
       [l:addr                ] (a? @ l | ptr l)
   = "ext#malloc"
```

Swap

- Safe swap

```
extern fun malloc
       {a:t@ype}
       (s:sizeof_t a):
       [l:addr | l > null] (a? @ l | ptr l)
   = "ext#malloc"
```

Swap

- Safe swap

```
implement main0 () = let
  val (      i) = malloc (sizeof<int>)




in



end
```

Swap

- Safe swap

```
implement main0 () = let
  val (     | i) = malloc (sizeof<int>)




in



end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)




in



end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)



in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val             = ptr_set(      i, 1)


in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val           = ptr_set(pfi | i, 1)


in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (       ()) = ptr_set(pfi | i, 1)


in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)


in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)

in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)
  val                    = swap(                 i, j, sizeof<int>)
in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)
  val                   = swap(           | i, j, sizeof<int>)
in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)
  val                   = swap(pfi1       | i, j, sizeof<int>)
in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)
  val                    = swap(pfi1, pfj2 | i, j, sizeof<int>)
in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)
  val (             ()) = swap(pfi1, pfj2 | i, j, sizeof<int>)
in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)
  val (pfi2      | ()) = swap(pfi1, pfj1 | i, j, sizeof<int>)
in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)
  val (pfi2,pfj2| ()) = swap(pfi1, pfj1 | i, j, sizeof<int>)
in


end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)
  val (pfi2,pfj2| ()) = swap(pfi1, pfj2 | i, j, sizeof<int>)
in
  free(pfi2 | i);

end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi | i) = malloc (sizeof<int>)
  val (pfj | j) = malloc (sizeof<int>)
  val (pfi1 | ()) = ptr_set(pfi | i, 1)
  val (pfj1 | ()) = ptr_set(pfj | j, 2)
  val (pfi2,pfj2| ()) = swap(pfi1, pfj1 | i, j, sizeof<int>)
in
  free(pfi2 | i);
  free(pfj2 | j);
end
```

Swap

- Safe swap

```
implement main0 () = let
  val (pfi    ) = malloc
       ^^^
  val (pfi1 | ()) = ptr_set(pfi |     )
       ^^^^                   ^^^

in


end
```

Swap

- Safe swap

```
implement main0 () = let


  val (pfi1 | ()) =
       ^^^^
  val (pfi2,     | ()) == swap(pfi1,       |                    )
in     ^^^^                    ^^^^


end
```

Swap

- Safe swap

```
implement main0 () = let




  val (pfi2,     | ()) =
in      ^^^^
  free(pfi2 |  );
       ^^^^
end
```

Swap

- Idiomatic swap

```
fun {a:t@ype}
    swap{l1,l2:addr}
    (..) : void =
  let
    val tmp = !p1
  in
    !p1 := !p2;
    !p2 := tmp
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
        .<n>.
        (acc : double, i : int (n)) : double =
      case- i of
      | 1 => acc
      | i when i > 1 => loop(acc * i, i - 1)

  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial


  let
    fun loop









  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }

  let
    fun loop
```

```
  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
```

```
  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }

        (acc : double, i : int (n)) : double =




  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
        .<n>.
        (acc : double, i : int (n)) : double =




  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
        .<n>.
        (acc : double, i : int (n)) : double =
      case- i of



  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
        .<n>.
        (acc : double, i : int (n)) : double =
      case- i of
      | 1 => acc
      |

  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
        .<n>.
        (acc : double, i : int (n)) : double =
      case- i of
      | 1 => acc
      | i

  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
        .<n>.
        (acc : double, i : int (n)) : double =
      case- i of
      | 1 => acc
      | i when i > 1

  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
        .<n>.
        (acc : double, i : int (n)) : double =
      case- i of
      | 1 => acc
      | i when i > 1 => loop(acc * i, i - 1)

  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial


  let
    fun loop
        { n : int | n >= 1 } <---


      case- i of
      |
      | i when i > 1 => loop(acc * i, i - 1)
          ^^^^^^^^^^
  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial


  let
    fun loop
        { n : int | n >= 1 } <---


      case- i of
      |
      | i when i > 1 => loop(acc * i, i - 1)
                                     ^^^^^

  in
    loop(1.0, i)
  end
```

Factorial

- Factorial

```
fun factorial


  let
    fun loop

        .<n>. <---

      case- i of
      |
      | i when i > 1 => loop(acc * i, i + 1)
                                      ^^^^^

  in
    loop(1.0, i)
  end
```

Viewtype

- Remember 'swap'?

```
extern fun swap
  {a:t@ype}
  {l1: addr | l1 > null}
  {l2: addr | l2 > null}
  (a @ l1 , a @ l2 | i : ptr l1, j : ptr l2, s: sizeof_t a):
    (a @ l1, a @ l2 | void) = "ext#swap"
```

Viewtype

- Remember 'swap'?

```
extern fun swap
  {a:t@ype}
  {l1: addr | l1 > null}

  (a @ l1              | i : ptr l1                    ):
```

Viewtype

- Remember 'swap'?

```
extern fun swap
  {a:t@ype}
  {l1: addr | l1 > null}

  (a @ l1           | i : ptr l1                    ):

sortdef ...

viewtypedef ...
```

Viewtype

- Remember 'swap'?

```
extern fun swap
  {a:t@ype}
  {l1: addr | l1 > null}
  ^^^^^^^^^^^^^^^^^^^^^^
  (a @ l1           | i : ptr l1                              ):

sortdef agz = {l:addr | l > null}
              ^^^^^^^^^^^^^^^^^^

viewtypedef ...
```

Viewtype

- Remember 'swap'?

```
extern fun swap
  {a:t@ype}
  {l1: addr | l1 > null}
  ^^^^^^^^^^^^^^^^^^^^^^
  (a @ l1           | i : ptr l1                              ):
   ^^^^^^                 ^^^^^^

sortdef agz = {l:addr | l > null}
              ^^^^^^^^^^^^^^^^^^

viewtypedef safePtr(a:t@ype) = [l:agz] (a @ l | ptr l)
                                        ^^^^^   ^^^^^
```

Viewtype

- Remember 'swap'?

```
extern fun swap
  {a:t@ype}
  {l1: addr | l1 > null}

  (a @ l1           | i : ptr l1                              ):

sortdef agz = {l:addr | l > null}
        ^^^
viewtypedef safePtr(a:t@ype) = [l:agz] (a @ l | ptr l)
                               ^^^^^^^
```

Viewtype

- Remember 'swap'?

```
extern fun swap
  {a:t@ype}
  {l1: addr | l1 > null}

  (a @ l1              | i : ptr l1                    ):
```

Viewtype

- Remember 'swap'?

```
extern fun swap
  {a:t@ype}


  (                    i : safePtr a                    ):
```

Viewtypes

- Viewtypes are the basic building block
- Can create algebras of linear resources!

Algebraic datatypes

- Build ADTs top of view types!

```
dataviewtype option_vt (a:viewt@ype, bool)
  =  Some_vt(a, true) of a
   | None_vt(a, false)
```

Algebraic datatypes

- Build ADTs top of view types!

```
dataviewtype option_vt (a:viewt@ype, bool)
  =  Some_vt
   | None_vt
```

Algebraic datatypes

- Build ADTs top of view types!

```
dataviewtype option_vt (a:viewt@ype, bool)
  =  Some_vt(a, true) of a
   | None_vt
```

Algebraic datatypes

- Build ADTs top of view types!

```
dataviewtype option_vt (a:viewt@ype, bool)
  =  Some_vt(a, true) of a
   | None_vt(a, false)
```

Algebraic datatypes

- Linear lists

```
dataviewtype list_vt
  (a:viewt@ype, int) =
  | list_vt_nil(a, 0) of ()
  | {n:int | n > 0}
    list_vt_cons(a, n) of (a, list_vt(a, n-1))
```

Algebraic datatypes

- Linear lists

```
dataviewtype list_vt
  (a:viewt@ype, int) =
  | list_vt_nil
  |
    list_vt_cons
```

Algebraic datatypes

- Linear lists

```
dataviewtype list_vt
  (a:viewt@ype, int) =
  | list_vt_nil(a, 0) of ()
  |
    list_vt_cons
```

Algebraic datatypes

- Linear lists

```
dataviewtype list_vt
  (a:viewt@ype, int) =
  | list_vt_nil(a, 0) of ()
  |
    list_vt_cons(a, n)
```

Algebraic datatypes

- Linear lists

```
dataviewtype list_vt
  (a:viewt@ype, int) =
  | list_vt_nil(a, 0) of ()
  |
    list_vt_cons(a, n) of (a, list_vt(a, n-1))
```

Algebraic datatypes

- Linear lists

```
dataviewtype list_vt
  (a:viewt@ype, int) =
  | list_vt_nil(a, 0) of ()
  | {n:int | n > 0}
    list_vt_cons(a, n) of (a, list_vt(a, n-1))
```