galois

Exercise 1:

Make sure the directory in which `s1.saw` is located contains `SHA256.cry`, `SHA.cry`, and `sha512.bc` which is created using clang.

SAW file `s1.saw`:

```
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" [] false Sigma0_setup z3;
   Sigma1_ov <- llvm_verify m "Sigma1" [] false Sigma1_setup z3;
   sigma0_ov <- llvm_verify m "sigma0" [] false sigma0_setup z3;
   sigma1_ov <- llvm_verify m "sigma1" [] false sigma1_setup z3;
   Ch_ov <- llvm_verify m "Ch" [] false 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 `s2.saw`:

```
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" [] false Sigma0_setup z3;
   Sigma1_ov <- llvm_verify m "Sigma1" [] false Sigma1_setup z3;
   sigma0_ov <- llvm_verify m "sigma0" [] false sigma0_setup z3;
   sigma1_ov <- llvm_verify m "sigma1" [] false sigma1_setup z3;
   sha512_bdo_ov <- llvm_verify m "sha512_block_data_order"
      [Sigma0_ov, Sigma1_ov, sigma0_ov, sigma1_ov] false
      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 `s3.saw`:

```
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 n = do {
   h <- llvm_fresh_var "sha512_ctx.h" (llvm_array 8 (llvm_int 64));
   block <- if eval_bool {{ `n == 0 }} then do {
      return {{ [] : [0][8] }};
   } else do {
      llvm_fresh_var "sha512_ctx.block" (llvm_array n (llvm_int 8));
   };
   sz <- llvm_fresh_var "sha512_ctx.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 = do {
   (sha512_ctx,sha_ptr) <- pointer_to_fresh_sha512_state_st 0;
   (data,data_ptr) <- pointer_to_fresh "data" (llvm_array 127 (llvm_int 8));
   llvm_execute_func [sha_ptr, data_ptr, llvm_term {{ `127 : [64] }}];
   points_to_sha512_state_st sha_ptr {{ SHAUpdate sha512_ctx data }} 127;
   llvm_return (llvm_term {{ 1 : [32] }});
};
```

```
   let main : TopLevel () = do {
      m <- llvm_load_module "sha512.bc";
      Sigma0_ov <- llvm_verify m "Sigma0" [] false Sigma0_setup z3;
      Sigma1_ov <- llvm_verify m "Sigma1" [] false Sigma1_setup z3;
      sigma0_ov <- llvm_verify m "sigma0" [] false sigma0_setup z3;
      sigma1_ov <- llvm_verify m "sigma1" [] false sigma1_setup z3;
      sha512_bdo_ov <- llvm_verify m "sha512_block_data_order"
         [Sigma0_ov, Sigma1_ov, sigma0_ov, sigma1_ov] false
         sha512_block_data_order_setup
         (w4_unint_z3 ["SIGMA_0", "SIGMA_1", "sigma_0", "sigma_1"]);
      update_ov <- llvm_verify m "SHA512_Update"
         [sha512_bdo_ov] false SHA512_Update_setup
         (w4_unint_z3 ["processBlock_Common"]);
   print "Done!";
};
```

Running saw on the above file:

```
   [19:19:54.881] Loading file ".../s3.saw"
   ...
   [19:20:00.525] Verifying SHA512_Update ...
   [19:20:00.528] Simulating SHA512_Update ...
   [19:20:00.541] Registering overrides for `sha512_block_data_order`
   [19:20:00.541]   variant `Symbol "sha512_block_data_order"`
   [19:20:00.556] Checking proof obligations SHA512_Update ...
   [19:20:00.945] Proof succeeded! SHA512_Update
   [19:20:00.945] Done!
```

Exercise 4:

SAW file `s4.saw`:

```
   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 n = do {
   h <- llvm_fresh_var "sha512_ctx.h" (llvm_array 8 (llvm_int 64));
   block <- if eval_bool {{ `n == 0 }} then do {
      return {{ [] : [0][8] }};
   } else do {
      llvm_fresh_var "sha512_ctx.block" (llvm_array n (llvm_int 8));
   };
   sz <- llvm_fresh_var "sha512_ctx.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 = do {
   (sha512_ctx,sha_ptr) <- pointer_to_fresh_sha512_state_st 0;
   (data,data_ptr) <- pointer_to_fresh "data" (llvm_array 127 (llvm_int 8));
   llvm_execute_func [sha_ptr, data_ptr, llvm_term {{ `127 : [64] }}];
```

```
   points_to_sha512_state_st sha_ptr {{ SHAUpdate sha512_ctx data }} 127;
   llvm_return (llvm_term {{ 1 : [32] }});
};
let SHA512_Final_setup = do {
   out_ptr <- llvm_alloc (llvm_array 64 (llvm_int 8));
   (sha512_ctx, sha_ptr) <- pointer_to_fresh_sha512_state_st 127;
   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" [] false Sigma0_setup z3;
   Sigma1_ov <- llvm_verify m "Sigma1" [] false Sigma1_setup z3;
   sigma0_ov <- llvm_verify m "sigma0" [] false sigma0_setup z3;
   sigma1_ov <- llvm_verify m "sigma1" [] false sigma1_setup z3;
   sha512_bdo_ov <- llvm_verify m "sha512_block_data_order"
      [Sigma0_ov, Sigma1_ov, sigma0_ov, sigma1_ov] false
      sha512_block_data_order_setup
      (w4_unint_z3 ["SIGMA_0", "SIGMA_1", "sigma_0", "sigma_1"]);
   update_ov <- llvm_verify m "SHA512_Update"
      [sha512_bdo_ov] false SHA512_Update_setup
      (w4_unint_z3 ["processBlock_Common"]);
   final_ov <- llvm_verify m "SHA512_Final"
      [sha512_bdo_ov] false SHA512_Final_setup
      (w4_unint_z3 ["processBlock_Common"]);
   print "Done!";
};
```

Running saw on the above file:

```
[19:23:42.148] Loading file ".../s4.saw"
...
[19:23:48.165] Verifying SHA512_Final ...
[19:23:48.167] Simulating SHA512_Final ...
[19:23:48.172] Registering overrides for `sha512_block_data_order`
[19:23:48.172]   variant `Symbol "sha512_block_data_order"`
[19:23:48.173] Matching 1 overrides of  sha512_block_data_order ...
[19:23:48.175] Branching on 1 override variants of sha512_block_data_order ...
[19:23:48.176] Applied override! sha512_block_data_order
[19:23:48.179] Matching 1 overrides of  sha512_block_data_order ...
[19:23:48.181] Branching on 1 override variants of sha512_block_data_order ...
[19:23:48.182] Applied override! sha512_block_data_order
[19:23:48.209] Checking proof obligations SHA512_Final ...
[19:23:48.300] Proof succeeded! SHA512_Final
[19:23:48.300] Done!
```

Exercise 5:

SAW file `s5.saw`:

```
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 n = do {
   h <- llvm_fresh_var "sha512_ctx.h" (llvm_array 8 (llvm_int 64));
   block <- if eval_bool {{ `n == 0 }} then do {
      return {{ [] : [0][8] }};
   } else do {
      llvm_fresh_var "sha512_ctx.block" (llvm_array n (llvm_int 8));
   };
   sz <- llvm_fresh_var "sha512_ctx.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 = do {
   (sha512_ctx,sha_ptr) <- pointer_to_fresh_sha512_state_st 0;
   (data,data_ptr) <- pointer_to_fresh "data" (llvm_array 127 (llvm_int 8));

   llvm_execute_func [sha_ptr, data_ptr, llvm_term {{ `127 : [64] }}];

   points_to_sha512_state_st sha_ptr {{ SHAUpdate sha512_ctx data }} 127;
   llvm_return (llvm_term {{ 1 : [32] }});
};
let SHA512_Final_setup = do {
   out_ptr <- llvm_alloc (llvm_array 64 (llvm_int 8));
   (sha512_ctx,sha_ptr) <- pointer_to_fresh_sha512_state_st 127;

   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 = do {
   (data,data_ptr) <- pointer_to_fresh "data" (llvm_array 127 (llvm_int 8));
   out_ptr <- llvm_alloc (llvm_array 64 (llvm_int 8));

   llvm_execute_func [ data_ptr, llvm_term {{ `127 : [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" [] false Sigma0_setup z3;
   Sigma1_ov <- llvm_verify m "Sigma1" [] false Sigma1_setup z3;
   sigma0_ov <- llvm_verify m "sigma0" [] false sigma0_setup z3;
   sigma1_ov <- llvm_verify m "sigma1" [] false sigma1_setup z3;
   sha512_bdo_ov <- llvm_verify m "sha512_block_data_order"
      [Sigma0_ov, Sigma1_ov, sigma0_ov, sigma1_ov] false
      sha512_block_data_order_setup
      (w4_unint_z3 ["SIGMA_0", "SIGMA_1", "sigma_0", "sigma_1"]);
   update_ov <- llvm_verify m "SHA512_Update"
      [sha512_bdo_ov] false SHA512_Update_setup
      (w4_unint_z3 ["processBlock_Common"]);
   final_ov <- llvm_verify m "SHA512_Final"
      [sha512_bdo_ov] false SHA512_Final_setup
      (w4_unint_z3 ["processBlock_Common"]);
   llvm_verify m "SHA512"
      [update_ov, final_ov] false SHA512_setup
      (w4_unint_z3 ["processBlock_Common"]);
   print "Done!";
};
```

Running saw on the file

```
[19:27:00.340] Loading file ".../s5.saw"
...
[19:27:06.775] Verifying SHA512 ...
[19:27:06.776] Simulating SHA512 ...
[19:27:06.778] Registering overrides for `SHA512_Final`
[19:27:06.778]   variant `Symbol "SHA512_Final"`
[19:27:06.778] Registering overrides for `SHA512_Update`
[19:27:06.778]   variant `Symbol "SHA512_Update"`
[19:27:06.780] Matching 1 overrides of  SHA512_Update ...
[19:27:06.783] Branching on 1 override variants of SHA512_Update ...
[19:27:06.787] Applied override! SHA512_Update
[19:27:06.787] Matching 1 overrides of  SHA512_Final ...
[19:27:06.790] Branching on 1 override variants of SHA512_Final ...
[19:27:06.792] Applied override! SHA512_Final
[19:27:06.793] Symbolic simulation completed with side conditions.
[19:27:06.795] Checking proof obligations SHA512 ...
[19:27:07.035] Proof succeeded! SHA512
[19:27:07.035] Done!
```

Exercise 6:

Modified `sha512.c`:

```
...
int main (int argc, char **argv) {
   int i=0;
   uint8_t *out = (uint8_t *)malloc(SHA512_DIGEST_LENGTH);
   int len = strlen(argv[1]);
   out = SHA512(argv[1], len, out);
   printf("0x");
   for (i=0 ; i < 64 ; i++)
      if (out[i] < 16) printf("0%1x",out[i]); else printf("%2x",out[i]);
   printf("\n");
}
```

Running sha512:

```
prompt> make sha512
cc sha512.c -o sha512
prompt> sha512 “Hello World Folks”
0xf9255b38dcc5b3538012414153f932042397215b9733a6a569a3405569aa17ce23ddd4eb9872f4
b8d3356bd06d7e38aaadff364ade2c6ca8d6465bded5c1b8cc
```

In Cryptol:

```
:l SHA512.cry
Loading module Cryptol
Loading module `where` argument of SHA512
Loading interface module `parameter` interface of SHA
Loading module SHA
Loading module SHA512
SHA512> SHAImp "Hello World Folks"
0xf9255b38dcc5b3538012414153f932042397215b9733a6a569a3405569aa17ce23ddd4eb9872f4
b8d3356bd06d7e38aaadff364ade2c6ca8d6465bded5c1b8cc
SHA512>
```