EECS498-003
Formal Verification of Systems Software

Material and slides created by

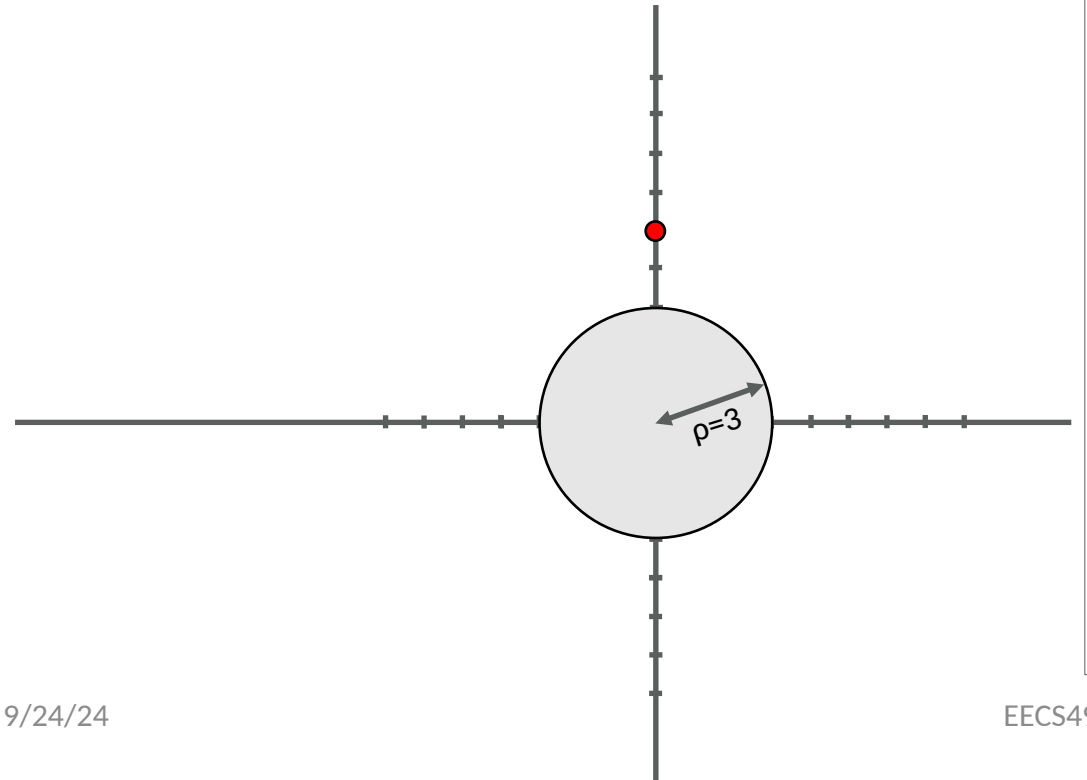
Jon Howell and Manos Kapritsos

Chapter 4

Inductive Invariants

A simple application: Crawler

- Crawler starts at (0,5)
- It can move 1 step north or 1 step south-east
- Can it ever fall in the hole?

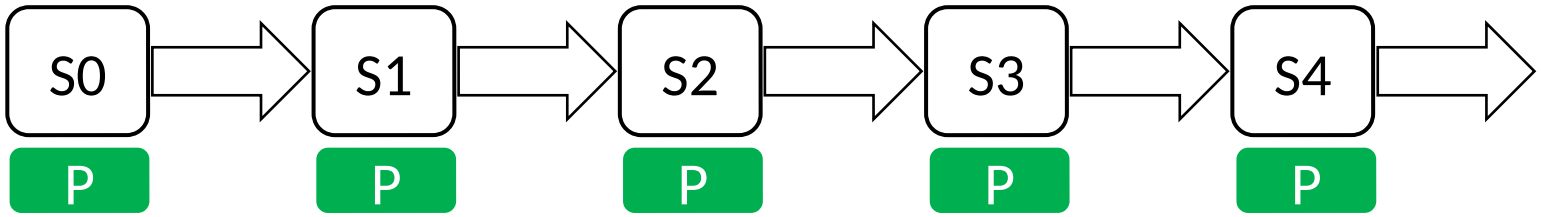



```
datatype Variables = Variables(x:int, y:int)
predicate Init(v:Variables) {
  && v.x == 0
  && v.y == 5
}

predicate MoveNorth(v:Variables, v':Variables)

{
  && v'.x == v.x
  && v'.y == v.y + 1
}

predicate MoveSouthEast(v:Variables,

v':Variables) {
  && v'.x == v.x + 1
  && v'.y == v.y - 1
}
```

Proving invariants

Proof by induction

- Prove it holds on the first state
- Prove it holds during a transition



Init(v) ==> P(v)

P(v) && Next(v, v') ==> P(v')

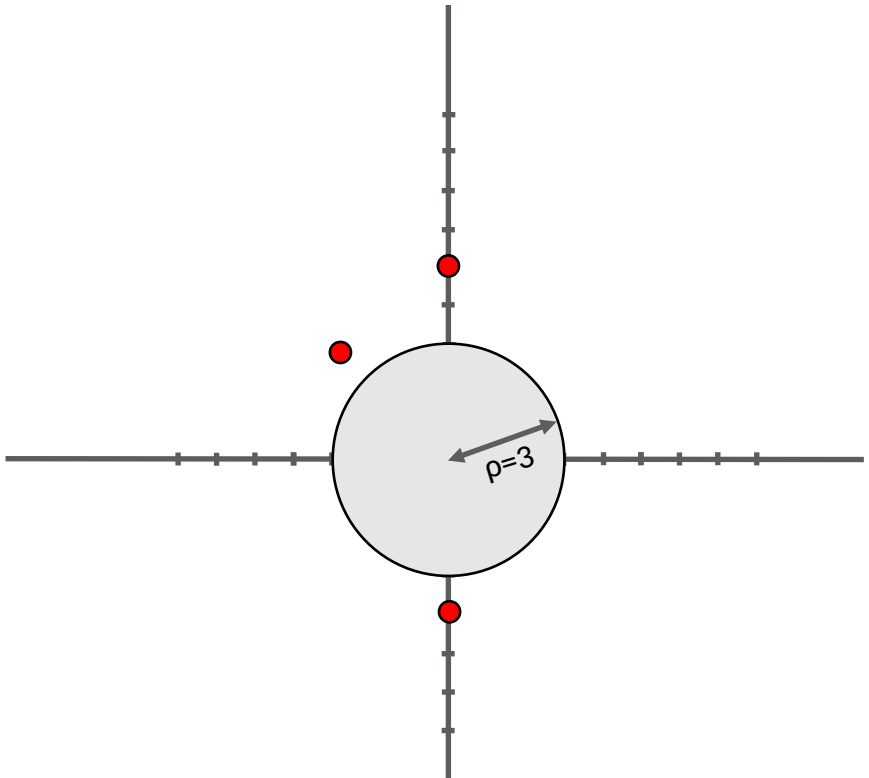
Proving the Crawler

```
datatype Variables = Variables(x:int, y:int)
predicate Init(v:Variables) {
  && v.x == 0
  && v.y == 5
}

predicate MoveNorth(v:Variables, v':Variables)

{
  && v'.x == v.x
  && v'.y == v.y + 1
}

predicate MoveSouthEast(v:Variables,

v':Variables) {
  && v'.x == v.x + 1
  && v'.y == v.y - 1
}
```

```
predicate Safety(v:Variables) {
}    v.x*v.x + v.y*v.y > 3*3
}
```

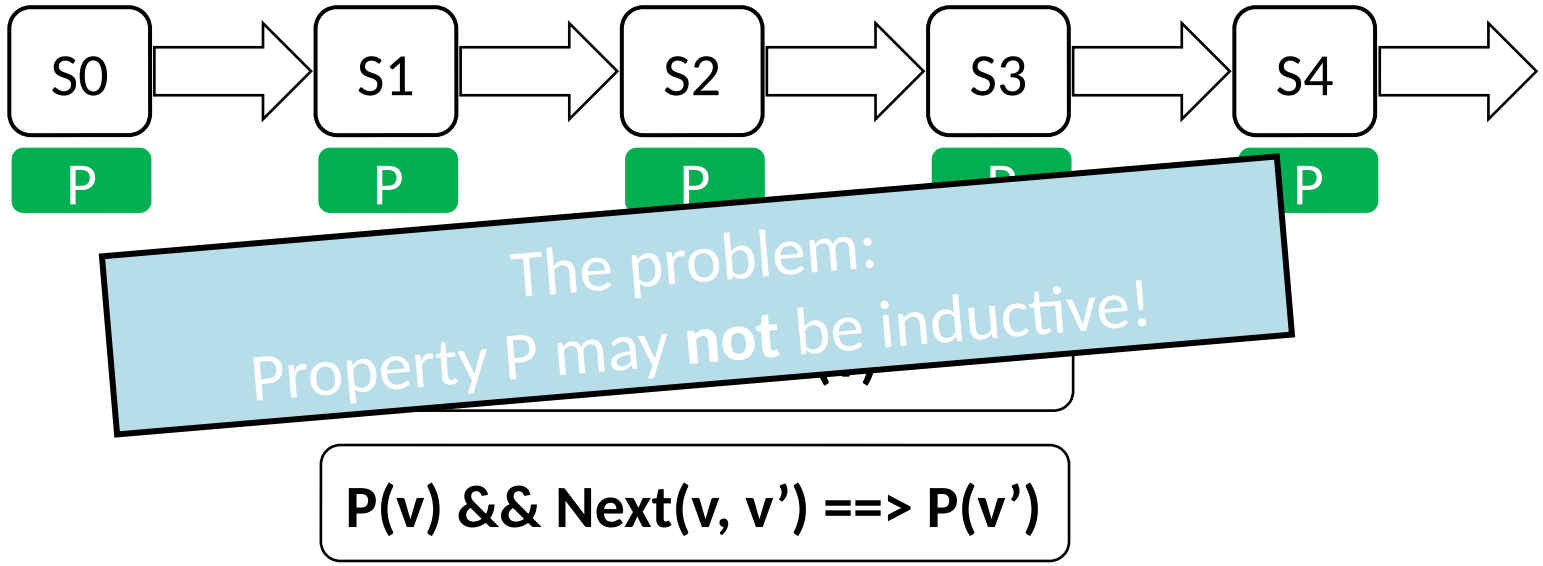
Init(v) ==> P(v) ✓

P(v) && Next(v, v') ==> P(v') ✗



Inductive invariants

Safety property (a.k.a. invariant):
a property that **always** holds

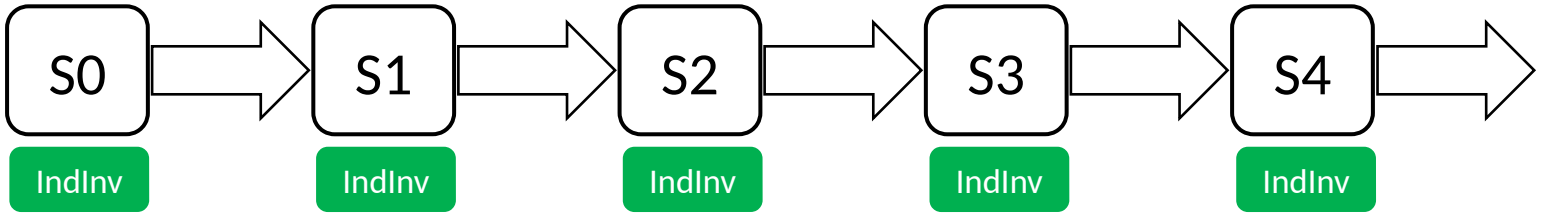


Proving safety with inductive invariants

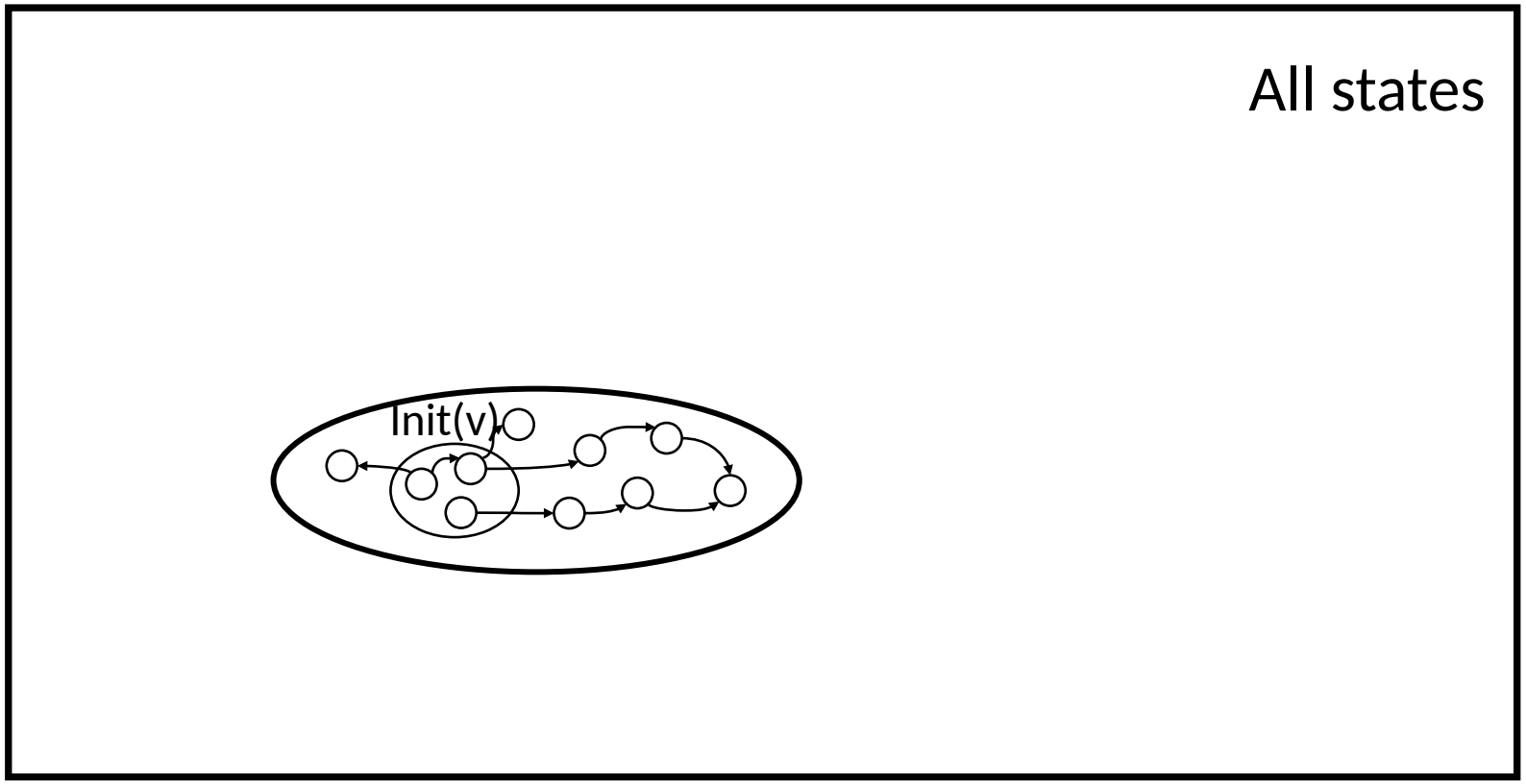
IndInv(v) ==> Safety(v)

Init(v) ==> IndInv(v)

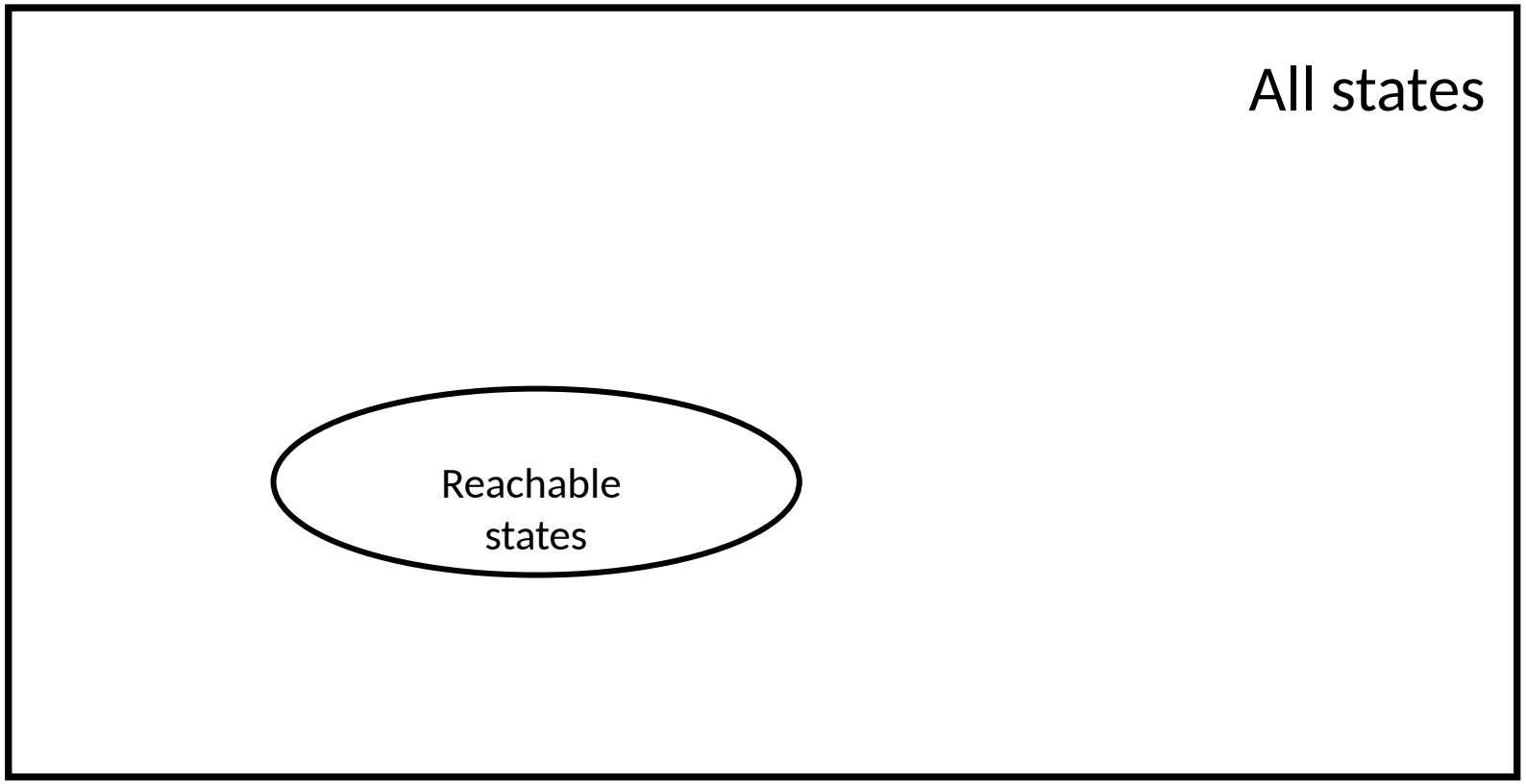
IndInv(v) && Next(v, v') ==> IndInv(v')



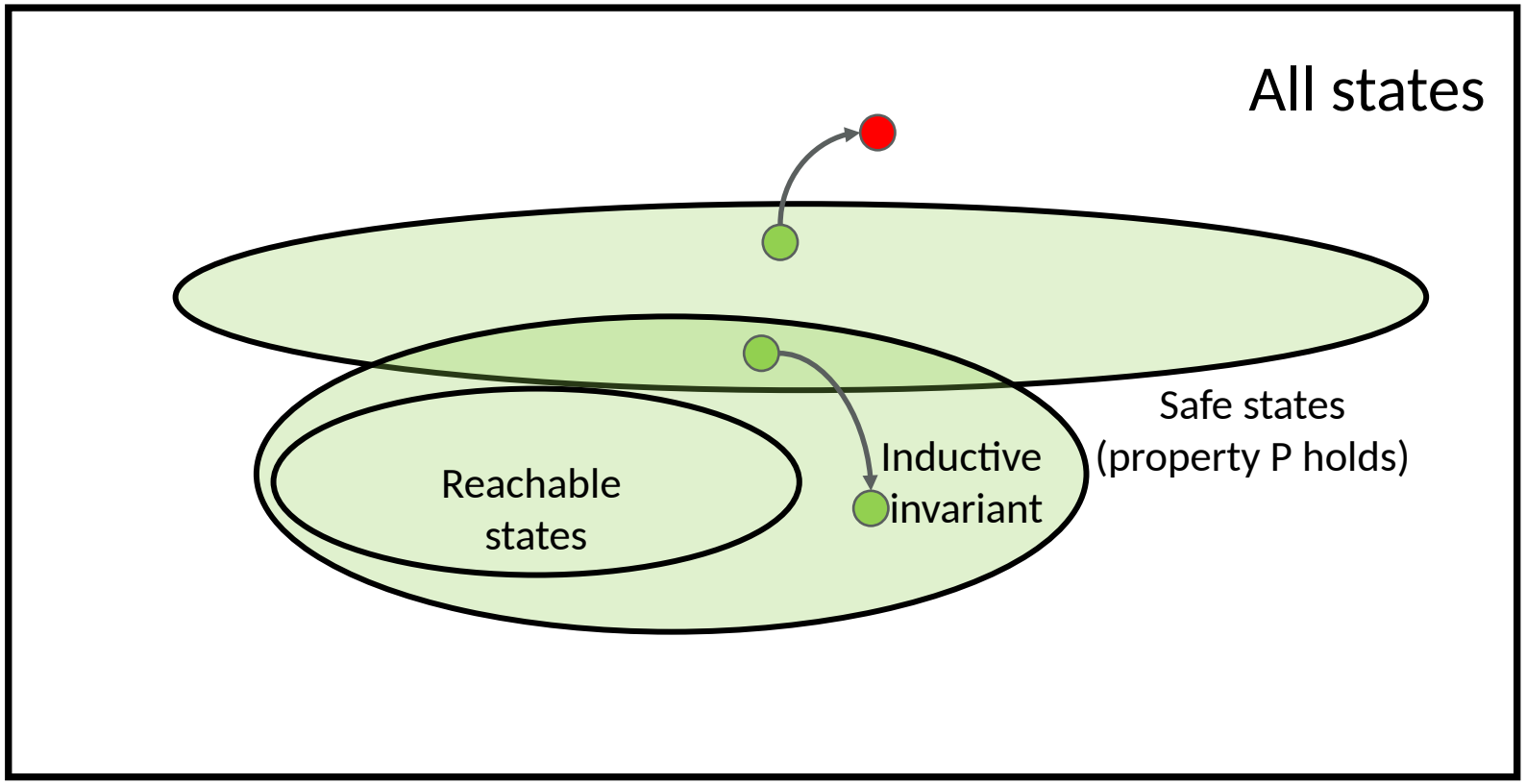
Invariants vs Inductive invariants



Invariants vs Inductive invariants

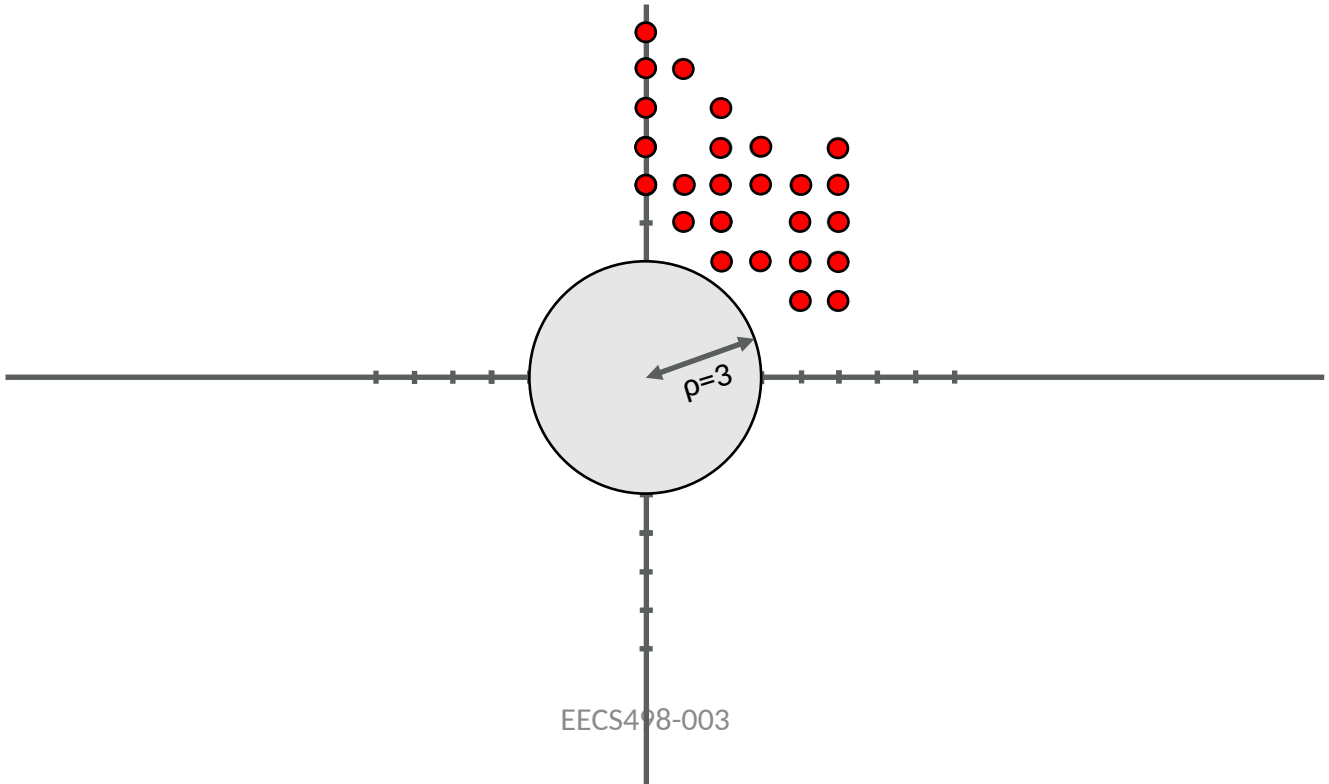


Invariants vs Inductive invariants



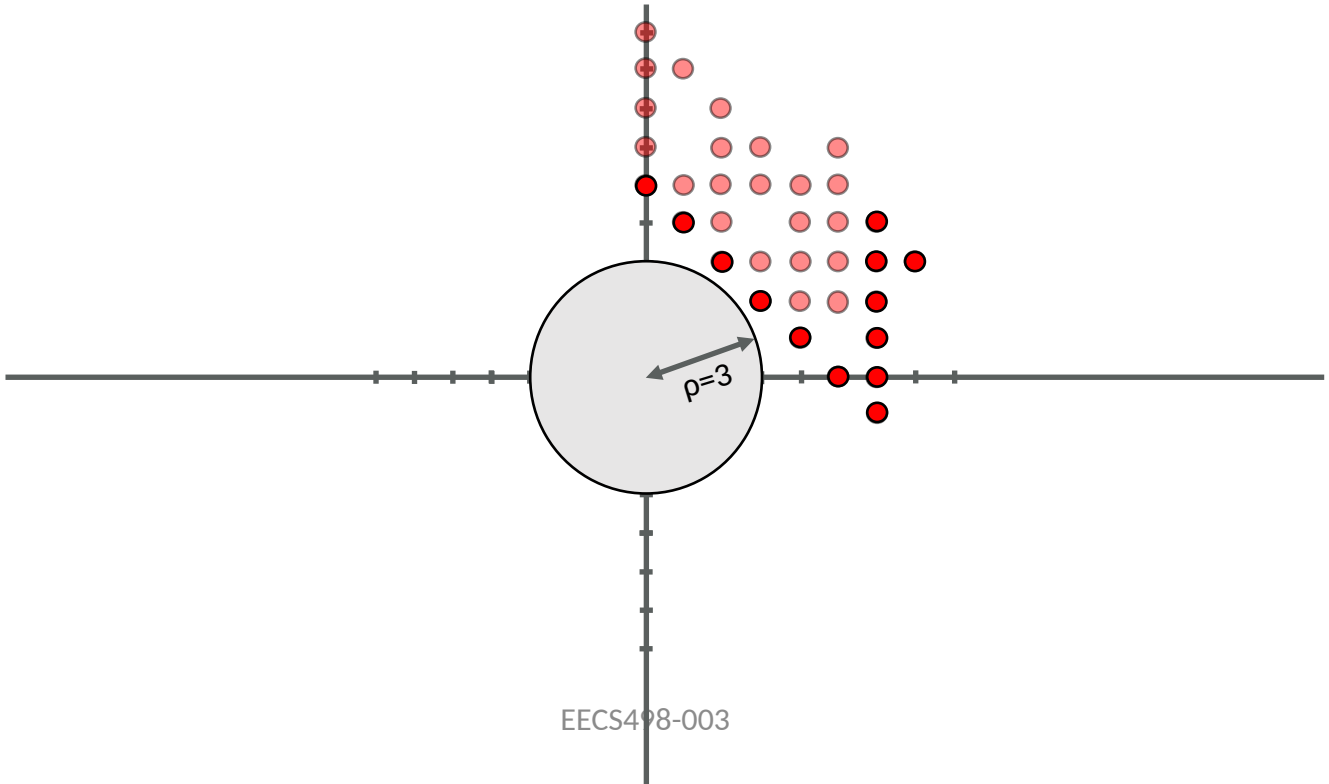
Proving the Crawler

Can the crawler ever fall in the hole?



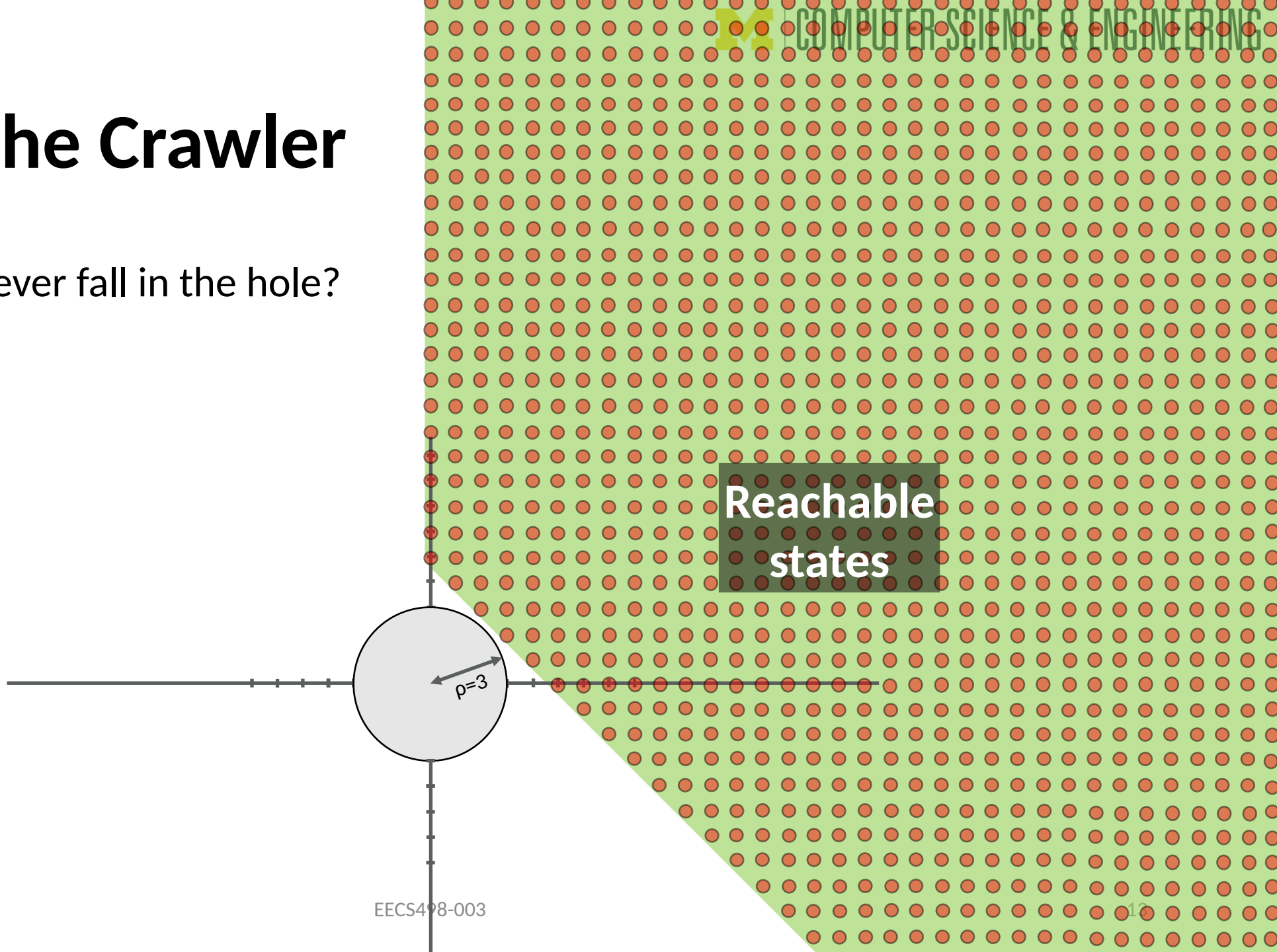
Proving the Crawler

Can the crawler ever fall in the hole?



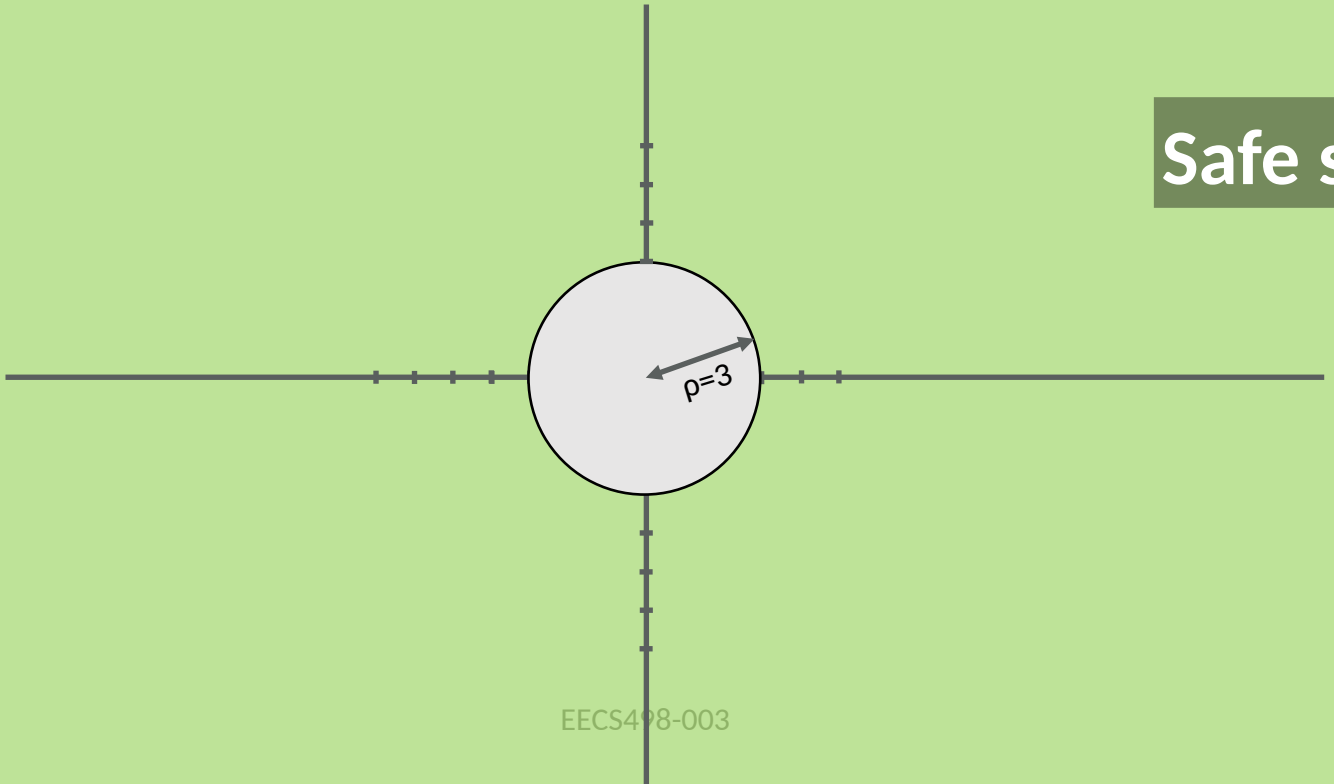
Proving the Crawler

Can the crawler ever fall in the hole?



Naïve safety proof

Safe(v) && Next(v, v') ==>
Safe(v')

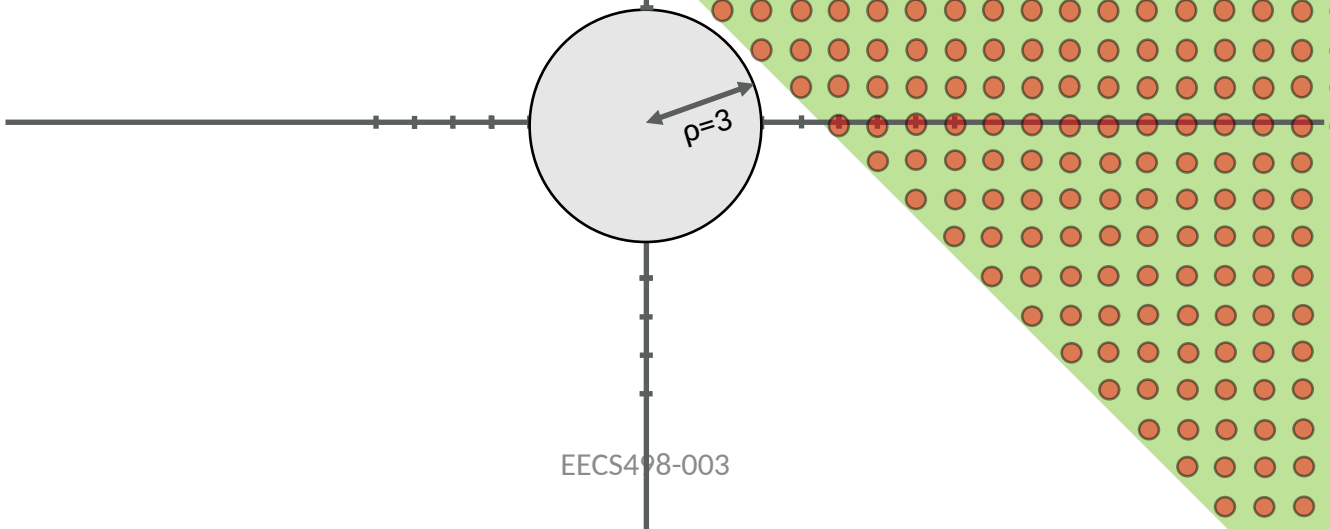


Safe states

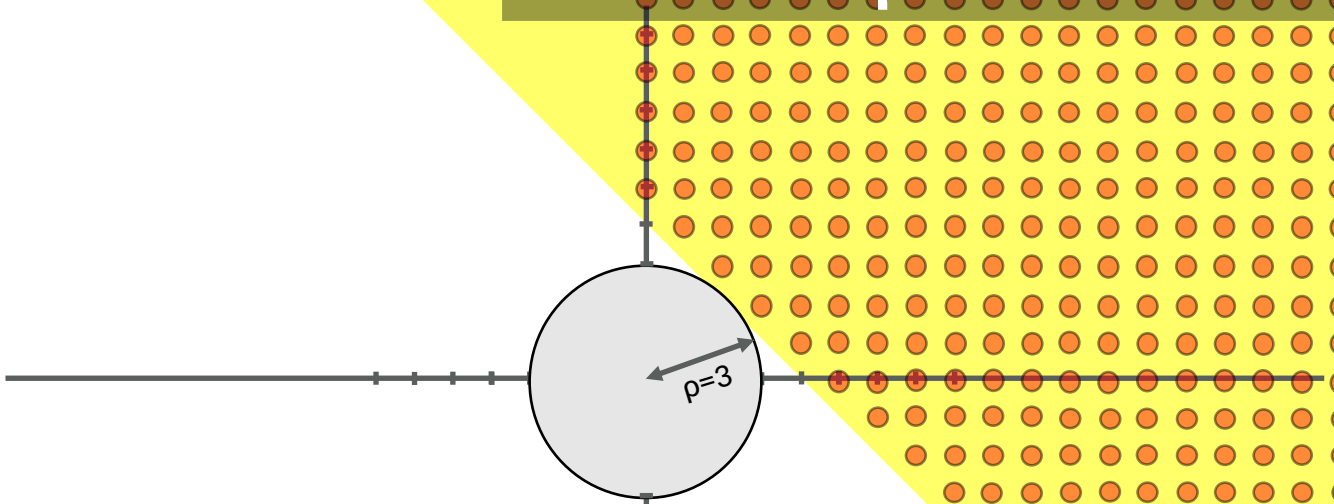
Safety proof using a stronger invariant

InGreenRegion(v) && Next(v, v') ==>
InGreenRegion(v')

The set of reachable states is always an inductive invariant

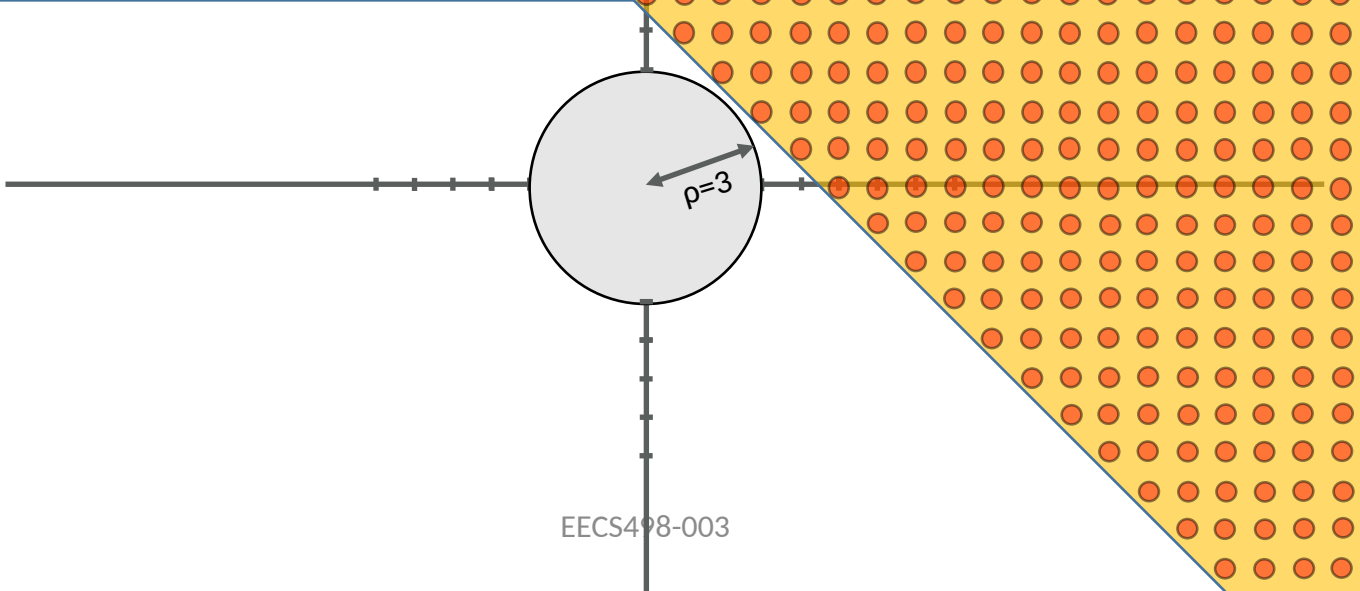


A simpler inductive invariant

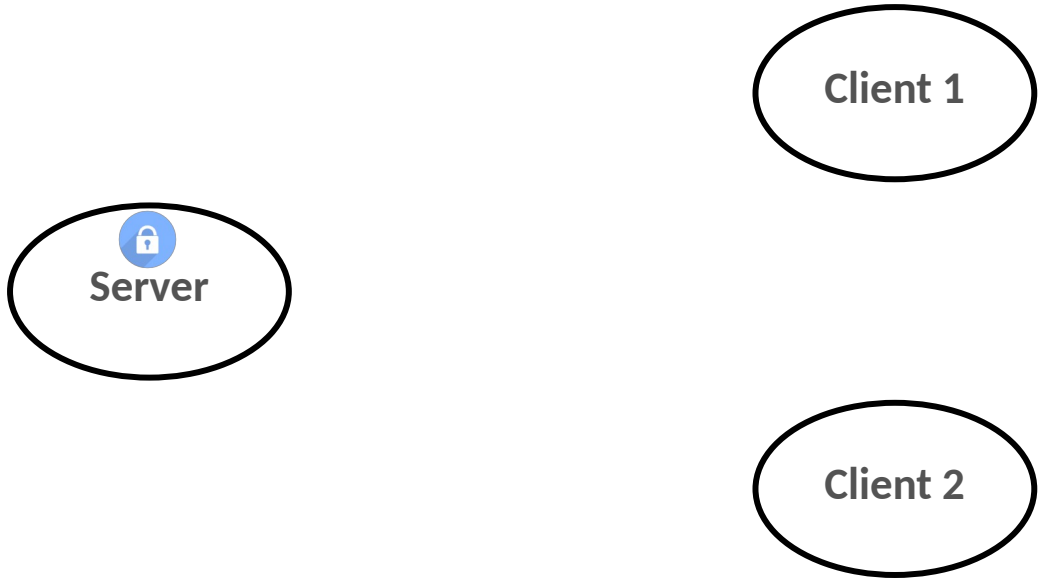


InYellowRegion(v) && Next(v, v') ==>
InYellowRegion(v')

A candidate inductive invariant



Example: lock server

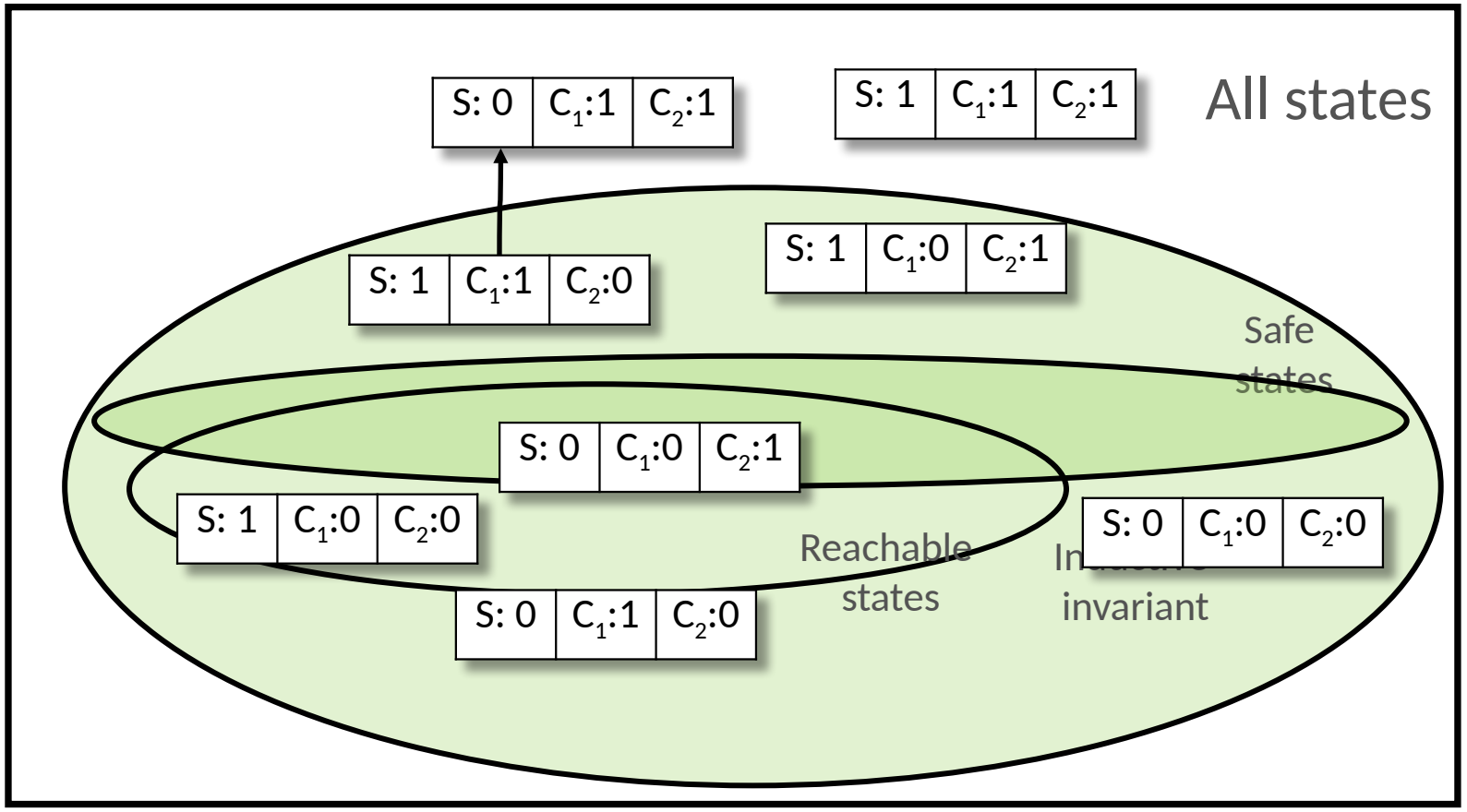


datatype Variables = Variables(S: bool, C1: bool, C2:bool)

ghost predicate Safety(v) { !(v.C1 && v.C2) }

Both clients cannot hold the lock at the same time

Example: lock server



Administrivia

- Chapter 4 will be released today
  - PS2 is due October 3

- Keep in mind that each chapter is a separate submission on the AG
  - And thus consumes late days separately

Some useful boilerplate

```
datatype Constants = Constants(capacity:int)
datatype Variables = Variables(numCokes:int)

predicate Init(c:Constants, v:Variables) { ... }

predicate Next(c:Constants, v:Variables, v':Variables) { ... }
```

Some useful boilerplate

```
datatype Constants = Constants(tableSize:nat)
{
  predicate WellFormed() {
      && 0 < tableSize
  }
}
```

```
datatype Variables = Variables()
{
  predicate WellFormed(c:
Constants) {
    && c.WellFormed()
  }
}
```

Typical examples:

- Length constraints on sequences
- Indices fit into a sequence length
- Domains of maps

Non-linear arithmetic

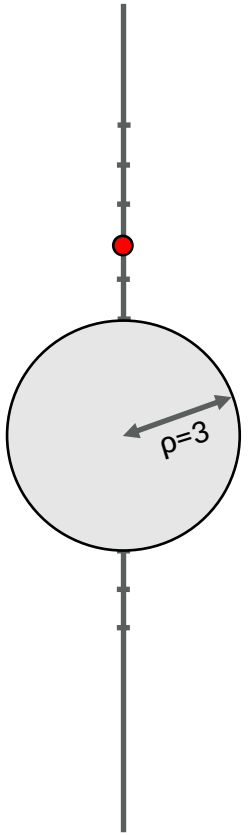
Dafny runs without non-linear reasoning by default

Beware of modulo operations

- Think of alternatives, if you run into trouble

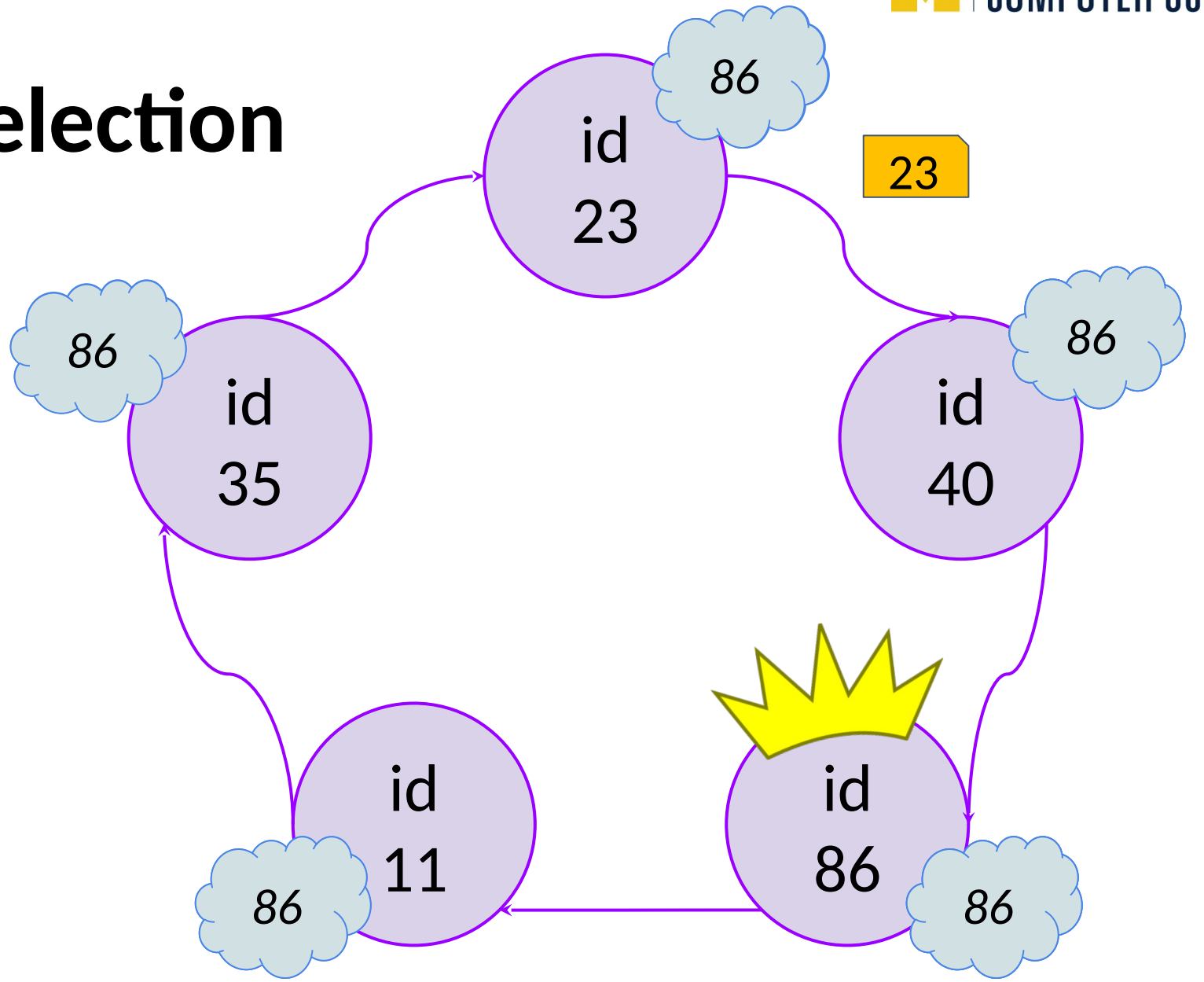
Crawler 2: Revenge of the inductive invariant

- The crawler can now only move North/South
  - Initially it can only move North
- It can also Flip(), teleporting to the symmetric point on the y-axis and changing direction of movement

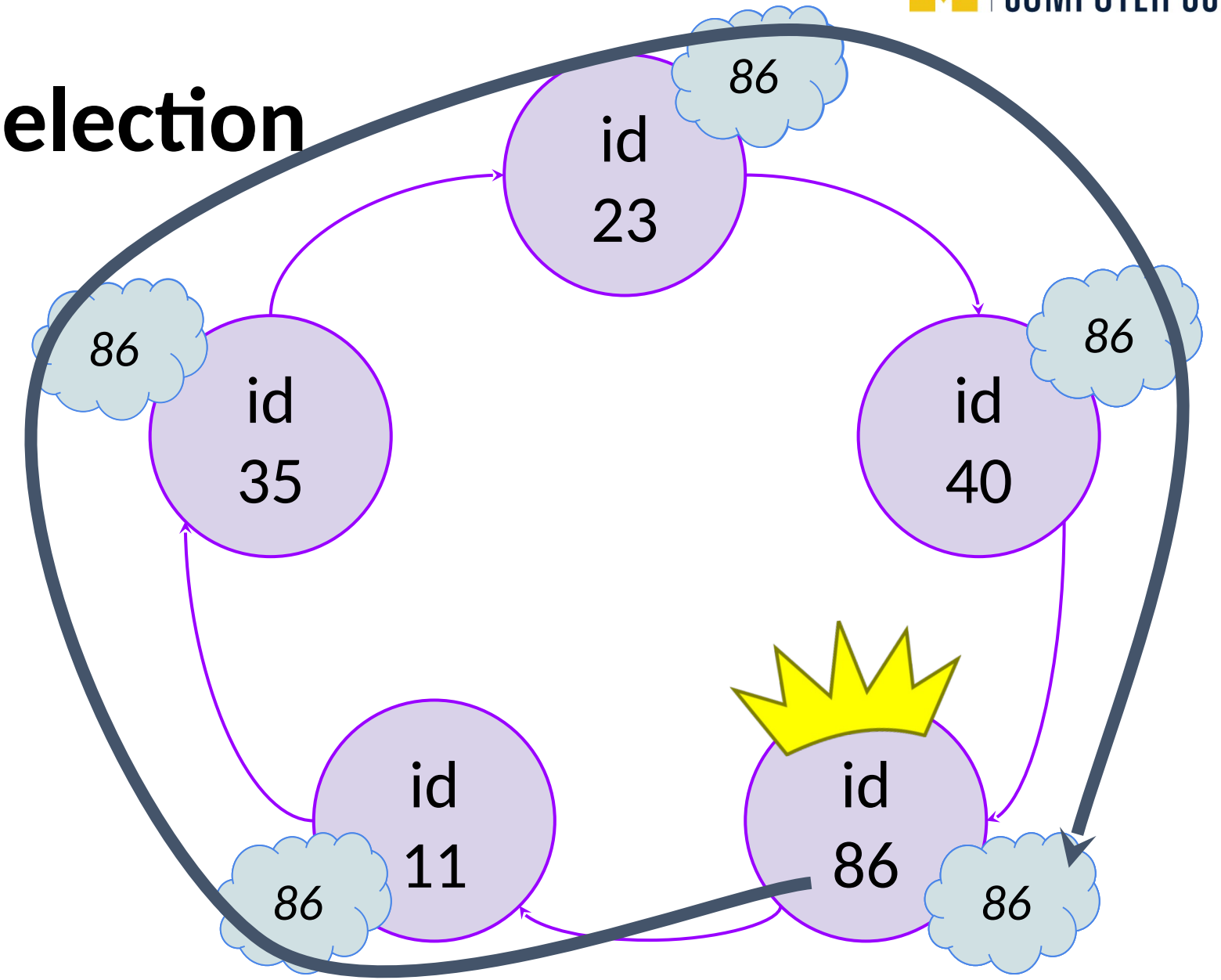


VSCode transition

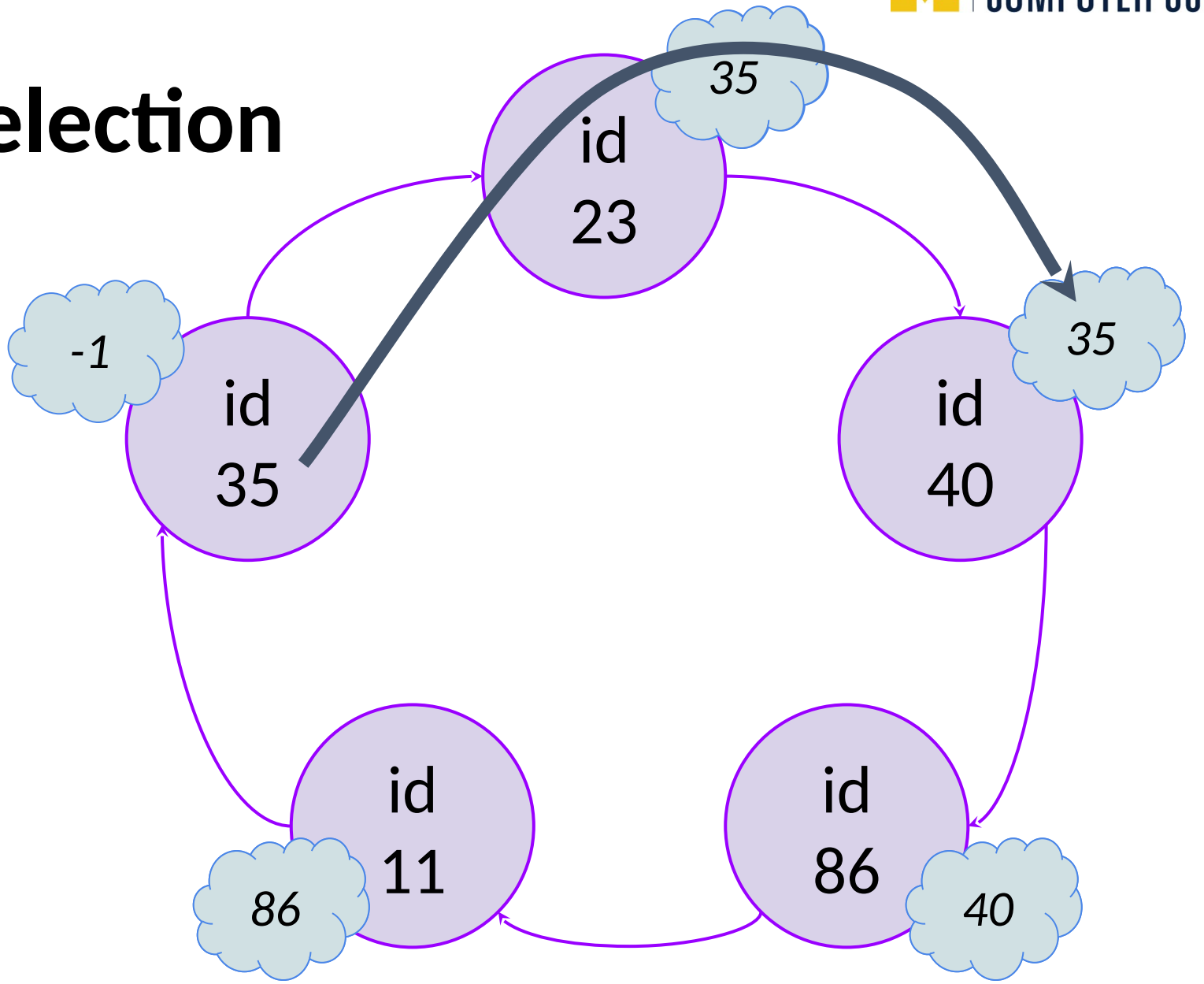
Leader election



Leader election



Leader election



Leader election