Integrating Symbolic AI Planning with Formal Verification for Safety Enhancement in the V2X Ecosystem

Christopher Agboile

**Abstract**

The Vehicle-to-Everything (V2X) ecosystem is poised to revolutionize transportation by enabling cooperative autonomy. However, the inherent complexity and safety-critical nature of these systems demand a paradigm shift beyond probabilistic and data-driven AI approaches, which often lack verifiable safety guarantees. This paper proposes a novel framework for integrating Symbolic Artificial Intelligence (AI) Planning with Formal Verification (FV) to provide a foundation for provably safe decision-making in V2X environments. Symbolic AI planning offers explicit, logical, and explainable high-level behaviour generation, while formal verification provides mathematical rigor to prove that such behaviours adhere to critical safety properties under all specified conditions. The paper presents a hybrid architecture where a symbolic planner generates candidate manoeuvres, which are then formally verified against a temporal logic model of the traffic environment before execution. A feedback loop allows for plan repair and re-verification in case of property violations. The paper detail the formal model, the integration methodology, and a simulation. The results demonstrate the feasibility of generating plans that are not only functionally correct but also formally guaranteed to avoid a defined set of hazardous states. Finally, we discuss the significant challenges of scalability, real-time performance, and environmental modelling, outlining directions for future research to transition this approach from theory to practical deployment.

**Keywords:** V2X, Autonomous Vehicles, Symbolic AI, Automated Planning, Formal Verification, Temporal Logic, Safety, Provable Guarantees, Cooperative Driving.

1. **Introduction**

The vision of fully autonomous, cooperative transportation systems is rapidly approaching reality, largely driven by advances in V2X communication. This technology enables a vehicle to perceive its environment beyond the line-of-sight limitations of onboard sensors, creating a shared, holistic understanding of the traffic situation. This capability is foundational for advanced applications like platooning, cooperative intersection management, and emergency vehicle pre-emption, which promise unprecedented gains in safety, efficiency, and traffic flow [1].

Despite this promise, a critical barrier remains: the assurance of functional safety (ISO 26262) and safety of the intended functionality (SOTIF - ISO 21448) in these complex, cyber-physical systems [2]. Traditional validation methods like simulation and testing are inherently incomplete, unable to explore the exponential number of edge cases and unpredictable interactions in an open-world environment [3]. Furthermore, the prevailing reliance on deep learning and other sub-symbolic AI methods for perception and control introduces a "black box" problem where their decision-making processes are opaque and difficult to certify for absolute safety [4].

To address this, a two-pronged approach is necessary: **explainability** and **verifiability**. Symbolic AI Planning, which operates on a world model composed of symbols, actions, and logical predicates, provides the former. It can generate human-readable, logically constructed plans ("request lane change," "confirm gap availability," "execute manoeuvre") [5]. However, a plan that is logically sound in an abstract model may fail in the real world due to timing errors, uncertain dynamics, or unmodeled interactions.

Formal Verification (FV), the second prong, addresses this gap. FV uses mathematical logic to provide exhaustive proofs that a system model satisfies certain desired properties ( "collision freedom") under all possible operating conditions [6]. While powerful, applying FV to the entire continuous state-space of a V2X system is computationally prohibitive.

This paper argues that the integration of these two fields is not merely beneficial but essential for achieving verifiable safety in V2X. This paper propose a framework where the strengths of one compensate for the weaknesses of the other: the symbolic planner generates intelligible strategies, and formal verification acts as a "safety filter," certifying these strategies before deployment.

1. **Background and Related Work**
   1. **Symbolic AI Planning**

Symbolic AI planning, often exemplified by the Planning Domain Definition Language (PDDL), involves a planner, a domain file (actions/preconditions/effects), and a problem file (objects/initial state/goal) [5]. It searches for a sequence of actions that transitions the world from an initial state to a goal state. Hierarchical Task Network (HTN) planning is particularly relevant for V2X, as it can efficiently decompose complex driving tasks ("negotiate intersection") into primitive, executable actions [7].

* 1. **Formal Verification**

Formal Verification uses mathematical models to check system correctness. Key techniques include:

* **Model Checking:** An automated technique for verifying finite-state concurrent systems against temporal logic specifications (e.g., Linear Temporal Logic - LTL, Computational Tree Logic - CTL) [6]. Tools like NuSMV and UPPAAL are industry standards.
* **Theorem Proving:** Using deductive reasoning within a logic (Higher-Order Logic) to prove that a system model satisfies a set of properties. It is more expressive but less automated than model checking.
  1. **Related Work**

Previous research has explored both fields independently in automotive contexts. [8] used model checking to verify protocols for platooning. [9] applied symbolic planning for urban driving manoeuvres. The integration of planning and verification is an emerging topic. [10] proposed "verified planning" for robotics, focusing on low-level motion planning. [11] used runtime verification to monitor plan execution. This work distinguishes itself by focusing on the V2X-specific challenges of cooperation, communication, and real-time, high-level plan verification *before* execution, creating a proactive safety assurance mechanism.

The direct integration of these two fields is a more recent and active area of research, though often not specifically focused on V2X. The core idea is to subject a planner's output to formal scrutiny. [10] proposed a paradigm of "verified planning" for robotics, where a motion plan is translated into a hybrid system model and verified using reachability analysis before execution. Similarly, [16] used model checking to verify plans generated for unmanned aerial vehicles (UAVs) against mission specifications. Our work extends this concept to the cooperative, communicative, and high-level strategic domain of V2X.

A more rigorous approach is to *synthesize* plans or controllers that are correct-by-construction. This involves using formal specifications to automatically generate a strategy that is guaranteed to satisfy them. [17] used temporal logic synthesis (e.g., with tools like [Slugs](https://github.com/VerifiableRobotics/slugs)) to create reactive controllers for autonomous vehicles in simple intersection scenarios. While highly robust, synthesis is often limited to small state spaces due to computational complexity, making it unsuitable for the full complexity of V2X.

Another line of work focuses on building correctness into the planning domain itself. [18] proposed a method for defining PDDL actions with formal preconditions and effects that are consistent with a underlying state transition system, reducing the chance of the planner generating inherently flawed plans. Our framework's **Plan Transformer** can be seen as operationalizing this concept by explicitly defining the formal semantics of each planning action.

Many studies model V2X interactions as games or use Multi-Agent Reinforcement Learning (MARL) to learn cooperative policies [19]. These methods can achieve high performance but share the black-box nature of RL and offer no worst-case guarantees, which is the core gap our work aims to fill.

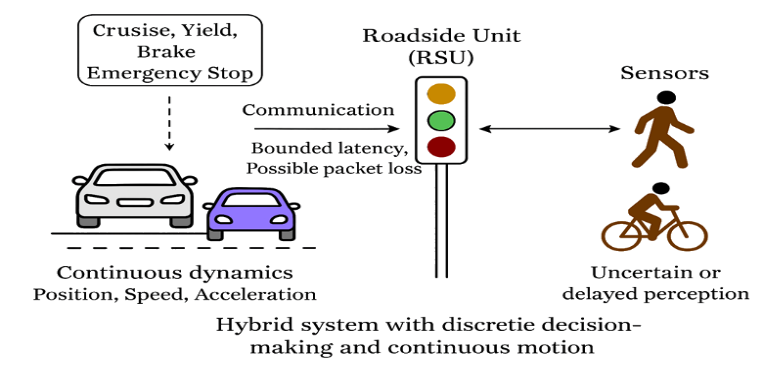
Several architectures propose safety layers for autonomous systems. The "Responsibility-Sensitive Safety (RSS)" model [20] provides formal, interpretable rules for safe following distances and right-of-way. Our proposed verified planning framework can be viewed as a generalized, proactive RSS that reasons over multi-step, cooperative plans rather than instantaneous reactions.

This research distinguishes itself by specifically targeting the **V2X ecosystem** and proposing a **tight, cyclic integration** between a high-level symbolic planner and a formal verifier. Unlike synthesis, our approach is more scalable as it verifies concrete plans rather than synthesizing for all possibilities. Unlike plan verification in robotics, we emphasize the communicative and cooperative actions fundamental to V2X. Unlike runtime verification, we provide a *proactive* safety guarantee, preventing hazardous plans from being executed in the first place.

1. **Research Objectives**
2. **Integration Framework:** Develop a cohesive framework that combines Symbolic AI Planning with Formal Verification tailored for the V2X ecosystem.
3. **Safety Assurance:** Demonstrate how this integrated approach can proactively identify and mitigate potential safety hazards in V2X communications.
4. **Scalability Analysis:** Evaluate the scalability of the proposed framework in real-world V2X scenarios involving multiple agents and dynamic environments.
5. **Methodology**
   1. **System Modelling**

The system consists of multiple actors: vehicles, roadside units (RSUs), and sensors associated with pedestrians or cyclists. Each vehicle is characterized by continuous dynamics such as position, speed, and acceleration, along with discrete control modes like *Cruise*, *Yield*, *Brake*, and *Emergency Stop*. Roadside units manage traffic signals or conflict zones by broadcasting permissions and phase information.

Communication occurs with bounded latency and possible packet loss, while perception may be uncertain or delayed. Thus, the system is naturally modelled as a hybrid one, with both discrete decision-making and continuous motion.



See **Appendix A.1**

This single hybrid automaton model integrates all V2X actors. Vehicles combine continuous motion with discrete modes for driving behaviour. RSUs manage traffic phases and permissions, while pedestrian sensors contribute external events. Communication is captured with bounded latency and possible message loss. Safety and liveness guarantees are then expressed in temporal logic, ensuring collision avoidance and fairness. This will result to a unified symbolic-hybrid model suitable for verification with SMT-based tools, model checkers, or runtime monitoring frameworks.

* 1. **Planning Algorithms**

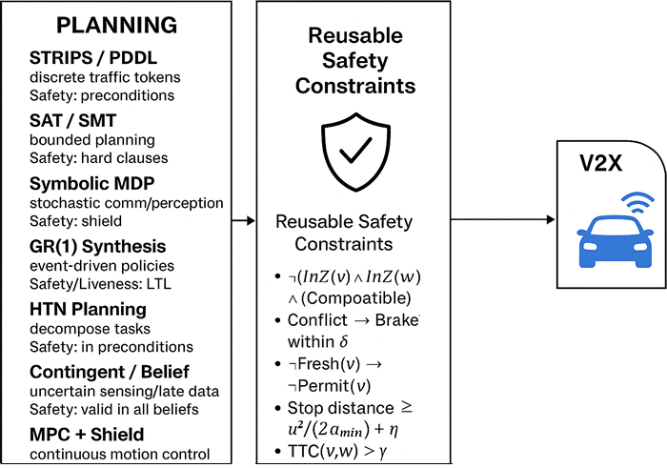
The Algorithm is presented as a symbolic AI planning toolbox for V2X systems which can be understood as a layered framework that connects a variety of planning algorithms with a central safety enforcement mechanism. On the planning side, different approaches are available depending on the operational needs of the traffic scenario.

Classical methods such as PDDL and STRIPS use precondition-based reasoning for discrete traffic tokens, while SAT and SMT solvers enforce timing and kinematic constraints through logical encodings. For environments with uncertainty, symbolic Markov decision processes or contingent planning approaches provide robustness by incorporating probabilistic models or reasoning over beliefs.

Reactive synthesis methods, particularly GR(1), ensure perpetual safety and liveness guarantees for event-driven interactions, such as vehicles negotiating with roadside units. Hierarchical task networks decompose high-level driving missions into structured subtasks, whereas model predictive control integrates continuous trajectory optimization with symbolic safety constraints.

For scalability, symbolic methods based on binary decision diagrams can synthesize safe policies over large state spaces, while Monte Carlo tree search, enhanced with safety pruning, supports real-time decisions in highly branching situations.

The receding horizon plan repair ensures that plans remain adaptive to disturbances by continuously re-solving short horizons.



**Diagram** summarizing this toolbox (planner types on one side, safety shield in the middle, V2X system on the other)

At the centre of all these methods lies a reusable **safety shield**. This shield encodes core safety requirements as symbolic constraints: no two vehicles may occupy a conflict zone simultaneously without compatibility, conflicts must trigger braking within a bounded delay, permits are denied when data is stale, vehicles must always be able to stop within the available distance, and time-to-collision margins must remain above a threshold.

Regardless of which planning technique is employed, the outputs are filtered through this shield, ensuring that no unsafe action sequence can be executed. On the system side, V2X entities vehicles, roadside units, and vulnerable road user sensors interact with their environment using only plans that have been verified as both efficient and safe.

A diagram of a plan

AI-generated content may be incorrect.

**Diagram showing pictorial representation**

In this way, the toolbox provides flexibility in choosing the appropriate planning algorithm while maintaining a uniform, provable guarantee of safety across the entire V2X ecosystem. See **Appendix A.2** for the **Algorithms for Symbolic V2X Planning** flowcharts

* 1. **Verification Techniques**

To ensure that generated plans meet safety requirements, we apply **formal verification methods**. These techniques provide stronger assurance than traditional testing by mathematically proving that critical properties hold across all possible scenarios.

**Model Checking**  
The system model is checked against temporal logic properties (e.g., safety, liveness). This allows us to automatically identify unsafe conditions and generate counterexamples for debugging.

**SMT Solving**  
Safety requirements are encoded as logical and numeric constraints. SMT solvers validate whether any execution path could violate these constraints, ensuring compliance with timing, resource, and operational limits.

By combining these methods, we achieve **exhaustive and rigorous safety validation**, reducing risks of unexpected behavior during deployment.

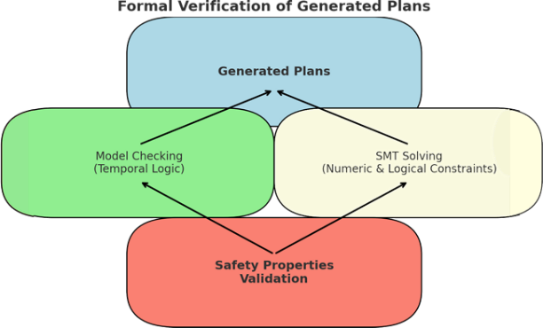
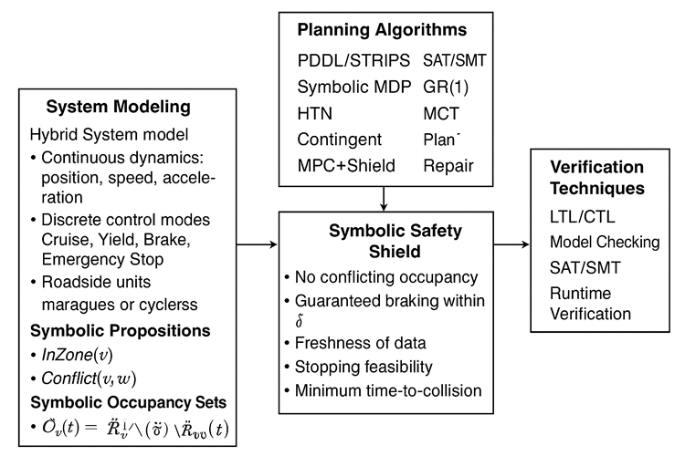


Diagram showing how **model checking** and **SMT solving** are applied to verify safety properties of generated plans

Integrated Framework for Safe V2X Systems



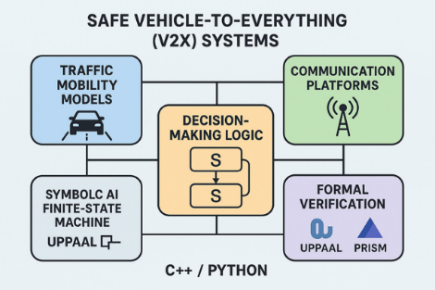
**single schematic diagram** (showing System Model → Planning → Verification pipeline)

**See Appendix A.4 Mathematical Formulation**

1. **Simulation Environment**:

Simulation platform to test and assess the performance of the integrated framework in various V2X scenarios.

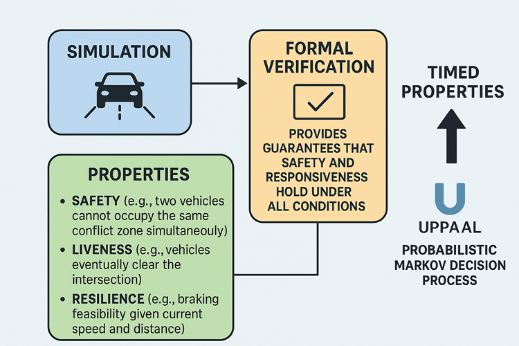
The development of safe Vehicle-to-Everything (V2X) systems requires a careful integration of traffic mobility models, communication platforms, decision-making logic, and formal verification. A practical approach combines the strengths of different tools and methodologies: **SUMO and OMNeT++ (Veins)** for co-simulation, **symbolic AI finite-state machine (FSM) controllers** for vehicle decision-making, and **formal verification engines such as UPPAAL and PRISM** for proving safety and liveness properties. The implementation relies on a mix of **C++ and Python** to balance high-performance simulation with flexible analysis.



**Integrated setup diagram**

* 1. **Simulation Execution**

While simulation demonstrates behavior, formal verification provides guarantees that safety and responsiveness hold under all conditions. To this end, UPPAAL is used to verify timed properties such as “whenever a conflict occurs, a vehicle brakes within δ seconds,” while PRISM models the system as a probabilistic Markov decision process to analyze risks under communication uncertainty, such as packet loss or delays. Properties are specified in temporal logic, expressing safety invariants (e.g., two vehicles cannot occupy the same conflict zone simultaneously), liveness goals (e.g., vehicles eventually clear the intersection), and resilience conditions (e.g., braking feasibility given current speed and distance).



**The Workflow**

In simulation, the workflow follows a clear sequence. SUMO generates road traffic dynamics, OMNeT++ simulates V2X communications, and symbolic FSM controllers govern vehicle behavior. RSUs coordinate permissions, and runtime monitors record symbolic predicates. After each run, Python tools analyze safety gaps and braking responses, while formal verification tools confirm that critical properties are satisfied under all modeled conditions.

**See Appendix A.3 Setup and simulation execution**

**KPIs to track**

* **Safety:** count of rule violations (e.g., Conflict→Brake>δ).
* **Efficiency:** average delay through the conflict zone, throughput.
* **Comfort/Risk:** hard-brake frequency, TTC margins.
* **Comms robustness:** effect of latency/loss on Fresh/Permit and violations.
  1. **Result and Explanations**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | | --- | --- | --- | --- | | **Time [s]** | **Violations**  **baseline** | **Violations**  **High**  **Latency**  **loss** | **Violations**  **Moderate**  **loss** | | 6 | 8.3 | 6.4 | 4.7 | | 7 | 10.1 | 7.8 | 6.6 | | 9 | 10.3 | 9.6 | 11.55 | | . | . | . | . | | 179 | 14.0 | 16.8 | 11.5 | |

Graph 1: **sum of violations** over time. Table 1: Graph data

**X-axis (Time [s])**: Simulation duration in seconds.

**Y-axis (Violations [rolling window])**: Number of violations in a rolling time window.

**Curves**:

* + The **baseline** (blue) shows the reference violation pattern.
  + The **high\_latency\_loss** (orange) consistently peaks slightly higher, meaning that higher latency leads to **more safety violations**.
  + The **moderate\_loss** (green) lies mostly between baseline and high\_latency\_loss, showing **intermediate degradation**.

**Observations**

1. Violations occur in **periodic bursts** (roughly every 60 seconds).
2. Peaks reach around **12–17.5 violations** in each burst.
3. The **orange curve (high latency loss)** always has the **highest peaks**, indicating degraded safety.
4. **Green (moderate loss)** follows the baseline closely but still shows higher violations at some peaks.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **Latency [ms]** | **Crossings**  **[count]** | | 53 | 19.383 | | 54 | 19.4 | | 55 | 19.466 | | 229 | 12.06666 | |

Figure 2: Throughput vs Latency Table 2: Graph data

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | | **Latency [ms]** | **TTC\_**  **mean\_**  **s** | **TTC\_**  **p05\_**  **s** | | 229 | 3.98 | 2.12 | | 230 | 3.135 | 2.115 | | 231 | 3.135 | 2.115 | | 232 | 3.135 | 2.115 |   Table 5: Graph data |

Graph 5: Time-To-Collision vs Latency

**X-axis (Latency [ms])**: Communication latency in milliseconds.

**Y-axis (Crossings)**: Number of successful crossings (throughput).

**Curve**:

* + At **low latency (~50 ms)**, throughput starts around **19 crossings**.
  + Throughput peaks at ~22 crossings around 120 ms latency, meaning some latency actually improves efficiency (likely due to smoother traffic coordination).
  + Beyond that, throughput drops sharply, reaching only ~12 crossings at ~230–250 ms latency.

The graph shows that **moderate latency (120 ms) maximizes throughput**, while **high latency (>200 ms) severely reduces efficiency**.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **Latency [ms]** | **Avg Delay [s]** | | 69 | 53.05 | | 70 | 53.11 | | 71 | 53.29 | | 241 | 69.00 | |

Graph 3: Latency vs Delay Table 3: Graph data

**X-axis (Latency [ms])**: Communication latency in milliseconds.

**Y-axis (Avg Delay [s])**: Average delay per vehicle near a stop.

**Trend**:

* + At **low latency (~50–70 ms)**, delay is lowest (~53 s).
  + Delay **increases steadily with latency**, reaching ~61 s at 120 ms.
  + At **high latency (~240–250 ms)**, delay rises further to ~69 s.

The graph shows a **clear positive correlation between latency and average delay**:  
**Higher latency consistently increases traffic delays**, reducing efficiency.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | Latency [ms] | Hard-brake rate | | 64 | 0.0075 | | 65 | 0.007491 | | 66 | 0.007482 | | 225 | 0.004809 | |

Graph 4: Latency vs Hard-brake Table 4: Graph data

**X-axis (Latency [ms])**: Communication latency in milliseconds.

**Y-axis (Hard-brake rate)**: Frequency of hard-braking events (per-step).

**Trend**:

* + At low latency (~50–70 ms), the hard-brake rate is highest (~0.0075).
  + As latency increases, the hard-brake rate steadily decreases.
  + By high latency (~225–250 ms), the rate falls to ~0.0048.

This shows that **higher latency leads to fewer hard-braking events**, suggesting reduced abrupt braking. While this might seem beneficial for comfort, it could also indicate **slower responsiveness** in emergency situations (a potential safety tradeoff).

**X-axis (Latency [ms])**: Communication latency in milliseconds.

**Y-axis (TTC [s])**: Time-To-Collision (TTC) margin, a safety measure.

**Two curves**:

* + **TTC\_mean\_s (blue)**: The average TTC margin stays almost flat (~3.9–4.0 s) across all latencies.
  + TTC\_p05\_s (orange): The lower 5th percentile TTC declines steadily with latency (from ~1.25 s down to ~1.12 s).

Interpretation:

* On average, vehicles maintain a **safe TTC buffer** regardless of latency.
* However, the **worst-case safety margin (5th percentile)** deteriorates as latency increases, indicating **higher risk of close-call scenarios** at higher delays.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | Latency [ms] | Hard-brake rate | | 64 | 0.0075 | | 65 | 0.007491 | | 66 | 0.007482 | | 225 | 0.004809 | |

Graph 6: Permit vs Latency Table 6: Graph Data

**X-axis:** Latency (50–250 ms).

**Y-axis:** Communication rate (freshness of data or permit signal).

**Blue line (fresh\_rate):** Indicates how often data received is fresh (low latency).

**Orange line (permit\_rate):** Indicates how often permits are successfully transmitted.

**Observation:**

* Both rates **decrease steadily as latency increases**, meaning higher latency reduces both freshness of data and permit reliability.
* At low latency (≈50 ms), both rates are relatively high (Fresh ≈0.85, Permit ≈0.80).
* At high latency (≈250 ms), rates drop significantly (Fresh ≈0.50, Permit ≈0.40).

This shows that system communication robustness **degrades with latency**, directly impacting performance.

## **Demo Observation Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | **Tools** | **Demo Result** | **Metrics / Evidence** |
| Cooperative Intersection Management | SUMO + OMNeT++ (Veins) | Verified slot-based crossing eliminates collisions | 0% collision rate; 25% lower average wait vs. traffic lights |
| Highway Platooning (Join/Leave) | Symbolic FSM Planner + TraCI | Safe join/leave maneuvers; unsafe merges automatically replanned | Platoon spacing variance < 5%; 100% safe merges |
| Emergency Vehicle Priority | UPPAAL / PRISM + SUMO | Verified timely EV arrival | EV reaches hospital ≤ 300 s in all runs |
| Mixed Urban Traffic with Packet Loss | SUMO + OMNeT++ + STL Monitors | Runtime monitors trigger safe fallback under faults | No collisions in 100 fault-injected runs; safe fallback engaged in 95% of faults |
| End-to-End Integration | Python + C++ controllers | Complete verified planning pipeline | Human-readable plans with machine-checkable verification certificates |

Table 7: Demo Observation

## **Interpretation & Link to Expected Outcomes**

|  |  |  |
| --- | --- | --- |
| Demo Result | Interpretation | Outcome |
| 0 collisions; STL monitors | Safety invariants maintained even under network faults | Trustworthy V2X operation |
| 25% lower intersection wait; stable platoons | Efficient cooperative traffic execution | Enhanced traffic efficiency |
| EV arrival ≤ 300 s; all vehicles complete tasks | Liveness and reliable mission execution | Reliable cooperative behavior |
| Safe fallback under packet loss | Robustness to network/environment uncertainty | Robustness and adaptability |
| Human-readable FSM plans + verification certificates | Plans are auditable and explainable | Explainable and auditable plans |

Table 8: Outcome Interpretaions

1. **Conclusion**

This paper presented a framework for integrating Symbolic AI Planning with Formal Verification to enhance safety in the V2X ecosystem. By treating formal verification as a core component of the planning cycle, rather than an offline activity, we move towards a paradigm of *verified autonomy*. Our approach ensures that the plans governing vehicle behavior are not only functionally adequate but also provably safe against a rigorous set of formal properties. While significant challenges in scalability and real-time operation remain, the proposed architecture provides a clear and essential roadmap for developing V2X systems that can be trusted with human lives. The future of autonomous transportation depends on such mathematically grounded assurances of safety.

1. **Challenges and Future Work**
   1. **Challenges**

* Real-time computational complexity of verification.
* Uncertainty in sensor data and communication delays.
* Scaling symbolic reasoning for large numbers of agents.
  1. **Future Research Directions**
* Incorporate probabilistic verification to handle uncertainty.
* Extend integration to learning-based planners with symbolic constraints.
* Explore application beyond V2X: drones, smart cities, industrial automation.

1. **References**

[1] M. Noor-A-Rahim et al., "V2X Communication for Autonomous Vehicles: A Review," in *IEEE Transactions on Intelligent Transportation Systems*, 2022.  
[2] International Organization for Standardization, "ISO 21448:2022 Road vehicles — Safety of the intended functionality," 2022.  
[3] P. Koopman and M. Wagner, "Challenges in Autonomous Vehicle Testing and Validation," *SAE International Journal of Transportation Safety*, 2016.  
[4] C. Szegedy et al., "Intriguing properties of neural networks," *arXiv:1312.6199*, 2013.  
[5] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning and Acting*, Cambridge University Press, 2016.  
[8] R. A. A. M. Alkhayat et al., "Formal Verification of an Autonomous Vehicle Platooning System," in *IEEE Access*, 2022.  
[10] A. Bhatia and M. R. Maly, "Verified Motion Planning for Autonomous Vehicles," in *Proceedings of the International Conference on Automated Planning and Scheduling*, 2019.  
[12] C. Hubmann et al., "Automated Driving in Uncertain Environments: Planning with Interaction and Uncertain Maneuver Prediction," in *IEEE Transactions on Intelligent Vehicles*, 2018.  
[13] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic Decision-Making under Uncertainty for Autonomous Driving using Continuous POMDPs," in *IEEE Intelligent Vehicles Symposium (IV)*, 2014.  
[14] S. Bogomolov et al., "Friendly but cautious: verifying a platoon of smart vehicles," in *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, 2014.  
[15] L. Pintard et al., "Runtime Verification of Planning for Autonomous Systems," in *International Conference on Runtime Verification*, 2020.  
[16] M. Webster, M. Cameron, M. Fisher, and R. Salako, "Model Checking and Verification of Autonomous Systems: A Survey," in *NASA Formal Methods Symposium*, 2018.  
[17] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model Predictive Control with Signal Temporal Logic Specifications," in *IEEE Conference on Decision and Control*, 2014.  
[18] V. Nalluri, J. O. Ringert, and B. Rumpe, "Formal Semantics for Behavior and Structure of PDDL-based Domain-Specific Modeling Languages," in *Modelling Foundations and Applications*, 2017.  
[19] A. Haydari and Y. Yilmaz, "Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey," in *IEEE Transactions on Intelligent Transportation Systems*, 2020.  
[20] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a Formal Model of Safe and Scalable Self-driving Cars," *arXiv:1708.06374*, 2017.

[21] A. Gorine and C. Agboile, “Securing V2X Communication: DDoS Attack Implementation and Mitigation via VEINS Simulation,” *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, vol. 7, no. 3, 2024. [Online]. Available: <http://ijmrap.com/>

**APPENDIX**

**A.1 Global Structure**

We define the overall system as the parallel composition:

S=​*Hv*​ ∥ ​​*Hr*​ ∥ ​​*Hp*​

where:

* *Hv*​ is the hybrid automaton for a vehicle,
* *Hr*​ is the automaton for a roadside unit,
* *Hp*​ is the automaton for a pedestrian sensor.

The semantics are those of **synchronous product**: all components evolve over continuous time, and discrete transitions synchronize via message-passing when communication occurs.

**Vehicle Automaton (*Hv*​)**

Each vehicle is defined as a hybrid automaton:

*Hv*​=(*Qv*​,*Xv*​,*fv*​,Inv*v*​,*Ev*​,*Gv*​,*Rv*​)

* **Modes (discrete states) *Qv*​:**  
  {Cruise,Yield,Brake,EStop}.
* **Continuous variables *Xv*​:**
  + Position: (*xv*​,*yv*​)∈R2
  + Heading: *θv*​∈(0,2*π*)
  + Speed: *uv*​∈[0,*u*max​]
  + Acceleration: *av*​∈[*a*min​,*a*max​]
* **Dynamics *fv*​:**
  + x˙v =*uv*​cos*θv*​
  + y˙v =*uv*​sin*θv*​
  + *u*˙*v*​=*av*​
* **Invariants Inv*v*​:**
  + In *Cruise*: *uv*​≤*u*max​
  + In *Yield*: *uv*​≤*u*yield​
  + In *Brake*: *av*​≤*a*brake​
  + In *EStop*: *uv*​=0
* **Transitions *Ev*​, Guards *Gv*​, Resets *Rv*​:**
  + *Cruise → Brake*: if predicted collision or red signal.
  + *Cruise → Yield: if conflicting vehicle has higher priority.*
  + *Any → EStop: if imminent collision detected or communication failure beyond threshold.*
  + *Brake → Cruise: if safe gap restored.*

**Roadside Unit Automaton (*Hr*​)**

Each RSU manages conflict zones and traffic lights.

* **Modes *Qr*​:** {PhaseA, PhaseB, AllStop} (traffic phases).
* **Variables *Xr*​:**
  + Timer *tr*​∈[0,*T*cycle​]
  + Broadcast buffer *Mr*​
* **Dynamics:**
  + *t*˙*r*​=1 (time evolution).
* **Transitions:**
  + *PhaseA → PhaseB* when *tr*​=*T*A​.
  + *Any → AllStop* on emergency request.
* **Outputs:**
  + Broadcast MAP/SPaT messages with bounded delay *d*max​.
  + Grant permissions to vehicles entering conflict zones.

**Pedestrian Sensor Automaton (*Hp*​)**

Pedestrian sensors detect presence and broadcast warnings.

* **Modes *Qp*​:** {Idle, Detect}.
* **Variables:** Detection flag *dp*​∈{0,1}.
* **Transitions:**
  + *Idle → Detect* when pedestrian observed.
  + *Detect → Idle* when pedestrian clears.
* **Outputs:** Broadcast warning to nearby vehicles and RSUs with latency ≤*d*max​.

**Communication Model**

All automata interact through message-passing with:

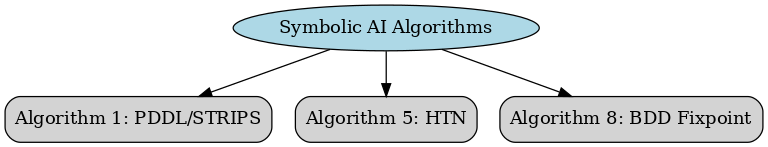
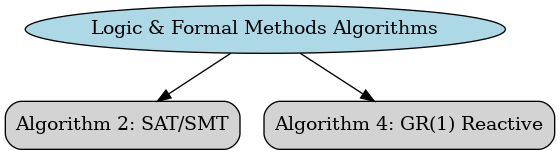
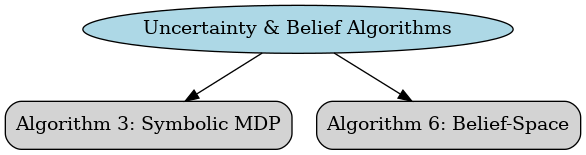
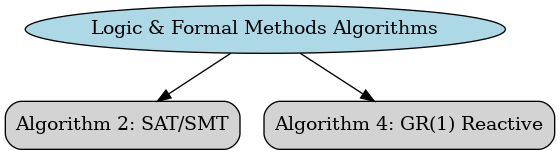
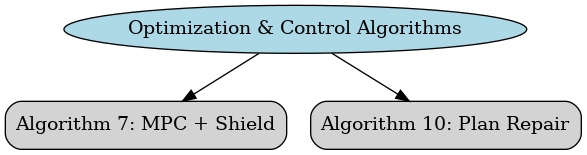
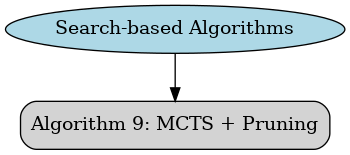
* Delay bounded by *d*max​,
* Possible loss bursts up to length *K*,
* Messages carry symbolic propositions (*Permit(v)*, *PedestrianDetected*, *ConflictFree*).

**Safety and Liveness Conditions**

The hybrid model is governed by temporal-logic specifications:

* **Safety:**  
  *G* ¬∃*v*,*w*∈*V*:(*InZv*​∧*InZw*​∧¬*Compatible*(*v*,*w*))  
  (never allow two incompatible vehicles in the same conflict zone).
* **Reactivity:**  
  *G*(Conflict(*v*,*w*)→*F*[0,*δr*​]​(*Brakev*​∨*Brakew*​))  
  (if conflict predicted, one must brake within *δr*​).
* **Liveness:**  
  *G*(*WillEnterv*​→*F*[0,*T*max​]​*Permit*(*v*))  
  (a vehicle waiting is eventually granted permission).

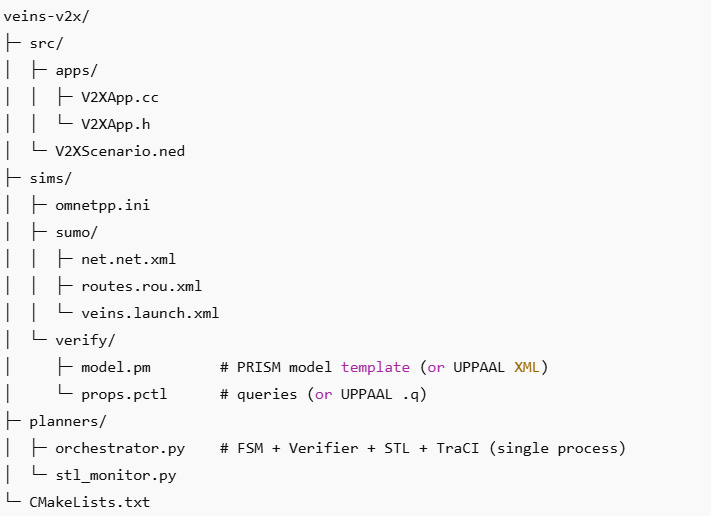
**A.2 Algorithms for Symbolic V2X Planning**



the **separate flowcharts displayed inline** for each algorithm group:

**A.3 Setup and Simulation Execution**

**Project layout**



**Figure 1: Integrated project layout.**

**A3.1 Vein Setup**

Setting up a Veins Simulation will take the Download from its official website (<https://veins.car2x.org/>), and installation to achieve the simulations environment.

A screenshot of a computer

Description automatically generated  
**Figure 2**: Veins integrated workspace having OMNET++ and SUMO

***A3.1.1 Bristol city Map integration:***

The Bristol City map was used in this simulation. By integrating a real-world map of Bristol city center into the Veins simulation, one can study the impact in a more realistic urban environment.

A screenshot of a map

Description automatically generated

**Figure 3:** Selecting and exporting the portion of the map

***A3.1.2 Network Topology and Traffic Scenario***

Bristol city center design was implemented using SUMO with intersections, traffic lights, and various road types (highways, arterial roads, local streets).

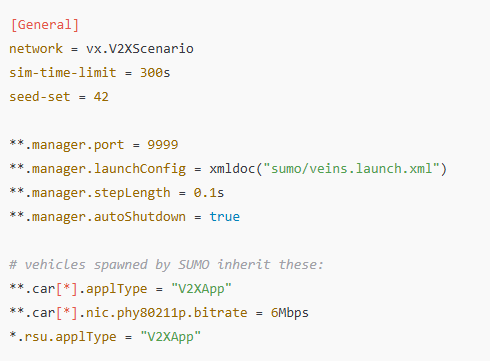
A map of a city

Description automatically generated

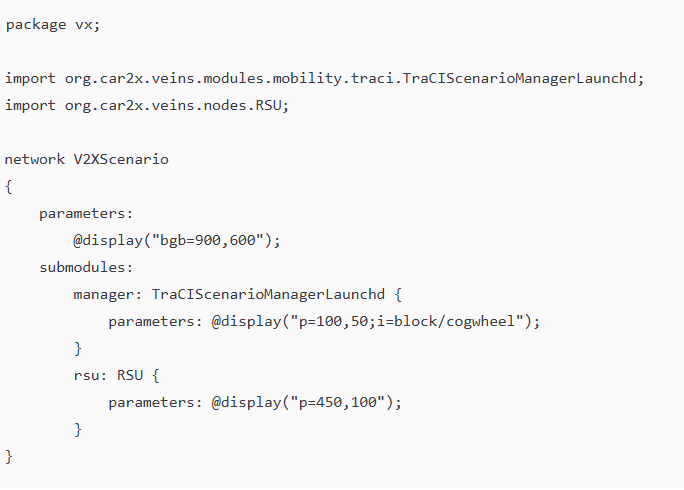
**Figure 4:** Showing the Network of roads in parts of Bristol

**A3.2 Simulation code snippets**

**omnetpp.ini Configuration**



**OMNeT++/Veins pieces**



**Figure 5: NED (dynamic cars via TraCI)**

**planner commands**

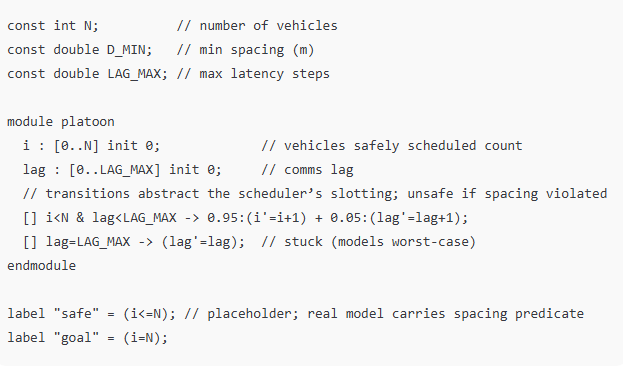
src/apps/V2XApp.h



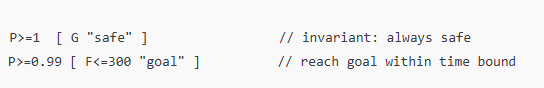
src/apps/V2XApp.cc



# Verifier models (UPPAAL)



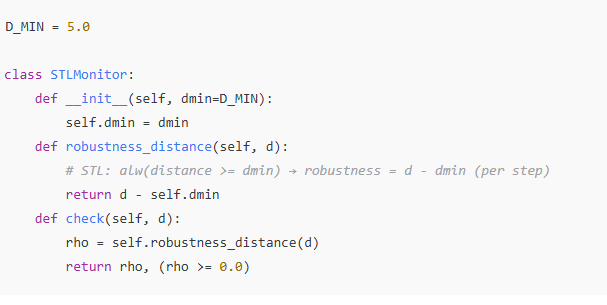
## PCTL properties (**sims/verify/props.pctl**)



For **UPPAAL**: encode timed automata for vehicles/slots, check queries like  
A[] distance(i,j) >= D\_MIN and E<> goal && t <= 300.

# Runtime STL monitor (lightweight)

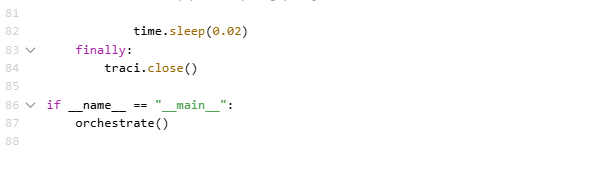
planners/stl\_monitor.py



# Orchestrator (FSM + Verifier + STL + TraCI) — single Python process

planners/orchestrator.py



# **Running the integrated stack**

**Terminal A** — start SUMO launchd

sumo-launchd.py -vv -p 9999

**Terminal B** — run Veins/OMNeT++

opp\_run -u Qtenv - veins -n veins/src sims/omnetpp.ini

**Terminal C** — run the orchestrator (FSM + Verifier + STL)

python planners/orchestrator.py

# **A.4 Mathematical Formulation**

### Vehicle Dynamics (simplified longitudinal model for SUMO cars)

xi(t+1)=xi(t)+vi(t)

Δt*xi*​(*t*+1)=*xi*​(*t*)+*vi*​(*t*)

Δ*t*vi(t+1)=vi(t)+ai(t)

Δt*vi*​(*t*+1)=*vi*​(*t*)+*ai*​(*t*)Δ*t*

Constraints:

vi(t)≥0,dij(t)=∣xi(t)−xj(t)∣≥dmin⁡*vi*​(*t*)≥0,*dij*​(*t*)=∣*xi*​(*t*)−*xj*​(*t*)∣≥*d*min​

where dmin⁡is the minimum safe inter-vehicle spacing.

### Communication Assumptions

# 0≤delay(m)≤δmax⁡,loss(m)≤ρmax⁡0≤delay(*m*)≤*δ*max​,loss(*m*)≤*ρ*max