Automatically Proving the Correctness of Vehicle Coordination

Mikael Asplund

Department of Computer and Information Science,
Linköping University, Sweden

Abstract

In the next generation of road-based transportation systems, where vehicles exchange information and coordinate their actions, a major challenge will be to ensure that the interaction rules are safe and lead to progress. In this paper we address the problem of automatically verifying the correctness of such distributed vehicular coordination protocols. We propose a novel modeling approach for communicating mobile entities based on the concept of satisfiability modulo theories (SMT). We apply this method to an intersection collision avoidance protocol and show how the method can be used to investigate the settings under which such a protocol achieves safety and progress.

Keywords: formal verification, vehicular coordination, SMT, intersection collision avoidance

1. Introduction

Advanced driver assistance systems (ADAS) are becoming increasingly sophisticated and connected. Emerging applications include vehicle platoons, collision avoidance, and emergency vehicle awareness. Despite the increasing interactions between vehicles, the industry currently lacks methods to ensure safety and correctness for collaborative vehicle systems. For example the current ISO 26262 functional safety standard is only concerned with in-vehicle functions.

In this paper, we present a novel approach to the formal modeling and automatic verification of vehicular coordination, including models of the environment and unreliable wireless communication. We propose using the concept of satisfiability modulo theories (SMT) which allows complex domain-specific models to be expressed while also supporting automatic verification of correctness properties. We discuss different modeling choices regarding the expressitivity/tractability tradeoff, and present a system model that we demonstrate to achieve a useful balance.

In previous work [1], we formalized a coordination protocol [10] designed to achieve intersection collision avoidance (ICA) using inter-vehicle communication. In that work, we created a very basic model of the environment, which only encompassed a single intersection. We used this to prove the safety of the algorithm (i.e., guaranteeing that it did not end up in a bad state). We now present a general modeling framework for describing *sets* of roads, intersections, vehicles, and shared resources. We expand the case study from our previous work to account for this new more general environment model. Moreover, we show the parameter conditions under which the case study achieves both logical safety and progress (as opposed to just logical safety in our previous work).

The rest of this paper is organized as follows. Section 2 presents our system model and our approach for formalizing vehicle coordination. Section 3 contains a validation of our approach. Finally, Section 4 describes related work, and Section 5 concludes the paper.

2. System Model

Our approach to formalize the vehicle coordination problem is to model the system as a set of time-dependent constraints in combination with a traditional hybrid automaton describing a specific subject vehicle. In this section we describe our basic modeling framework including how the physical environment and communication capabilities are represented.

The goal of our modeling phase is to provide a set of basic building blocks with which a sufficiently detailed representation of a system with collaborating vehicles can be constructed. Because we will use this model to formally prove the correctness of a coordination approach it is important to balance the need of ex-

Email address: mikael.asplund@liu.se (Mikael Asplund)

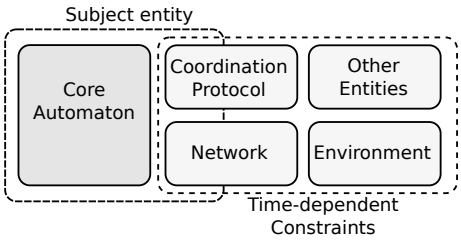

Figure 1: System model overview

pressitivity (which allows realistic representations) with tractability (i.e., to keep the model abstract enough for automated proofs).

Figure 1 shows an overview of how our system model is constructed. It is composed of a "core" automaton, which encodes the behavior of a vehicle under different circumstances and a set of time-dependent constraints, which capture the properties of the surroundings (including other vehicles). We model the system as a tuple $M = (\mathbf{E}, \mathbf{R}, \mathbf{X}, \mathbf{\Pi}, \mathbf{M}, \mathbf{S}, \mathbf{I}, T, \mathbf{F}, \mathbf{C})$ composed of a number of infinite and finite sets, and a mapping, as shown in Table 1.

Table 1: Components of the system model

Component	Description
$\mathbf{E}$	a set of entities (vehicles)
$\mathbf{R}$	a set of routes
$\mathbf{X}$	a set of intersections
$\mathbf{\Pi}$	a set of resources
$\mathbf{M}$	a set of messages
$\mathbf{S}$	a set of states
$\mathbf{I} \subset \mathbf{S}$	a set of initial states
$T : \mathbf{S} \times \mathbf{S} \rightarrow \mathbf{Bool}$	a transition function
$\mathbf{F}$	a finite set of predicates
$\mathbf{C}$	a finite set of constraints

Note that the sets $\mathbf{E}, \mathbf{R}, \mathbf{X}, \mathbf{\Pi}, \mathbf{M}, \mathbf{S}$, and $\mathbf{I}$ can all be infinite, thereby allowing us to model an unbounded number of cars, routes, intersections and communication messages. The set of predicates (or uninterpreted functions), $\mathbf{F}$, provides the semantics for the states of the core automaton. The allowed domains and ranges of the functions are the real numbers (time), integers, and any of the sets in our model. An example of an uninterpreted function that we use in our model is $t_{snd} : \mathbf{M} \rightarrow \mathbb{R}$, which denotes the sending time of a given message.

The constraints in $\mathbf{C}$ provide us with a way to describe the properties of the environment and other assumptions that we need to adopt. The constraints apply over the same domains as the uninterpreted functions, $\mathbf{F}$, and may also contain quantifiers. An example of a constraint (which we do not use) could be $\forall m \in \mathbf{M} : t_{snd}(m) \leq 10$, which would say that no message is sent after the time point 10.

We let the states in $\mathbf{S}$ and the transition function T denote the state and behavior of the specific subject entity. The behavior of other entities in the system is modeled using the constraints in $\mathbf{C}$. This allows us to provide a more detailed internal model of a single entity, and model other entities using assumptions regarding their observable behavior (including communication).

Finally, consider the transition function $T(i, j)$, where i and j are states, which is used to characterize the behavior of the subject entity. We encode the hybrid automaton as a transition function that alternates between timed and non-timed transitions which is a common procedure when modeling hybrid systems. The full model cannot be described here due to space restrictions, but can be shared with the research community.

3. Validation

We implemented our model using Z3Py, which provides a Python API to the Z3 v4.3.1 theorem prover. Essentially, we have a number of first-order predicate logic formulae which we express as python functions. These can be combined into a model M. The goal of the verification is to show that the model M logically entails a safe state for all reachable states $\mathbf{S_r} \subset \mathbf{S}$:

$$M \models \forall i \in \mathbf{S_r} : safe^i$$

where $safe^i$ is a safety predicate. The most basic definition of the safety predicate is to require no collisions between entities. We call this the *noCollision* predicate which states that if the subject entity is in an intersection, then no other car can be in the same physical resource. We employed limited k-induction [9] and safety invariants to ensure that the model could be tractably handled by the Z3 theorem prover [1].

In the remainder of this section we present a validation of our approach using three important aspects: consistency, crash-freedom, and progress. We use the term consistency to mean that the model is logically sound so that at least some basic behaviors are supported by the model. Crash-freedom means we can prove that if the vehicles adhere to the coordination protocol, then no crashes will occur due to faults in the protocol. Note that we cannot guarantee crash-freedom in the general case, because this depends on many other factors in a real-life traffic scenario. We simply prove that the coordination protocol works as intended. Finally, we prove that the protocol also guarantees progress in the sense

that vehicles must not wait forever to pass the intersection.

3.1. Scenario

We have applied our formal modeling approach to an intersection collision avoidance case study. The scenario is based on a four-way intersection where vehicles can arrive from all four directions. The vehicle under study approaches the intersection, and executes a coordnation protocol (CwoRIS) to agree with the other vehicles when it is safe to pass the intersection. Details of the core automaton and protocol formalization is published elsewhere [1], albeit for a simpler environment and communication model than we have used in this work.

The CwoRIS protocol ensures vehicle coordination through the use of resources that correspond to a physical area of the road. Every entity is responsible for not entering a resource without having made sure that it has exclusive access to that resource. The protocol uses a series of exchanged messages and local data structures to infer whether there are potential conflicts for a resource. The protocol also uses priorities to avoid circular deadlocks.

3.2. Consistency

The first step of the validation is to ensure that the model is sound in the sense that we have not made it overly restrictive. In particular, it should always be possible to transition to a new state in the automaton of the system. If the model is stuck, this means that there is an error in the representation of the real world.

We introduce a successor function $succ : \mathbf{S} \rightarrow \mathbf{S}$, where for each state this returns a new state to which there is a valid transition. The successor function can be derived from the definition of the transition function without major effort. Because *succ* is always guaranteed to provide an output for every input state, we can prove freedom from deadlock by proving the following formula:

$$M \models \forall i, j : T(i, j) \Rightarrow T(j, succ(j))$$

3.3. Crash-freedom

The protocol studied here can be shown to guarantee freedom from crashes under certain assumptions on the parameters. Figure 2 shows a matrix of parameter combinations for which the system can be proven to avoid collisions. The rows (y-axis) in the matrix represent the distance to the intersection center at which the vehicle must brake unless it has negotiated access to the intersection. The columns (x-axis) represents the maximum braking (deceleration) capacity of the vehicle.

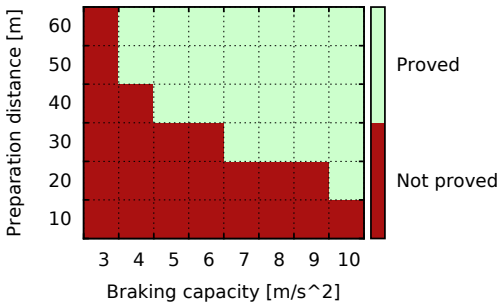

Figure 2: Matrix showing the combination of parameters (the distance from the center of the intersection at which to start preparing for an intersection, and the braking capacity of the vehicle) for which the system can be proven to avoid collisions.

The results are intuitive and expected. If the car has a higher braking capacity, then it can approach closer to the intersection before it has to brake. However, what is interesting in these results is that the light green areas represent values where we have mathematically and automatically *proved* that the system is correct, in the sense that collisions cannot occur (under the assumptions of the formal model).

3.4. Progress

It is easy to design a provably safe protocol for intersection collision avoidance by forcing some or all cars to stop indefinitely. However, such a scheme is obviously faulty, because it does not guarantee that the vehicles will reach their respective destinations. Therefore, it is important to be able to prove not only freedom from collisions, but also that there is a bound on the time that vehicles must wait "in limbo." To achieve this, we add the assumption that all messages are delivered (an assumption that is not required to prove safety). The protocol can only guarantee progress during periods when messages are not dropped.

Figure 3 shows a matrix with the cases for which it is possible to prove progress in the system. By progress we mean that an entity will not be stuck in a state where its request for access is infinitely delayed. The rows in the matrix (y-axis) represents the maximum time for which a vehicle can be forced to wait before deciding on whether its request to access the intersection is granted or not. The columns (x-axis) represent how long an acknowledgment message can be delayed for before being sent. The interpretation of the green areas is that if the acknowledgment delay is below the given value, then

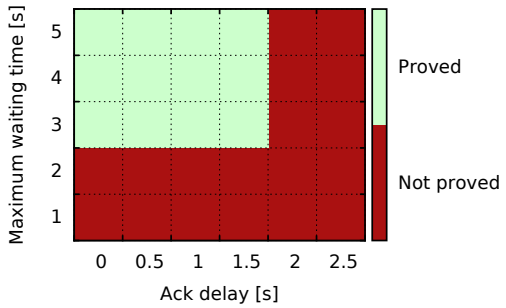

Figure 3: Matrix showing the combination of parameters (the maximum acceptable waiting time and the maximum time from receiving a message to sending an acknowledgment) for which the system can be proven to maintain progress.

the system can be proved to achieve progress, with waiting times no longer than the value shown on the x-axis.

The bottom line of this investigation is that for all the parameter combinations resulting in green cells in the figures, we have demonstrated that the coordination protocol is provably safe and without deadlock. Given the complexities involved with an unbounded number of participants, an explicit model of communication and continuous time and space, this is clearly a step forward in what has been previously described in the literature.

4. Related Work

Several studies such as that by Huang et al. [7], use SMT solvers to verify real-time communication protocols, but do not consider mobility and spatial safety constraints. The problem of how autonomous traffic agents should avoid collisions has also been treated formally with manual proof strategies. For example, Damm et al. [4] present a proof rule for the collision freedom of two vehicles. Other studies have also employed SMT to verify vehicular applications (e.g., [6, 5]). However, to our knowledge we are the first to show how a system with an unbounded number of participants, explicit modeling of communication, and continuous time and space can be automatically verified using a constraint solver.

Autonomous intersection management has been extensively explored in the intelligent transportation community [3], although usually not with a focus on proving correctness. The Comhordú coordination scheme, on which the coordination approach presented here is based, was formalized by Bhandal et al. [2] using a process algebraic approach. Vehicular platoons have been studied in several works on vehicular verification, including recently Kamali et al. [8], who use a combination of the Uppaal model checker and logic programming.

Our work differs from existing studies that treat the safety of coordinating vehicles, by explicitly modeling communication protocols and message passing as well as using an approach that allows mostly automatic rather than manual verification.

5. Discussion and Conclusion

In this paper, we have proposed a modeling approach for vehicular coordination algorithms. We showed that, with the help of SMT and carefully choosing appropriate model abstractions, it is possible to automatically prove properties of the system that would not otherwise be attainable. We have been able to verify the correctness of a fully distributed intersection collision avoidance protocol, in terms of both the logical safety and progress. The model is not without limitations. For example, we have not included lateral movements and control, and the longitudinal model is very basic. However, we have made significant progress over the current state-of-the-art by modeling and verifying an intelligent transportation system with an unbounded number of vehicles, explicitly modeled communication messages, and continuous time and space.

Acknowledgment

This work was supported by Centrum för industriell informationsteknologi (CENIIT), project 14.04.

[1] M. Asplund, A. Manzoor, M. Bouroche, S. Clarke, and V. Cahill. A formal approach to autonomous vehicle coordination. In D. Giannakopoulou and D. Mry, editors, *FM 2012: Formal Methods*, volume 7436 of *Lecture Notes in Computer Science*, pages 52–67. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-32759-9˙8.

[2] C. Bhandal, M. Bouroche, and A. Hughes. A process algebraic description of a temporal wireless network protocol. In *Proceedings of the Fourth International Workshop on Formal Methods for Interactive Systems*, 2011.

[3] L. Chen and C. Englund. Cooperative intersection management: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 17(2), 2016. doi: 10.1109/TITS.2015.2471812.

[4] W. Damm, H. Hungar, and E.-R. Olderog. Verification of cooperating traffic agents. *International Journal of Control*, 79(5), 2006. doi: 10.1080/00207170600587531.

[5] S. Gulwani and A. Tiwari. Constraint-based approach for analysis of hybrid systems. In A. Gupta and S. Malik, editors, *Computer Aided Verification*, volume 5123 of *Lecture Notes in Computer Science*, pages 190–203. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-70545-1_18.

[6] C. Herde, A. Eggers, M. Franzle, and T. Teige. Analysis of hybrid systems using hysat. In *Third International Conference on Systems (ICONS)*, 2008. doi: 10.1109/ICONS.2008.17.

[7] J. Huang, J. Blech, A. Raabe, C. Buckl, and A. Knoll. Static scheduling of a time-triggered network-on-chip based on SMT solving. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 509 – 514, 2012.

[8] M. Kamali, L. A. Dennis, O. McAree, M. Fisher, and S. M. Veres. Formal verification of autonomous vehicle platooning. *Science of Computer Programming*, 2017. doi: http://dx.doi.org/10.1016/j.scico.2017.05.006.

[9] M. Sheeran, S. Singh, and G. Stålmarck. Checking safety properties using induction and a sat-solver. In W. Hunt and S. Johnson, editors, *Formal Methods in Computer-Aided Design*, volume 1954 of *Lecture Notes in Computer Science*, pages 127–144. Springer Berlin / Heidelberg, 2000. doi: 10.1007/3-540-40922-X˙8.

[10] M. L. Sin, M. Bouroche, and V. Cahill. Scheduling of dynamic participants in real-time distributed systems. In *30th IEEE Symposium on Reliable Distributed Systems, SRDS*, 2011. doi: 10.1109/SRDS.2011.37.