Alcino Cunha

SPECIFICATION AND MODELING

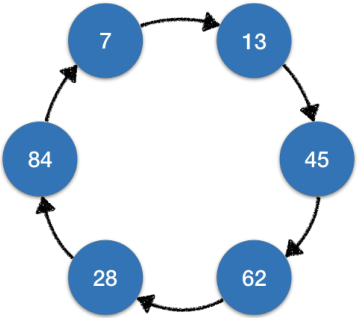
SAFETY, LIVENESS, AND FAIRNESS

Universidade do Minho & INESC TEC

2019/20

LEADER ELECTION IN A RING

LEADER ELECTION IN A RING



Verify the correctness of the protocol:

- One leader will be elected

CONFIGURATION AND STATE

```
open util/ordering[Id]

sig Id {}

sig Node {
    id : one Id,
    succ : one Node,
    var inbox : set Id,
    var outbox : set Id
}

fact ring {
    all i : Id | lone id.i
    all n : Node | Node in n.^succ
}
```

ELECTION

```
fun elected : set Node {
    { n : Node | once n.id in n.inbox }
}
```

OPERATIONS

```
pred send [n : Node] { ... }

pred compute [n : Node] {
    some i : n.inbox {
        n.inbox' = n.inbox - i
        n.outbox' = n.outbox + (i - n.id.*(~next))
    }
    all m : Node - n | m.inbox' = m.inbox
    all m : Node - n | m.outbox' = m.outbox
}

pred skip { ... }
```

BEHAVIOR

```
fact init {
    no inbox
    outbox = id
}

fact transitions {
    always (skip or some n : Node | send[n] or compute[n])
}
```


DEMO

SAFETY VS LIVENESS

SAFETY PROPERTIES

- Something “bad” will never happen
- A trace that violates a safety property has a “bad” prefix
  - A prefix such that every possible continuation violates the property
  - To understand a counter-example it suffices to inspect such prefix

```
always p
always (p implies once q)
always (p implies after always not p)
```

LEADER ELECTION

```
assert safety {
  always lone elected
}
```


DEMO

LIVENESS PROPERTIES

- Something “good” will eventually happen
- A property is a liveness property if any prefix can be extended to an infinite trace satisfying it
  - Much harder to check than safety properties
  - Inspection of a prefix is not sufficient to understand a counter-example
  - The full (infinite) trace must be observed

```
eventually p
always (p implies eventually q)
always eventually p
```

LEADER ELECTION

```
assert liveness {
  eventually some elected
}
```


DEMO

FAIRNESS

FAIRNESS ASSUMPTIONS

- Necessary for verifying most liveness properties
- Exclude traces where an event becomes “continuously” enabled but never occurs
  - continuously = infinitely often (strong)
  - continuously = permanently (weak)

Strong fairness

```
(always eventually enabled) implies (always eventually happens)
```

Weak fairness

```
(eventually always enabled) implies (always eventually happens)
always ((always enabled) implies (eventually happens))
```

LEADER ELECTION SPECIFICATION FIXED

```
pred sendEnabled [n : Node] { some n.outbox }
pred computeEnabled [n : Node] { some n.inbox }
pred fairness {
  all n : Node {
    (eventually always sendEnabled[n]) implies
      (always eventually send[n])
    (eventually always computeEnabled[n]) implies
      (always eventually compute[n])
  }
}
assert liveness {
  fairness implies eventually some elected
}
```


DEMO