3/2 - Traffic Lights in Forge

Wellness advocates:

- wellness.advocates@cs.brown.edu
- Email them - accessibility, health, wellness, etc.
- Also hiring!

Software verification: Traffic lights

- Four-way intersection with traffic lights north-south and east-west
- What do the states of the system look like?
    - Have to know what color your traffic lights are e.g. (R, R)
        - Light could be off
        - We could have multiple colors on one light
          > We are abstracting this out
    - We start off at (R, R), what should the rest of the rest of the transitions be?
        - No event fields, simply a tick forward in the system
- What property do we need to hold for the intersection to be **safe**?
    - It's always true that at least one of the lights is red
    - We'd like to never get to a state that could result in an accident
    - *Bad things never happen.*
- What other property do we want?
    - We'd like everyone to eventually get to move at some point - each of the lights should at some point transition to green
        - **Liveness**
    - *Good things happen at some point*
- What do the counterexample traces look like?
    - For **safety**: e.g. RR → RG → GG
        - We can find this with search (e.g. depth-first search)
        - Has finite counterexamples
    - For **liveness**: we need an infinite trace!
        - In a finite graph, we just need to find a cycle
        - Pigeonhole principle
            - We only have 9 states, but traces can be arbitrarily long or infinite, so some states must repeat
        - Has infinite-size counterexamples (it suffices to find a 'prefix' of the violating property, all extensions of which also violate the counterexample)
- It's very difficult to check liveness in Forge
    - We have finite states to work with and counterexamples to liveness are infinite