EECS 219C: Formal Methods — Assignment 2

Parker Ziegler

March 18, 2022

1. Interrupt-Driven Program

a. Describing properties of the `Sys` module

We can describe the properties of the `Sys` module as follows:

1. `invariant main_ISR_mutex` — this property requires that execution of `main` and `ISR` is mutually exclusive. That is, if `main` is executing, `ISR` cannot be executing at the same time (or vice versa).
2. `property[LTL] one_step_ISR_return` — this property requires that, globally, if `ISR` has just returned then, in the next state, `ISR` will not return.
3. `property[LTL] main_after_ISR` — this property requires that, globally, if `ISR` is currently enabled to run and, in the next state, `main` is enabled to run, this implies that `ISR` has just returned.
4. `property[LTL] ISR_after_main` — this property requires that, globally, if `main` is enabled and, in the next state, `ISR` is enabled, this implies that an interrupt has occurred.

b. Interpreting counterexamples from the verifier

Running `uclid` with all properties commented out *except* for `main_after_ISR` results in the following counterexample:

```
CEX for vobj [Step #3] property main_after_ISR:safety @ IntSW.ucl, line 105
==================================
Step #0
  mode : main_t
  M_enable : true
  I_enable : false
  return_ISR : false
  assert_intr : initial_1570_assert_intr
[Assertion Failure]: More than one definition found!
```

This counterexample is found for step 3 in our transition system. In this case

Likewise, running `uclid` with all properties commented out *except* for `ISR_after_main` results in the following counterexample:

```
CEX for vobj [Step #2] property ISR_after_main:safety @ IntSW.ucl, line 106
==================================
Step #0
  mode : main_t
  M_enable : true
  I_enable : false
  return_ISR : false
  assert_intr : false
```

```
[Assertion Failure]: More than one definition found!
```