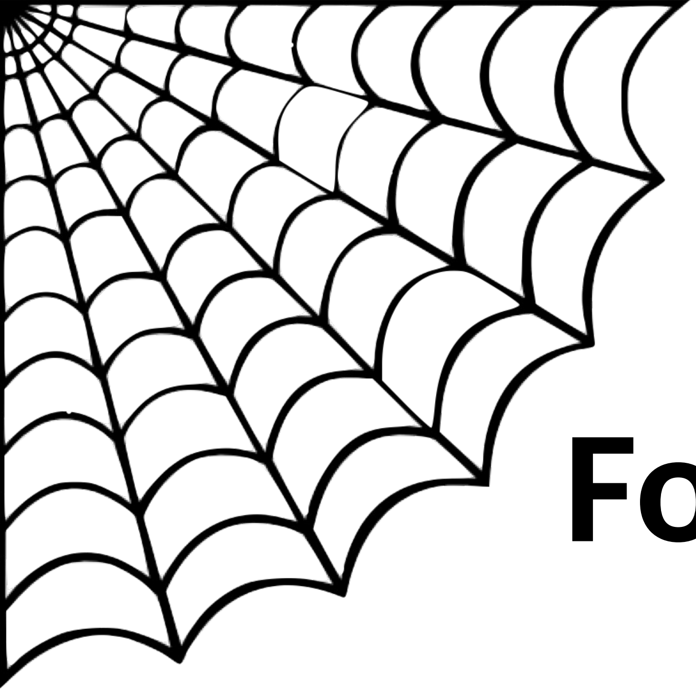
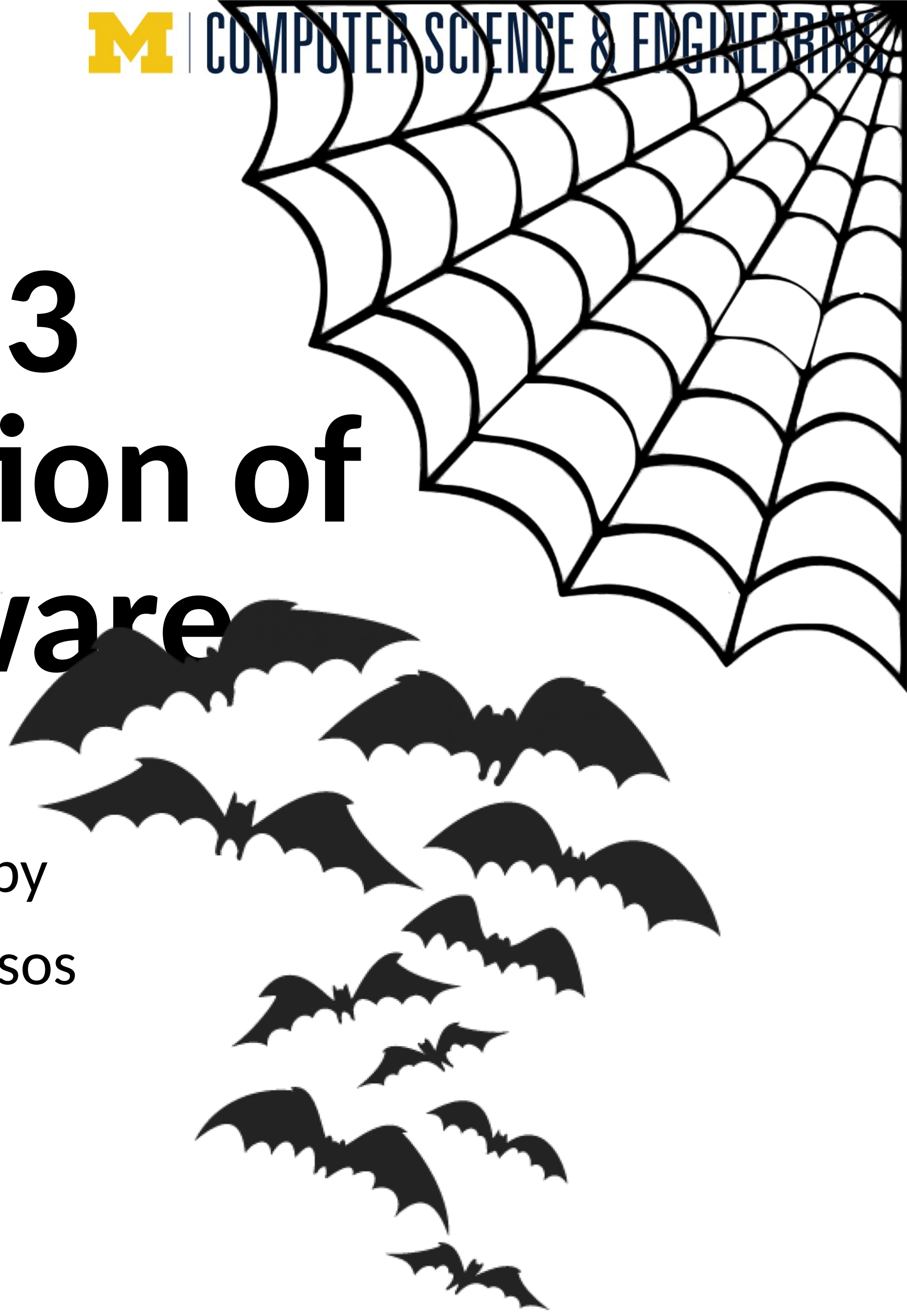
COMPUTER SCIENCE & ENGINEERING

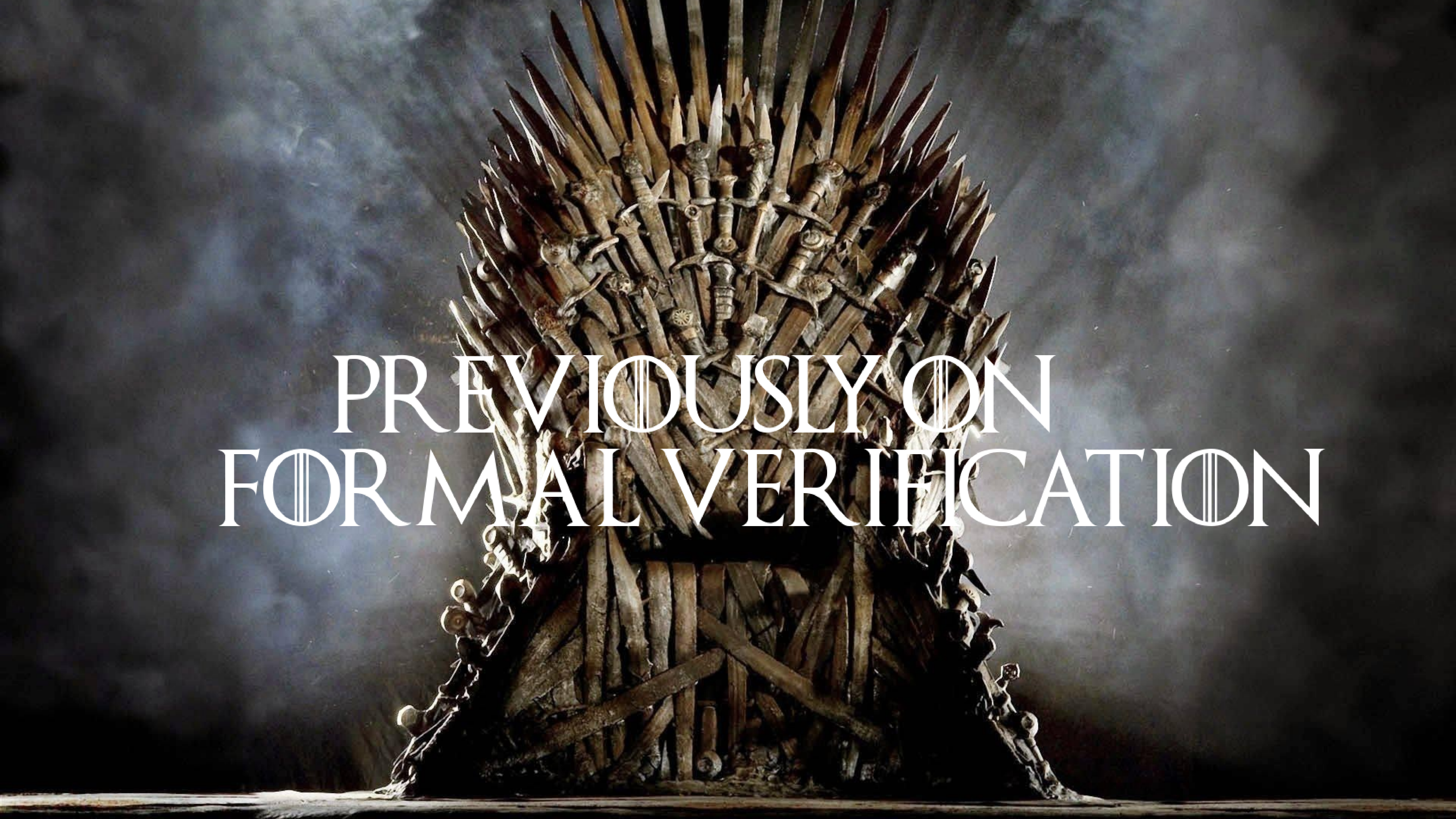




EECS498-003 Formal Verification of Systems Software

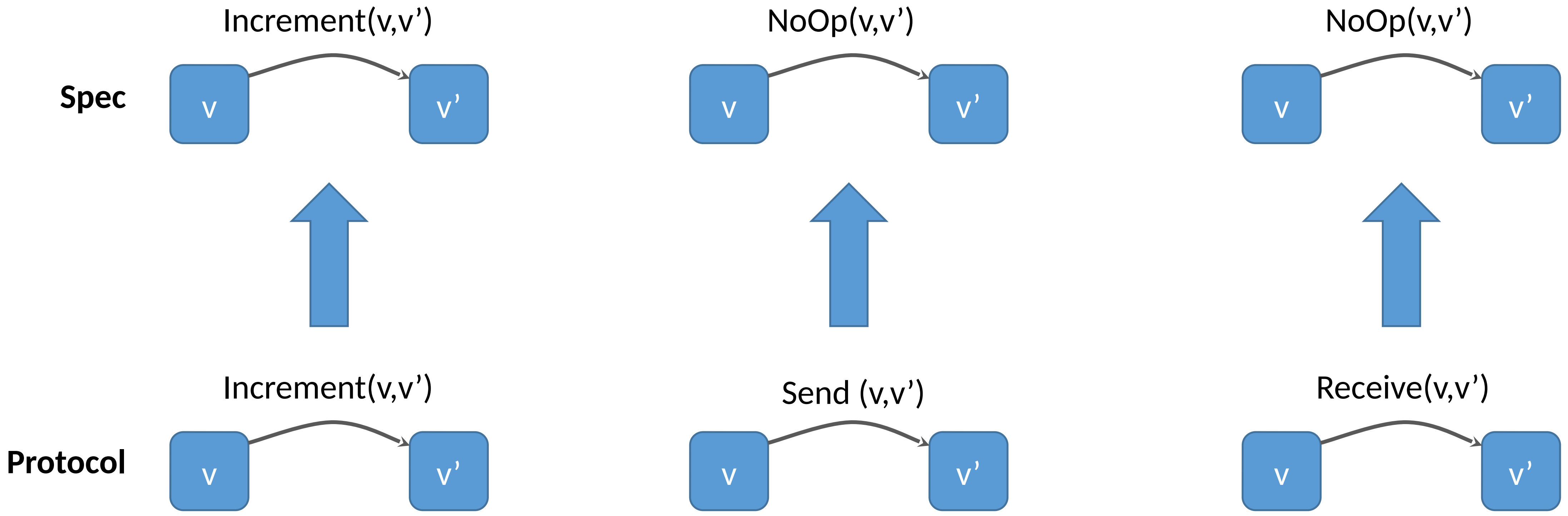
Material and slides created by

Jon Howell and Manos Kapritsos



PREVIOUSLY ON FORMAL VERIFICATION

Case study: a moving counter

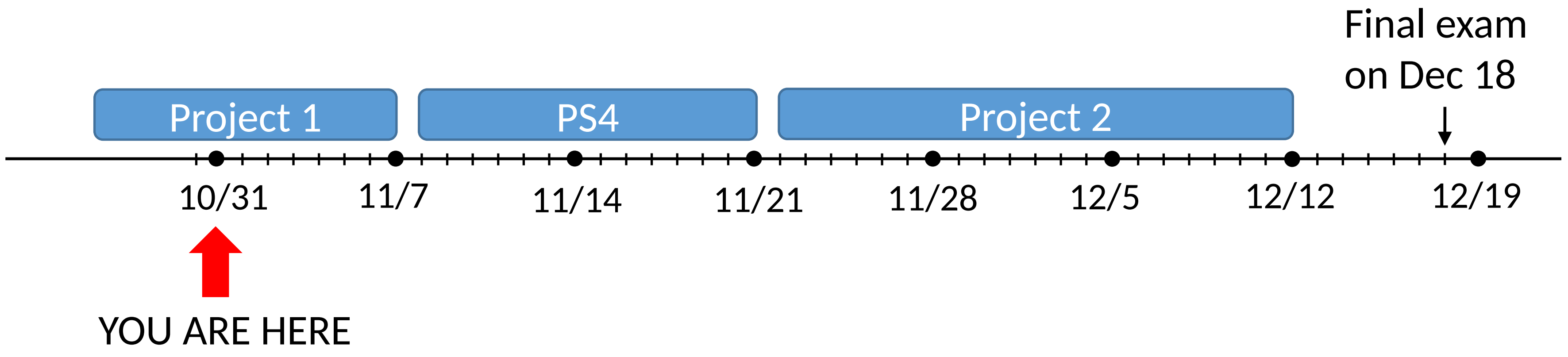


VSCode transition

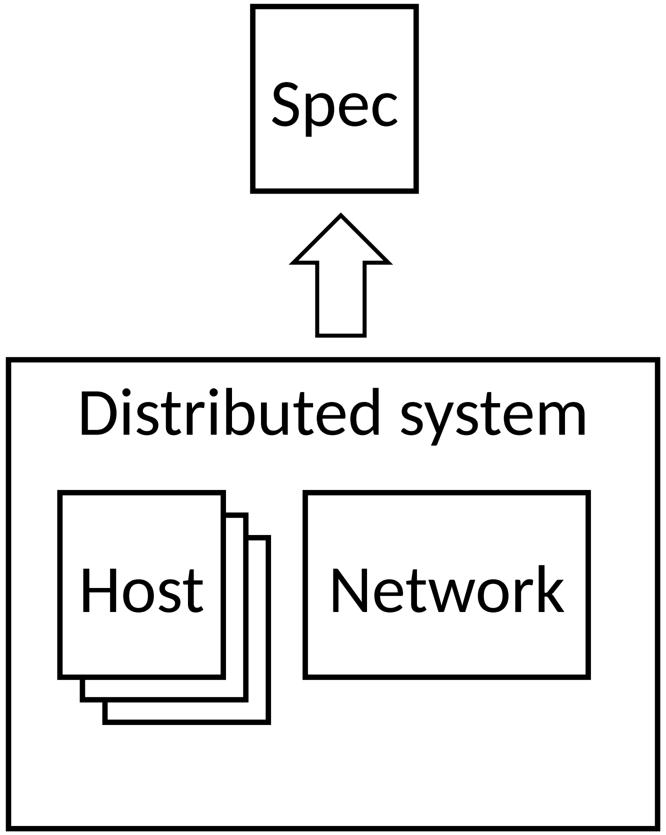
Administrivia

- No class this Tuesday, November 5

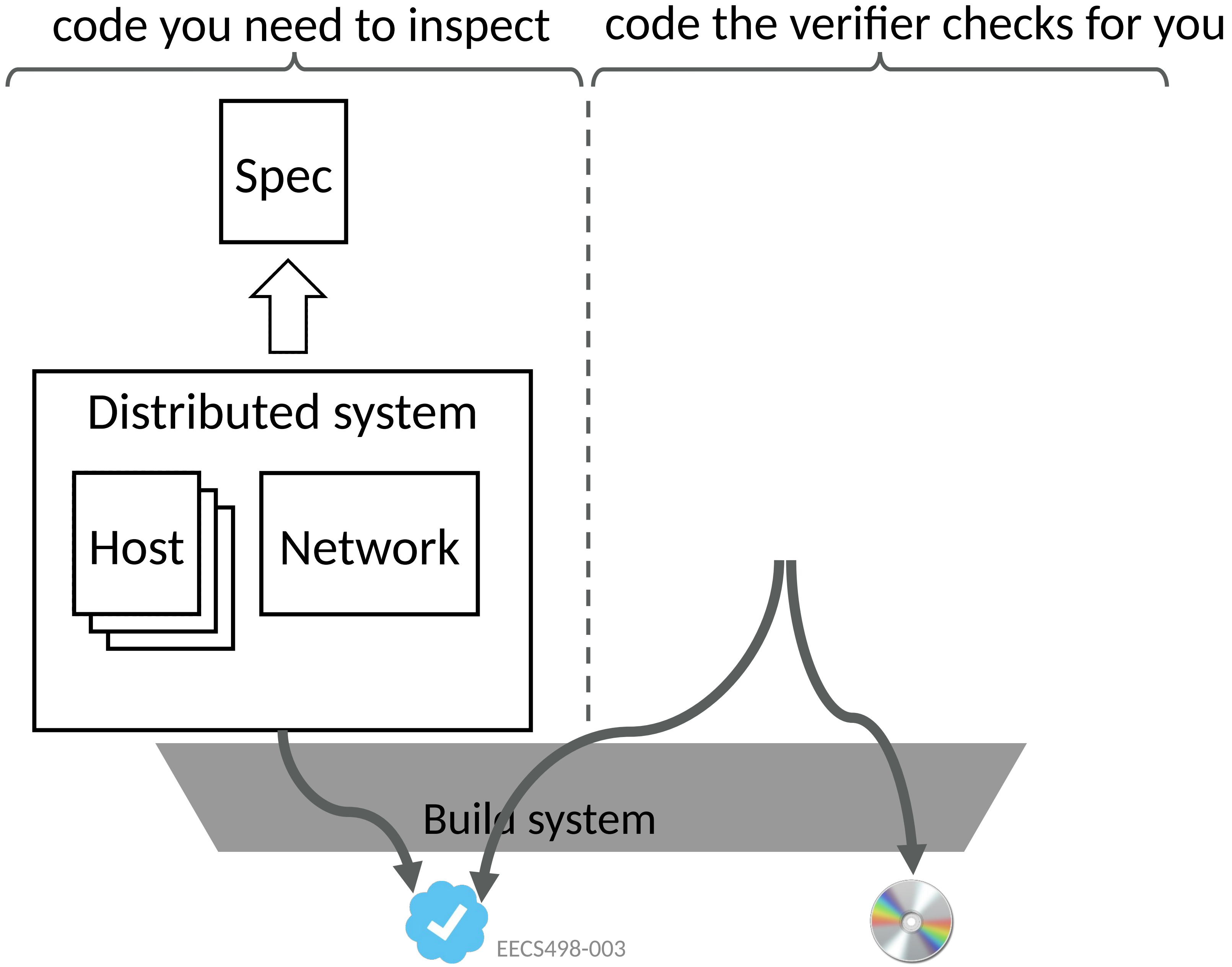
- Project 1 is due November 7
- PS4 (Chapter 6 – Refinement) will be released on November 8

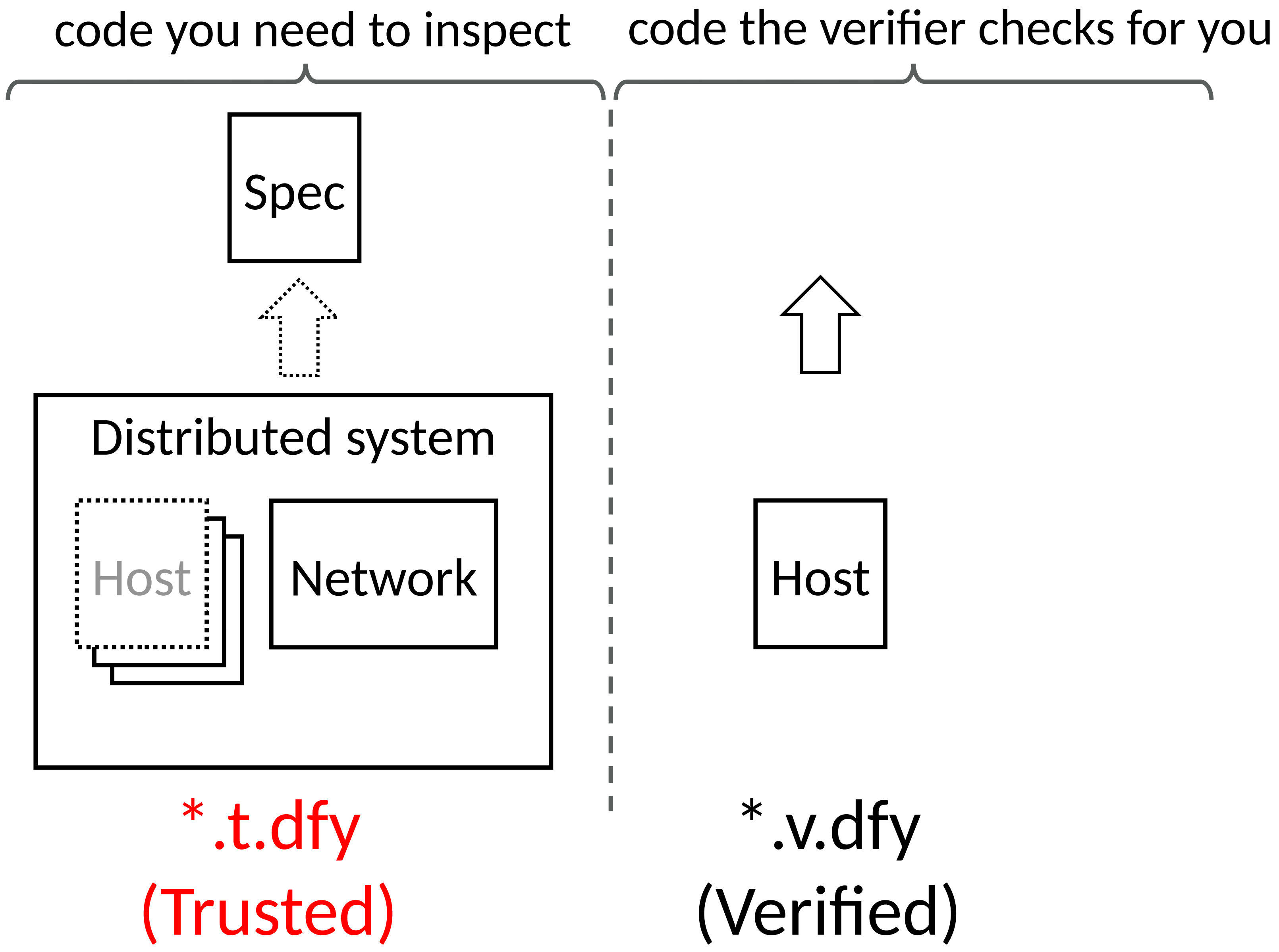


Refinement recap



```
ghost function Abstraction(v:Variables) : Spec.Variables
predicate Inv(v:Variables)

lemma RefinementInit(v:Variables)
    requires Init(v)
    ensures Inv(v) // Inv base case
    ensures Spec.Init(Abstraction(v))  // Refinement base case

lemma RefinementNext(v:Variables, v':Variables)
    requires Next(v, v’, evt)
    requires Inv(v)
    ensures Inv(v')  // Inv inductive step
    ensures Spec.Next(Abstraction(v), Abstraction(v’), evt)
          || Abstraction(v) == Abstraction(v’) && evt == NoOp
```


code you need to inspect
code the verifier checks for you
Spec
Distributed system
Host
Network
Build system


code you need to inspect
code the verifier checks for you
Spec
Distributed system
Host
Network
Host
*.t.dfy
(Trusted)
*.v.dfy
(Verified)

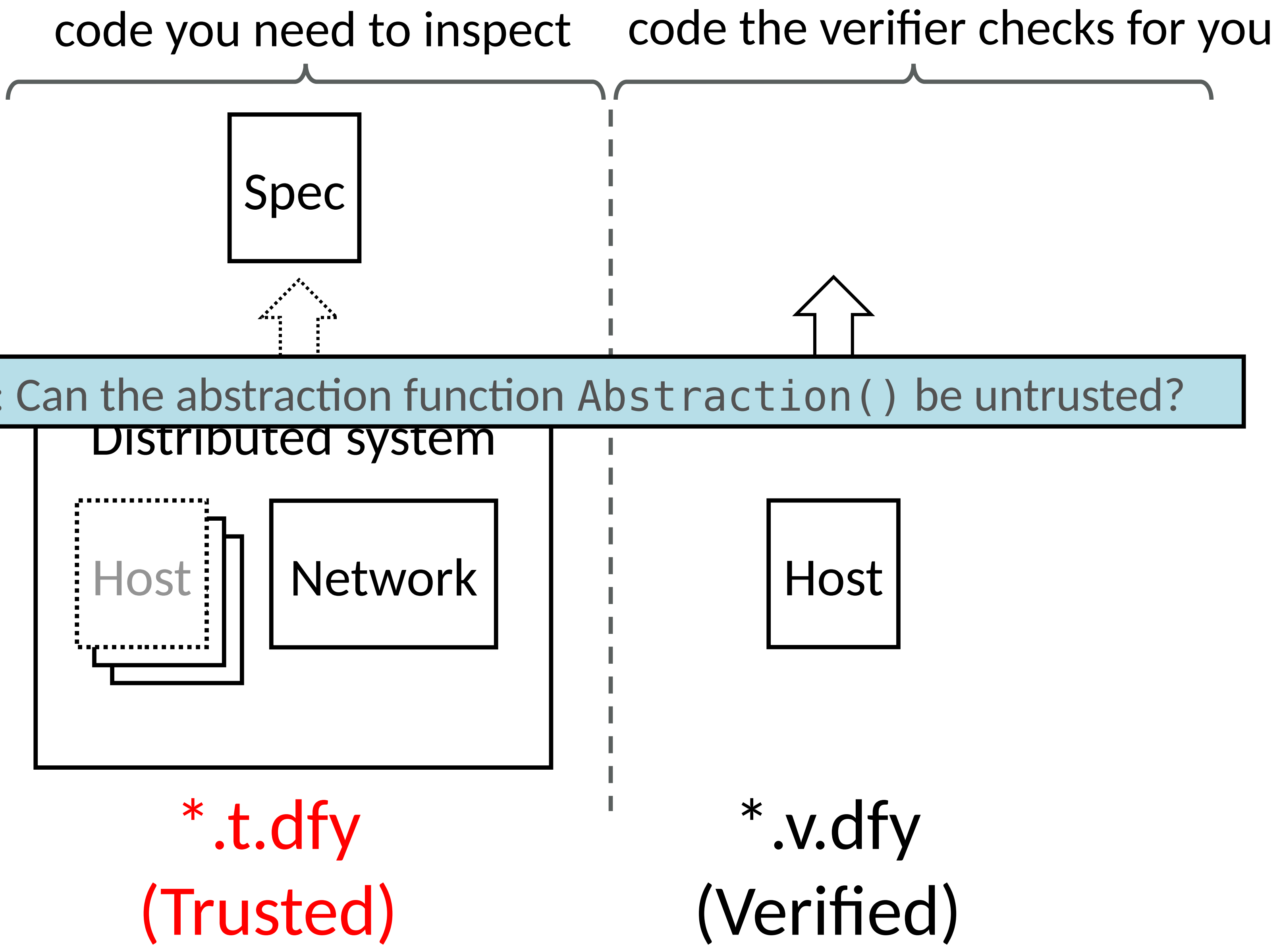
The verification game

- Player 1: the benign verification expert
- Player 2: the malicious engineer

Player 1 sets up the trusted environment
(i.e. all `.t.dfy` files)

Player 2 writes the implementation and proof
(i.e. all `.v.dfy` files)

Player 1 runs the build system



Q: Can the abstraction function `Abstraction()` be untrusted?

What if the abstraction function pretended nothing ever happened?

Always returns the initial state

```
function Abstraction(v:Variables) :
                Spec.Variables {
  var a0 :| SpecInit(a0);
  a0
}

predicate Inv(v:Variables) { true }
```

…or just made up a fake story?

```
datatype Variables =
Variables(actualState: Stuff, fakeState:
HostState)

function Abstraction(v:Variables) :
Spec.Variables {
      v.fakeState
}
```

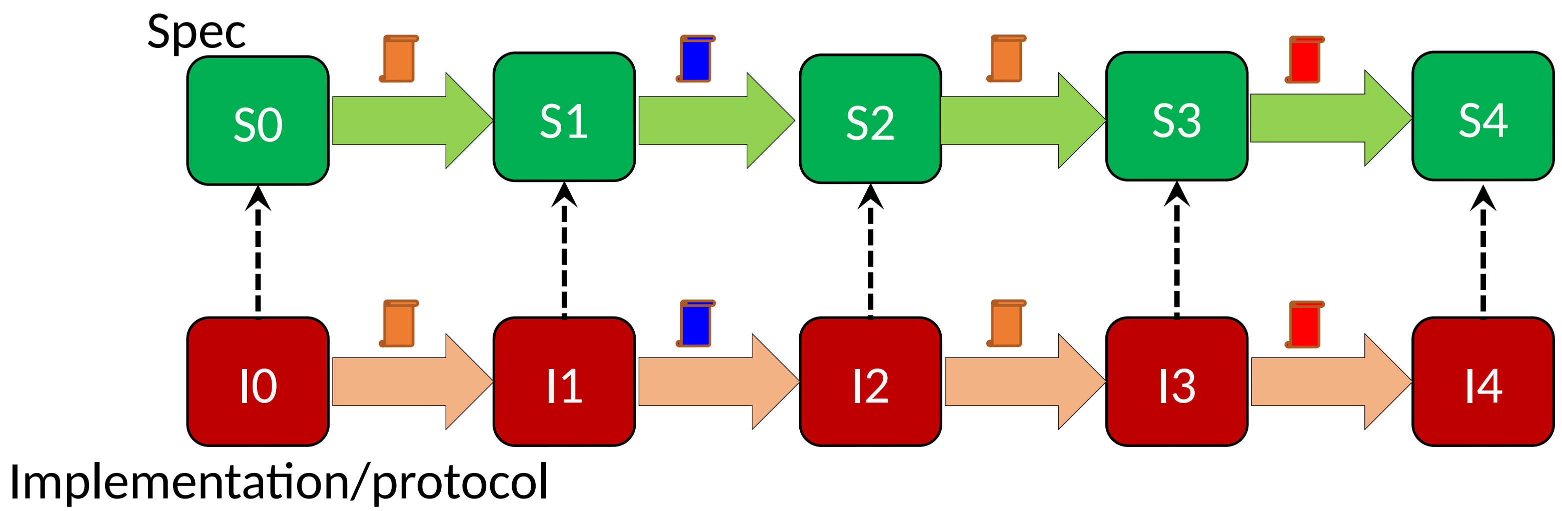
Returns fake state

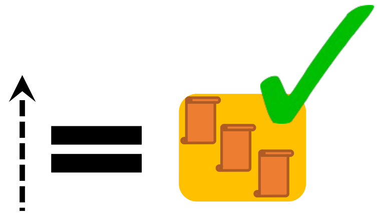
Events to the rescue

```
ghost function Abstraction(v:Variables) : Spec.Variables
predicate Inv(v:Variables)

lemma RefinementInit(v:Variables)
    requires Init(v)
    ensures Inv(v) // Inv base case
    ensures Spec.Init(Abstraction(v))  // Refinement base case

lemma RefinementNext(v:Variables, v':Variables)
    requires Next(v, v’, evt)
    requires Inv(v)
    ensures Inv(v')  // Inv inductive step
    ensures Spec.Next(Abstraction(v), Abstraction(v’), evt) // Refinement
inductive step
        || Abstraction(v) == Abstraction(v’) && evt == NoOp // OR stutter step
```

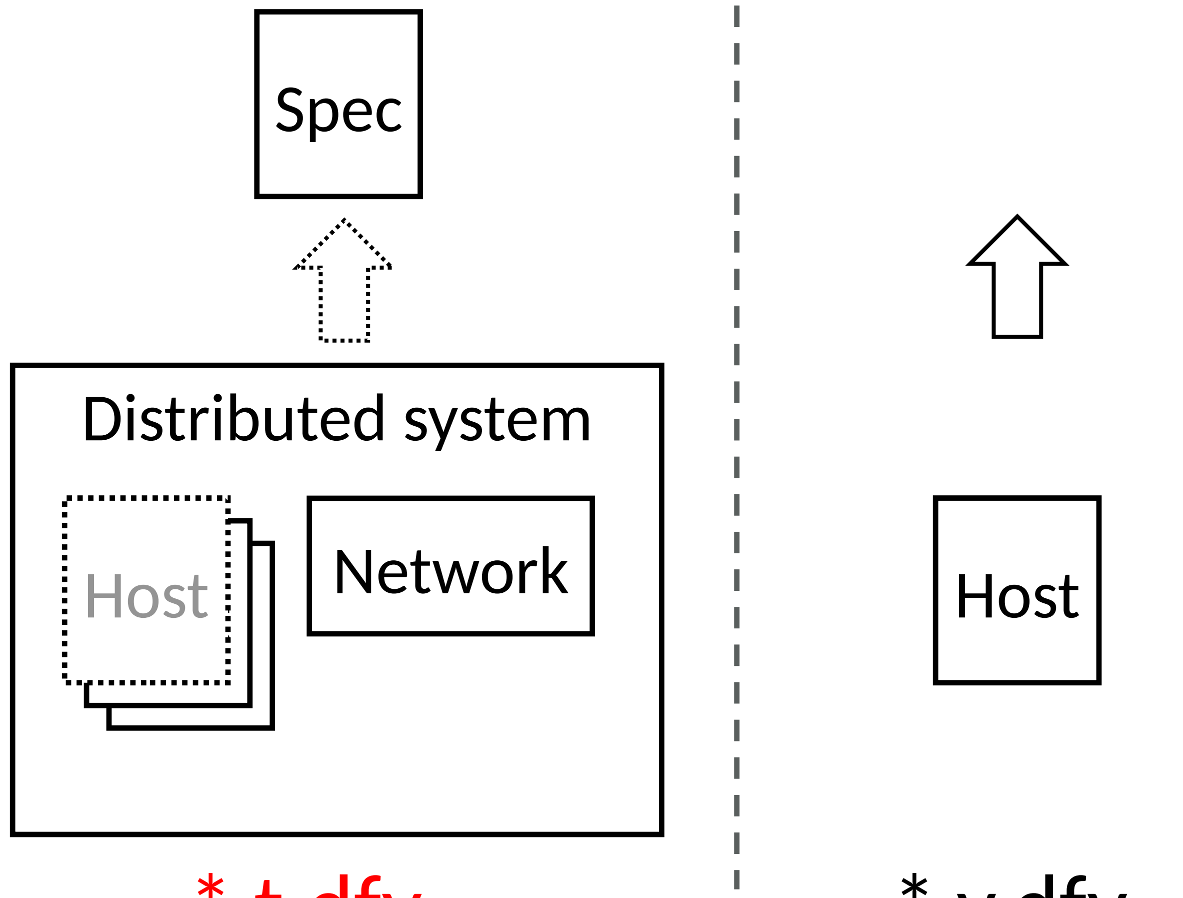
Application correspondence

aka Events to the rescue!





Revisiting the big picture



*.t.dfy
(Trusted)

*.v.dfy
(Verified)