Compatibility Checking for Autonomous Lane-Changing Assistance Systems

Po-Yu Huang[1], Kai-Wei Liu[1], Zong-Lun Li[1], Sanggu Park[2], Edward Andert[2], Chung-Wei Lin[1], Aviral Shrivastava[2]
[1]National Taiwan University, Taipei 106319, Taiwan, [2]Arizona State University, Tempe, AZ 85281
Emails: r10922091@ntu.edu.tw, b06902042@ntu.edu.tw, cwlin@csie.ntu.edu.tw

***Abstract*—Different types of lane-changing assistance systems are usually developed separately by different automotive makers or suppliers. A lane-changing model can meet its own requirements, but it may be incompatible with another lane-changing model. In this paper, we verify if two lane-changing models are compatible so that the two corresponding vehicles on different lanes can exchange their lanes successfully. We propose a methodology and an algorithm to perform the verification on the combinations of four lane-changing models. Experimental results demonstrate the compatibility (or incompatibility) between the models. The verification results can be utilized during runtime to prevent incompatible vehicles from entering a lane-changing road segment. To the best of our knowledge, this is the first work considering the compatibility issue for lane-changing models.**

I. Introduction

As the development of vehicular technology, Advanced Driver Assistance Systems (ADAS) have become popular in modern vehicles. These systems will continue playing an important role (or be part of an autonomous system) when vehicles become more autonomous. However, even for the same ADAS, different automotive makers (and suppliers) develop different types of systems, and it is questionable if these systems are "compatible," *i.e.*, if they work properly together. A compatibility issue may be resolvable by human drivers, but it can cause a functional failure for autonomous vehicles. In this paper, we focus on lane-changing assistance systems and answer if two lane-changing models are compatible so that the two corresponding vehicles on different lanes can exchange their lanes successfully.

Existing work has proposed different kinds of algorithms for lane-changing maneuvers. Song and Li [13] divided them into two different categories: machine-learning approaches [1], [4], [7] and model-based approaches [2], [8], [10]. Machine-learning approaches imitate human-like decision making or estimate intentions of other vehicles. Model-based approaches model the process of lane-changing decision making, where a cost function or a gain function is usually used to evaluate how good a particular maneuver is. If the connectivity between vehicles is available, the information that vehicles share also plays a role in the decision-making process [3], [5], [9], where the safety, lane-changing success rate, human comfort, and fuel efficiency can be further analyzed and improved in a cooperative setting. Other studies focused on some specific environments (*e.g.*, highway). Wang [6] established a lane-changing model to mimic aggressive behaviors on highway. Zheng *et al.* [11] proposed a cooperative lane-changing strategy for connected and autonomous vehicles to improve traffic operations and safety at a diverging area near a highway off-ramp. Kumaravel *et al.* [12] modeled the problem of cooperative merging at a highway on-ramps as a job-shop scheduling problem and presented an optimal scheduling algorithm for the problem.

The existing work above assumes that all vehicles use the same lane-changing model, and the compatibility issue has not been considered, no matter for the same type or different types of lane-changing models[1]. In this paper, we consider the compatibility issue and verify if two lane-changing models are compatible so that the two corresponding vehicles on different lanes can exchange their lanes successfully. We propose a methodology and an algorithm to perform the verification on the combinations of four lane-changing models. Experimental results demonstrate the compatibility (or incompatibility) between them. The verification results can be utilized during runtime to prevent incompatible vehicles from entering a lane-changing road segment. To the best of our knowledge, this is the first work considering the compatibility issue for lane-changing models.

The rest of this paper is organized as follows. Section II defines the problem. Section III introduces the methodology. Section IV explains how we perform the compatibility checking. Section V provides experimental results, and Section VI concludes this paper.

II. Problem Definition

The lane-changing scenario is as follows. There is a road segment with a finite length. There are two lanes and two vehicles, and each lane has exactly one vehicle on it. Both vehicles intend to change their lanes, *i.e.*, exchange their lanes. Our goal is to verify if the two vehicles can exchange their lanes before the end of the road segment (the liveness property), which is defined as "*compatible*."

We want to emphasize that a lane-changing model itself is designed to provide safe lane-changing maneuvers, so the collision freeness (the safety property) is guaranteed by each lane-changing model. Here, we are concerned if the maneuver of one vehicle prevents the other vehicle from changing its lane,

This work is partially supported by Ministry of Education (MOE) in Taiwan under Grant Number NTU-110V0901, Ministry of Science and Technology (MOST) in Taiwan under Grant Number MOST-110-2636-E-002-026, National Science Foundation under Grant Numbers CPS 1645578 and CCF 1723476, and the NSF/Intel joint research center for Computer Assisted Programming for Heterogeneous Architectures (CAPA). Po-Yu Huang and Kai-Wei Liu contributed equally.

[1]In Section V, we will demonstrate that, even with the same lane-changing model, the two corresponding vehicles on different lanes may not exchange their lanes successfully.

978-3-9819263-6-1/DATE22/©2022 EDAA

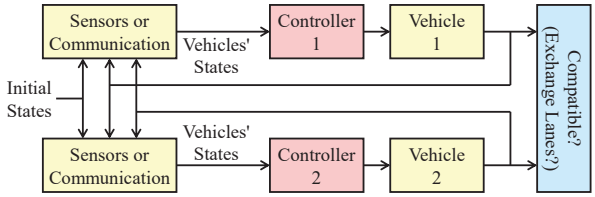

Fig. 1. The proposed methodology.

and vice versa. As a motivation example of incompatibility, *the two vehicles always accelerate or decelerate together, keep the same longitude along the road segment, and fail to exchange their lanes before the end of the road segment*.

Regarding the usage, the verification results can be utilized (1) during design time to trigger redesign of lane-changing models or (2) during runtime to prevent incompatible vehicles from entering the road segment with the incompatible conditions (by traffic lights or instruction messages). However, as lane-changing assistance systems are usually developed separately by different automotive makers or suppliers, we believe that the second case is more applicable in practice.

III. Methodology

The proposed methodology is shown in Figure 1. The two vehicles have their initial states. The initial or current states are obtained via sensors or communication, depending on the available functions of the two vehicles. The two vehicles' states are the inputs of the controllers. Each controller makes decision based on its lane-changing model and controls the corresponding vehicle. Then, the states are updated, and the process repeats iteratively until the two vehicles exchange their lanes ("compatible") or at least one of them fails to exchange its lane before the end of the road segment ("incompatible"). The methodology can work, no matter the controllers and their lane-changing models are the same or different.

More specifically, we use a 3-tuple $S = (v, x, l)$ to describe the state of a vehicle, where v is its velocity, x is its longitude along the road segment, and l is the lane that the vehicle is on. We use a 2-tuple $D = (a, c)$ to describe the decision of a vehicle, where a is the acceleration and c is the lane that the vehicle will be on, *i.e.*, the vehicle will change its lane if and only if $c \neq l$. Given the state of the ego vehicle, S, and the state of the other vehicle, S', a controller can be generally described as a function F to make the decision D as $F : S \times S' \mapsto D$.

IV. Compatibility Checking

Our algorithm for compatibility checking is listed in Algorithm 1. The states and the time steps are discrete as the control command generated by a controller is discrete and periodic. The inputs of the algorithm includes the initial lanes that the two vehicles are on (l_1, l_2), the controllers of the two vehicles (F_1, F_2), and the initial ranges of (v_1, v_2, x_1, x_2). We consider the initial ranges here because they can provide flexible granularity for incompatible conditions which can be utilized during runtime, as mentioned in Section II. Also, if the upper bounds and the lower bounds are given based on traffic

Algorithm 1 Compatibility Checking

Input: l_1, l_2, F_1, F_2 and Initial Ranges of (v_1, v_2, x_1, x_2)
Output: p ▷ Success rate

```
1:  function COMPATIBILITY-CHECKING
2:      (Θ⁻, Θ⁺, p) ← (∅, ∅, 0)
3:      for (v1, v2, x1, x2) ∈ Initial Ranges do
4:          Θ ← ∅
5:          (S1, S2) ← ((v1, x1, l1), (v2, x2, l2))
6:          (p1, p2) ← (PROB(S1), PROB(S2))
7:          while (S1, S2) ∉ Θ⁻ ∪ Θ⁺ do
8:              Θ ← Θ ∪ (S1, S2)              ▷ Update traversed set
9:              if FAIL(S1, S2) or SUCCEED(S1, S2) then
10:                 break
11:             end if
12:             (D1, D2) ← (F1(S1, S2), F2(S2, S1))
13:             S1* ← NEXT-STATE(S1, D1)
14:             S2* ← NEXT-STATE(S2, D2)
15:             (S1, S2) ← (S1*, S2*)          ▷ Update states
16:         end while
17:         if (S1, S2) ∈ Θ⁻ or FAIL(S1, S2) then
18:             Θ⁻ ← Θ⁻ ∪ Θ                   ▷ Update failed set
19:         else
20:             Θ⁺ ← Θ⁺ ∪ Θ                   ▷ Update successful set
21:             p ← p + p1p2                  ▷ Update success rate
22:         end if
23:     end for
24:     return p
25: end function
```

regulations or physical limitations, the initial ranges can still be set to consider all possible values. The output of the algorithm is the success rate p of the lane changing, given the inputs. The verification result is "compatible" if and only if the success rate is 1. We consider the success rate here because, if it is not 1, it can be used to evaluate the ratio of incompatible conditions over all possible conditions, *i.e.*, the level of incompatibility.

Algorithm 1 is based on depth-first search (DFS) and state memorization which can be implemented by a hash with constant time complexity or a binary tree with logarithmic time complexity for each access. We use Θ^- to memorize the failed set (of the vehicles' states (S_1, S_2)), Θ^+ to memorize the successful set, and Θ to memorize the traversed set from a specific initial 4-tuple (v_1, v_2, x_1, x_2).

For each possible 4-tuple (v_1, v_2, x_1, x_2) in the initial ranges, Algorithm 1 initializes Θ and the vehicles' states S_1 and S_2 as well as the probabilities of S_1 and S_2 (Line 4–6), where the probabilities depend on the distributions (*e.g.*, uniform distributions, normal distributions, etc.) of the values of (v_1, v_2, x_1, x_2) in the initial ranges. If (S_1, S_2) has not been checked before (Line 7), Algorithm 1 updates Θ (Line 8) and checks if (S_1, S_2) fails, *i.e.*, at least one vehicle fails to exchange its lane before the end of the road segment, or succeeds, *i.e.*, both vehicles have exchanged their lanes. If (S_1, S_2) fails or succeeds, Algorithm 1 breaks the while-loop (Lines 9–10); otherwise, Algorithm 1 uses the controllers F_1 and F_2 to compute decisions (Line 12), computes the next states S_1^* and S_2^* (Lines 13–14), updates

TABLE I
SUCCESS RATES WITH DIFFERENT LENGTHS OF ROAD SEGMENT, WHERE $\surd$ MEANS 1 (COMPATIBLE).

	Controller in [6]			Controller in [10]			Controller in [11]			Ours		
Length of Road Segment (m)	50	100	200	50	100	200	50	100	200	50	100	200
Controller in [6]	0.524	0.742	0.841	0.506	0.551	0.551	0.670	0.882	0.916	0.687	0.975	$\surd$
Controller in [10]	0.506	0.551	0.551	0.745	0.869	0.909	0.703	0.855	0.859	0.657	0.866	0.883
Controller in [11]	0.670	0.882	0.916	0.703	0.855	0.859	0.802	0.849	0.849	0.779	0.893	0.905
Ours	0.717	0.982	$\surd$	0.673	0.864	0.876	0.804	0.888	0.888	0.854	$\surd$	$\surd$

TABLE II
SUCCESS RATES WITH DIFFERENT INITIAL RANGES OF VELOCITIES, WHERE $\surd$ MEANS MEANS 1 (COMPATIBLE).

	Controller in [6]			Controller in [10]			Controller in [11]			Ours		
Initial Ranges of Velocities (m/s)	±1	±5	±10	±1	±5	±10	±1	±5	±10	±1	±5	±10
Controller in [6]	0.216	0.742	0.857	0.000	0.551	0.743	0.920	0.882	0.804	0.972	0.975	0.968
Controller in [10]	0.000	0.551	0.743	0.568	0.869	0.922	0.852	0.855	0.796	0.855	0.866	0.895
Controller in [11]	0.920	0.882	0.804	0.852	0.855	0.796	0.944	0.849	0.819	0.929	0.893	0.829
Ours	0.991	0.982	0.973	0.873	0.864	0.890	0.914	0.888	0.831	$\surd$	$\surd$	0.998

(S_1, S_2) (Line 15), and continues the while-loop. Note that the next state $S_i^* = (v_i^*, x_i^*, l_i^*)$ of $S_i = (v_i, x_i, l_i)$ with $D_i = (a_i, c_i)$ can be computed by

$$v_i^* = v_i + a_i \cdot \Delta t; \tag{1}$$

$$x_i^* = x_i + v_i \cdot \Delta t + \frac{1}{2} \cdot a \cdot \Delta t^2; \tag{2}$$

$$l_i^* = c_i, \tag{3}$$

where Δt is the step size of time or the period of control commands. After the while-loop, if (S_1, S_2) fails, no matter it has been checked before or it is just decided, Algorithm 1 updates Θ^- (Lines 17–18); otherwise, Algorithm 1 updates Θ^+ and p (Lines 19–21). After this, Algorithm 1 continues the DFS for another possible 4-tuple (v_1, v_2, x_1, x_2) in the initial ranges.

V. EXPERIMENTAL RESULTS

A. Experimental Setting

We implement the proposed algorithm in C++ programming language and perform the experiments on a MacBook Pro with 2.3GHz dual-core Intel Core i5 CPU and 8 GB memory. We consider three recent controllers [6], [10], [11] and design one controller giving a higher priority to a vehicle as follows:

- If the longitudes of the two vehicles are different, we can define one front vehicle and one back vehicle. The front vehicle accelerates with the maximum acceleration $a_{\max}$ without exceeding the velocity limit. If the longitudinal distances between the two vehicles is larger than a constant safe distance $d_{\min}$, the back vehicle accelerates with the maximum acceleration $a_{\max}$ without exceeding the velocity limit; otherwise, the back vehicle decelerates with the maximum deceleration $b_{\max}$.
- If the longitudes of the two vehicles are the same, we can define one right vehicle and one left vehicle. The right vehicle accelerates with the maximum acceleration $a_{\max}$ without exceeding the velocity limit. The left vehicle decelerates with the maximum deceleration $b_{\max}$. At the next time step, the longitudes of the two vehicles will become different.
- If the target lane has a space larger than $d_{\min}$, the ego vehicle changes to the target lane; otherwise, it stays on the same lane.

In the following experiments, $d_{\min} = 10$m, $a_{\max} = 2\text{m/s}^2$, $b_{\max} = 4\text{m/s}^2$. Also, in Algorithm 1, the step sizes of velocities are 1m/s (Line 3), the step sizes of longitudes are 1m (Line 3), and the uniform distributions are applied to the values of (v_1, v_2, x_1, x_2) in the initial ranges (Line 6), and the step size of time is $\Delta t = 0.1$s (Lines 13–14 and Equations (1) and (2)).

B. Experiment on Different Lengths of Road Segment

In this experiment, we set the initial ranges of (v_1, v_2, x_1, x_2) to be $(10 \pm 5, 10 \pm 5, 0 \pm 5, 0 \pm 5)$ in m/s or m. The length of the road segment is set to be 50m, 100m, or 200m. We pair the three recent controllers [6], [10], [11] and ours, resulting in 16 combinations.

The experimental results are listed in Table I, where the controllers listed at the left and top sides mean that the corresponding vehicles are on the left and right lanes, respectively. We can observe the trend that the success rate increases as the length of the road segment increases, which is because the two corresponding vehicles have longer space and thus more opportunities to exchange lanes. However, even if the length of the road segment reaches 200m, the success rates do not reach 1 in most cases, which is because it is possible that the two vehicles always keep the same longitude along the road segment, as mentioned in the motivation example in Section II.

When the length of the road segment is long enough (100m in this experiment), the success rate of the pair of our controllers reaches 1 (compatible), which is because our controller defines one right vehicle and one left vehicle and give the right vehicles a higher priority to accelerate first. This suggests that prioritization (or a tiebreaker) is crucial to the compatibility in the lane-changing scenario. This is also the reason that the results with our controller are asymmetric, *i.e.*, when it is paired with another controller, using it for the right vehicle and the left vehicle has different results. Here, we are not claiming that our controller is the only applicable controller (there are other perspectives such as performance, human comfort, and fuel efficiency), but it can narrow or even clear the incompatible conditions and thus reduce interference during runtime.

C. Experiment on Different Initial Ranges of Velocities

In this experiment, we set the initial ranges of longitudes to be 0 ± 5m and the length of the road segment to be

TABLE III
RUNTIMES (SECOND) WITH DIFFERENT LENGTHS OF ROAD SEGMENT.

	Controller in [6]			Controller in [10]			Controller in [11]			Ours		
Length of Road Segment (m)	50	100	200	50	100	200	50	100	200	50	100	200
Controller in [6]	0.241	0.290	0.346	0.218	0.245	0.278	0.201	0.215	0.216	0.220	0.232	0.234
Controller in [10]	0.210	0.243	0.278	0.193	0.219	0.256	0.184	0.185	0.189	0.205	0.216	0.220
Controller in [11]	0.190	0.204	0.213	0.167	0.178	0.182	0.159	0.156	0.164	0.180	0.185	0.209
Ours	0.203	0.224	0.229	0.193	0.210	0.212	0.185	0.179	0.199	0.199	0.202	0.211

TABLE IV
RUNTIMES (SECOND) WITH DIFFERENT INITIAL RANGES OF VELOCITIES.

	Controller in [6]			Controller in [10]			Controller in [11]			Ours		
Initial Ranges of Velocities (m/s)	±1	±5	±10	±1	±5	±10	±1	±5	±10	±1	±5	±10
Controller in [6]	0.027	0.286	0.881	0.021	0.238	0.832	0.016	0.224	0.679	0.016	0.230	0.784
Controller in [10]	0.020	0.239	0.772	0.020	0.217	0.692	0.013	0.185	0.595	0.015	0.206	0.706
Controller in [11]	0.015	0.208	0.634	0.013	0.172	0.578	0.010	0.155	0.473	0.013	0.179	0.595
Ours	0.014	0.213	0.723	0.016	0.200	0.699	0.016	0.172	0.586	0.013	0.194	0.682

100m. The initial ranges of velocities are set to 10 ± 1m/s, 10 ± 5m/s, and 10 ± 10m/s. The experimental results are listed in Table II, where the controllers listed at the left and top sides mean that the corresponding vehicles are on the left and right lanes, respectively. For the controllers in [6], [10], the success rate tends to increase as the initial ranges of velocities increase. On the contrary, for the other controllers, the success rate tends to decrease as the initial ranges of velocities increase. The difference is because close velocities are easier to make the controllers in [6], [10] incompatible, but they are easier to make the others compatible. Due to the limitation of space, we do not report the experimental results with different parameters or different initial ranges of longitudes, but the proposed methodology and algorithm are still applicable to them.

D. Runtimes

We also report the runtimes for the experiments above. As shown in Tables III, the runtime increases as the length of the road segment increases, but it also depends on the success rate as the two vehicles may exchange their lanes very early in some cases. As shown in Tables IV, the runtime increases as the initial ranges of velocities increase. The number of initial states increases quadratically as there are two velocity variables, but the runtime is lower than a quadratic growth as we utilize state memorization to perform the compatibility checking efficiently.

VI. CONCLUSION

A lane-changing model can meet its own requirements, but it may be incompatible with another lane-changing model as they are usually developed separately by different automotive makers or suppliers. In this paper, we verified if two lane-changing models are compatible. The verification results can be utilized during runtime to prevent incompatible vehicles from entering a lane-changing road segment. To the best of our knowledge, this is the first work considering the compatibility issue for lane-changing models. Future directions of compatibility checking include modeling with hybrid systems (combinations of differential equations and state machines), lane-changing scenarios with more vehicles, other ADAS, more complicated applications such as intersection management, and deadlock analysis and prevention which are challenging at the design stages of individual vehicles but addressable as a compatibility problem of multiple vehicles.

REFERENCES

[1] V. Popescu and S. Nedevschi, "Cut-in maneuver recognition and behavior generation using bayesian networks and fuzzy logic," in *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, Aug. 2012, pp. 123–130.

[2] W. Yao, H. Zhao, P. Bonnifait, and H. Zha, "Lane change trajectory prediction by using recorded human driving data," in *IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2013, pp. 430–436.

[3] M. Düring and P. Pascheka, "Cooperative decentralized decision making for conflict resolution among autonomous agents," in *IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, Jun. 2014, pp. 154–161.

[4] M. Schreier, V. Willert, and J. Adamy, "Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2014, pp. 334–341.

[5] J. Nie, J. Zhang, W. Ding, X. Wan, X. Chen, and B. Ran, "Decentralized cooperative lane-changing decision-making for connected autonomous vehicles," in *IEEE Access*, vol. 4, 2016, pp. 9413–9420.

[6] Y. Wang, "Modeling and simulation of aggressive lane-changing behavior for highway driver training," in *International Conference on Computer and Communications (ICCC)*, Dec. 2017, pp. 2894–2898.

[7] H. Woo, Y. Ji, H. Kono, Y. Tamura, Y. Kuroda, T. Sugano, Y. Yamamoto, A. Yamashita, and H. Asama, "Lane-change detection based on vehicle-trajectory prediction," in *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 2, Apr. 2017, pp. 1109–1106.

[8] Z. Wang, S. Cui, and T. Yu, "Automatic lane change control for intelligent connected vehicles," in *International Conference on Electromechanical Control Technology and Transportation (ICECTT)*, Apr. 2019, pp. 286–289.

[9] X. Liu, N. Masoud, and Q. Zhu, "Impact of sharing driving attitude information: A quantitative study on lane changing," in *IEEE Intelligent Vehicles Symposium (IV)*, Nov. 2020, pp. 1998–2005.

[10] K. Ouyang, Y. Wang, Y. Li, and Y. Zhu, "Lane change decision planning for autonomous vehicles," in *Chinese Automation Congress (CAC)*, Nov. 2020, pp. 6277–6281.

[11] Y. Zheng, B. Ran, X. Qu, J. Zhang, and Y. Lin, "Cooperative lane changing strategies to improve traffic operation and safety nearby freeway off-ramps in a connected and automated vehicles environment," in *IEEE Transaction on Intelligent Transportation Systems (T-ITS)*, vol. 21, no. 11, Nov. 2020, pp. 4605–4614.

[12] S. D. Kumaravel, A. A. Malikopoulos, and R. Ayyagari, "Decentralized cooperative merging of platoons of connected and automated vehicles at highway on-ramps," in *American Control Conference (ACC)*, Feb. 2021, pp. 2055–2060.

[13] R. Song and B. Li, "Surrounding vehicles' lane change maneuver prediction and detection for intelligent vehicles: A comprehensive review," in *IEEE Transaction on Intelligent Transportation Systems (T-ITS)*, May 2021, pp. 1–17.