Egocentric abstractions for modeling and safety verification of distributed cyber-physical systems

Sung Woo Jeon
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Champaign, USA
sjeon12@illinois.edu

Sayan Mitra
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Champaign, USA
mitras@illinois.edu

***Abstract*—Modeling is a significant piece of the puzzle in achieving safety certificates for distributed IoT and cyber-physical systems. From smart home devices to connected and autonomous vehicles, several modeling challenges like dynamic membership of participants and complex interaction patterns, span across application domains. Modeling multiple interacting vehicles can become unwieldy and impractical as vehicles change relative positions and lanes. In this paper, we present *an egocentric abstraction* for succinctly modeling local interactions among an *arbitrary number* of agents around an ego agent. These models abstract away the detailed behavior of the other agents and ignore present but physically distant agents. We show that this approach can capture interesting scenarios considered in the responsibility sensitive safety (RSS) framework for autonomous vehicles. As an illustration of how the framework can be useful for analysis, we prove safety of several highway driving scenarios using egocentric models. The proof technique also brings to the forefront the power of a classical verification approach, namely, inductive invariant assertions. We discuss possible generalizations of the analysis to other scenarios and applications.**

I. Introduction

Creating solutions for the development of safe and secure IoT and cyber-physical systems (CPS) will require research endeavors spanning disciplinary boundaries and styles of enquiry. Mathematical models have a role in this enterprise. Models are not only necessary for rigorous guarantees, but also for precise communication, intelligibility, and abstraction [34]. However, models for distributed IoT and CPS systems are not without complications. Interactions among agents (e.g., cars or devices) have to be captured without compromising compositionality and dynamic membership has to be maintained even as participating agents move in and out. In this paper, we present *egocentric abstractions* for modeling local interactions among an *arbitrary number* of agents. Egocentric abstractions remove detailed information about behavior of distant or irrelevant agents while preserving the semantics of the ego agent. As a result, egocentric models are smaller and also require less-bookkeeping than full-blown interaction models. The price is that the rules for interaction are absorbed in the ego agent model, and this potentially leads to more complexity.

This work was supported in part by the National Science Foundation through research grants NSF FMITF: 1918531 and NSF CCF 2008883.

We believe that the egocentric view will be useful for applications where each agent interacts with many agents over a lifetime, but only with a handful over a short time span. Examples domain include urban air-traffic management [2, 15], microgrids [32], and [28, 13]. Here we show that this approach can capture interesting scenarios considered in the *responsibility sensitive safety (RSS)* framework for autonomous vehicles [29]. Safety analyses of autonomous vehicles, and even simpler driver assistance systems, remain challenging [14, 29] and test driving alone is inadequate for providing acceptable levels of confidence. *RSS* formalizes simple traffic rules by which agents can avoid being responsible for accidents. This way *RSS* suggests a scalable "inductive" method of making safe decisions under different multi-agent settings. RSS has been adopted as a safety framework at Mobileye and then Intel, it has been used to formulate temporal logic-based monitors [9], its safety conditions have been generalized [20], and a C++ library has been built to integrate it with the Baidu Apollo driving stack [8]. With the exception of the short paper [23], neither the original article [29] nor any of the follow-up works aim to provide a model of vehicle interactions that can be used for verification.

The challenge of modeling highway traffic, and for that matter any distributed system, is to properly keep track of the agent information. There can be hundreds of cars on the highway which can affect each other. Managing the information of every agent and the relation between different agents can be complex and costly, as we will explain further in Section II. To resolve the problem, *egocentric abstraction* happens in two different directions. First, we reduce the number of variables in the model by focusing on the neighboring agents and trimming away the agents that are not directly related to the safety of the ego car. Here, 'neighboring agents' are simply the cars that are in front, front-left, rear-left, etc., of the *ego car*. In general, this notion of neighborhood could be defined to suite the application. Since this neighborhood relation can change as agents move and pass, it can be challenging to define this relation precisely. Therefore, instead of keeping track of the entities bound to each specific neighborhood relation, the model relies on the information that the ego agent can collect from the neighbors at any given time. Second, we make the model permissive to the behaviors of the neighboring

agents, giving more freedom to the neighboring agent than they have in the real world. The neighboring agent are allowed to perform any behaviors as long as they follow minimal rules that prevent from endangering the ego agent. With the extended freedom of the neighbors, the safety analysis of the model becomes simpler.

As an illustration of how the modeling framework can be useful for safety analysis, we prove safety of several highway driving scenarios using egocentric models. The proof technique also brings to the forefront the power of a classical verification approach, namely, inductive invariant assertions. We discuss possible generalizations of the analysis to other scenarios and applications.

Related works: Modeling of "smart" vehicles and platoons have received the attention of researchers for many decades [4, 31, 33]. Recent advances cover verification of controllers for individual vehicles [11, 35, 30], all the way up to design of optimal strategies for large-scale traffic measurement [10] and flow control in mixed-autonomy networks [17, 3]. Within these different levels of abstractions, we study safety analysis problems concerning interactions of a small neighborhood of vehicles, while abstracting away the details of the control of an individual vehicle.

II. Interaction modeling approaches for CPS

We focus on models for formal analysis, however, the discussion is relevant for simulations as well. For the sake of concreteness, consider the instantaneous snapshot of a three-lane scenario (Figure 1): Three cars c_1, c_2, c_3 are moving along their respective lanes; c_1 is longitudinally *behind* c_2 and c_3. In the course of time, other vehicles may enter and leave the neighborhoods of each of these vehicles. The following paragraphs illustrates different possible approaches in modeling the scenario.

Flat models: A quick and naïve approach for modeling dynamic scenarios is to create a flat array of agents where all the information about each agent is available to every other agent. This is lightweight, but lacks *data encapsulation.* and leads to usage errors like one agents accessing state of a distant agent. The approach also lacks *support for dynamic membership.* All vehicle states have to be maintained all the time.

Interacting automata: An alternative is to consider each car as an independent automaton that makes decisions based only on the information available to it from *neighbors* [16]. For instance, for scenario above, an automaton could represent each of the agents; a single world automaton would collect all the *observable data* from all the agents, maintain the neighborhood relationship, and only make those information available to agent c_1 that are from its neighbors. The flow of information is shown in Figure 1. This approach does not suffer from the the data encapsulation problem, but the size of the model still grows with the number agents and the world automaton has to maintain complex state dependent neighborhood relations.

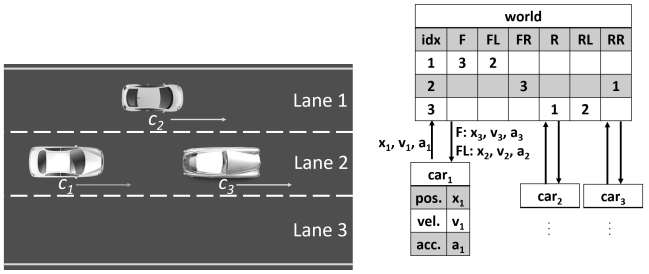

Fig. 1. Interacting automata approach. The world automaton has a table that keeps track of the relation between different car automata. The world automaton collects position, velocity, acceleration from every car automaton and gives neighbors' information to each car.

Egocentric automaton: This approach allows dynamic changes in the neighborhood and provides better data encapsulation, at the expense of requiring more complex coordination logic. This type of approach is also used in [18, 26], however, our work is the first egocentric highway model. Unlike the above approaches where all the cars were treated equally, here we have an *ego car*, say c_1, and the model is from the ego car's perspective (Figure 2) and it keeps track of the information of its finitely many neighbors. The ego car accesses the information of the neighbors using pointers that were assigned to the neighbors based on their relative position to the ego car. As the cars are moving and the relative positions are changing, the entity that a specific pointer refers to can change over time. For example, the front car in Figure 2 can move to lane 3 and become the front-right car, or it can even disappear from the table by moving to lane 1. The egocentric model *does not* keep track of which specific agent it is referring to. Any such changes on the neighboring relation appears as a discrete update of the information. The egocentric model naturally provides better data encapsulation. Since the number of neighbors is finite at any given time, the model tracks a constant number of variables, regardless of the total number of member cars in the system.

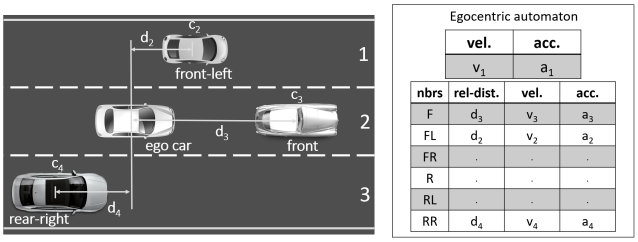

Fig. 2. Egocentric three-lane scenario.

III. Background: Automata and Invariants

Egocentric models are automata or state machines. For the purpose of defining a model for interactive vehicle systems, the state variables need to evolve both continuously and discretely with time. Here we provide backgrounds that are required to specify a hybrid model.

A *variable* is an named quantity that is associated with a type such as integer, float, enumeration, etc. A *valuation* for a set of variables V maps each variable name $v \in V$ to a value in its type. $val(V)$ is the set of all possible valuations of the variables in V. As usual, a *transition* over a set of variables V is a relation $R \subseteq val(V) \times val(V)$ that specifies a discrete state change relation. For $T \in \mathbb{R}^+$ and a set of variables V, a *trajectory* τ of duration T for V is a mapping from each point in the interval $[0, T]$ to $val(V)$ and it models continuous change.

A hybrid automaton (HA) $\mathcal{A}$ is a state machine whose variables can change through transitions or trajectories. Formally, it is a tuple (V, Θ, A, $\mathcal{D}$, $\mathcal{T}$) where (a) V is a set of variables or state variables. $val(V)$is the set of states; (b) $\Theta \subseteq val(V)$ is a nonempty set of start states; (c) A is a set of actions or transition labels; (d) $\mathcal{D} \subseteq val(V) \times A \times val(V)$ is the set of transitions. A transition (v, a, v') is written as $v \xrightarrow{a} v'$. Finally, (e) $\mathcal{T}$ is a set of trajectories for V. For a closed trajectory $\tau \in \mathcal{T}$, we write $\tau.\mathsf{fstate}$ and $\tau.\mathsf{lstate}$ the first and the last state of τ.

An *execution fragment* of an automaton $\mathcal{A}$ is an alternating, possibly infinite, sequence of actions and trajectories $\alpha = \tau_0 a_1 \tau_1 a_2 ...$, where each τ_i is a trajectory in $\mathcal{T}$ and if τ_i is not the last trajectory in the sequence then $\tau_i.\mathsf{lstate} \xrightarrow{a_{i+1}} \tau_{i+1}.\mathsf{fstate}$. An execution fragment is an *execution* if $\tau_0.\mathsf{fstate} \in \Theta$. A state $v \in val(V)$ is *reachable* if it is the last state of some execution of $\mathcal{A}$. An invariant is a set of states $\mathcal{S}$ that contains all reachable states. Proving an invariant I of $\mathcal{A}$ that is disjoint from the unsafe states, therefore, proves safety. We will use the following classical inductive method for proving invariants.

Theorem 1. *Given an HA $\mathcal{A}$, if a set of states $I \subseteq val(V)$ satisfies the following:*

(a) *(Start condition) For any starting state $x \in \Theta, x \in I$,*
(b) *(Transition closure) For any action $a \in A$, if $x \xrightarrow{a} x'$ and $x \in I$ then $x' \in I$,*
(c) *(Trajectory closure) For any trajectory $\tau \in \mathcal{T}$, if $\tau.\mathsf{fstate} \in I$ then $\tau.\mathsf{lstate} \in I$.*

then I is an invariant of $\mathcal{A}$.

IV. Modeling of a Three-lane Scenario

We will demonstrate the egocentric abstraction on a model of a three-lane, one-way scenario. For simplicity, we measure the longitudinal distance between the center of two cars to obtain the relative longitudinal distance. We assume that the ego car has access to the velocity and acceleration of the cars surrounding it. The safety requirement of the ego car is to proceed without colliding. The model uses 6 parameters:

a_{max}	: maximum acceleration of every car
b_{min}	: minimum braking of the ego car
b_{max}	: maximum braking of every car
ρ	: reaction delay of the ego car
d_{detect}	: maximum sensing distance of the ego car
v_{max}	: maximum velocity of every car
l_{max}	: maximum length of every car(center to an end)

The ego car has the maximum reaction delay ρ, which would be further described in the following section. The ego car has limited sensory capability so it cannot detect neighboring cars that are farther than d_{detect}. Also, we assume that the length of any cars from the center to the front or rear end does not exceed l_{max}.

A. Safe distance and reaction delay

In our model, the ego car avoids collision to the front car by braking once the relative distance to the front car becomes less than a certain threshold. However, sensors and actuators in real world are noisy and has finite frequency, meaning that even if the distance between the two cars is less than the threshold, the ego car might not react for a certain period. To reflect such limitations, we assume that once the actual distance between the ego car and the front car becomes less than a certain threshold, the ego car realizes it within time at most the maximum reaction delay ρ.

One of the key concepts used in the model is the notion of minimum safe distance defined below.

Definition 1. *Given the rear car's velocity v_r, the front car's velocity v_f, and the rear car's maximum reaction delay ρ, the minimum safe distance between the two is*

$$d_{safe}(v_r, v_f, \rho) = \max\Big(v_r \cdot \rho + \frac{1}{2} a_{max} \rho^2 - \frac{(v_r + a_{max}\rho)^2}{2b_{min}} + \frac{v_f^2}{2b_{max}} + l_{max},\ l_{max}\Big)$$

The minimum safe distance d_{safe} has a property that is useful for safety. If d_i is the initial value of d_{rf} and is no less than $d_{safe}(v_r, v_f, \rho)$, then $d_{rf} \geq l_{max}$ until c_r stops if the rear car applies maximum brake even with reaction delay. For any two cars c_r behind c_f, we say the distance d_{rf} between c_r and c_f is **safe** if $d_{rf} \geq d_{safe}(v_r, v_f, \rho)$. Otherwise, d_{rf} is **unsafe**.

B. Identifying key Assumptions

One of the benefits of modeling is that it can help identify key assumptions for safety. We present our assumptions here as an illustration of what is sufficient for this particular analysis to go through, rather than to claim that they are necessary for a real system.

Assumption 1. *The cars change lane only if after changing lane the relative distances to its front and rear cars are safe.*

This constrains the behavior of both the ego car and the neighbors. The ego car controls its acceleration to avoid colliding with the front car. However, if a car cuts in front of the ego car when it is too close to the ego car, then it might not be possible for the ego car to avoid collision.

Assumption 2. *Neighboring cars do not collide with the ego car from behind.*

Throughout RSS, it is the rear car's responsibility to keep enough distance to the ego car.

Assumption 3. *Lane changes occur instantaneously.*

This is a simplification, and more realistic lane change models can be approximated by this with additional safety buffers around vehicles.

C. Egocentric lane change model

We now describe the details of the egocentric model of the three-lane scenario. Figures 3, 4, 5 give the full details of the parameters, types, actions and variables defining the hybrid automaton. We note that with additional effort and some simplifications it is possible to create these models in existing verification tools [22, 27, 7, 6].

automaton Egocentric($b_{max}, b_{min}, a_{max}, \rho, d_{detect}, v_{max}, l_{max}$)
type $Modes$: *enum* [freeDriving, dangerous, braking, stopped]
type $Lanes$: *enum* [0,1,2]
type $Nbrs$: *enum* [*FL*, *F*, *FR*, *BL*, *BR*]
type Dir: *enum* [*L*, *R*]
type $Dist : [0, d_{detect}] \cup \{\infty, \emptyset\}$
type $Vel : [0, v_{max}] \cup \{\infty, \emptyset\}$
type $Acc : [b_{max}, a_{max}] \cup \{\infty, \emptyset\}$
actions
internal
freeDrive($acc' : Acc$)
danger
brake($acc' : Acc$)
stop
depart($acc' : Acc$)
chLane($j : Dir, d_1, d_2, d_3 : Dist, v_1, v_2, v_3 : Vel$)
maxSense($n : Nbrs$)
appear($n : Nbrs, v_1 : Vel$)
updateNbr($n : Nbrs, x : Dist, y : Vel$)

variables
internal
$mode : Modes$
$Lane : Lanes$
$v : Vel$
$v : Nbrs \to Vel$
$d : Nbrs \to Dist$
$acc : Acc$
$acc : Nbrs \to Acc$
t_ρ : Real

Fig. 3. Actions and variables of the three lane model

transitions

freeDrive(acc')
pre $d[F] > d_{safe}(v, v[F], \rho)$
$mode =$ dangerous **or** braking
$b_{max} \le acc' \le a_{max}$
eff $mode =$ freeDriving
$acc = acc'$
$t_\rho = 0$

danger
pre $d[F] \le d_{safe}(v, v[F], \rho)$
$mode =$ freeDriving
eff $mode =$ dangerous
$t_\rho = 0$

brake(acc')
pre $d[F] \le d_{safe}(v, v[F], \rho)$
$mode =$ dangerous
$b_{max} \le acc' \le b_{min}$
eff $mode =$ braking
$acc = acc'$
$t_\rho = \rho$

stop
pre $v = 0$
eff $mode =$ stopped
$acc = 0$
$t_\rho = 0$

depart(acc')
pre $mode =$ stopped
$d[F] > d_{safe}(v, v[F], \rho)$
$acc' > 0$
eff $mode =$ freeDriving
$acc = acc'$
$t_\rho = 0$

chLane($j, d_1, d_2, d_3, v_1, v_2, v_3$)
pre $d[Fj] \ge d_{safe}(v, v[Fj], \rho) \wedge$
$d[Bj] \ge d_{safe}(v[Bj], v, \rho) \wedge$
$((Lane = 1 \wedge d_1 = d_2 = \emptyset) \vee$
$(Lane \ne 1 \wedge d[Fj] \ne \emptyset \wedge$
$d_1 \ne \emptyset \wedge d_2 \ne \emptyset))$
eff *update* $Lane$
$mode =$ freeDriving
$t_\rho = 0$
$d[Fj*] = d[F]$
$d[F] = d[Fj]$
$d[Fj] = d_1$
$v[Fj*] = v[F]$
$v[F] = v[Fj]$
$v[Fj] = v_1$
$d[Bj] = d_2; v[Bj] = v_2$
$d[Bj*] = d_3; v[Bj*] = v_3;$

maxSense(n)
pre $d[n] \ne \emptyset \wedge d[n] > d_{detect}$
eff $d[n] = \infty$

appear(n, v_1)
pre $d[n] = \infty$
$v_1 \in Vel$
eff $d[n] = d_{detect}$
$v[n] = v_1$

updateNbr(n, d_1, v_1)
pre $d[n] \ne \emptyset \wedge$
$(n \ne f \vee d_1 \ge d_{safe}(v, v_1, \rho))$
eff $d[n] = d_1$
$v[n] = v_1$

Fig. 4. Transitions of the three lane model.

As appears in Figure 3 the model has variables v, d, acc to store the information of the ego car and its neighboring cars. It also has variables that represents the discrete mode, lane of the ego car, and the reaction delay. The discrete mode switches between four different values, freeDriving, dangerous, braking, stopped, depending on the distance to the front car. In each mode, the ego car has different range of acceleration. The reaction delay is used in dangerous mode to assure that the ego car starts decelerating within a certain time period once the distance to the front car becomes unsafe. The number of neighbors is fixed as 5 regardless of the number of cars on the road. This shows that the egocentric implementation is more scalable compared to the full-blown model.

Figure 4 describes discrete transitions of the model. The transitions freeDrive, danger, brake captures the discrete changes of the mode, determined by the distance to the front car. The transition stop and depart keeps the velocity of the ego car non-negative, and prevents the ego car from departing when it is too close to the front car. Lane change of the ego car is captured by chLane. the precondition checks the distance to the front and the rear car after changing lane, and also assures that the parameters passed to the transition are valid, using the $Lane$ variable. For example, when the ego car moves to the left lane, there can be no neighbor on the left lane, so the corresponding data must be set to $\emptyset$. The effects updates the lane information of the ego car and also updates the information of the neighbors, by changing the mapping between the neighbors and updating information of any newly observed neighbors. Discrete updates that can occur due to the neighbors' behaviors are abstracted down to the remaining three transitions, maxSense, appear, and updateNbr. The actions maxSense, appear handles a neighbor disappearing beyond the sensor distance and appearing at the maximum sensor distance. Any other discrete updates are captured by updateNbr. The model allows any discrete changes of the neighboring cars as long as they do not make $d[F]$ unsafe. This adds flexibility to the model so it can simulate various scenarios that can occur in real environment using only the three transitions.

Figure 5 describes the continuous changes of the model. The dynamics that applies regardless of the mode appears at the beginning of Figure 5, without a label. The rest of the dynamics applies to each discrete mode. Depending on the mode, the ego car takes different range of acceleration.

The price of egocentric abstraction is the additional work needed to keep track of the relation between the ego car and the neighboring cars. As an example, a single chLane action not only needs to update the ego car's lane information but also the information of all its neighbors. This makes the precondition and the effect complex, while in the full-blown model the same action can be modeled simply.

trajectories

evolve

$\mathrm{d}(v) = acc$

$\forall n \in Nbrs$,

if $v[n] \geq v_{max}$

$acc[n] = [b_{max}, 0]$

else if $v[n] \geq 0$

$acc[n] = [0, a_{max}]$

else

$acc[n] = [b_{max}, a_{max}]$

$\mathrm{d}(v[n]) = acc[n]$

if $d[n] \neq \infty$

$\mathrm{d}(d[n]) = v[n]$

stop when

neighbor leaves $d_{detect} \vee$

neighbor appears

freeDriving

evolve

$acc \in [b_{max}, a_{max}]$

stop when

$(d[F] \leq d_{safe}(v, v[F], \rho)) \vee$

$(v \leq 0 \wedge acc \leq 0)$

dangerous

evolve

$acc \in [acc_{b,max}, acc_{max}]$

$\mathrm{d}(t_\rho) = 1$

stop when

$(d[F] > d_{safe}(v, v[F], \rho)) \vee$

$(v \leq 0 \wedge acc \leq 0) \vee$

$(t_\rho \geq \rho)$

braking

evolve

$acc \in [acc_{b,max}, acc_{b,min}]$

stop when

$(d[F] > d_{safe}(v, v[F], \rho)) \vee$

$(v \leq 0 \wedge acc \leq 0)$

stopped

evolve

$acc = 0$

Fig. 5. Trajectories of the three lane model.

V. Safety Analysis via Invariant Assertions

Egocentric abstractions together with standard invariant assertions can be used for establishing safety of distributed cyber-physical systems. Here we state the key invariants we have proved for the model presented in Section IV. Due to limited space we omit most of the proofs. It is worth noting that this proof approach can be and has been used earlier to partially automate verification using theorem provers [1, 19, 27].

Invariant 1 states that the velocity of any car in the model meets the velocity constraints described in Section IV in any mode. Invariant 2 states that the reaction delay t_ρ does not exceed ρ under any mode. We show the proof of this invariant to illustrate how these proofs typically proceed by an application of Theorem 1 followed by a case analysis on the actions.

Invariant 1. *For any reachable state s, and for any $n \in Nbrs$, $0 \leq s.v \leq v_{max}$ and $0 \leq s.v[n] \leq v_{max}$.*

Invariant 2. *For any reachable state s,*

- *If $s.mode =$ freeDriving, $s.t_\rho = 0$.*
- *If $s.mode =$ dangerous, $s.t_\rho \in [0, \rho]$.*
- *If $s.mode =$ braking, $s.t_\rho = \rho$.*
- *If $s.mode =$ stopped, $s.t_\rho = 0$.*

Proof. Let I_2 be the set of states that satisfies Invariant 2. Initially the automaton has $mode =$ freeDriving and $t_\rho = 0$, so for any $s \in \Theta$, $s \in I_2$.

Consider any action a and states s, s' such that $s \xrightarrow{a} s'$.

Case $a =$ freeDrive, depart : $s'.mode =$ freeDriving and $s'.t_\rho = 0$, so $s' \in I_2$.

Case $a =$ danger : $s'.mode =$ dangerous and $s'.t_\rho = 0$, so $s' \in I_2$.

Case $a =$ brake : $s'.mode =$ braking and $s'.t_\rho = \rho$, so $s' \in I_2$.

Case $a =$ stop : $s'.mode =$ stopped and $s'.t_\rho = 0$, so $s' \in I_2$.

Case $a =$ maxSense, appear, updateNbr : These actions set $s'.t_\rho$ to 0 only when the parameter $n = f$. In this case, $s'.mode =$ freeDriving. Otherwise, t_ρ remains the same. Therefore, if $s \in I_2$, then $s' \in I_2$.

Consider any trajectory τ such that $\tau(0) \in I_2$. t_ρ increases only while in the mode dangerous, and by the stopping condition of dangerous, $t_\rho \leq \rho$. Therefore if $\tau(0) \in I_2$ then for any t, $\tau(t) \in I_2$. □

The freeDriving mode captures the states where the ego car is safe and therefore can drive with acceleration no greater than a_{max}. Invariant 3 states that the distance to the front car is no less than $d_{safe}(s.v, s.v[F], \rho)$ while in freeDriving mode.

Invariant 3. *In any reachable state s, if $s.mode =$ freeDriving then*

$$s.d[F] \geq d_{safe}(s.v, s.v[F], \rho).$$

The dangerous mode captures the states where $d[F]$ is unsafe but the ego car has not yet reacted. We assume that the ego car reacts to unsafe $d[F]$ within time ρ, so if at some point $d[F]$ has been unsafe for time $t \leq \rho$ and the ego car has not yet reacted, it would start braking within time $\rho - t$ since then. The variable t_ρ serves as a timer that keeps track of the time elapsed while in dangerous mode. Therefore, by Definition 1, if $d[F] \geq d_{safe}(v, v[F], \rho - t_\rho)$, then the ego car would not collide with the front car. Invariant 4 states that this condition indeed holds while in dangerous mode.

Invariant 4. *In any reachable state s, if $s.mode =$ dangerous then*

$$s.d[F] \geq d_{safe}(s.v, s.v[F], \rho - s.t_\rho)$$

Proof. Let $I_4 \subset val(X)$ be the set of states that satisfies Invariant 4. By Assumption 3, the automaton initially has $mode =$ freeDriving and it can reach $mode =$ dangerous only by the discrete transition danger. Therefore, $\forall s \in \Theta$, $s \in I_4$.

Consider any action $a \in \mathcal{A}$ and states $s, s' \in val(X)$ such that $s \xrightarrow{a} s'$ and $s'.mode =$ dangerous.

Case $a =$ danger : The action does not affect the velocities or the relative distances. By the precondition, $s.mode =$ freeDriving and $s.t_\rho = s'.t_\rho = 0$. By Invariant 3, $s.d[F] \geq d_{safe}(s.v, s.v[F], \rho)$ so it follows that $s'.d[F] \geq d_{safe}(s'.v, s'.v[F], \rho - s'.t_\rho)$.

Case $a =$ maxSense, appear, updateNbr : From the proof of Invariant 3 and the definition of safe distance, it follows that $s'.d[n] \geq d_{safe}(s'.v, s'.v[n], \rho) \geq d_{safe}(s'.v, s'.v[n], \rho - s'.t_\rho)$. Therefore, $s' \in I_4$.

Now consider any trajectory τ such that $\tau(0).d[F] \geq d_{safe}(\tau(0).v, \tau(0).v[F], \rho - \tau(0).t_\rho)$, and $\tau(0).mode =$ dangerous. By Invariant 2, $t \leq \rho$. The minimum of $\tau(t).d[F]$ can be obtained when the ego car accelerates with a_{max} during

the t period and the front car decelerates with b_{max}. Therefore, for $t \leq -\frac{\tau(0).v[F]}{b_{max}}$,

$$\begin{aligned}\tau(t).d[F] \geq & \tau(0).d[F] - (\tau(0).v \cdot t + \frac{1}{2}a_{max}t^2) \\ & + (\tau(0).v[F] \cdot t + \frac{1}{2}b_{max}t^2)\end{aligned}$$

By the inductive hypothesis, if we let $C = \rho - \tau(0).t_\rho$,

$$\begin{aligned}\tau(0).d[F] \geq & \tau(0).v \cdot C + \frac{1}{2}a_{max} \cdot C^2 \\ & - \frac{\tau(0).v + a_{max} \cdot C}{2b_{min}} + \frac{\tau(0).v[F]^2}{2b_{max}} + l_{max}.\end{aligned}$$

Also, due to the acceleration constraints,

$$\begin{aligned}\tau(t).v &\leq \tau(0).v + a_{max}t \\ \tau(t).v[F] &\geq \tau(0).v[F] + b_{max}t\end{aligned}$$

and therefore,

$$\begin{aligned}& d_{safe}(\tau(0).v + a_{max}t,\ \tau(0).v[F] + b_{max}t,\ C - t)) \\ & \quad \geq d_{safe}(\tau(t).v, \tau(t).v[F], \rho - \tau(t).t_\rho)\end{aligned}$$

Combining the above we get

$$\begin{aligned}\tau(t).d[F] \geq & d_{safe}(\tau(0).v + a_{max}t,\ \tau(0).v[F] + b_{max}t,\ C - t)) \\ \geq & d_{safe}(\tau(t).v, \tau(t).v[F]\rho - \tau(t).t_\rho)\end{aligned}$$

Now for $t > -\frac{\tau(0).v[F]}{b_{max}}$, since $\tau(t).v[F] \geq 0$ by Invariant 1,

$$\tau(t).d[F] \geq \tau(0).d[F] - (\tau(0).v \cdot t + \frac{1}{2}a_{max}t^2) - \frac{\tau(0).v[F]^2}{2b_{max}}$$

and by the acceleration constraints,

$$\begin{aligned}& d_{safe}(\tau(0).v + a_{max}t,\ 0,\ C - t)) \\ & \quad \geq d_{safe}(\tau(t).v, \tau(t).v[F], \rho - \tau(t).t_\rho)\end{aligned}$$

Therefore we get

$$\begin{aligned}\tau(t).d[F] \geq & d_{safe}(\tau(0).v + a_{max}t,\ 0,\ C - t)) \\ \geq & d_{safe}(\tau(t).v, \tau(t).v[F], \rho - \tau(t).t_\rho)\end{aligned}$$

□

The braking mode captures the states where the ego car is slowing down to avoid collision. Note that when the ego car is decelerating it can be considered as the ego car reacting to unsafe state with 0 response time. Therefore, by Definition 1, the ego car can avoid collision if $d[F] \geq d_{safe}(v, v[F], 0)$. Invariant 5 states that the model satisfies this condition while in braking mode.

Invariant 5. *In any reachable state s, if* $s.mode$=*braking then*

$$s.d[F] \geq d_{safe}(s.v, s.v[F], 0)$$

Proof. The proof is similar to that of Invariant 4, and therefore is omitted due to the limited space. □

stopped mode captures the states where the ego car is stopped. Invariant 6 that whenever the ego car is not moving, $d[F] \geq l_{max}$.

Invariant 6. *In any reachable state s, if* $s.mode =$ *stopped then* $s.d[F] \geq l_{max}$.

A. Safety invariant

We have proven mode-specific invariants for the four discrete modes. Combining the four invariants gives a safety invariant for the model.

Invariant 7. *For any reachable state s,*

- *If* $s.mode =$ *freeDriving*, $s.d[F] \geq d_{safe}(s.v, s.v[F], \rho)$.
- *If* $s.mode =$ *dangerous*, $s.d[F] \geq d_{safe}(s.v, s.v[F], \rho - s.t_\rho)$.
- *If* $s.mode =$ *braking*, $s.d[F] \geq d_{safe}(s.v, s.v[F], 0)$.
- *If* $s.mode =$ *stopped*, $s.d[F] \geq l_{max}$.

Since by definition the minimum safe distance is no less than l_{max}, Invariant 7 shows that for any mode in the model, $s.d \geq l_{max}$. With the constraint on the maximum longitudinal length of the cars, it is guaranteed that the ego car would not collide with its front car. Given Assumption 1 and 2 which assures that there is no collision from side or behind of the ego car, we have shown that the ego car drives without any collision.

VI. Generalization

In this section, we suggest how the egocentric three-lane model could easily be applied on scenarios with more than three lanes. We also provide ideas for simulating merging lanes without significantly modifying the three-lane model. We then discuss the discrete lane change, which is our assumption for simplification, and briefly describe how the model with discrete lane changes can be extended to capture continuous lane changes.

A. Extending the number of lanes

Suppose we wish to extend the model so that it has more than 3 lanes. Since the number of neighbors of the ego car does not change as the number of lanes increases, it is trivial to add lanes to the egocentric three-lane model. The type $Lanes$ must be modified so that it has the desired number of lanes. The precondition for the action chLane must be modified to allow the ego car to move to the extended lanes. The same analysis of invariants applies to the extended model.

B. Merging

Consider a merging scenario illustrated by figure 6. Lane 1 disappears at a certain location so the ego car must switch lane to proceed. This maneuver of the ego car required for a safe merge can be simulated using the three-lane model. Consider the scenario described in figure 7. The front car is at the location where the merging happens with velocity $v[F] = 0$. If the front car stays in that location then the ego car must switch lane to proceed, which is the same as its desired behavior in the merging scenario. Therefore, to model the merging scenario, we only need to introduce variables that indicate the availability of each lane and allow the ego car to detect the distance to the merging at least d_{detect} ahead. The ego car can simply treat the merging as a front car with 0 velocity. This way, the same analysis of invariant applies on the merging scenario.

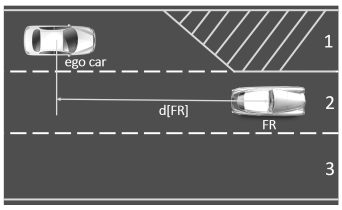

Fig. 6. Merging scenario

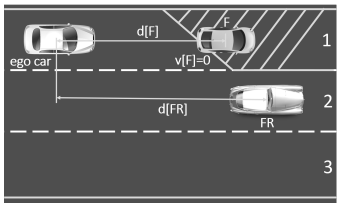

Fig. 7. Simulation of merging scenario

C. Continuous lane change

Suppose we wish to extend the model so the change of lane occurs continuously over time. One way to achieve this is to allow the cars that are changing lane to occupy two lanes, as appears in figure 8.

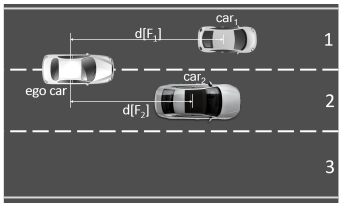

Fig. 8. Intermediate state while changing lane

We can set a parameter ϕ which is the maximum time that a single lane change can take. The model must have another discrete mode M that captures the states where the ego car is changing lane. There must be a timer that counts the time spent in M to guarantee that the change of lane occurs within ϕ. While in M, the ego car would be considered as occupying two lanes, lane 1 and lane 2 in the specific case of figure 8. Both car_1 and car_2 are considered as the front car, so the ego car must keep enough distance from them. In this model, the relative distance to the neighboring cars can change while the ego car is changing lane. Thus, there might be a case where $d[FL]$ was greater than $d_{safe}(v, v[FL], \rho)$ before the ego car started changing lane, but it becomes less than $d_{safe}(v, v[FL], \rho)$ while the ego car was changing lane to the left. To avoid such situations, we can modify the precondition of chLane so the ego car won't change lane to the left unless $d[FL]$ is greater than $d_{safe}(v, v[FL], \rho) + \delta$. The constant δ must be chosen in accordance with the constant ϕ and the acceleration constraints so that it guarantees enough distance margin.

VII. Discussions and Conclusion

We presented an abstraction for modeling complex, distributed, cyber-physical systems that reduces information and bookkeeping details about distant and irrelevant agents from the point of view of an ego agent. This egocentric abstraction can be useful for modeling and safety analysis of distributed systems in which agents dynamically join, leave, fail, and interact with continuously changing neighbors.

We presented a detailed formal model of highway driving based on the popular responsibility sensitive safety (RSS) framework [29]. We see that the egocentric abstraction dramatically reduces the size of the model—instead of an arbitrary number of vehicles it only tracks a small number of neighbors at any given time. At the same time, the price of this reduction is slightly more complex transition rules that track the neighbors. We proved safety of the ego car by showing a safe inductive invariant of the egocentric model. This case study also shows that the classical proof technique of inductive invariant assertions can help prove safety in this dynamic setting. In the process, we also identify the key assumptions needed for the safety proof. These types of assumptions will ultimately define what are called the *Operational Design Domains (ODD)* or the conditions under which the autonomous system is assured to be safe.

There are several limitations in the current work. For example, lane changes are modeled as instantaneous events rather than a continuous process. This semantic simplification can approximate more realistic lateral dynamics. The straight line lanes can be naturally generalized to more complex lane geometries.

Another application of the egocentric abstraction could be in the urban traffic management (UTM) and the integration of unmanned aerial systems (UAS) in the national airspace. Collision avoidance of airborne traffic has been widely studied, with many results on safety analysis (see, for example, [21, 12, 25, 5]). However, analysis of distributed and dynamic scenarios remains a major challenge. Egocentric abstraction can be a step toward a scalable analysis of collision avoidance system. Relatively simple problems such as waypoint tracking could be modeled with an egocentric approach. The goal of the vehicles would be to maintain distance between each other while following predefined waypoints, which is similar to the highway scenario as the ego aircraft (ownship) makes evading maneuvers based only on the neighboring aircraft. The challenge here is to properly define the notion of neighbors and the decision making process.

Another practical research direction is to integrate traffic simulators like SUMO [24] with this model. The lightweight nature of the egocentric model makes it appropriate for use as a real-time monitor for the ego agent. The ego agent equipped with such a monitor based on the egocentric model and could predicatively evaluate safety and raise alarms. Finally, it would be interesting to see whether variants of egocentric models can be used for analyzing other types of requirements such as noninterference, anonymity, and intrusion detection.

References

[1] M. Archer, H. Lim, N. A. Lynch, S. Mitra, and S. Umeno. Specifying and proving properties of timed I/O automata in the TIOA toolkit. In *In 4th ACM-IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'06)*. IEEE, 2006.

[2] S. Bharadwaj, S. Carr, N. Neogi, H. Poonawala, A. B. Chueca, and U. Topcu. Traffic management for urban air mobility. In *NASA Formal Methods Symposium*, pages 71–87. Springer, 2019.

[3] S. Coogan and M. Arcak. A compartmental model for traffic networks and its dynamical behavior. *IEEE Trans. Autom. Control.*, 60(10):2698–2703, 2015.

[4] E. Dolginova and N. A. Lynch. Safety verification for automated platoon maneuvers: A case study. In *HART'97 (International Workshop on Hybrid and Real-Time Systems)*, volume 1201 of *LNCS*. Springer Verlag, March 1997.

[5] P. S. Duggirala, L. Wang, S. Mitra, M. Viswanathan, and C. A. Muñoz. Temporal precedence checking for switched models and its application to a parallel landing protocol. In *FM'14: Proceedings of the 19th International Symposium on Formal Methods*, pages 215–229. Springer, 2014.

[6] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala. Automatic reachability analysis for nonlinear hybrid models with C2E2. In *CAV'16: Proceedings of the 28th International Conference on Computer Aided Verification, Part I*, pages 531–538. Springer, 2016.

[7] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *CAV'11: Proceedings of the 23rd International Conference on Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer, 2011.

[8] B. Gassmann, F. Oboril, C. Buerkle, S. Liu, S. Yan, M. S. Elli, I. Alvarez, N. Aerrabotu, S. Jaber, P. van Beek, D. Iyer, and J. Weast. Towards standardization of av safety: C++ library for responsibility sensitive safety. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2265–2271, 2019.

[9] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos. Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, MEMOCODE '19, New York, NY, USA, 2019. Association for Computing Machinery.

[10] T. Hunter, T. Moldovan, M. Zaharia, S. Merzgui, J. Ma, M. J. Franklin, P. Abbeel, and A. M. Bayen. Scaling the mobile millennium system in the cloud. In *SOCC'11: Proceedings of the 2nd ACM Symposium on Cloud Computing*, pages 28:1–28:8, New York, NY, USA, 2011. ACM.

[11] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Case study: verifying the safety of an autonomous racing car with a neural network controller. In A. D. Ames, S. A. Seshia, and J. Deshmukh, editors, *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, pages 28:1–28:7. ACM, 2020.

[12] J. Jeannin, K. Ghorbal, Y. Kouskoulas, A. Schmidt, R. W. Gardner, S. Mitsch, and A. Platzer. A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system. *Int. J. Softw. Tools Technol. Transf.*, 19(6):717–741, 2017.

[13] H. S. Kang, J. Y. Lee, S. Choi, H. Kim, J. H. Park, J. Y. Son, B. H. Kim, and S. Do Noh. Smart manufacturing: Past research, present findings, and future directions. *International journal of precision engineering and manufacturing-green technology*, 3(1):111–128, 2016.

[14] P. Koopman, H. Choset, R. Gandhi, B. Krogh, D. Marculescu, P. Narasimhan, J. Paul, R. Rajkumar, D. Siewiorek, A. Smailagic, et al. Undergraduate embedded system education at Carnegie Mellon. *ACM Transactions on Embedded Computing Systems (TECS)*, 4(3):500–528, 2005.

[15] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson. Unmanned aircraft system traffic management (utm) concept of operations. In *AIAA aviation forum*, 2016.

[16] L. Lamport. Real-time model checking is really simple. In *Correct Hardware Design and Verification Methods, 13th IFIP WG (CHARME)*, volume 3725 of *Lecture Notes in Computer Science*, pages 162–175. Springer, 2005.

[17] D. A. Lazar, S. Coogan, and R. Pedarsani. Optimal tolling for heterogeneous traffic networks with mixed autonomy. In *58th IEEE Conference on Decision and Control, CDC 2019, Nice, France, December 11-13, 2019*, pages 4103–4108. IEEE, 2019.

[18] S. M. Loos, A. Platzer, and L. Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In M. J. Butler and W. Schulte, editors, *FM 2011: Formal Methods - 17th International Symposium on Formal Methods, Limerick, Ireland, June 20-24, 2011. Proceedings*, volume 6664 of *Lecture Notes in Computer Science*, pages 42–56. Springer, 2011.

[19] S. Mitra. *A Verification Framework for Hybrid Systems*. PhD thesis, Massachusetts Institute of Technology, September 2007.

[20] P. F. Orzechowski, K. Li, and M. Lauer. Towards responsibility-sensitive safety of automated vehicles with reachable set analysis. *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pages 1–6, 2019.

[21] M. P. Owen, A. Panken, R. Moss, L. Alvarez, and C. Leeper. Acas xu: Integrated collision avoidance and

detect and avoid capability for uas. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pages 1–10, 2019.

[22] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. Srivas. PVS: Combining specification, proof checking, and model checking. In R. Alur and T. A. Henzinger, editors, *CAV'96: Proceedings of the International Conference on Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 411–414. Springer-Verlag, July/August 1996.

[23] N. Ozay. Inter-triggering hybrid automata: a formalism for responsibility-sensitive safety. In A. D. Ames, S. A. Seshia, and J. Deshmukh, editors, *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, pages 32:1–32:2. ACM, 2020.

[24] H. Paulino. Sumo: A framework for prototyping distributed and mobile software. In A. G. Bourgeois and S. Zheng, editors, *Algorithms and Architectures for Parallel Processing, 8th International Conference, ICA3PP 2008, Cyprus, June 9-11, 2008, Proceedings*, volume 5022 of *Lecture Notes in Computer Science*, pages 269–281. Springer, 2008.

[25] R. B. Perry, M. M. Madden, W. Torres-Pomales, and R. W. Butler. *The simplified aircraft-based paired approach with the ALAS alerting algorithm*. Technical Report NASA/TM-2013-217804, NASA, Langley Research Center, 2013.

[26] A. Platzer. Quantified differential dynamic logic for distributed hybrid systems. In A. Dawar and H. Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 469–483. Springer, 2010.

[27] A. Platzer and J.-D. Quesel. KeYmaera: A hybrid theorem prover for hybrid systems (system description). In A. Armando, P. Baumgartner, and G. Dowek, editors, *Automated Reasoning*, pages 171–178. Springer Berlin Heidelberg, 2008.

[28] M. Potok, C.-Y. Chen, S. Mitra, and S. Mohan. Sdcworks: a formal framework for software defined control of smart manufacturing systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 88–97. IEEE, 2018.

[29] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint:1708.06374*, 2017.

[30] X. Sun, H. Khedr, and Y. Shoukry. Formal verification of neural network controlled autonomous systems. In N. Ozay and P. Prabhakar, editors, *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, pages 147–156. ACM, 2019.

[31] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on automatic control*, 38(2):195–207, 1993.

[32] V. Venkataramanan, A. K. Srivastava, A. Hahn, and S. Zonouz. Measuring and enhancing microgrid resiliency against cyber threats. *IEEE Transactions on Industry Applications*, 55(6):6303–6312, 2019.

[33] H. B. Weinberg and N. A. Lynch. Correctness of vehicle control systems—a case study. In *RTSS'96: Proceedings of the 17th IEEE Real-Time Systems Symposium*, pages 62–72. IEEE, December 1996.

[34] J. M. Wing. A symbiotic relationship between formal methods and security. In *Proceedings Computer Security, Dependability, and Assurance: From Needs to Solutions (Cat. No. 98EX358)*, pages 26–38. IEEE, 1998.

[35] T. Wongpiromsarn, S. Mitra, R. M. Murray, and A. Lamperski. Periodically controlled hybrid systems: Verifying a controller for an autonomous vehicle. In R. Majumdar and P. Tabuada, editors, *HSCC'09: Proceedings of the International Workshop on Hybrid Systems: Computation and Control*, volume 5469 of *Lecture Notes in Computer Science*, pages 396–410. Springer, 2009.