Verifying Platoons in Intelligent Connected Vehicles

Jim Woodcock and Pedro Ribeiro

Southwest University | Aarhus University | University of York

1st July 225 | IFIP WG 1.9 | Paris

with Pedro Antonino and Augusto Sampaio and others

Outline

Outline

Foundations

Project work based on

1. Pedro R. G. Antonino, Marcel Vinícius Medeiros Oliveira, Augusto Sampaio, Klaus E. Kristensen, Jeremy W. Bryans: Leadership Election: An Industrial SoS Application of Compositional Deadlock Verification. NASA Formal Methods 2014: 31-45 213

2. Pedro R. G. Antonino, Augusto Sampaio, Jim Woodcock: A Refinement Based Strategy for Local Deadlock Analysis of Networks of CSP Processes. FM 2014: 62-77

3. Pedro Antonino, Augusto Sampaio, Jim Woodcock: A Pattern-based deadlock-freedom analysis strategy for concurrent systems. CoRR abs/227.8854 (2022)

The Challenge of Autonomy in ICVs

- Intelligent connected vehicles (ICVs) drive on a smart highway. A platoon.
- One vehicle must lead; the others must follow.
- There's no central controller. Just local communication.
- One of the vehicles must be elected to coordinate the platoon.
- The comms links are unreliable. Vehicles must be controlled in real-time.

What Could Possibly Go Wrong?

No outcome	Leadership election deadlocks or livelocks.
Multiple leadership	Command conflict to followers, uncoordinated control.
No leadership	No global decisions, drift, inconsistent manoeuvres.
Conflicting manoeuvres	Vehicles take incompatible actions: lane changes or turns.
Message loss or delay	Mis-coordination, leader brakes, follower doesn't.
Failure to respond	Vehicle joining/leaving isn't integrated/removed.
Partial failure conflict	Violates safety properties.

Example

- Failure Scenario: Conflicting Leadership Decisions
- A 5-vehicle platoon is led by `Vehicle0`.
- `Vehicle0` suddenly loses power and drops out.
- `Vehicle1` detects this and initiates a leadership election.
- `Vehicle2`, has a temporary communication partition.
- It doesn't detect `Vehicle0`'s failure and continues as if `Vehicle0` is still leader.
- `Vehicle1` becomes the new leader and slows the platoon.
- `Vehicle2`, unaware of the change, maintains speed.
- This directly leads to unsafe spacing and a potential rear-end collision.

The Challenge of Autonomy in ICVs

- Leadership Election is Critical
  - Leader controls speed, spacing, manoeuvres. One leader per platoon.
- But Verifying It Is Hard
  - Asynchronous execution No global clock.
  - Unreliable communication Messages may be delayed, lost, or reordered.
  - Dynamic topology Vehicles can join or leave at any time.
  - Partial failures Some vehicles or links may fail independently.
- Formal Methods to the Rescue!
  - Model protocols using CSP (Communicating Sequential Processes).
  - FDR4 model checker: deadlock-, livelock-freedom, consistency.
- Scalability is the Core Challenge
  - Global verification doesn't scale beyond 2–3 nodes.
  - Pattern-based compositionality enables local verification of vehicles and comms.
- Result: Verified ICV Platoons
  - We verify leadership election protocols for up to 32 vehicles (2,048 processes).
  - Deadlock-freedom is guaranteed by design, even with faults and delays.

Outline

Motivation

- We tackle the challenge of verifying leadership election algorithms.
- Inherently complex due to concurrency and dynamic networked nodes.
- Traditional global verification approaches are systematic.
- But they do not scale because they treat every system composition as monolithic.
- This fails to reuse previous verification results.
- We use a compositional approach based on CSP + pattern-based strategy.
- We enable local verification of components, significantly improving scalability.
- We verify deadlock-freedom by constraining the interaction between components.
- Architectural patterns formalised using refinement and predicate abstraction.
- Our main contribution is the analysis of cyclic networks in ICVs.
- Our novel verification pattern enables local verification using FDR.

CSP

- Describes systems as collections of independent processes interacting via events.
- We use CSPM, machine-readable dialect supported by tools like FDR.
- Key constructs in CSP include
  - Basic processes: `STOP` (deadlock) and `SKIP` (termination).
  - Prefixing: `a -> P`: event `a` occurs before process `P`.
  - Choice operators: `P [] Q` (external choice), `P |~| Q` (internal choice).
  - Parallel composition: `P [|X|] Q` synchronises on events in `X`
  - `P ||| Q` runs processes `P` and `Q` independently.
  - Guards (`b & P`), sequencing (`P ; Q`), and recursion (`P = F(P)`) are also available.
  - Hiding and renaming enable abstraction and interface management.
- We use stable failures semantics, which captures:
  - Traces (event sequences process may perform).
  - Failures (traces/refusal-set pairs) indicate events process can refuse after trace.
- `Q` refines `P` under stable failures `P [F= Q` if `Q` doesn't have unexpected failures.

Outline

Platoons in ICVs

- Formations of vehicles that travel together in close coordination.
- Behaviour governed by principles of cooperation, communication, control, safety.
- Dynamically elected leader: velocity, spacing, overall coordination.
- Communication ensures real-time exchange of control and situational data.
- Vehicles can join or leave the platoon dynamically.
- Triggers reconfiguration and potentially leadership re-election.
- Leadership roles may rotate to distribute fuel costs and computational load fairly.
- Safety mechanisms handle faults, delays, dropouts without compromising stability.
- Deadlock-freedom + correct leadership: asynchronous and dynamic conditions.

Platoons in ICVs

1. Formation and Structure

- Platoon Leader vehicle and one or more follower vehicles.
- Vehicles maintain short but safe inter-vehicle distance.
- Enabled by vehicle-to-vehicle (V2V) communication and sensor data (radar, LiDAR).
- Joining ICV may request to join a platoon.
- Request coordinated with leader and adjacent vehicles.
- Leaving Vehicle may leave platoon due to route changes, malfunction, low battery.
- Triggers reconfiguration.

2. Leadership and Decision-Making

- Leader determines speed, route, and manoeuvres (lane changes, braking).
- Followers adapt behaviour using real-time data from leader and neighbours.
- Leadership election protocols used when leader departs or fails.
- Select new leader based on position, fuel level, sensor capability.

Platoons in ICVs

3. Communication and Coordination

- Shared state information: speed, acceleration, brake status, location, intentions.
- Vehicles share using V2V protocols.
- Low-latency, high-reliability communication is critical to maintain stability and safety.
- Consensus mechanisms used coordinate decisions: overtaking or obstacle response.

4. Safety and Control

- Vehicle obeys commands from platoon leader.
- Continuously performs autonomous sensing and actuation.
- Collision avoidance algorithms ensure safety even in communication failure scenarios.
- Fallback modes: revert to autonomous control if platoon integrity compromised.

Platoons in ICVs

5. Efficiency and Benefits

- Fuel and energy savings: reduced aerodynamic drag from close proximity.
- Traffic throughput: improved by reducing vehicle gaps safely.
- Shared tasks (navigation, sensing) reduce load on individual vehicles.

6. Challenges and Research Topics

- Dynamic membership Coping with vehicles joining or leaving.
- Heterogeneity Managing different vehicle types, capabilities, or manufacturers.
- Privacy and security Ensuring messages are authentic, timely, and not spoofed.
- Fault tolerance

  Ensure robustness to sensor failure, communication loss, misbehaving nodes.

Outline

Networks of Processes

- Network set of atomic tuples: identifier, CSP process, alphabet of events.
- Network behaviour defined on alphabetised parallel composition of components.
- Key assumptions for a live network
  1. Busyness — Each component is individually deadlock-free.
  2. Non-termination — Components do not terminate.
  3. Triple-disjointness — No event is shared by more than two components.
- Ungranted requests process waits for another process that cannot respond.
- Two important results underpin compositional analysis
  - Theorem 1 Any deadlock involves a cycle of ungranted requests.
  - Theorem 2 A network is deadlock-free if all its essential components (obtained by removing disconnecting edges in the communication graph) are deadlock-free, and no conflicts exist on the edges.
- Basis for decomposing verification, especially useful in acyclic topologies.

Pattern Based Approach to Cyclic Network Verification

- Architectural pattern designed to enable local deadlock analysis in cyclic networks.
- Key ideas
  - The system is structured with two kinds of nodes:
    1. Participants — System components that send/receive data.
    2. Transport entities — Intermediary components that manage communication. Detect participant on/off states.
  - Pattern constraints both behavioural and structural
    - Participants interact only via transport entities.
    - Transport layers are unidirectional and monitor sender status.
    - Message exchange and node state changes follow a predefined order.
- Formalisation
  - Behaviour specified using parametrised CSP processes.
  - Conformance to the pattern is checked using stable failures refinement $[F=$.
  - Additional structural constraints ensure clean separation between participants and transport, well-formed communication alphabets, and order-preserving message flow.

Theorem 3: Deadlock-Freedom by Pattern Conformance

Statement

- If a network satisfies all pattern constraints, then it is deadlock-free.

What This Means

- The system is structured using two node types:
  - Participants (e.g., vehicles): send/receive messages.
  - Transport entities (e.g., bus cells): relay messages, detect on/off status.
- Communication is:
  - Indirect only Via transport layers.
  - Unidirectional With failure detection.
  - Ordered Participants follow a fixed send/receive sequence.

Consequence

- Local refinement checks (`[F=]`) of each node and transport suffice.
- Entire cyclic network is guaranteed deadlock-free.
- Enables scalable verification even with 32 vehicles and 2,048 processes.
- Theorem 3: Pattern adherence $\Rightarrow$ network is deadlock-free

Why Theorem 3 Works: Informal Illustration

- Key Idea Deadlocks in CSP arise from cycles of ungranted requests. The pattern prevents such cycles by construction.
- How the Pattern Prevents Deadlock
- No peer-to-peer entanglement

  Participants can interact only via well-behaved transport entities.
- No cycles of confusion If a participant waits, it's only for a transport entity that:
  - either has data to send,
  - or reports that the source is off.
- Ordered interaction Participants send and receive in a known, fixed sequence.
- Their expectations cannot deadlock against each other.
- Transport always responds or times out Bus cells never block silently.
- This breaks potential cycles.
- Result Every request is either fulfilled, refused, or timed out in a finite way.
- No circular waiting means no deadlock.

Applying the Three Theorems: Ensuring Deadlock Freedom

- Theorem 1 Cycle Characterisation
  - Any deadlocked state in a live network involves a cycle of ungranted requests.
  - Implication: Deadlock prevention reduces to avoiding such cycles.
- Theorem 2 Decomposition into Essential Components
  - A network is deadlock free if:
    1. It decomposes into essential components via disconnecting edges,
    2. Each component is deadlock free.
    3. All disconnecting edges are conflict-free.
  - Enables compositional verification in tree-like or sparse topologies.
- Theorem 3 Pattern-Based Verification for Cyclic Networks
  - A novel communication pattern ensures deadlock freedom by:
    - Structurally preventing cycles of ungranted requests.
    - Enforcing behavioural conformance using stable failures refinement.
  - Enables fully local analysis, even for cyclic topologies with dynamic behaviour.

Outline

Fairness

- Platoon leader vehicle consumes more fuel than followers.
- Aerodynamic drag Leader breaks air resistance.
- Followers benefit from slipstream lower air pressure and turbulence.
- Reduces energy expenditure.
- Is that fair? No! Not in a naïvely managed platoon.
- Four strategies to address this imbalance.

Fairness

1. Leadership Rotation
   - Vehicles take turns as leader. Ensures fuel burden is equitably distributed over time.
   - Common in biological systems like bird flocks or cycling pelotons.
   - Leadership rotation protocols

     Formalised using distributed algorithms with fairness and safety guarantees.
2. Incentive Mechanisms
   - Vehicles owned by different users or fleet operators.
   - Compensate leader using credits or micropayments.
   - Leads to economic fairness without requiring every vehicle act as leader.

Fairness

3. Leader Selection Based on Capability
   - Volunteer elected as leader.
   - Vehicles agree to lead based on preferences and capabilities.
   - Greater fuel reserves, better sensor capabilities, lower energy cost per mile.
4. Weighted Cost Models
   - Platoons use cost-sharing model.
   - Vehicle's cost (fuel, wear-and-tear) estimated.
   - Algorithm determines fair compensation or task allocation.
   - Handled by contractual agreements or fleet-level optimisation policies.

Outline

Isn't the Leadership Election Problem Already Solved?

- Selecting a unique coordinator in a distributed system is indeed "solved".
- Well-known algorithms offer strong distributed consensus consistency guarantees:
  - Bully Algorithm (Garcia-Molina, 1982).
  - Ring-based election.
  - Paxos (Lamport) and Raft (Ongaro and Ousterhout).
- But it isn't completely solved in all contexts, particularly when you account for:
  1. New application domains.
  2. Scalability and verification.
  3. Safety guarantees in design.
  4. Adaptability.

Problems

1. New Application Domains

- ICVs, audio/video entertainment systems, IoT, cyber-physical systems.
- All involve domain-specific constraints like:
  - Dynamic topologies (vehicles or nodes can leave/join)
  - Real-time or soft/hard deadlines
  - Fault-tolerance in asynchronous communication
  - Energy constraints or partial observability

2. Scalability and Verification

- Leadership election protocols can validated using traditional model checkers.
- Suffers from state explosion, particularly in cyclic or asynchronous topologies.
- Pattern-based compositional verification technique enables local reasoning.
- Deadlock freedom scales to industrial-sized systems.
- For example, 32-node networks at Bang & Olufsen (Antonio et al.).

Problems

3. Safety Guarantees in Design

- Functional logic of leadership election can be correctly implemented.
- Guarantees important properties
  - Deadlock-freedom, livelock-freedom, eventual consistency under failures.
- Requires domain-specific formal verification.

4. Adaptability

- Existing algorithms assume specific communication models: synchronous, reliable.
- They assume failure models crash, byzantine.
- They include assumptions total order on IDs.
- These may not hold in every application.
- ICVs message loss, delay, and partial knowledge are common.

Summary: Isn't the Leadership Election Problem Already Solved?

- Abstract leadership election problem is algorithmically understood.
- Concrete deployment contexts: safety-critical, dynamic, real-time systems.
- Require new modelling, refinement, and verification strategies.
- These ensure correctness and robustness.
- Further research is needed and relevant.

Outline

Objective

Scientific research objective

Model and verify a deadlock-free leadership election protocol for a platoon of ICVs using CSP and FDR4.

Engineering research objective

Create push-button solution that works for 32 vehicles.

Current research

- CSP model of leadership election protocol for a platoon of ICVs.
- Asynchronous vehicle behaviour
- Communication over bus cells.
- Dynamic process of electing unique leader.
- Explain architecture, components, and formal verification of deadlock-freedom.
- All checks carried out using FDR4 model checker.

System Overview

- 3 Vehicle processes `Vehicle0`, `Vehicle1`, `Vehicle2`
- 6 Bus cells unidirectional communication channels
- Communication via channels `cp_pack`, `switchOn`, `timeout`, etc.
- Asynchronous, interleaved execution.
- Verification uses stable failures model nondeterminism.
- Pattern-based decomposition reduces verification to local refinement checks.
- Model asynchronous execution, dynamic vehicle participation, message timeouts.
- Same approach scales to industrial strength.

Steps of the Protocol

1. Vehicle Discovery
   - Vehicles broadcast a "Hello" message upon detecting others nearby.
   - Example: "I am Vehicle1, ID=1, Pref=High".
2. Exchange and Comparison
   - Each vehicle shares metadata (e.g., speed, heading, priority).
   - Vehicles construct a list of candidates.
3. Leader Election
   - Run a deterministic protocol (e.g., highest ID wins).
   - Optional fallback: consensus via voting, or timeout for re-election.
4. Confirmation
   - Elected leader broadcasts confirmation: 'Leader = VehicleX'.
   - Followers acknowledge and begin formation.
5. Formation Complete
   - Platoon enters normal operation mode, maintaining inter-vehicle spacing.

Initial Platoon Formation & Leader Election Scenario

- First stage in platooning protocol.
- Vehicles come into proximity and negotiate who should be the leader.
- Goal: Vehicles that come into range must:
  - Discover each other.
  - Exchange identifiers or priority metadata (e.g., fuel level, destination, timestamp).
  - Agree on a unique leader.
  - Form a communication topology for platooning.

Kernel Model Overview

- Model represents platoon of three vehicles `Vehicle0`, `Vehicle1`, and `Vehicle2`.
- Each vehicle can be in one of four roles: `undecided`, `leader`, `follower`, or `off`.
- Vehicle communication unidirectional bus cells simulating point-to-point links.
- Fully asynchronous system vehicles operate independently.
- They join and leave the platoon at any time.
- Vehicles initiate communication by broadcasting status and listening to others.
- Bus cells simulate message loss and timeouts: real-world nondeterminism.
- Leadership is determined based on priority values and tie-breaking by ID.
- Local observation triggers dynamic role change: new leader, follower, re-election.
- Model provides compact setting to verify deadlock-freedom using FDR4.

Election Logic Summary

- Initial State: Each node starts in the undecided state with a priority value.
- Broadcast Cycle:
  - Node broadcasts its current state and priority.
  - Receives state and priority from peers (or timeout if peer is off).
- Decision Making:
  - If any leader exists: become follower.
  - If none:
    - Become leader if node has highest priority.
    - Tie-break using highest ID.
- State Evolution:
  - Leader: Increments priority if still leader after a full cycle.
  - Follower: Reverts to undecided if no leader detected.
- Resilience: Nodes may rejoin or fail at any time; the protocol ensures convergence.

System Composition

- Complete network, `ICVPlatoon`, constructed by interleaving vehicles and bus cells.
- Ensures each vehicle executes independently and asynchronously.
- Model 3 vehicles and 6 unidirectional bus cells (two between every ordered pair).
- Communication structure reflects a fully connected topology.
- Every vehicle talks to every other.
- Dynamic behaviour preserved vehicles can join and leave platoon at runtime.
- Model supports message loss, timeouts, re-elections, emulating realistic faults.
- Composition used for asserting system-wide properties, like deadlock freedom.

Assertion: Deadlock Freedom

```
assert ICVPlatoon :[deadlock free]

-- checks ICVPlatoon system is deadlock free.
-- Always at least one enabled event in any reachable state.
-- Ensures liveness of the protocol.
-- System doesn't get permanently stuck.
```

Verification

- Model concludes with assertion `assert ICVPlatoon :[deadlock free]`.
- This is successfully verified using FDR4 the platoon is indeed free from deadlock.
- Proves system cannot enter deadlock state under any execution schedule.
- System ensures leadership election proceeds...

  ...even with communication failures or dynamic vehicle membership!
- `nleaders.a!n`: vehicle `a` reports number of known leaders (consistency check).
- `hpetition.a!p`, `hpetitionid.a!b`: highest priority and ID candidate seen by `a`.
- Channels are directionally indexed: `a.b` indicates sender `a`, receiver `b`.
- Communication is asynchronous and decoupled.
- Each vehicle maintains local knowledge and responds accordingly.
- Channels collectively implement a decentralised coordination protocol.
- Enables robust leader election under faults.

Correctness Properties to Verify

- Using CSP/FDR or model checking, you can verify:
- Uniqueness Only one leader elected.
- Agreement All vehicles agree on the leader.
- Liveness Election eventually terminates.
- Fairness No vehicle with higher priority is ignored.

Outline

Local Strategy

- Evaluate leadership election model using three verification strategies in FDR:

1. Global verification
   - Entire system modelled and verified monolithically.
   - Fails to scale due to state-space explosion.
   - Exceeds memory for > 2 nodes.
2. Decomposition-based verification
   - Uses Theorem 2 to verify essential components separately.
   - Helps with some components (e.g., memory cells).
   - But still not scalable for the subnetwork of nodes and buses.
3. Pattern-based local verification (proposed approach)
   - Applies the pattern to reduce analysis to local checks.
   - Each node and bus cell is verified for conformance via CSP refinement.
   - Verifies deadlock-freedom compositionally and efficiently.

Local Strategy & Experimental Results

- Compared three verification strategies using FDR:
  1. Global model checking fails beyond 2 nodes (state explosion).
  2. Decomposition (Theorem 2) helps, but can't handle cyclic subnetwork.
  3. Proposed pattern-based approach: only viable strategy.
- Each node and bus cell verified locally via CSP refinement.
- Scalability:
  - Realistic platoon (32 nodes).
  - Verification time ≈ 1.43 hours.
  - Structural constraints + conformance assertions proved sufficient.
- Demonstrates efficiency and scalability of local, pattern-based analysis.

Operational Scenarios

- Leadership Change
- Platoon Formation
- Platoon Splitting
- Platoon Merging (Rejoining)
- Vehicle Insertion and Removal
- Obstacle Handling
- Junction and Roundabout Navigation
- Lane Change and Multi-lane Navigation
- Toll Booths and Checkpoints
- Communication Failure
- Sensor Failure
- Mechanical Failure or Emergency Braking
- Security and Trustworthiness
- Route and Destination Mismatch
- Traffic Law Compliance
- Energy Optimisation and Fairness
- Scalability
- Mixed Autonomy
- Handover with Infrastructure or Other Platoons

Outline

Conclusion and Related Work

Conclusion

- Architectural pattern Local analysis: deadlock-free cyclic distributed systems.
- Formalised behavioural conformance using CSP refinement: stable failures model.
- Adaptation of B&O's leadership election protocol.
- Successfully verifies model with up to 32 nodes and 2,048 processes.
- Method is scalable. Allows verification early in development.
- Real-world applications and implementations.

Looking Ahead

- At scale. Can we reach 55 members?
- Deadlock-free platoons. Totally machine checked.
- Verified by design. Totally machine checked.