KICS The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express ▮ (▮▮▮▮) ▮▮▮–▮▮▮

www.elsevier.com/locate/icte

Automatically proving the correctness of vehicle coordination

Mikael Asplund

Department of Computer and Information Science, Linköping University, Sweden

Received 1 September 2017; accepted 3 January 2018
Available online xxxx

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 expressivity/tractability trade-off, and present a system model that we demonstrate to achieve a useful balance.

In previous work [1], we formalized a coordination protocol [2] 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.

E-mail address: mikael.asplund@liu.se.
Peer review under responsibility of The Korean Institute of Communications Information Sciences.

https://doi.org/10.1016/j.icte.2018.01.013

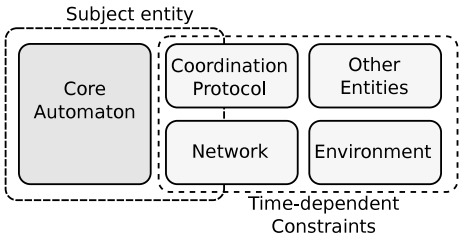

Fig. 1. System model overview.

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

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

Fig. 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.

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 formulas 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 [3] 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 coordination protocol (CwoRIS) to agree with the other vehicles when it is safe to pass the intersection. Details of the core automaton and protocol formalization are published elsewhere [1], albeit for a simpler environment and communication model than what 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

Please cite this article in press as: M. Asplund, Automatically proving the correctness of vehicle coordination, ICT Express (2018), https://doi.org/10.1016/j.icte.2018.01.013.