Distributed Computing

José Orlando Pereira

Grupo de Sistemas Distribuídos
Departamento de Informática
Universidade do Minho

2007/2008

Asynchronous systems

- Assume no bounds on:
  - clock drift
  - processing time
  - message passing time
- Real world considerations:
  - Load and processor scheduling
  - Network delays
  - ...
- Without loss of generality, assume a reliable fully connected network

Asynchronous systems

- Relax the synchronous system:
  - Unbounded message loss
  - Large/unknown graph diameter
  - Dynamic graph
- Tight synchronous limits are dangerous:
  - Low coverage, expensive systems
- Large synchronous limits are not useful:
  - Taking advantage of synchrony causes a very large penalty

I/O Automata

- Very general model:
  - Describes also non-distributed and even non-concurrent systems
- Powerful tools:
  - Composable specifications
  - Hierarchical specifications
- Very widespread use in DS research

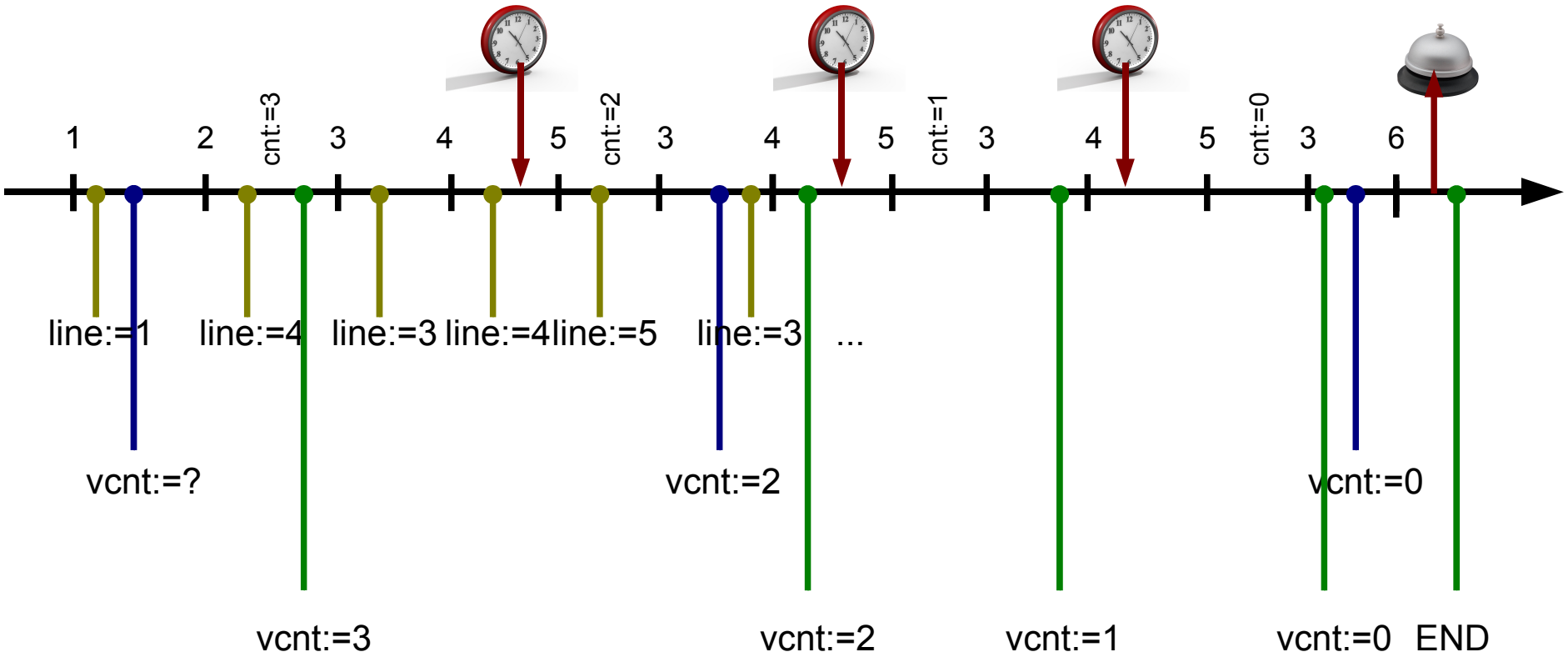
Sample computation

- An alarm clock program:

```
main:                          // line 1
    cnt:=3                     // line 2
    while cnt>0:               // line 3
        sleep 1s               // line 4
        cnt := cnt-1           // line 5
    ring                       // line 6
```

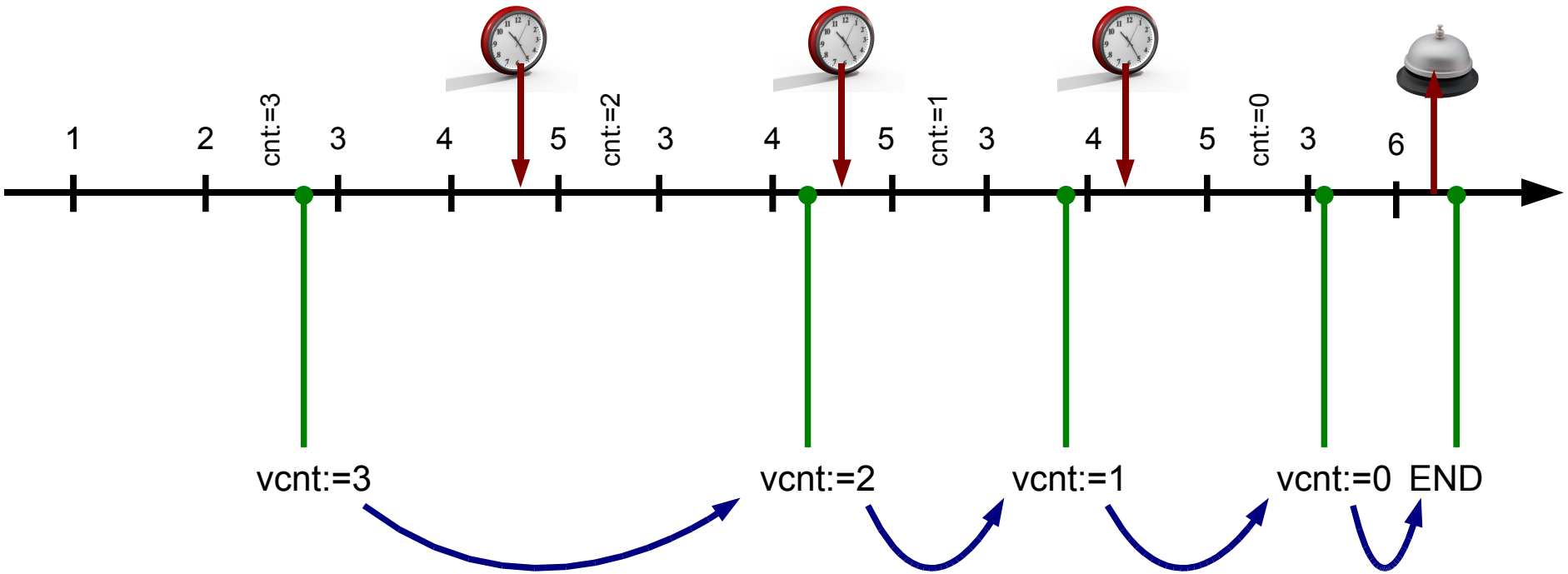
Observation

- Select model variables and periodically observe the system:



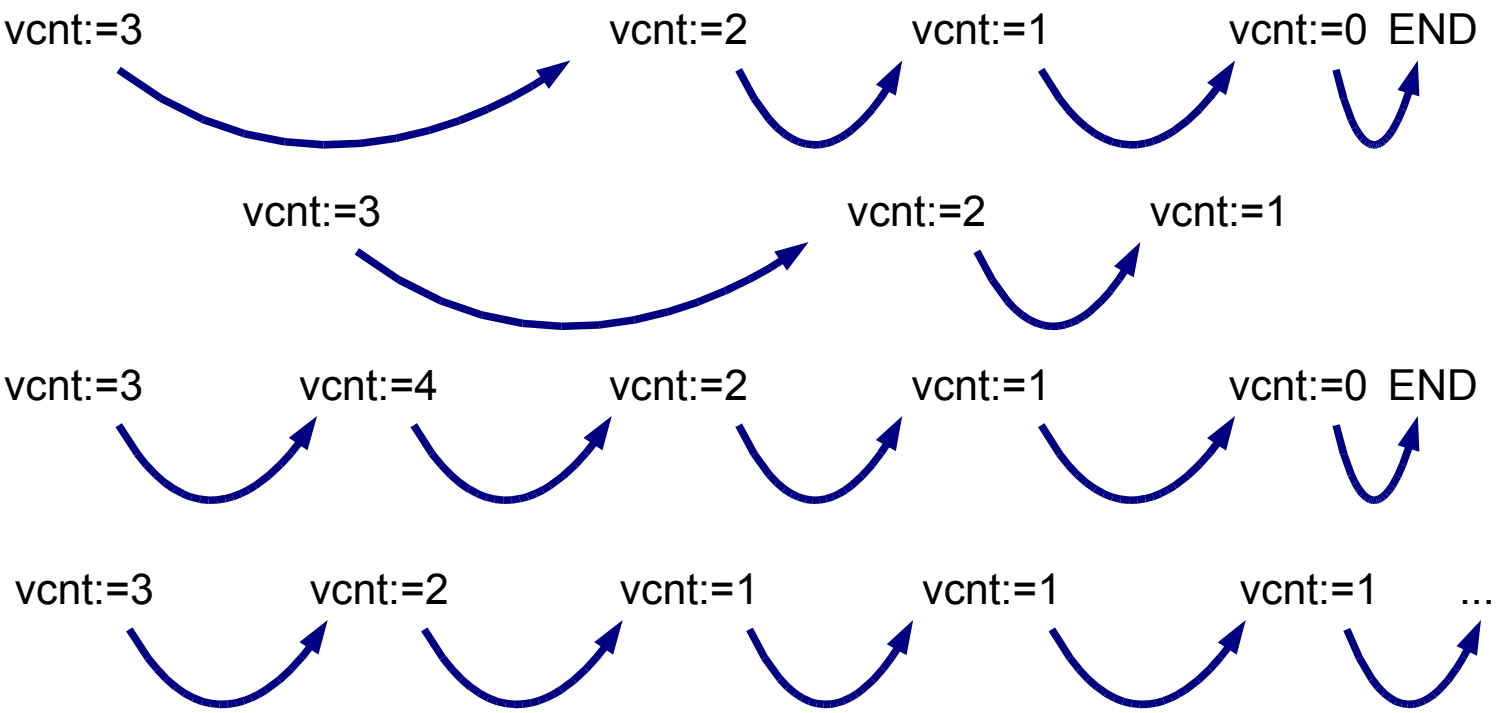
Abstraction

- Choose observation that conveys interface, not implementation:



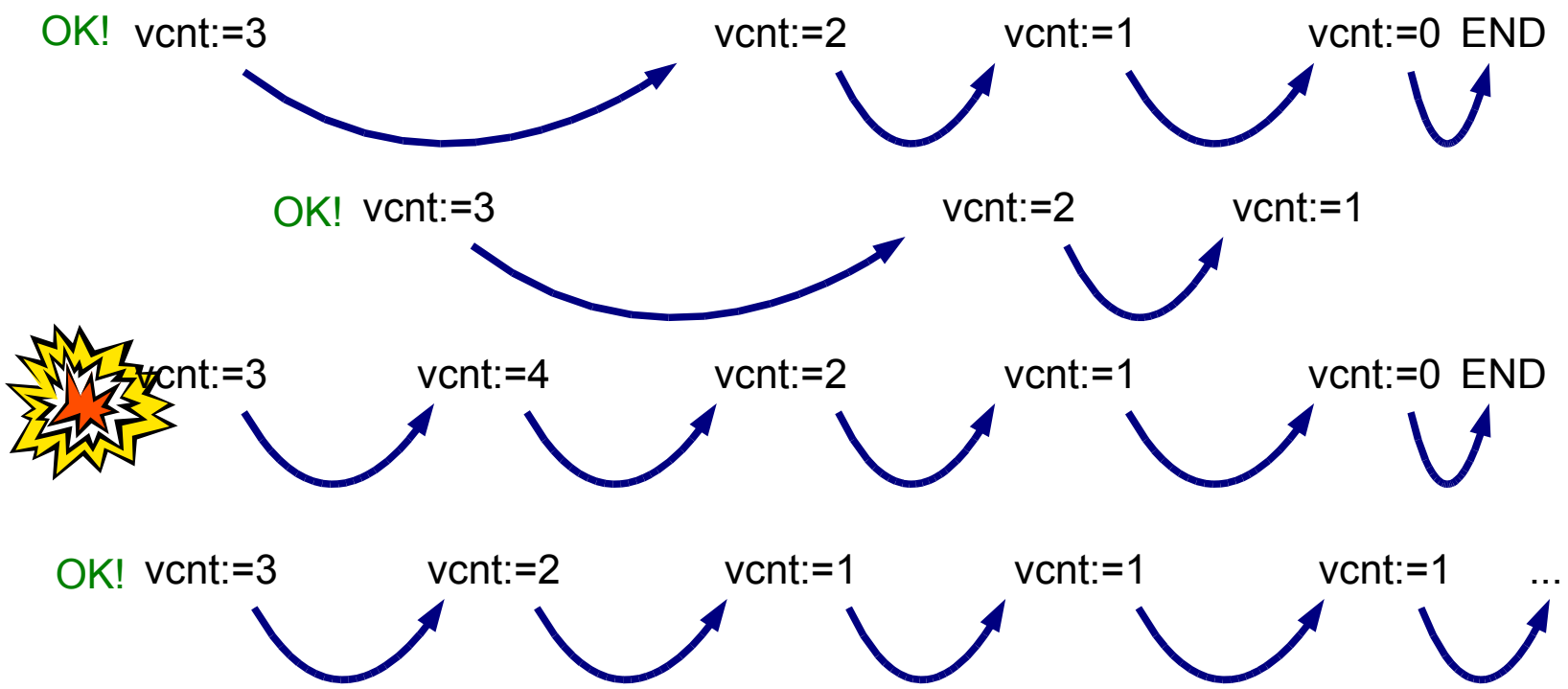
Behaviors/Executions

- Consider all possible sequences of chosen atomic actions:



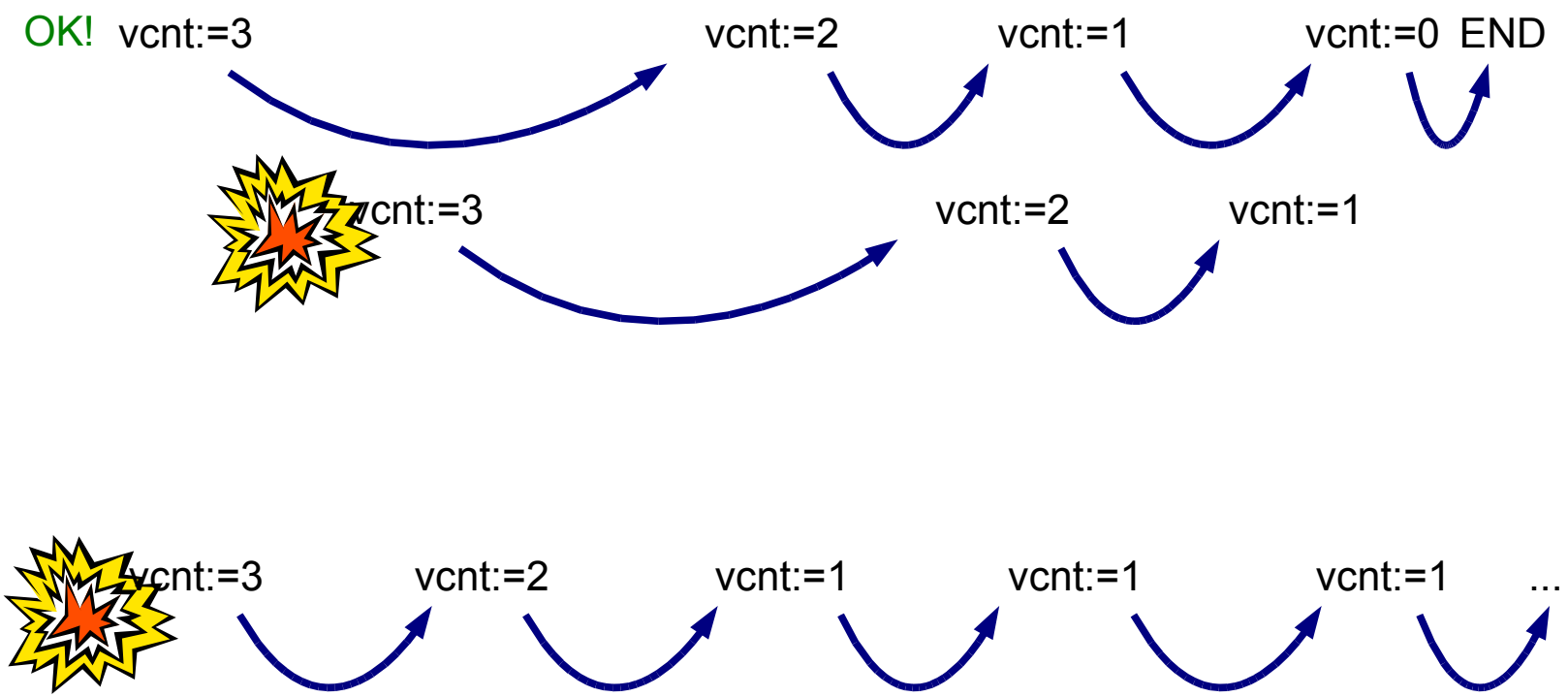
Safety properties

- Nothing bad ever happens:



Liveness properties

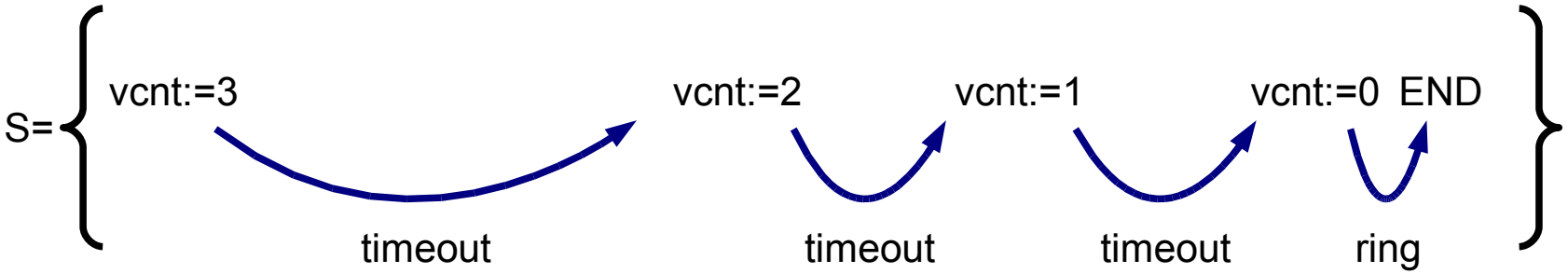
- Something good eventually(*) happens:



(*) eventually = inevitavelmente ≠ eventualmente

Specification

- Specification is a set of allowable behaviors:



- An automaton provides a compact and practical representation
  - Infinite sets of behaviors

Automaton definition

- An automaton A has five components:
  - sig(A), a triplet S of disjoint sets of actions:
    - in(S), the input actions
    - out(S), the output actions
    - int(S), the internal actions
  - states(A), a (possibly infinite) set of states
  - start(A), a non-empty subset of states(A)
  - trans(A), a subset of
    states(A) x acts(sig(A)) x states(A)
  - tasks(A), a partition off local(sig(A))

Automaton definition

- Additional definitions:
  - ext(S) = in(S) U out(S)
  - local(S) = out(S) U int(S)
  - extsig(S) = (in(S), out(S), {})
- Short-hands:
  - ext(A) for ext(sig(A))
  - ...

Transitions

- A transition is enabled in state s if there is some π,s' such that (s,π,s') ∈ trans(A)
- Input transitions are required to be enabled in all reachable states of A
- A state in which only input transitions are enabled is said to be quiescent

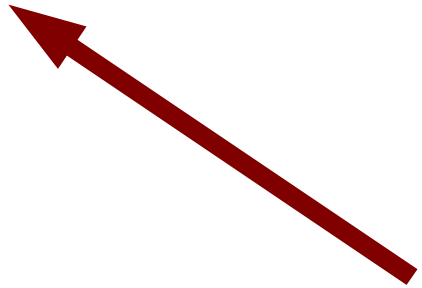
Signature and State

- Input:
  - Timeout
- Output:
  - Ring

- States:
  - vcnt, integer, initially 3
  - END, boolean, initially false

Transitions

- Timeout:
  - Pre-condition:
    - ¬END and vcnt>0
  - Effect:
    - vcnt := vcnt - 1
- Ring:
  - Pre-condition:
    - ¬END and vcnt = 0
  - Effect:
    - END := True



This is an equation,
not an attribution!

Effects

- Effect equation:
  - vcnt := vcnt - 1
- Read this as:
  - “vcnt-after = vcnt-before – 1 and the state otherwise unchanged”
- Could be written as:
  - vcnt-after + 1 = vcnt-before
  - vcnt-before - vcnt-after = 1
  - ...

Invariants

- Goal: Prove that always vcnt < 4 (safety!).
- Proof by induction:
  - Base step: True for all initial states?
    - 3<4: Yes!
  - Induction step: True for any next step?
    - Timeout transition:
      - vcnt-after = vcnt-before - 1
      - vcnt-before < 4
vcnt-after+1 < 4
vcnt-after < 3 < 4: Done
    - Ring transition:
      - always vcnt-after = vcnt-before = 0
      - 0<4: Done