A Decomposition Method for the Verification of a Real-time Safety-critical Protocol

Tamás Tóth

András Vörös

István Majzik

Overview of the talk

- Safety critical systems
  - Case-study
- Background
  - Formal methods
  - Model checking
  - Temporal logic specification
- Verification approach
  - Decomposition method

Motivation: safety critical systems

- A system-level failure may result in
  - damage to people’s health
  - serious environmental or financial harm
- Example:
  - Railway interlocking systems
- Characteristics:
  - Time-dependent behavior
  - Parametric behavior
- Ensuring correct behavior is crucial
  - In the presence of failures

Case-study

A master election and ID assignment protocol

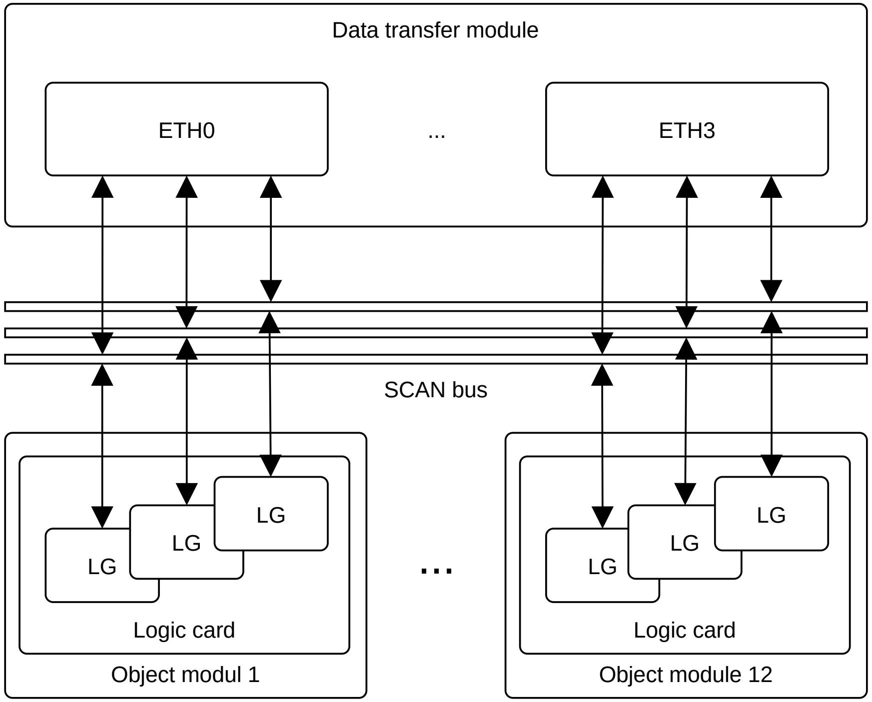
The case study

- Protocol in a railway SCADA (*supervisory control and data acquisition*) system
- Ensures stable and fault tolerant communication between components
- Roles: MASTER-SLAVE
- Communication is performed in two layers:
  - the lower layer serves for administration,
  - while the upper layer transmits information between the components

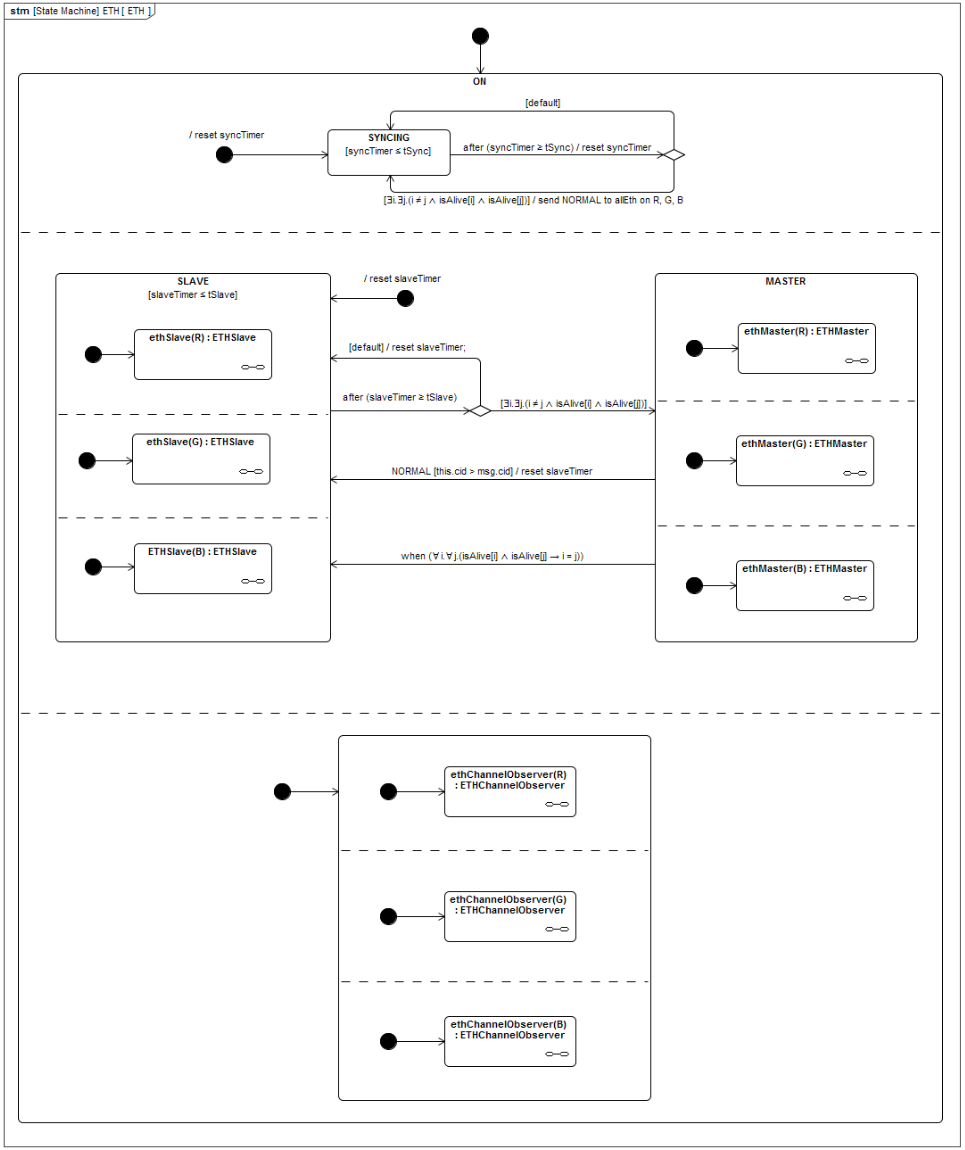
The case study

- Protocol in a railway SCADA (*supervisory control and data acquisition*) system
- Components:
  - *ETH* units [1 .. 4]
  - *LIO* units [0 .. 10]
- Goal:
  1. Election of a unique *ETH* master
  2. Assignment of unique logical addresses (*CIDs*) to *LIOs*

Management and administration

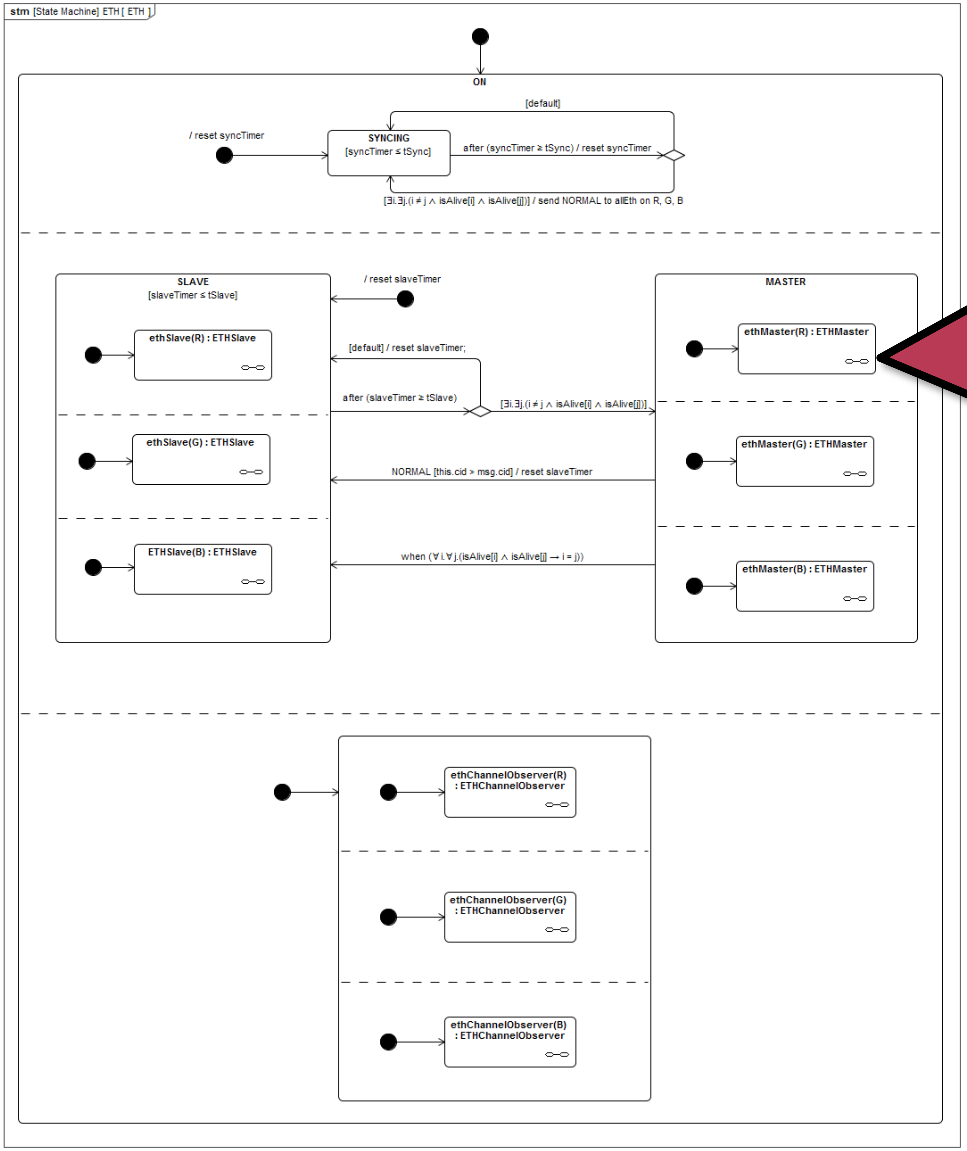


ETH module

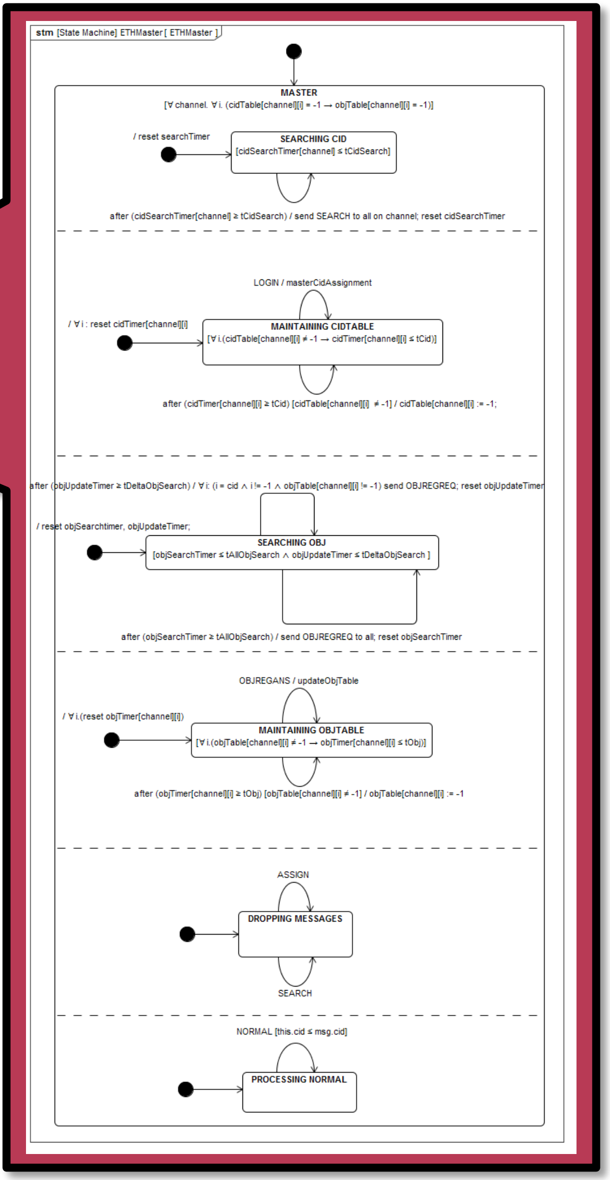


- 3 channels for communication
- MASTER and SLAVE roles

ETH module

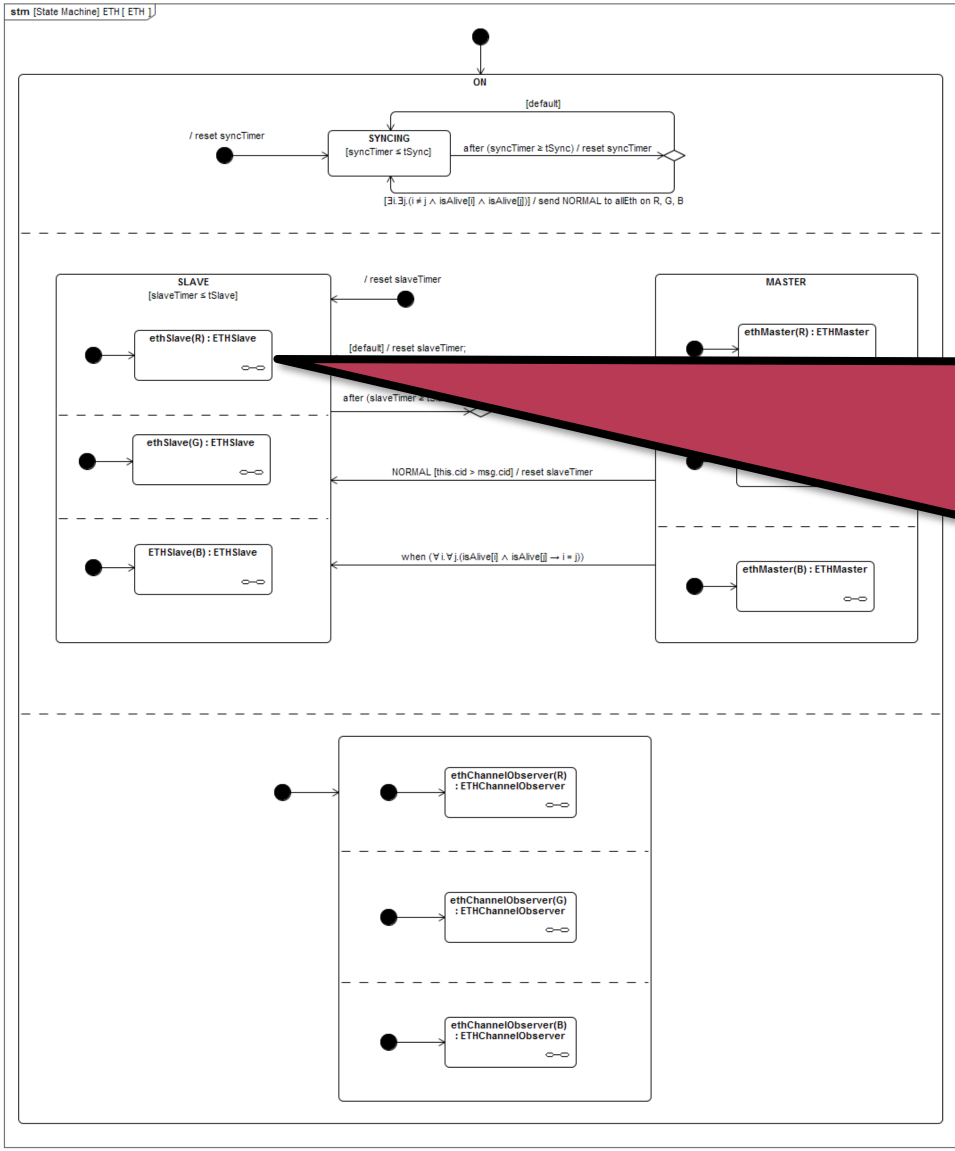


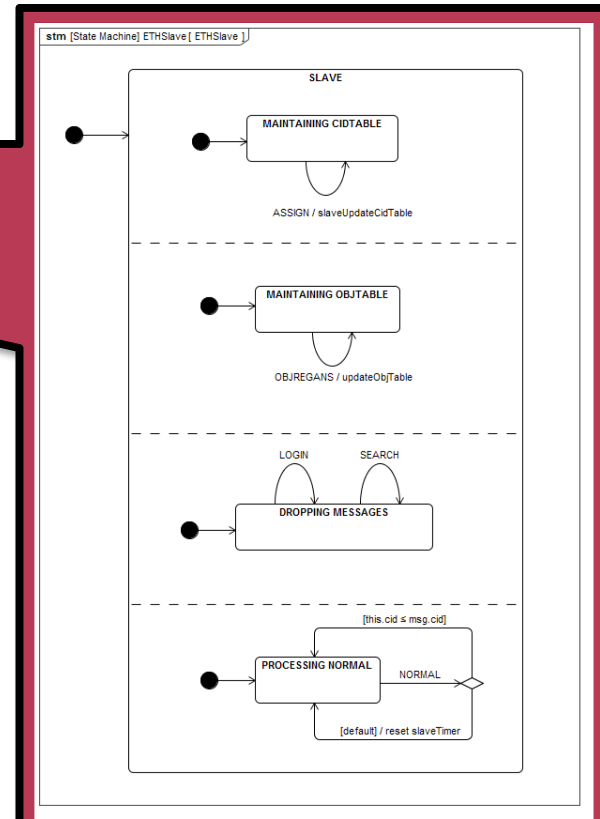
stm [State Machine] ETH [ETH]
ON
[default]
/ reset syncTimer
SYNCING
[syncTimer ≤ tSync]
after (syncTimer ≥ tSync) / reset syncTimer
[∃i.∃j.(i ≠ j ∧ isAlive[i] ∧ isAlive[j])] / send NORMAL to allEth on R, G, B
SLAVE
[slaveTimer ≤ tSlave]
/ reset slaveTimer
ethSlave(R) : ETHSlave
[default] / reset slaveTimer;
after (slaveTimer ≥ tSlave)
[∃i.∃j.(i ≠ j ∧ isAlive[i] ∧ isAlive[j])]
ethSlave(G) : ETHSlave
NORMAL [this.cid > msg.cid] / reset slaveTimer
ETHSlave(B) : ETHSlave
when (∀i.∀j.(isAlive[i] ∧ isAlive[j] → i = j))
MASTER
ethMaster(R) : ETHMaster
ethMaster(G) : ETHMaster
ethMaster(B) : ETHMaster
ethChannelObserver(R) : ETHChannelObserver
ethChannelObserver(G) : ETHChannelObserver
ethChannelObserver(B) : ETHChannelObserver



stm [State Machine] ETHMaster [ETHMaster]
MASTER
[∀ channel. ∀ i. (cidTable[channel][i] = -1 → objTable[channel][i] = -1)]
/ reset searchTimer
SEARCHING CID
[cidSearchTimer[channel] ≤ tCidSearch]
after (cidSearchTimer[channel] ≥ tCidSearch) / send SEARCH to all on channel; reset cidSearchTimer
LOGIN / masterCidAssignment
/ ∀ i : reset cidTimer[channel][i]
MAINTAINING CIDTABLE
[∀ i.(cidTable[channel][i] ≠ -1 → cidTimer[channel][i] ≤ tCid)]
after (cidTimer[channel][i] ≥ tCid) [cidTable[channel][i] ≠ -1] / cidTable[channel][i] := -1;
after (objUpdateTimer ≥ tDeltaObjSearch) / ∀ i: (i = cid ∧ i != -1 ∧ objTable[channel][i] != -1) send OBJREGREQ; reset objUpdateTimer
/ reset objSearchtimer, objUpdateTimer;
SEARCHING OBJ
[objSearchTimer ≤ tAllObjSearch ∧ objUpdateTimer ≤ tDeltaObjSearch]
after (objSearchTimer ≥ tAllObjSearch) / send OBJREGREQ to all; reset objSearchTimer
OBJREGANS / updateObjTable
/ ∀ i.(reset objTimer[channel][i])
MAINTAINING OBJTABLE
[∀ i.(objTable[channel][i] ≠ -1 → objTimer[channel][i] ≤ tObj)]
after (objTimer[channel][i] ≥ tObj) [objTable[channel][i] ≠ -1] / objTable[channel][i] := -1
ASSIGN
DROPPING MESSAGES
SEARCH
NORMAL [this.cid ≤ msg.cid]
PROCESSING NORMAL

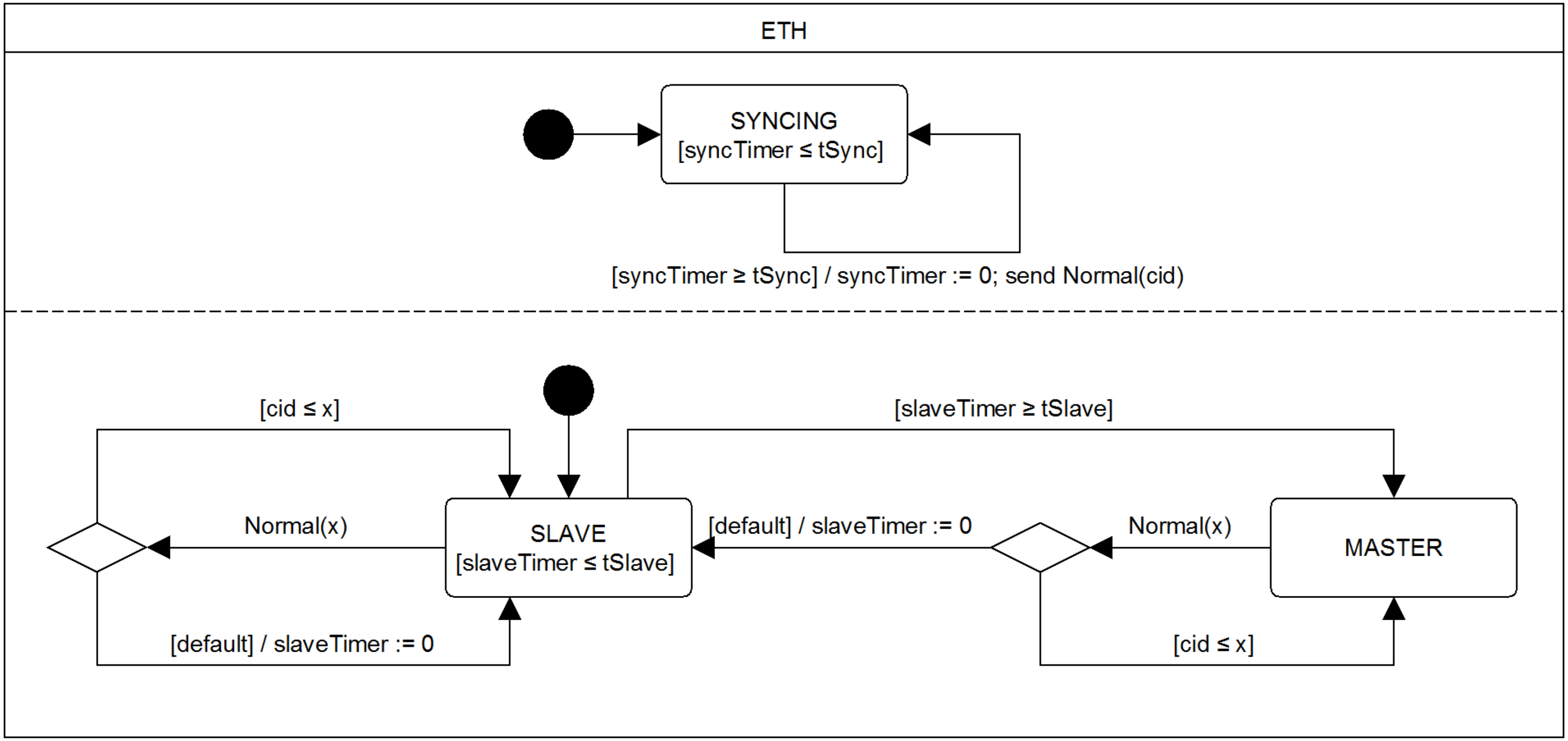
ETH module


stm [State Machine] ETH [ETH]
ON
/ reset syncTimer
SYNCING
[syncTimer ≤ tSync]
[default]
after (syncTimer ≥ tSync) / reset syncTimer
[∃i.∃j.(i ≠ j ∧ isAlive[i] ∧ isAlive[j])] / send NORMAL to allEth on R, G, B
SLAVE
[slaveTimer ≤ tSlave]
/ reset slaveTimer
MASTER
ethSlave(R) : ETHSlave
ethSlave(G) : ETHSlave
ETHSlave(B) : ETHSlave
[default] / reset slaveTimer;
NORMAL [this.cid > msg.cid] / reset slaveTimer
when (∀i.∀j.(isAlive[i] ∧ isAlive[j] → i = j))
ethMaster(R) : ETHMaster
ethMaster(B) : ETHMaster
ethChannelObserver(R)
: ETHChannelObserver
ethChannelObserver(G)
: ETHChannelObserver
ethChannelObserver(B)
: ETHChannelObserver

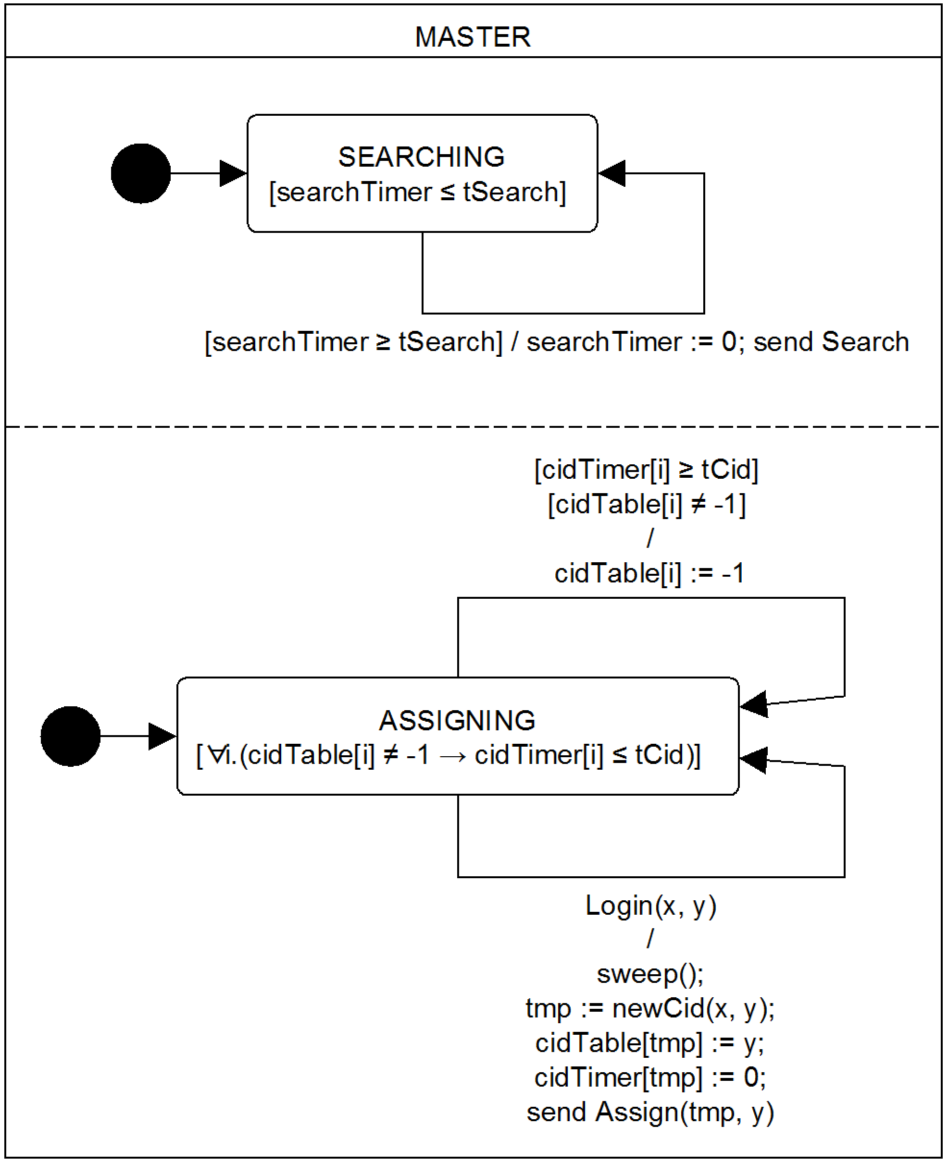

stm [State Machine] ETHSlave [ETHSlave]
SLAVE
MAINTAINING CIDTABLE
ASSIGN / slaveUpdateCidTable
MAINTAINING OBJTABLE
OBJREGANS / updateObjTable
LOGIN
SEARCH
DROPPING MESSAGES
[this.cid ≤ msg.cid]
PROCESSING NORMAL
NORMAL
[default] / reset slaveTimer

SysML Model of master election

- Reducing the model to the master election and CID assignment



SysML Model of CID-assignment



Background

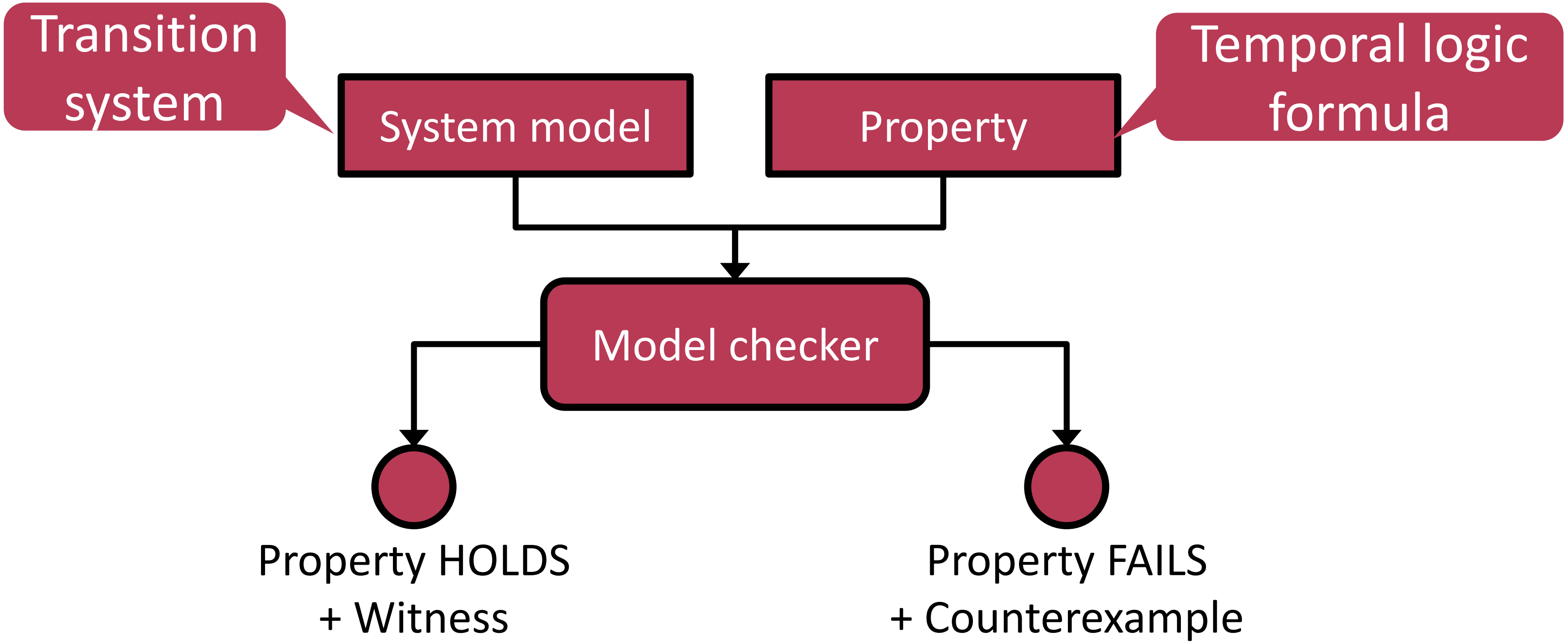
Verification

Formal methods

- Mathematical techniques for
  - Specifying systems
    - Hardware, software, continuous dynamics, ...
  - Reasoning about systems
- Advantages:
  - Applicable in early phase of develoment
  - Unambiguous
  - Automatic (?)

Model checking

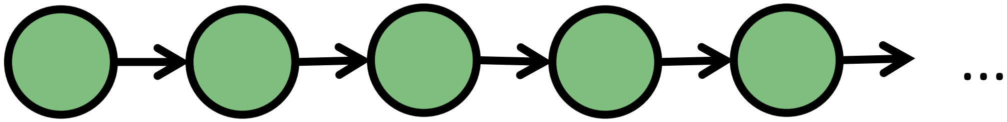
- Automatic property checking
- Exhaustive exploration of the state space



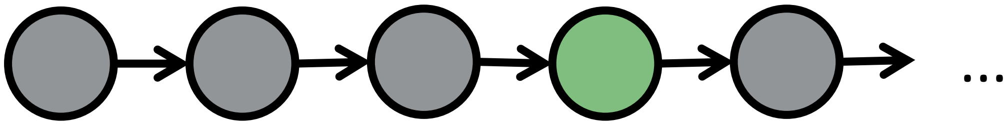
- Advantage: generates counterexample

Linear Temporal Logic (LTL)

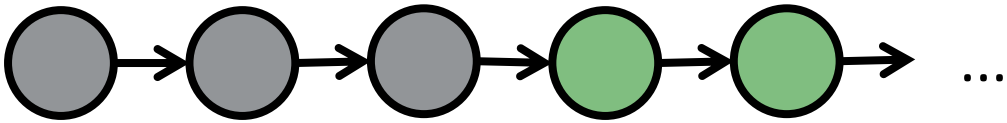
- Gp – p holds **globally** along all paths



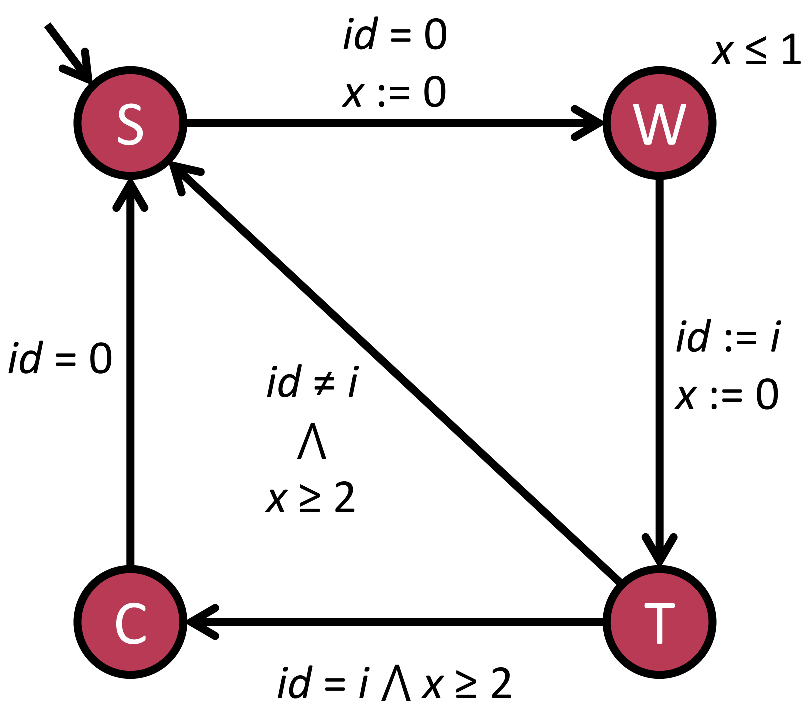
- Fp – p holds in the **future** along all paths



- Example: FGp – **persistence** property

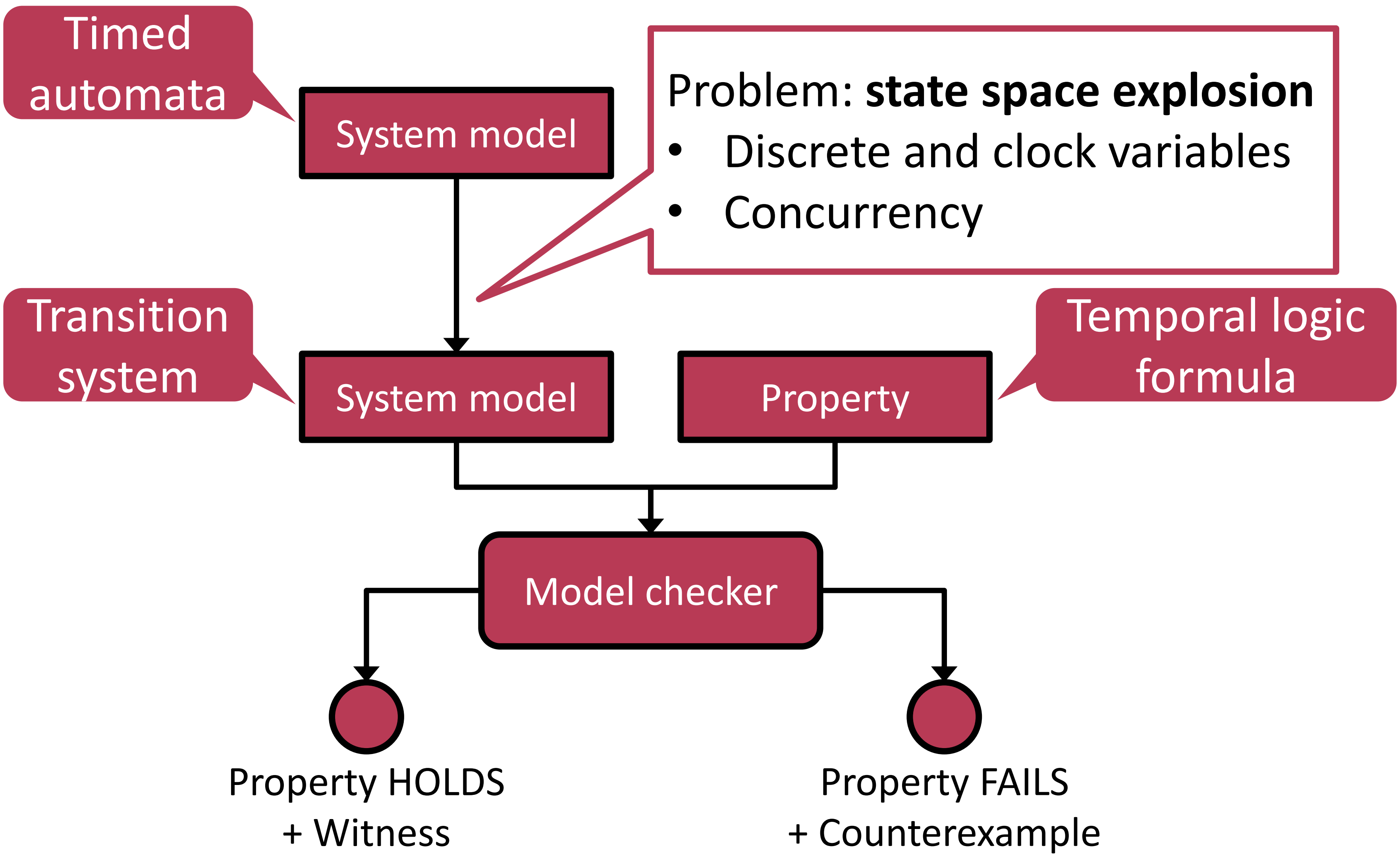


Timed automata



- Program graph +
  - Clock variables
  - Clock constraints
    - Invariants
    - Guards
  - Clock reset

Model checking timed automata

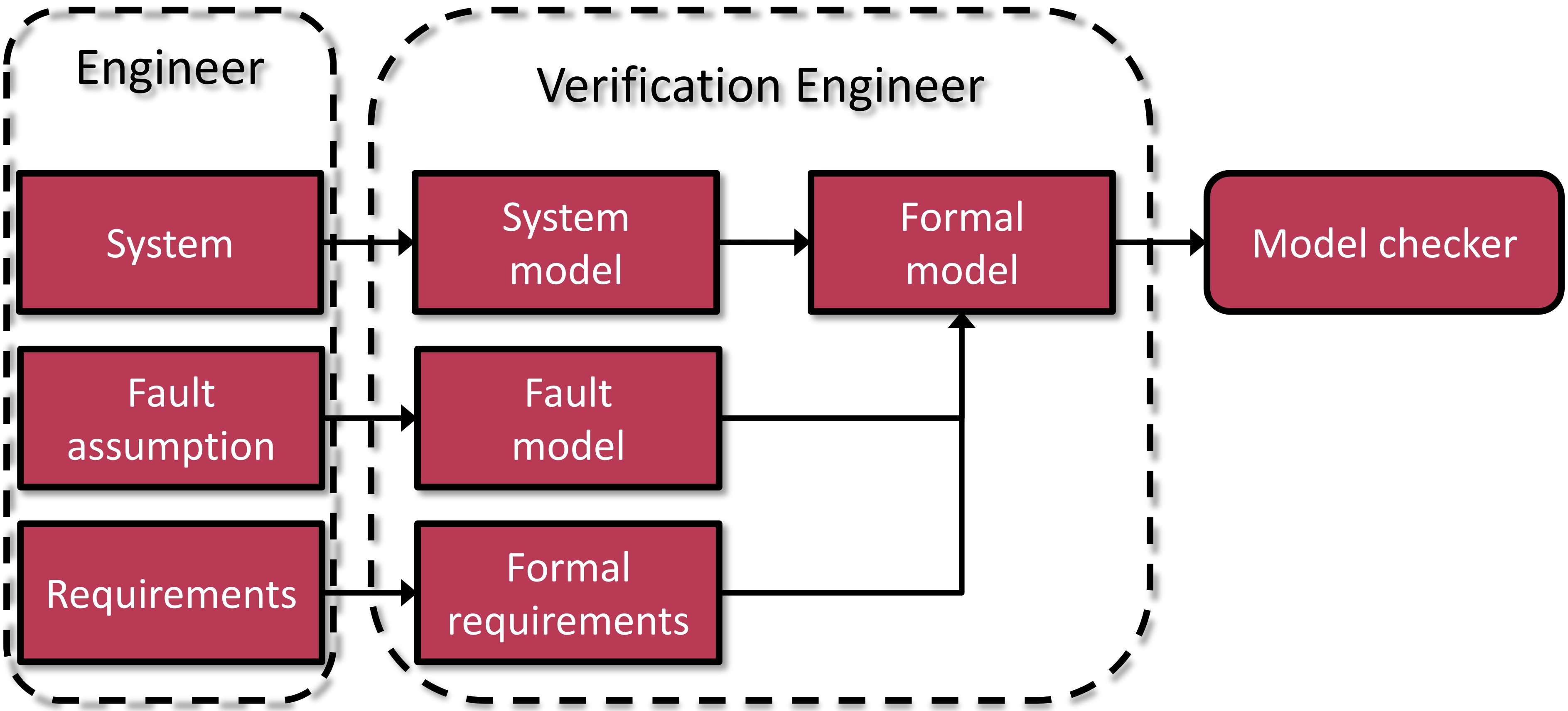


Verification approach

Contribution

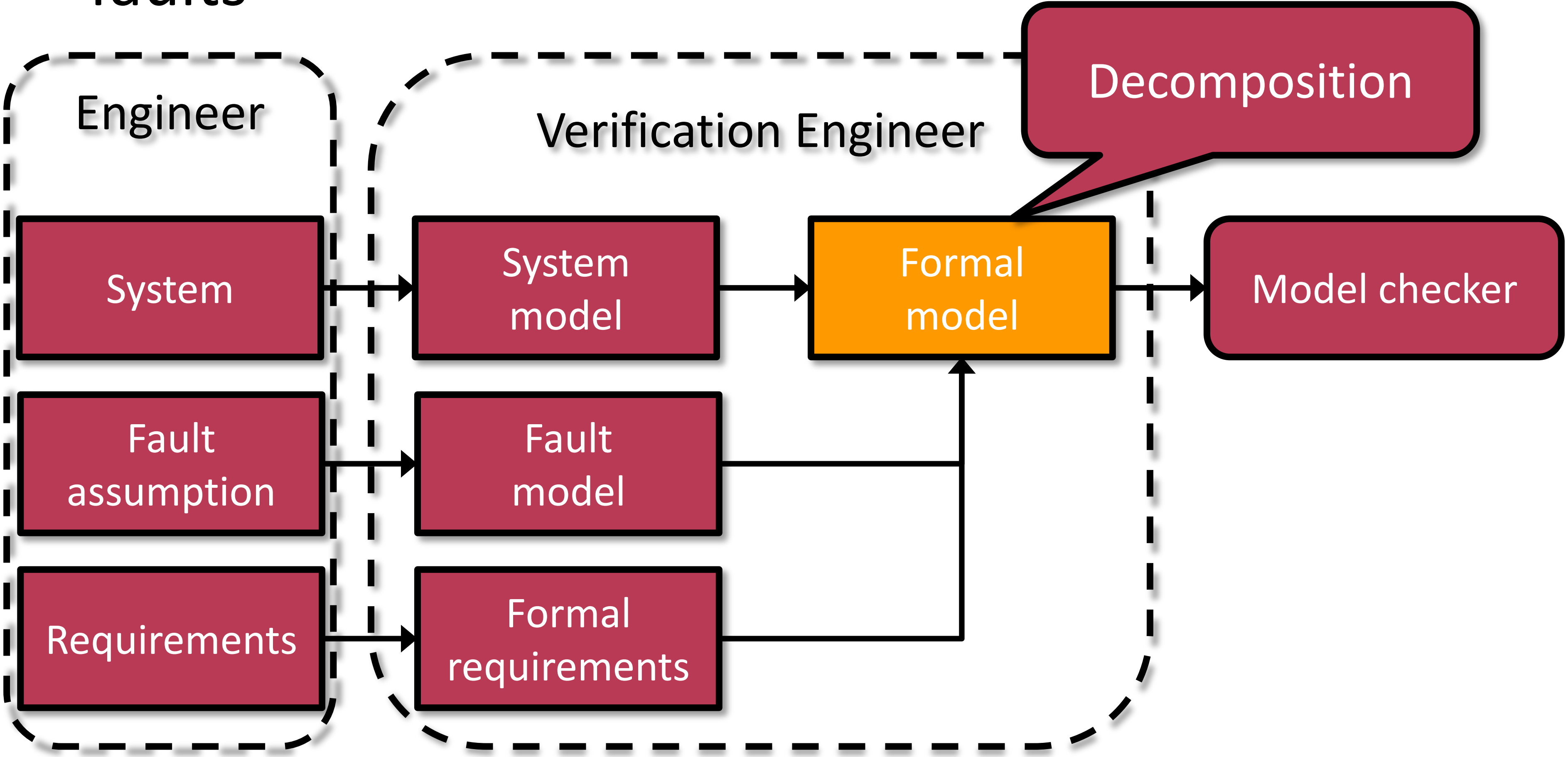
Verification approach

- Goal: Ensure correct behavior in the presence of faults



Verification approach

- Goal: Ensure correct behavior in the presence of faults



Overview

- System modeled as a network of timed automata
- Fault model
  - **Transient faults** as unexpected change of state
- Goal of verification:
  - The system will **finally** work **correctly**

FG (*master election is successfull*
&&
CID assignment is successfull)

Verification approach

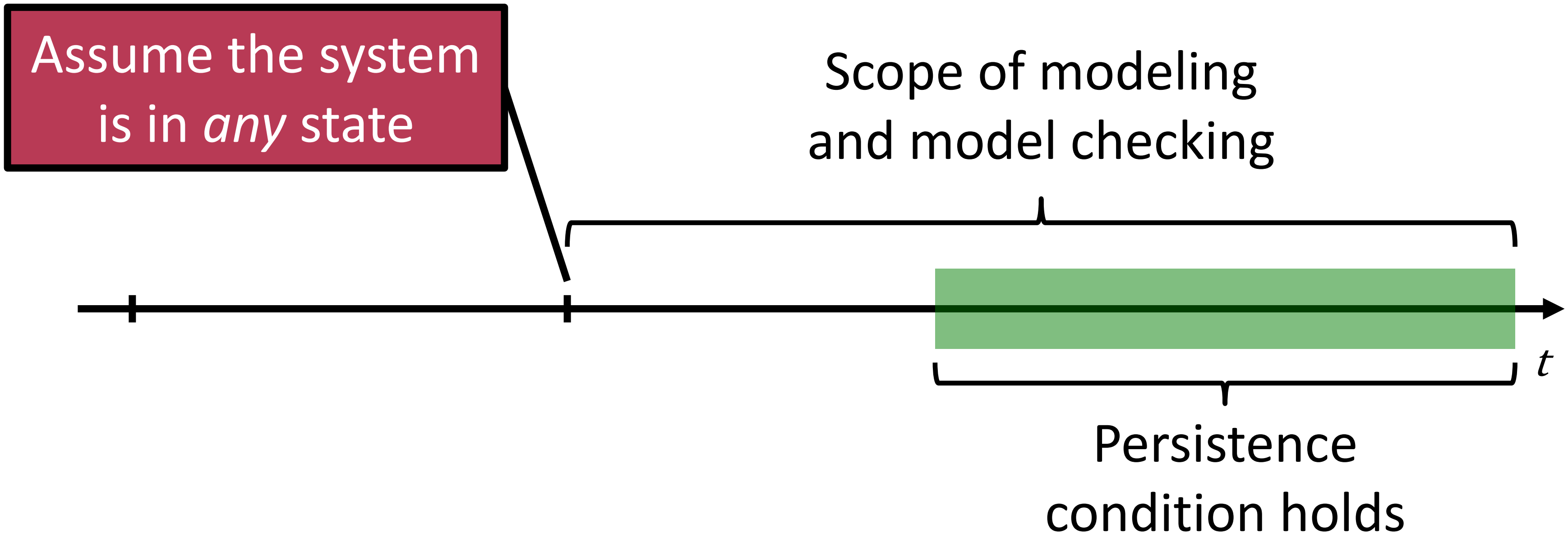
- Goal of verification:
  - The system will **finally** work **correctly**
- Battling state space explosion with decomposition
  - One property depends on the other
    - **Split** the problem into two subproblems
    - Apply **property-preserving simplification** to the systems
  - Both subproperties are persistence properties
    - **Strengthen** to a conjunction of two simpler properties

Modeling faults

- Consider faults that can be modeled as nondeterministic change of model state, e.g.
  - Loss, modification or creation of a message
  - Restart of a unit
  - Modification of a variable
  - ...
- Allow a finite number of occurrences

Fault abstraction

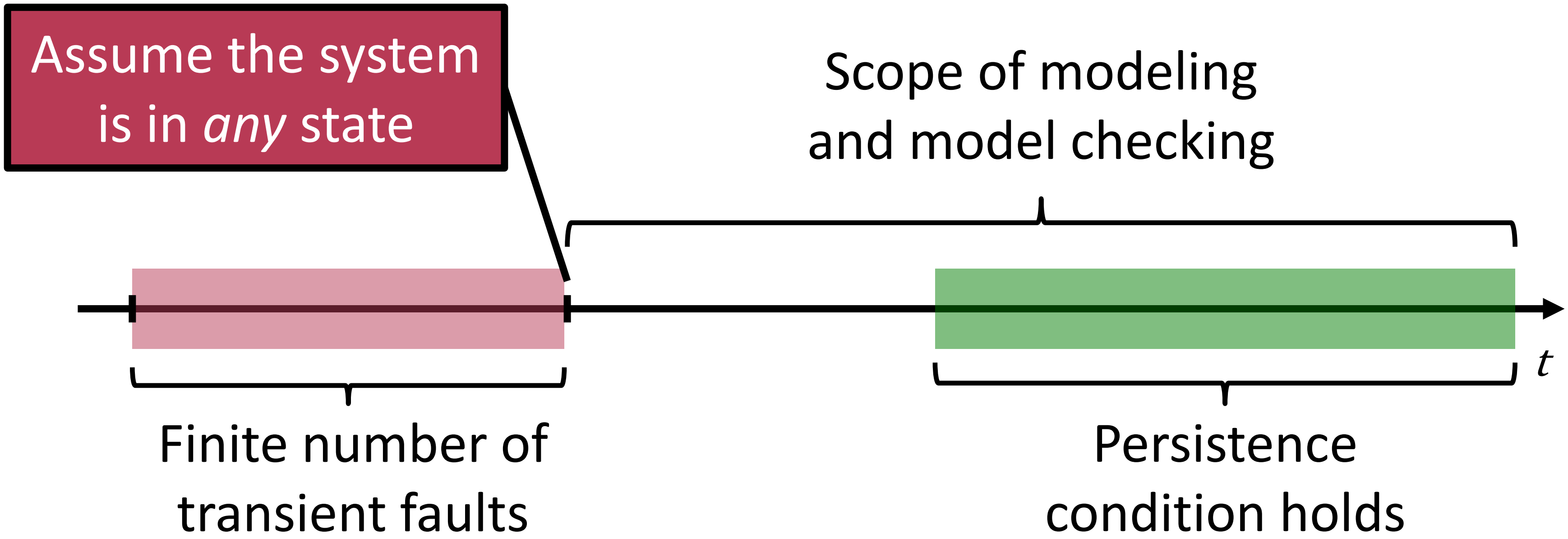
- Instead of modeling faults, we apply abstraction



- If the persistence property holds in the fault free model from any (initial) state,

Fault abstraction

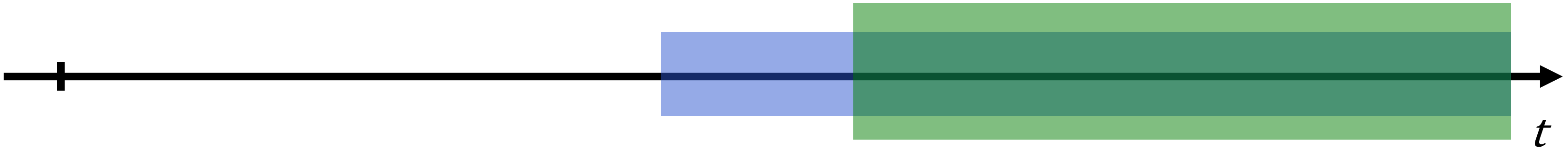
- Instead of modeling faults, we apply abstraction



- If the persistence property holds in the fault free model from any (initial) state,
- It holds after any finite number of transient faults

Decomposition by FG-detachment

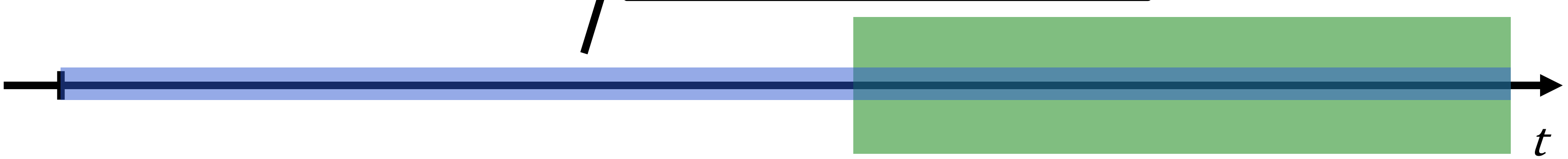
- Instead of checking FG($p \land q$),



- Check 1: FG(p)

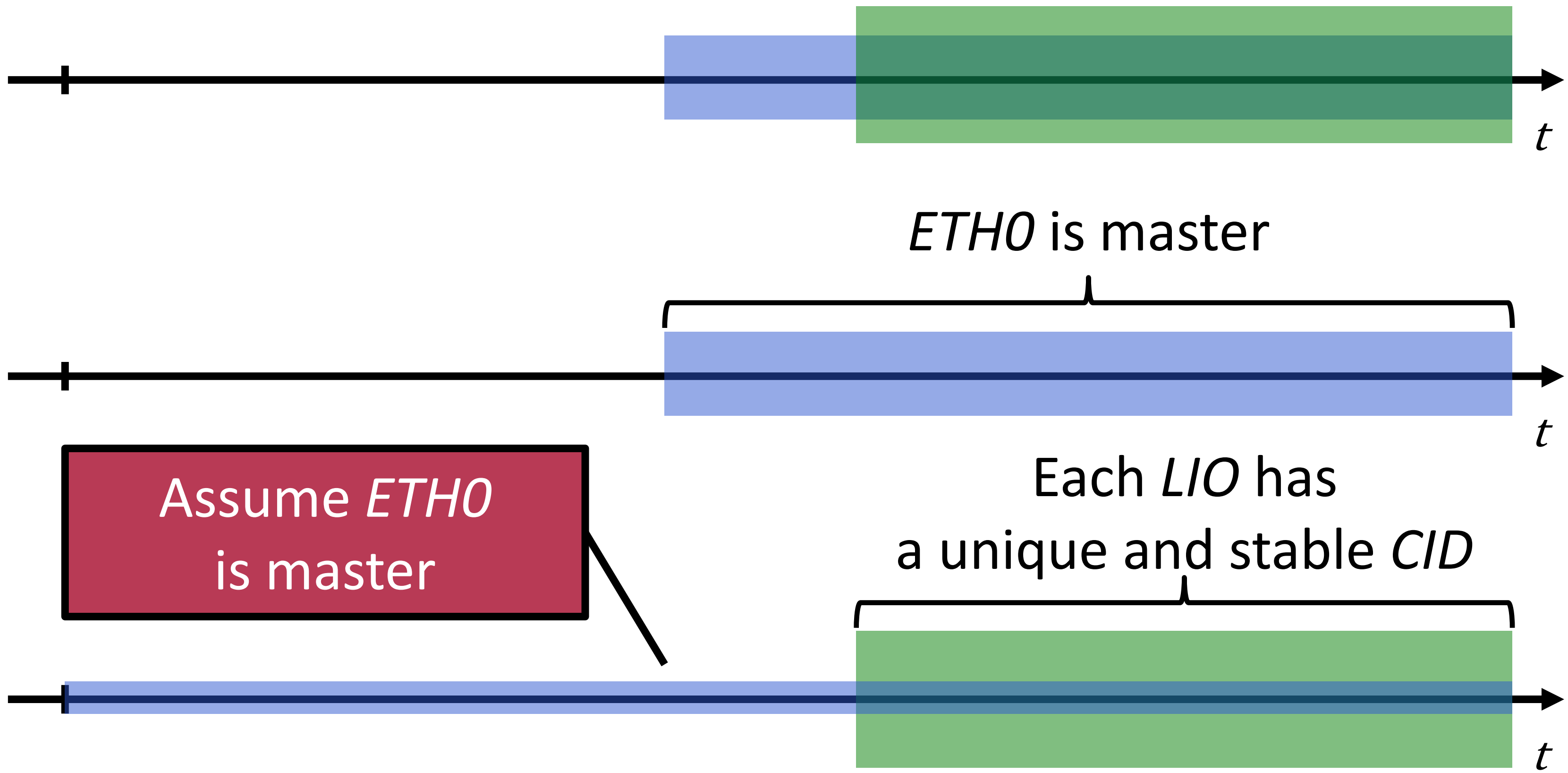
- Check 2: FG(q)

Assume the system only has p-states



- This way, the system to be checked can be significantly reduced

Decomposition by FG-detachment

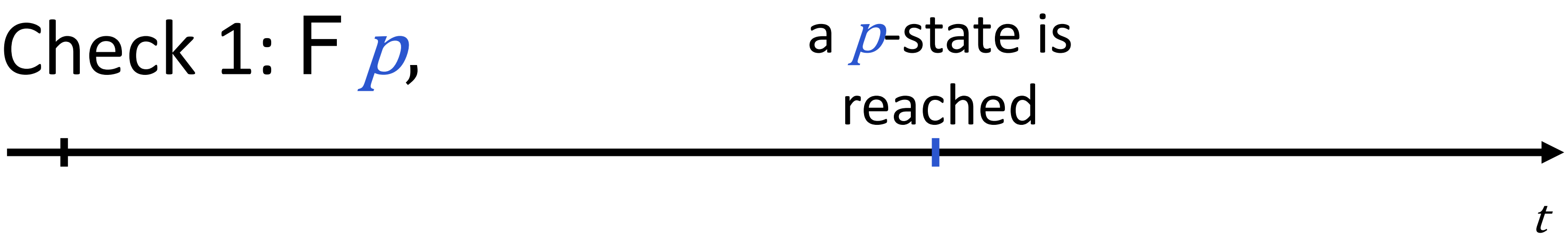


- Master election is not in the cone of influence
- *ETH1*, *ETH2*, *ETH3* is not in the cone of influence

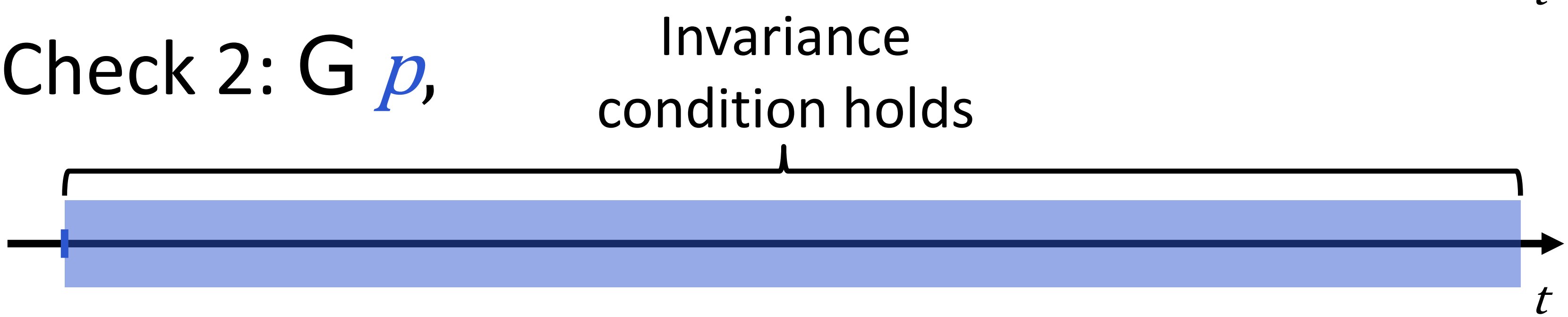
Decomposition by G-detachment

- Instead of checking FG p,

- Check 1: F p,



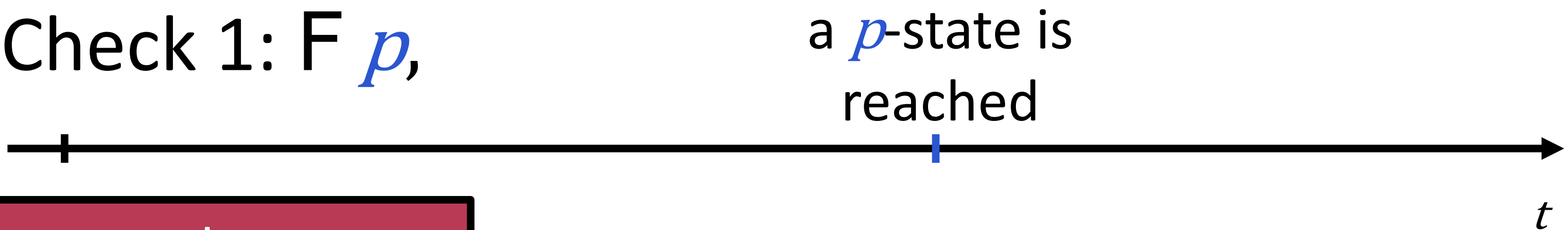
- Check 2: G p,

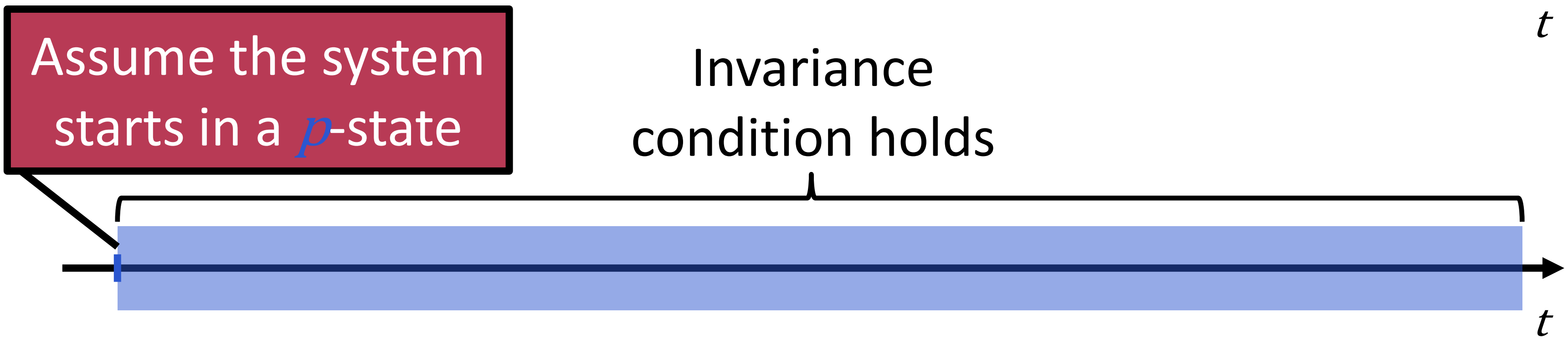


Decomposition by G-detachment

- Instead of checking FG p,

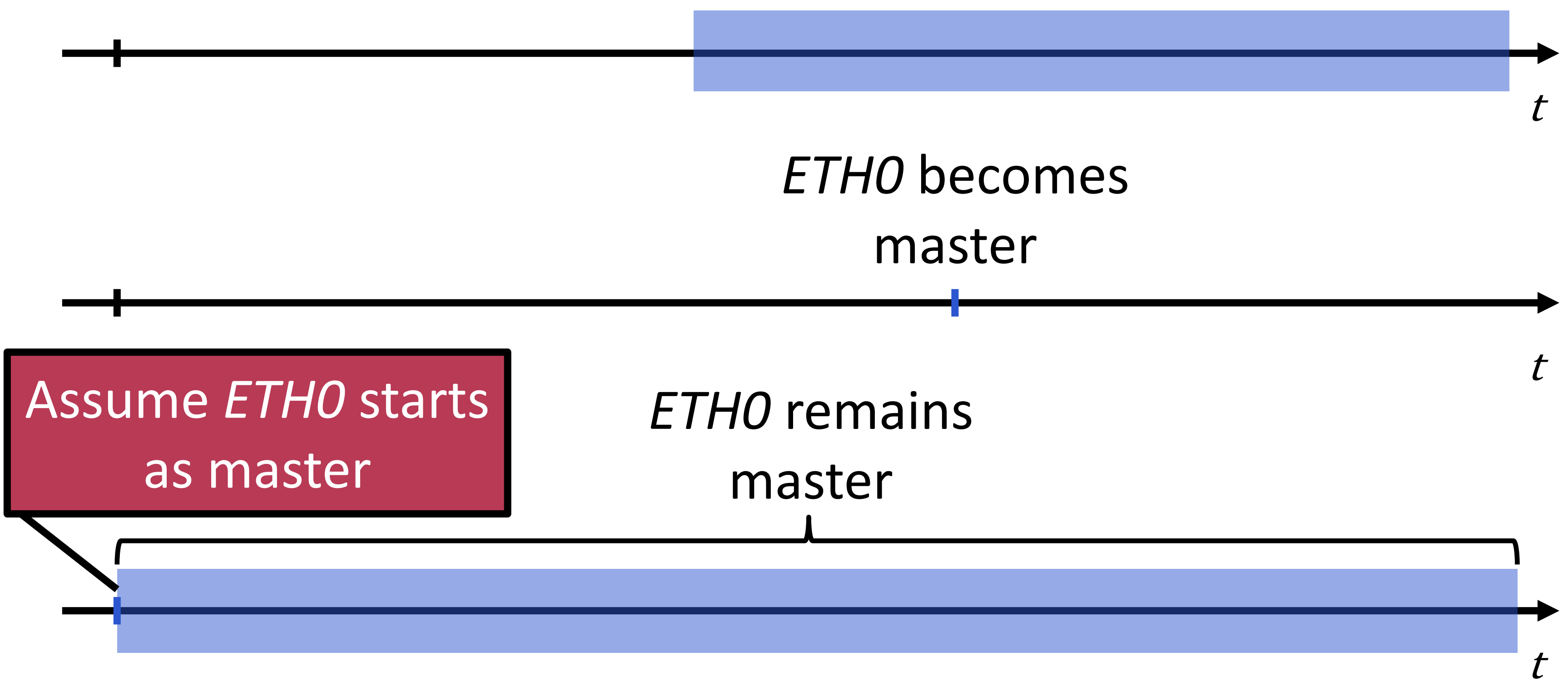
- Check 1: F p,





- Decompose an expensive query into two less expensive ones

Decomposition by G-detachment



Complete verification process

Assume any starting state

ETH0 becomes master eventually

Assume *ETH0* starts as master

ETH0 remains master

Assume *ETH0* is master

Valid *CIDs* are assigned eventually

Assume *ETH0* is master and assigned *CIDs* are valid

Assigned valid *CIDs* are stable

G-detachment + reduction

G-detachment + reduction

Assume any starting state

Master election works as expected

Assume *ETH0* is master

CID-assignment works as expected

FG-detachment + reduction

Assume any starting state

The protocol works as expected

Fault abstraction

Assume a finite number of transient faults may occur

The protocol works as expected

Summary

- Modeled the complete system as a network of timed automata
- Formalized and applied decomposition rules to obtain smaller subtasks
- During verification, **discovered bugs** have been corrected
- The protocol has been successfuly verified in UPPAAL
  - Each query completed in seconds (instead of OOM)