galois

Exercise 1:

SAW file:

```
import "SHA512.cry";

let Sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_0 x }});
};

let Sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_1 x }});
};

let sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_0 x }});
};

let sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_1 x }});
};

let Ch_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   y <- llvm_fresh_var "y" (llvm_int 64);
   z <- llvm_fresh_var "z" (llvm_int 64);
   llvm_execute_func [llvm_term x, llvm_term y, llvm_term z];
   llvm_return (llvm_term {{ Ch x y z }});
};

let main : TopLevel () = do {
   m <- llvm_load_module "sha512.bc";
   Sigma0_ov <- llvm_verify m "Sigma0" [] true Sigma0_setup z3;
   Sigma1_ov <- llvm_verify m "Sigma1" [] true Sigma1_setup z3;
   sigma0_ov <- llvm_verify m "sigma0" [] true sigma0_setup z3;
   sigma1_ov <- llvm_verify m "sigma1" [] true sigma1_setup z3;
   Ch_ov <- llvm_verify m "Ch" [] true Ch_setup z3;
   print "Done!";
};
```

Running saw on the above file:

```
[16:28:31.471] Loading file ".../s1.saw"
[16:28:31.711] Verifying Sigma0 ...
[16:28:31.728] Simulating Sigma0 ...
[16:28:31.735] Checking proof obligations Sigma0 ...
[16:28:31.806] Proof succeeded! Sigma0
[16:28:31.860] Verifying Sigma1 ...
[16:28:31.876] Simulating Sigma1 ...
[16:28:31.882] Checking proof obligations Sigma1 ...
```

```
[16:28:32.000] Proof succeeded! Sigma1
[16:28:32.053] Verifying sigma0 ...
[16:28:32.072] Simulating sigma0 ...
[16:28:32.077] Checking proof obligations sigma0 ...
[16:28:32.152] Proof succeeded! sigma0
[16:28:32.207] Verifying sigma1 ...
[16:28:32.225] Simulating sigma1 ...
[16:28:32.230] Checking proof obligations sigma1 ...
[16:28:32.310] Proof succeeded! sigma1
[16:28:32.366] Verifying Ch ...
[16:28:32.384] Simulating Ch ...
[16:28:32.388] Checking proof obligations Ch ...
[16:28:32.420] Proof succeeded! Ch
[16:28:32.420] Done!
```

Exercise 2:

SAW file:

```
import "SHA512.cry";

let alloc_init ty v = do {
   p <- llvm_alloc ty;
   llvm_points_to p v;
   return p;
};

let pointer_to_fresh n ty = do {
   x <- llvm_fresh_var n ty;
   p <- alloc_init ty (llvm_term x);
   return (x, p);
};

let Sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_0 x }});
};

let Sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_1 x }});
};

let sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_0 x }});
};

let sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_1 x }});
};
```

```
let sha512_block_data_order_setup = do {
   (state,state_ptr) <- pointer_to_fresh "state" (llvm_array 8 (llvm_int 64));
   (data,data_ptr) <- pointer_to_fresh "data" (llvm_array 128 (llvm_int 8));

   llvm_execute_func [state_ptr, data_ptr, llvm_term {{ 1 : [64] }}];

   llvm_points_to state_ptr
         (llvm_term {{ processBlock_Common state (split (join data)) }});
};

let main : TopLevel () = do {
   m <- llvm_load_module "sha512.bc";
   Sigma0_ov <- llvm_verify m "Sigma0" [] true Sigma0_setup z3;
   Sigma1_ov <- llvm_verify m "Sigma1" [] true Sigma1_setup z3;
   sigma0_ov <- llvm_verify m "sigma0" [] true sigma0_setup z3;
   sigma1_ov <- llvm_verify m "sigma1" [] true sigma1_setup z3;
   sha512_block_data_order_ov <- llvm_verify m "sha512_block_data_order"
      [Sigma0_ov, Sigma1_ov, sigma0_ov, sigma1_ov] true
      sha512_block_data_order_setup
      (w4_unint_z3 ["SIGMA_0", "SIGMA_1", "sigma_0", "sigma_1"]);
   print "Done!";
};
```

Running saw on the above file:

```
[16:47:12.133] Loading file ".../s2.saw"
[16:47:12.452] Verifying Sigma0 ...
[16:47:12.471] Simulating Sigma0 ...
[16:47:12.477] Checking proof obligations Sigma0 ...
[16:47:12.555] Proof succeeded! Sigma0
[16:47:12.606] Verifying Sigma1 ...
[16:47:12.623] Simulating Sigma1 ...
[16:47:12.628] Checking proof obligations Sigma1 ...
[16:47:12.747] Proof succeeded! Sigma1
[16:47:12.798] Verifying sigma0 ...
[16:47:12.817] Simulating sigma0 ...
[16:47:12.821] Checking proof obligations sigma0 ...
[16:47:12.896] Proof succeeded! sigma0
[16:47:12.946] Verifying sigma1 ...
[16:47:12.964] Simulating sigma1 ...
[16:47:12.968] Checking proof obligations sigma1 ...
[16:47:13.049] Proof succeeded! sigma1
[16:47:13.150] Verifying sha512_block_data_order ...
[16:47:13.169] Simulating sha512_block_data_order ...
[16:47:13.511] Registering overrides for `Sigma0`
[16:47:13.511]   variant `Symbol "Sigma0"`
[16:47:13.511] Registering overrides for `Sigma1`
[16:47:13.511]   variant `Symbol "Sigma1"`
[16:47:13.511] Registering overrides for `sigma0`
[16:47:13.511]   variant `Symbol "sigma0"`
[16:47:13.511] Registering overrides for `sigma1`
[16:47:13.511]   variant `Symbol "sigma1"`
```

```
[16:47:13.559] Matching 1 overrides of  Sigma1 ...
[16:47:13.560] Branching on 1 override variants of Sigma1 ...
[16:47:13.560] Applied override! Sigma1
[16:47:13.561] Matching 1 overrides of  Sigma0 ...
[16:47:13.561] Branching on 1 override variants of Sigma0 ...
[16:47:13.561] Applied override! Sigma0
  ...
[16:47:13.623] Matching 1 overrides of  sigma0 ...
[16:47:13.623] Branching on 1 override variants of sigma0 ...
[16:47:13.624] Applied override! sigma0
[16:47:13.624] Matching 1 overrides of  sigma1 ...
[16:47:13.624] Branching on 1 override variants of sigma1 ...
[16:47:13.624] Applied override! sigma1
  ...
[16:47:14.928] Checking proof obligations sha512_block_data_order ...
[16:47:16.810] Proof succeeded! sha512_block_data_order
[16:47:16.810] Done!
```

Exercise 3:

```
SAW file:
  import "SHA512.cry";
  let points_to_sha512_state_st_common ptr (h, sz, block, n) num = do {
     llvm_points_to (llvm_field ptr "h") (llvm_term h);
     llvm_points_to_at_type (llvm_field ptr "Nl") (llvm_int 128) (llvm_term sz);

     if eval_bool {{ `num == 0 }} then do {
        return ();
     } else do {
        llvm_points_to_untyped (llvm_field ptr "p") (llvm_term block);
     };

     llvm_points_to (llvm_field ptr "num") (llvm_term n);
     llvm_points_to (llvm_field ptr "md_len") (llvm_term {{ `64 : [32] }});
  };
  let points_to_sha512_state_st ptr state num = do {
     points_to_sha512_state_st_common ptr
     ({{ state.h }}, {{ state.sz }}, {{ take`{num} state.block }}, {{ state.n }})
     num;
  };
  let SHA512_Init_setup = do {
     sha_ptr <- llvm_alloc (llvm_struct "struct.sha512_state_st");
     llvm_execute_func [sha_ptr];
     points_to_sha512_state_st
        sha_ptr
        {{ { h = SHAH0, block = zero : [128][8], n = 0 : [32], sz = 0 : [128] } }} 0;
        llvm_return (llvm_term {{ 1 : [32] }});
  };
```

```
let main : TopLevel () = do {
   m <- llvm_load_module "sha512.bc";
   SHA512_Init_ov <- llvm_verify m "SHA512_Init" [] true SHA512_Init_setup z3;
   print "Done!";
};
```

Running saw on the above file:

```
[17:13:13.430] Loading file ".../s3.saw"
[17:13:13.995] Verifying SHA512_Init ...
[17:13:14.011] Simulating SHA512_Init ...
[17:13:14.020] Checking proof obligations SHA512_Init ...
[17:13:14.020] Proof succeeded! SHA512_Init
[17:13:14.020] Done!
```

Exercise 4:

SAW file:

```
import "SHA512.cry";

let alloc_init ty v = do {
   p <- llvm_alloc ty;
   llvm_points_to p v;
   return p;
};

let pointer_to_fresh n ty = do {
   x <- llvm_fresh_var n ty;
   p <- alloc_init ty (llvm_term x);
   return (x, p);
};

let points_to_sha512_state_st_common ptr (h, sz, block, n) num = do {
   llvm_points_to (llvm_field ptr "h") (llvm_term h);
   llvm_points_to_at_type (llvm_field ptr "Nl") (llvm_int 128) (llvm_term sz);
   if eval_bool {{ `num == 0 }} then do {
      return ();
   } else do {
      llvm_points_to_untyped (llvm_field ptr "p") (llvm_term block);
   };
   llvm_points_to (llvm_field ptr "num") (llvm_term n);
   llvm_points_to (llvm_field ptr "md_len") (llvm_term {{ `64 : [32] }});
};

let points_to_sha512_state_st ptr state num = do {
   points_to_sha512_state_st_common
      ptr
      ({{ state.h }}, {{ state.sz }}, {{ take`{num} state.block }}, {{ state.n }}) num;
};
```

```
let pointer_to_fresh_sha512_state_st name n = do {
   h <- llvm_fresh_var (str_concat name ".h") (llvm_array 8 (llvm_int 64));
   block <- if eval_bool {{ `n == 0 }} then do {
      return {{ [] : [0][8] }};
   } else do {
      llvm_fresh_var (str_concat name ".block") (llvm_array n (llvm_int 8));
   };
   sz <- llvm_fresh_var (str_concat name ".sz") (llvm_int 128);
   let state = {{ { h = h, block = (block # zero) : [128][8], n = `n : [32], sz = sz } }};
   ptr <- llvm_alloc (llvm_struct "struct.sha512_state_st");
   points_to_sha512_state_st_common ptr (h, sz, block, {{ `n : [32]}}) n;
   return (state, ptr);
};

let Sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_0 x }});
};

let Sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_1 x }});
};

let sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_0 x }});
};

let sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_1 x }});
};

let sha512_block_data_order_setup = do {
   (state, state_ptr) <- pointer_to_fresh "state" (llvm_array 8 (llvm_int 64));
   (data, data_ptr) <- pointer_to_fresh "data" (llvm_array 128 (llvm_int 8));

   llvm_execute_func [state_ptr, data_ptr, llvm_term {{ 1 : [64] }}];

   llvm_points_to state_ptr
       (llvm_term {{ processBlock_Common state (split (join data)) }});
};

let SHA512_Update_setup num len = do {
   (sha512_ctx, sha_ptr) <- pointer_to_fresh_sha512_state_st "sha512_ctx" num;
   (data, data_ptr) <- pointer_to_fresh "data" (llvm_array len (llvm_int 8));

   llvm_execute_func [sha_ptr, data_ptr, llvm_term {{ `len : [64] }}];

   points_to_sha512_state_st
      sha_ptr
      {{ SHAUpdate sha512_ctx data }} (eval_size {| (num + len) % 128 |});
      llvm_return (llvm_term {{ 1 : [32] }});
};

let main : TopLevel () = do {
   m <- llvm_load_module "sha512.bc";
   Sigma0_ov <- llvm_verify m "Sigma0" [] true Sigma0_setup z3;
```

```
    Sigma1_ov <- llvm_verify m "Sigma1" [] true Sigma1_setup z3;
    sigma0_ov <- llvm_verify m "sigma0" [] true sigma0_setup z3;
    sigma1_ov <- llvm_verify m "sigma1" [] true sigma1_setup z3;
    sha512_block_data_order_ov <- llvm_verify m "sha512_block_data_order"
                          [Sigma0_ov, Sigma1_ov, sigma0_ov, sigma1_ov] true
                          sha512_block_data_order_setup
                          (w4_unint_z3 ["SIGMA_0","SIGMA_1","sigma_0","sigma_1"]);
    SHA512_Update_0_240_ov <- llvm_verify m "SHA512_Update"
                          [sha512_block_data_order_ov] true
                          (SHA512_Update_setup 0 240)
                          (w4_unint_z3 ["processBlock_Common"]);
    SHA512_Update_0_127_ov <- llvm_verify m "SHA512_Update"
                          [sha512_block_data_order_ov] true
                          (SHA512_Update_setup 0 127)
                          (w4_unint_z3 ["processBlock_Common"]);
    SHA512_Update_127_241_ov <- llvm_verify m "SHA512_Update"
                          [sha512_block_data_order_ov] true
                          (SHA512_Update_setup 127 241)
                          (w4_unint_z3 ["processBlock_Common"]);
    print "Done!";
  };
```

Running saw on the above file:

```
  [17:45:50.642] Loading file ".../s4.saw"
  [17:45:50.884] Verifying Sigma0 ...
  [17:45:50.902] Simulating Sigma0 ...
  [17:45:50.908] Checking proof obligations Sigma0 ...
  [17:45:50.984] Proof succeeded! Sigma0
  [17:45:51.074] Verifying Sigma1 ...
  [17:45:51.091] Simulating Sigma1 ...
  [17:45:51.095] Checking proof obligations Sigma1 ...
  [17:45:51.169] Proof succeeded! Sigma1
  [17:45:51.222] Verifying sigma0 ...
  [17:45:51.239] Simulating sigma0 ...
  [17:45:51.242] Checking proof obligations sigma0 ...
  [17:45:51.316] Proof succeeded! sigma0
  [17:45:51.366] Verifying sigma1 ...
  [17:45:51.384] Simulating sigma1 ...
  [17:45:51.387] Checking proof obligations sigma1 ...
  [17:45:51.453] Proof succeeded! sigma1
  [17:45:51.550] Verifying sha512_block_data_order ...
  [17:45:51.568] Simulating sha512_block_data_order ...
  ...
  [17:45:53.300] Checking proof obligations sha512_block_data_order ...
  [17:45:55.140] Proof succeeded! sha512_block_data_order
  [17:45:55.893] Verifying SHA512_Update ...
  [17:45:55.913] Simulating SHA512_Update ...
  [17:45:55.921] Registering overrides for `sha512_block_data_order`
  [17:45:55.921]   variant `Symbol "sha512_block_data_order"`
  [17:45:55.941] Matching 1 overrides of  sha512_block_data_order ...
  [17:45:55.943] Branching on 1 override variants of sha512_block_data_order ...
  [17:45:55.944] Applied override! sha512_block_data_order
  [17:45:55.962] Checking proof obligations SHA512_Update ...
  [17:45:56.363] Proof succeeded! SHA512_Update
  [17:45:57.198] Verifying SHA512_Update ...
  [17:45:57.218] Simulating SHA512_Update ...
  [17:45:57.219] Registering overrides for `sha512_block_data_order`
```

```
[17:45:57.219]   variant `Symbol "sha512_block_data_order"`
[17:45:57.246] Checking proof obligations SHA512_Update ...
[17:45:57.595] Proof succeeded! SHA512_Update
[17:45:58.304] Verifying SHA512_Update ...
[17:45:58.325] Simulating SHA512_Update ...
[17:45:58.326] Registering overrides for `sha512_block_data_order`
[17:45:58.326]   variant `Symbol "sha512_block_data_order"`
[17:45:58.348] Matching 1 overrides of  sha512_block_data_order ...
[17:45:58.349] Branching on 1 override variants of sha512_block_data_order ...
[17:45:58.350] Applied override! sha512_block_data_order
[17:45:58.351] Matching 1 overrides of  sha512_block_data_order ...
[17:45:58.352] Branching on 1 override variants of sha512_block_data_order ...
[17:45:58.353] Applied override! sha512_block_data_order
[17:45:58.375] Checking proof obligations SHA512_Update ...
[17:45:58.859] Proof succeeded! SHA512_Update
[17:45:58.859] Done!
```

Exercise 5:

SAW file:

```
import "SHA512.cry";

let alloc_init ty v = do {
   p <- llvm_alloc ty;
   llvm_points_to p v;
   return p;
};

let pointer_to_fresh n ty = do {
   x <- llvm_fresh_var n ty;
   p <- alloc_init ty (llvm_term x);
   return (x, p);
};

let points_to_sha512_state_st_common ptr (h, sz, block, n) num = do {
   llvm_points_to (llvm_field ptr "h") (llvm_term h);
   llvm_points_to_at_type (llvm_field ptr "Nl") (llvm_int 128) (llvm_term sz);
   if eval_bool {{ `num == 0 }} then do {
      return ();
   } else do {
      llvm_points_to_untyped (llvm_field ptr "p") (llvm_term block);
   };
   llvm_points_to (llvm_field ptr "num") (llvm_term n);
   llvm_points_to (llvm_field ptr "md_len") (llvm_term {{ `64 : [32] }});
};

let pointer_to_fresh_sha512_state_st name n = do {
   h <- llvm_fresh_var (str_concat name ".h") (llvm_array 8 (llvm_int 64));
   block <- if eval_bool {{ `n == 0 }} then do {
      return {{ [] : [0][8] }};
   } else do {
      llvm_fresh_var (str_concat name ".block") (llvm_array n (llvm_int 8));
   };
   sz <- llvm_fresh_var (str_concat name ".sz") (llvm_int 128);
   let state = {{ { h = h, block = (block # zero) : [128][8], n = `n : [32], sz = sz } }};
   ptr <- llvm_alloc (llvm_struct "struct.sha512_state_st");
   points_to_sha512_state_st_common ptr (h, sz, block, {{ `n : [32]}}) n;
   return (state, ptr);
};
```

```
let Sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_0 x }});
};
let Sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_1 x }});
};
let sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_0 x }});
};
let sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_1 x }});
};
let sha512_block_data_order_setup = do {
   (state, state_ptr) <- pointer_to_fresh "state" (llvm_array 8 (llvm_int 64));
   (data, data_ptr) <- pointer_to_fresh "data" (llvm_array 128 (llvm_int 8));
   llvm_execute_func [state_ptr, data_ptr, llvm_term {{ 1 : [64] }}];
   llvm_points_to state_ptr
         (llvm_term {{ processBlock_Common state (split (join data)) }});
};
let SHA512_Final_setup num = do {
   out_ptr <- llvm_alloc (llvm_array 64 (llvm_int 8));
   (sha512_ctx, sha_ptr) <- pointer_to_fresh_sha512_state_st "sha512_ctx" num;
   llvm_execute_func [out_ptr, sha_ptr];
   llvm_points_to out_ptr (llvm_term {{ split`{64} (SHAFinal sha512_ctx) }});
   llvm_return (llvm_term {{ 1 : [32] }});
};
let main : TopLevel () = do {
   m <- llvm_load_module "sha512.bc";
   Sigma0_ov <- llvm_verify m "Sigma0" [] true Sigma0_setup z3;
   Sigma1_ov <- llvm_verify m "Sigma1" [] true Sigma1_setup z3;
   sigma0_ov <- llvm_verify m "sigma0" [] true sigma0_setup z3;
   sigma1_ov <- llvm_verify m "sigma1" [] true sigma1_setup z3;
   sha512_block_data_order_ov <- llvm_verify m "sha512_block_data_order"
       [Sigma0_ov, Sigma1_ov, sigma0_ov, sigma1_ov] true
       sha512_block_data_order_setup
       (w4_unint_z3 ["SIGMA_0", "SIGMA_1", "sigma_0", "sigma_1"]);
   SHA512_Final_111_ov <- llvm_verify m "SHA512_Final"
       [sha512_block_data_order_ov] true
       (SHA512_Final_setup 111)
       (w4_unint_z3 ["processBlock_Common"]);
   SHA512_Final_112_ov <- llvm_verify m "SHA512_Final"
       [sha512_block_data_order_ov] true
       (SHA512_Final_setup 112)
       (w4_unint_z3 ["processBlock_Common"]);
   print "Done!"; };
```

Running saw on the file

```
[18:26:49.225] Loading file ".../s5.saw"
[18:26:49.464] Verifying Sigma0 ...
[18:26:49.481] Simulating Sigma0 ...
[18:26:49.488] Checking proof obligations Sigma0 ...
[18:26:49.563] Proof succeeded! Sigma0
[18:26:49.612] Verifying Sigma1 ...
[18:26:49.628] Simulating Sigma1 ...
[18:26:49.632] Checking proof obligations Sigma1 ...
[18:26:49.747] Proof succeeded! Sigma1
[18:26:49.796] Verifying sigma0 ...
[18:26:49.815] Simulating sigma0 ...
[18:26:49.819] Checking proof obligations sigma0 ...
[18:26:49.891] Proof succeeded! sigma0
[18:26:49.944] Verifying sigma1 ...
[18:26:49.962] Simulating sigma1 ...
[18:26:49.966] Checking proof obligations sigma1 ...
[18:26:50.042] Proof succeeded! sigma1
[18:26:50.143] Verifying sha512_block_data_order ...
[18:26:50.161] Simulating sha512_block_data_order ...
[18:26:50.509] Registering overrides for `Sigma0`
[18:26:50.509]   variant `Symbol "Sigma0"`
[18:26:50.509] Registering overrides for `Sigma1`
[18:26:50.509]   variant `Symbol "Sigma1"`
[18:26:50.511] Registering overrides for `sigma0`
[18:26:50.511]   variant `Symbol "sigma0"`
[18:26:50.511] Registering overrides for `sigma1`
[18:26:50.511]   variant `Symbol "sigma1"`
...
[18:26:51.912] Checking proof obligations sha512_block_data_order ...
[18:26:53.755] Proof succeeded! sha512_block_data_order
[18:26:54.100] Verifying SHA512_Final ...
[18:26:54.119] Simulating SHA512_Final ...
[18:26:54.126] Registering overrides for `sha512_block_data_order`
[18:26:54.126]   variant `Symbol "sha512_block_data_order"`
[18:26:54.130] Matching 1 overrides of  sha512_block_data_order ...
[18:26:54.134] Branching on 1 override variants of sha512_block_data_order ...
[18:26:54.136] Applied override! sha512_block_data_order
[18:26:54.161] Checking proof obligations SHA512_Final ...
[18:26:54.247] Proof succeeded! SHA512_Final
[18:26:54.602] Verifying SHA512_Final ...
[18:26:54.621] Simulating SHA512_Final ...
[18:26:54.622] Registering overrides for `sha512_block_data_order`
[18:26:54.622]   variant `Symbol "sha512_block_data_order"`
[18:26:54.623] Matching 1 overrides of  sha512_block_data_order ...
[18:26:54.626] Branching on 1 override variants of sha512_block_data_order ...
[18:26:54.627] Applied override! sha512_block_data_order
[18:26:54.631] Matching 1 overrides of  sha512_block_data_order ...
[18:26:54.633] Branching on 1 override variants of sha512_block_data_order ...
[18:26:54.634] Applied override! sha512_block_data_order
[18:26:54.657] Checking proof obligations SHA512_Final ...
[18:26:54.740] Proof succeeded! SHA512_Final
[18:26:54.740] Done!
```

Exercise 6:

SAW file:

```
import "SHA512.cry";

let alloc_init ty v = do {
   p <- llvm_alloc ty;
   llvm_points_to p v;
   return p;
};

let pointer_to_fresh n ty = do {
   x <- llvm_fresh_var n ty;
   p <- alloc_init ty (llvm_term x);
   return (x, p);
};

let points_to_sha512_state_st_common ptr (h, sz, block, n) num = do {
   llvm_points_to (llvm_field ptr "h") (llvm_term h);
   llvm_points_to_at_type (llvm_field ptr "Nl") (llvm_int 128) (llvm_term sz);
   if eval_bool {{ `num == 0 }} then do {
      return ();
   } else do {
      llvm_points_to_untyped (llvm_field ptr "p") (llvm_term block);
   };
   llvm_points_to (llvm_field ptr "num") (llvm_term n);
   llvm_points_to (llvm_field ptr "md_len") (llvm_term {{ `64 : [32] }});
};

let points_to_sha512_state_st ptr state num = do {
  points_to_sha512_state_st_common
    ptr
    ({{ state.h }}, {{ state.sz }}, {{ take`{num} state.block }}, {{ state.n }}) num;
};

let pointer_to_fresh_sha512_state_st name n = do {
   h <- llvm_fresh_var (str_concat name ".h") (llvm_array 8 (llvm_int 64));
   block <- if eval_bool {{ `n == 0 }} then do {
      return {{ [] : [0][8] }};
   } else do {
      llvm_fresh_var (str_concat name ".block") (llvm_array n (llvm_int 8));
   };
   sz <- llvm_fresh_var (str_concat name ".sz") (llvm_int 128);
   let state = {{ { h = h, block = (block # zero) : [128][8], n = `n : [32], sz = sz } }};
   ptr <- llvm_alloc (llvm_struct "struct.sha512_state_st");
   points_to_sha512_state_st_common ptr (h, sz, block, {{ `n : [32]}}) n;
   return (state, ptr);
};

let Sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_0 x }});
};

let Sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ SIGMA_1 x }});
};
```

```
let sigma0_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_0 x }});
};
let sigma1_setup = do {
   x <- llvm_fresh_var "x" (llvm_int 64);
   llvm_execute_func [llvm_term x];
   llvm_return (llvm_term {{ sigma_1 x }});
};
let sha512_block_data_order_setup = do {
   (state, state_ptr) <- pointer_to_fresh "state" (llvm_array 8 (llvm_int 64));
   (data, data_ptr) <- pointer_to_fresh "data" (llvm_array 128 (llvm_int 8));
   llvm_execute_func [state_ptr, data_ptr, llvm_term {{ 1 : [64] }}];
   llvm_points_to state_ptr
         (llvm_term {{ processBlock_Common state (split (join data)) }});
};
let SHA512_Update_setup num len = do {
   (sha512_ctx, sha_ptr) <- pointer_to_fresh_sha512_state_st "sha512_ctx" num;
   (data, data_ptr) <- pointer_to_fresh "data" (llvm_array len (llvm_int 8));
   llvm_execute_func [sha_ptr, data_ptr, llvm_term {{ `len : [64] }}];
   points_to_sha512_state_st
      sha_ptr
      {{ SHAUpdate sha512_ctx data }} (eval_size {| (num + len) % 128 |});
      llvm_return (llvm_term {{ 1 : [32] }});
};
let SHA512_Final_setup num = do {
   out_ptr <- llvm_alloc (llvm_array 64 (llvm_int 8));
   (sha512_ctx, sha_ptr) <- pointer_to_fresh_sha512_state_st "sha512_ctx" num;
   llvm_execute_func [out_ptr, sha_ptr];
   llvm_points_to out_ptr (llvm_term {{ split`{64} (SHAFinal sha512_ctx) }});
   llvm_return (llvm_term {{ 1 : [32] }});
};
let SHA512_setup len = do {
   (data, data_ptr) <- pointer_to_fresh "data" (llvm_array len (llvm_int 8));
   out_ptr <- llvm_alloc (llvm_array 64 (llvm_int 8));
   llvm_execute_func [ data_ptr, llvm_term {{ `len : [64] }}, out_ptr];
   llvm_points_to out_ptr (llvm_term {{ split`{64} (SHAImp data) }});
   llvm_return out_ptr;
};

let main : TopLevel () = do {
   m <- llvm_load_module "sha512.bc";
   Sigma0_ov <- llvm_verify m "Sigma0" [] true Sigma0_setup z3;
   Sigma1_ov <- llvm_verify m "Sigma1" [] true Sigma1_setup z3;
   sigma0_ov <- llvm_verify m "sigma0" [] true sigma0_setup z3;
   sigma1_ov <- llvm_verify m "sigma1" [] true sigma1_setup z3;
   sha512_block_data_order_ov <- llvm_verify m "sha512_block_data_order"
      [Sigma0_ov, Sigma1_ov, sigma0_ov, sigma1_ov] true
      sha512_block_data_order_setup
      (w4_unint_z3 ["SIGMA_0", "SIGMA_1", "sigma_0", "sigma_1"]);
```

```
    Update_0_240_ov <- llvm_verify m "SHA512_Update"
       [sha512_block_data_order_ov] true
       (SHA512_Update_setup 0 240)
       (w4_unint_z3 ["processBlock_Common"]);
    Final_112_ov <- llvm_verify m "SHA512_Final"
       [sha512_block_data_order_ov] true
       (SHA512_Final_setup 112)
       (w4_unint_z3 ["processBlock_Common"]);
    llvm_verify m "SHA512" [Update_0_240_ov, Final_112_ov] true
       (SHA512_setup 240)
       (w4_unint_z3 ["processBlock_Common"]);
    print "Done!";
 };
```

Running saw on the above file

```
[18:43:42.481] Loading file ".../s6.saw"
[18:43:42.731] Verifying Sigma0 ...
[18:43:42.747] Simulating Sigma0 ...
[18:43:42.754] Checking proof obligations Sigma0 ...
[18:43:42.825] Proof succeeded! Sigma0
[18:43:42.917] Verifying Sigma1 ...
[18:43:42.935] Simulating Sigma1 ...
[18:43:42.939] Checking proof obligations Sigma1 ...
[18:43:43.011] Proof succeeded! Sigma1
[18:43:43.063] Verifying sigma0 ...
[18:43:43.079] Simulating sigma0 ...
[18:43:43.084] Checking proof obligations sigma0 ...
[18:43:43.151] Proof succeeded! sigma0
[18:43:43.203] Verifying sigma1 ...
[18:43:43.220] Simulating sigma1 ...
[18:43:43.226] Checking proof obligations sigma1 ...
[18:43:43.301] Proof succeeded! sigma1
[18:43:43.399] Verifying sha512_block_data_order ...
[18:43:43.417] Simulating sha512_block_data_order ...
[18:43:43.751] Registering overrides for `Sigma0`
[18:43:43.751]   variant `Symbol "Sigma0"`
[18:43:43.751] Registering overrides for `Sigma1`
[18:43:43.751]   variant `Symbol "Sigma1"`
[18:43:43.751] Registering overrides for `sigma0`
[18:43:43.751]   variant `Symbol "sigma0"`
[18:43:43.751] Registering overrides for `sigma1`
[18:43:43.751]   variant `Symbol "sigma1"`
...
[18:43:45.166] Checking proof obligations sha512_block_data_order ...
[18:43:47.042] Proof succeeded! sha512_block_data_order
[18:43:47.439] Verifying SHA512_Init ...
[18:43:47.458] Simulating SHA512_Init ...
[18:43:47.465] Checking proof obligations SHA512_Init ...
[18:43:47.466] Proof succeeded! SHA512_Init
[18:43:48.221] Verifying SHA512_Update ...
[18:43:48.241] Simulating SHA512_Update ...
[18:43:48.253] Registering overrides for `sha512_block_data_order`
[18:43:48.253]   variant `Symbol "sha512_block_data_order"`
[18:43:48.271] Matching 1 overrides of  sha512_block_data_order ...
[18:43:48.273] Branching on 1 override variants of sha512_block_data_order ...
[18:43:48.274] Applied override! sha512_block_data_order
[18:43:48.293] Checking proof obligations SHA512_Update ...
```

```
[18:43:48.775] Proof succeeded! SHA512_Update
[18:43:49.124] Verifying SHA512_Final ...
[18:43:49.142] Simulating SHA512_Final ...
[18:43:49.149] Registering overrides for `sha512_block_data_order`
[18:43:49.149]   variant `Symbol "sha512_block_data_order"`
[18:43:49.151] Matching 1 overrides of  sha512_block_data_order ...
[18:43:49.153] Branching on 1 override variants of sha512_block_data_order ...
[18:43:49.154] Applied override! sha512_block_data_order
[18:43:49.157] Matching 1 overrides of  sha512_block_data_order ...
[18:43:49.159] Branching on 1 override variants of sha512_block_data_order ...
[18:43:49.160] Applied override! sha512_block_data_order
[18:43:49.185] Checking proof obligations SHA512_Final ...
[18:43:49.269] Proof succeeded! SHA512_Final
[18:43:49.369] Verifying SHA512 ...
[18:43:49.388] Simulating SHA512 ...
[18:43:49.390] Registering overrides for `SHA512_Final`
[18:43:49.391]   variant `Symbol "SHA512_Final"`
[18:43:49.391] Registering overrides for `SHA512_Init`
[18:43:49.391]   variant `Symbol "SHA512_Init"`
[18:43:49.391] Registering overrides for `SHA512_Update`
[18:43:49.391]   variant `Symbol "SHA512_Update"`
[18:43:49.391] Matching 1 overrides of  SHA512_Init ...
[18:43:49.391] Branching on 1 override variants of SHA512_Init ...
[18:43:49.395] Applied override! SHA512_Init
[18:43:49.395] Matching 1 overrides of  SHA512_Update ...
[18:43:49.398] Branching on 1 override variants of SHA512_Update ...
[18:43:49.403] Applied override! SHA512_Update
[18:43:49.403] Matching 1 overrides of  SHA512_Final ...
[18:43:49.407] Branching on 1 override variants of SHA512_Final ...
[18:43:49.416] Applied override! SHA512_Final
[18:43:49.416] Symbolic simulation completed with side conditions.
[18:43:49.419] Checking proof obligations SHA512 ...
[18:43:49.725] Proof succeeded! SHA512
[18:43:49.725] Done!
```