A (Not So Gentle) Introduction To ATS

Aditya Siram

September 17, 2017

Outline

Overview

- A Lisp
- Pattern matching
- Optional Types
- Built in YACC

Swap

- A slightly non-standard swap

```
void swap(void *i, void *j, size_t size) {
  void* tmp = malloc(size);
  memcpy(tmp, j, size);
  memcpy(j, i, size);
  memcpy(i, tmp, size);
  free(tmp);
}
```

Swap

- A slightly non-standard swap

```
void swap(void *i, void *j, size_t size) {
  void* tmp = malloc(size);




}
```

Swap

- A slightly non-standard swap

```
void swap(void *i, void *j, size_t size) {
  void* tmp = malloc(size);
  memcpy(tmp, j, size);
  memcpy(j, i, size);
  memcpy(i, tmp, size);

}
```

Swap

- A slightly non-standard swap

```
void swap(void *i, void *j, size_t size) {
  void* tmp = malloc(size);
  memcpy(tmp, j, size);
  memcpy(j, i, size);
  memcpy(i, tmp, size);
  free(tmp);
}
```

Swap

- A slightly non-standard swap

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

Swap

- A slightly non-standard swap

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

Swap

- A slightly non-standard swap

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

- Safe swap

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

Swap

- Safe swap

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

Swap

- Safe swap

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

Swap

- Safe swap

```
extern fun swap
```

```
= "mac#swap"
```

Swap

- Safe swap

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

```
= "mac#swap"
```

Swap

- Safe swap

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

Swap

- Safe swap

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

```
= "mac#swap"
```

Swap

- Safe swap

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

Swap

- Safe swap

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

Swap

- Safe swap

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

Swap

- Safe swap

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

Swap

- Safe swap

```
extern fun malloc

    = "mac#malloc"
```

Swap

- Safe swap

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


   = "mac#malloc"
```

Swap

- Safe swap

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

   = "mac#malloc"
```

Swap

- Safe swap

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

Swap

- Safe swap

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

Swap

- Safe swap

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

Swap

- Safe swap

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

Swap

- Safe swap

```
extern fun malloc
       {a:t@ype}
       (s:sizeof_t a):
       [l:addr | l > null] (a? @ l | ptr l)
   = "mac#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)
  val _ = 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 _ = ptr_set(pfi | i, 1)
  val _ = 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 _ = ptr_set(pfi | i, 1)
  val _ = 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 _ = ptr_set(pfi | i, 1)
  val _ = ptr_set(pfj | j, 2)
  val                   = swap(pfi      | 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 _ = ptr_set(pfi | i, 1)
  val _ = ptr_set(pfj | j, 2)
  val                  = swap(pfi, pfj | 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 _ = ptr_set(pfi | i, 1)
  val _ = ptr_set(pfj | j, 2)
  val (             ()) = swap(pfi, pfj | 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 _ = ptr_set(pfi | i, 1)
  val _ = ptr_set(pfj | j, 2)
  val (           | ()) = swap(pfi, pfj | 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 _ = ptr_set(pfi | i, 1)
  val _ = ptr_set(pfj | j, 2)
  val (pfi1      | ()) = swap(pfi, pfj | 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 _ = ptr_set(pfi | i, 1)
  val _ = ptr_set(pfj | j, 2)
  val (pfi1,pfj1| ()) = swap(pfi, pfj | 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 _ = ptr_set(pfi | i, 1)
  val _ = ptr_set(pfj | j, 2)
  val (pfi1,pfj1| ()) = swap(pfi, pfj | i, j, sizeof<int>)
in
  free(pfi1 | i);

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)
  val _ = ptr_set(pfj | j, 2)
  val (pfi1,pfj1| ()) = swap(pfi, pfj | i, j, sizeof<int>)
in
  free(pfi1 | i);
  free(pfj1 | j);
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 =
      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
        { 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 }
        (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 }
        (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 }
        (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 }
        (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 }
        (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
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
        (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
    { n : int | n >= 1 }
    (i : int n) : double =
  let
    fun loop
        { n : int | n >= 1 }
        (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
```