Formalizing traffic rules for uncontrolled intersections

Abolfazl Karimi
Department of Computer Science
University of North Carolina at Chapel-Hill
Chapel Hill, United States
ak@cs.unc.edu

Parasara Sridhar Duggirala
Department of Computer Science
University of North Carolina at Chapel-Hill
Chapel Hill, United States
psd@cs.unc.edu

***Abstract*—One of the challenges in designing autonomous vehicles (AV's) is driving around humans (i.e. drivers, cyclists, pedestrians, etc.) In particular, the AV's and the humans must have a common set of traffic rules to follow. In this paper, we present a new approach to formalize and implement traffic rules. We use California's DMV driver handbook as a working example. Our approach provides a straightforward mapping from the rules in the handbook to its formal model, and from the model to its implementation. To demonstrate the efficiency of this approach, we formally model the traffic rules in the logic programming paradigm of Answer Set Programming (ASP) using a programming language called Clingo. We then integrate these rules into CARLA, a virtual test bed environment for autonomous vehicles. We simulate the behavior of autonomous vehicles at four way and three way uncontrolled intersections by correct reasoning of right-of-way rules for autonomous vehicles in real time. As a result, the behaviors of autonomous vehicles under our controller are more realistic compared to CARLA's default FIFO controller. This also improves the throughput of the traffic through the intersection.**

I. Introduction

According to a report by Governors Highway Safety Association [1], "all vehicles on the road will not be autonomous for a very long time, perhaps never. Until then, autonomous vehicles must share the road with vehicles driven by humans," so "the short-run challenge is to manage a world with a mix of driver-operated and autonomous vehicles." One approach to handle the mixture of autonomous vehicles (AVs) and human drivers is to change current rules and traffic infrastructure to incorporate AVs. This approach seems inadequate for at least two reasons: the high cost of new infrastructure and drivers' education, and lack of industry standards in AV technologies since the sector is still in its experimental stages. Another approach is to design AVs such that they abide by the current traffic rules. This calls for a set of rules that are both machine and human understandable, that is, it requires a set of statements in natural language as well as their mathematically precise representation.

Considering the ambiguity and complexity of natural language, the problem is to develop a formal version of traffic rules that is machine-understandable. The challenge is to preserve the human understanding of the rules as much as possible. A formalized version of the rules removes the ambiguities in natural language and makes all the assumptions explicit. This provides the ground truth (of legal behaviour) that can be used in designing AVs, testing and certifying AVs, fault determination in accidents, automating drivers' education, ADAS[1] features, etc.

Currently, self reported disengagement rates and traffic incidents[2] by the AV companies is used as a benchmark for measuring the efficacy of autonomous vehicles. For a safety critical system such as autonomous vehicles, we believe that disengagements and traffic incidents are a very weak form of evidence for safety assurance. A better form of evidence would be all incidents that are violations of traffic rules. Finally, for any safety critical system, one has to establish a rigorous basis for evaluating functional correctness. For all these reasons, we believe that a rigorous mathematical model of traffic rules is an essential part of certification procedures for autonomous vehicles.

In this paper, we present a framework to formalize traffic rules. We use California's driver handbook as a working example. While we focus here on a subset of the rules,[3] the goal is to formalize all of the driver handbook using this framework. We follow a top-down approach to formalize the rules. That is, we first study the driver handbook to identify the logical form of the rules and the concepts that they employ. We model the rules as formal logic sentences using atomic predicates that represent the concepts employed. The concepts referred to by the atomic predicates are modeled using spatio-temporal mathematical objects. The separation of logical form and mathematical concepts provides flexibility to incorporate various types of traffic rules in the driver handbook.

Intersections are the main area where accidents happen. According to the Fatality Analysis Reporting System (FARS) and National Automotive Sampling System-General Estimates System (NASS-GES) data, at least 47% of the estimated 11,275,000 crashes in the United States in 2015 [2] were intersection-related. Furthermore, most of the accidents involving autonomous vehicles also happened at intersections [3], [4]. According to Federal Highway Administration, "unsignalized intersections are of particular concern because they com-

[1] Advanced Driver-Assistance System
[2] https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/testing
[3] i.e. right-of-way rules at uncontrolled intersections

prise the vast majority of intersections in the U.S." [5] The rules for *uncontrolled intersections* (i.e. without "STOP" or "YIELD" signs) apply to all types of conventional unsignalized intersections. Furthermore, in case of a blackout, signalized intersections should be treated as all-way-stops. [6] Therefore, uncontrolled intersection is an interesting case study to demonstrate our framework.

To show the practicality of this approach, we encode the traffic rules in logic programming paradigm using answer set programming (ASP). Using this framework, one can decide whether a given action of a vehicle in a traffic scenario respects all the traffic rules. Further, one can also design a *correct controller* such that all of its actions will respect the traffic rules. We integrate this correct controller into the CARLA simulator and simulate various traffic scenarios at various four-way and three-way intersections. CARLA [7] is an open-source simulator for autonomous driving research. Starting at CVPR 2019 (Computer Vision and Pattern Recognition) conference, CARLA organizes the CARLA Autonomous Driving Challenge [8]. To create a traffic environment, CARLA has a built-in vehicle driver called *autopilot*. For unsignalized intersections, autopilot (not AI or manually driven) vehicles pass the intersection only one at a time. However, our controller implements several rules: first-in-first-out, yield-to-right, yield-to-inside, etc. This makes the traffic more realistic and also increases the traffic throughput.

The ambiguity of the natural language requires that we try different formalizations and study the emergent traffic behaviour from following each version. Our framework facilitates improving the model iteratively by specifying the rules as a declarative logic program. One can easily add conditions to the rules, or add more rules to see the interactions. Furthermore, changing the logic program does not require compiling CARLA's code.

We use Clingo to implement and evaluate the traffic rules. Clingo is an open-source toolchain for answer set programming (ASP) [9]. ASP is a logic programming paradigm. The syntax of Clingo is similar to Prolog but the semantics is different. The SAT-solving techniques used in ASP solvers give them a significant performance advantage over Prolog. Our choice of selecting ASP as the paradigm for modeling traffic rules is a result of several design choices. First, we want the mathematical representation to be fairly easy to understand. Second, we want the model to be transparent, that is, given a statement in the drivers manual provided by the DMV, we would like to map it to a corresponding sentence (or a collection of sentences) in the model. Finally, we want the rules to be modular, that is, the rules can be adopted to other traffic scenarios with minor modifications. As we shall see in the rest of the document, we provide an intuitive, transparent, and modular model for traffic rules using ASP.

We summarize the contributions of this paper as follows:

1) We provide a formalization of traffic rules that is straightforward from the human-understandable informal version to the machine-understandable formal version. This bridges the gap between the high-level human reasoning to the low-level machine instructions.
2) Our formalization and implementation of the rules is efficiently executable. Unreal Engine computes the spatio-temporal atomic predicates, and the Clingo ASP solver evaluates the logical formulas over the atomic predicates.
3) Using a declarative logic program to implement the rules creates great flexibility in iteratively improving the model.

II. Related work

Formalizing and monitoring of traffic rules for autonomous vehicles is studied by Rizaldi et al. [10]. They consider highway driving and safe distance, but we focus on intersections and right-of-way. Our formalization is similar in using a top-down approach. First they "codify" a rule in a temporal logic sentence using abstract atomic predicates and then "concretize" the atomic predicates by giving them a computable definition. Computation of the predicates are done by code generated by Isabelle. However, we codify the rules in a first-order logic sentence and implement them in an ASP logic program, and compute the concretized atomic predicates using Unreal Engine. In [10], given the trajectories of all vehicles, the monitor detects any violations of traffic rules. However, we determine the legal action at any point in time. This gives us a high-level controller. Furthermore, we easily get a monitor by comparing the required action with the action taken.

Safety at intersections in terms of collision freedom is studied by Hilscher et al. [11]. The main difference from our work is that they develop a logic to prove safety of controllers, whereas we formalize the traffic rules to determine the legal action. Another difference is that the rules that we formalize in this paper concern right-of-way, whereas they consider safety in terms of collision freedom. Our concept of *intersection-lane* is inspired by their paper, but our modeling and implementation is different: Our definition readily handles different number of lanes and streets connecting to the intersection and captures the geometry of the intersection more faithfully. In particular, they partition the intersection area into four disjoint regions called "crossing segments" and then define intersection-lanes to be certain subsets of this partition. Two lanes overlap if they share a segment. In our approach, each intersection-lane is a curved lane specified by a spline. Unreal Engine calculates whether two such lanes overlap.

Traffic protocols for safe behavior of autonomous vehicles at intersections were proposed in [12], [13]. Approaches to formally verify such protocols using dynamic logic were developed in [14]. However, these papers focus on the safety predicate (safe separation of vehicles). Unlike our paper, these works do not explicitly consider the high level traffic behaviors of vehicles. Furthermore, such protocols rely on vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) equipment and do not propose a solution where the agents are a mix of autonomous and human drivers.

OpenDRIVE [15] and OpenSCENARIO [16] are standards from Association for Standardization of Automation and

Measuring Systems (ASAM) for describing road networks and traffic scenarios in a tool-independent format. CARLA partially supports OpenSCENARIO. Traffic sequence charts, presented in [17], is a visual specification language for capturing scenarios, that is easy to understand and reason. Another project that provides schematics for describing traffic scenarios is Open Autonomous Safety from Voyage [18]. Extracting formal specification from natural language automatically has been proposed in ARSENAL [19] and VARED [20]. However, these tools only extract specification as LTL formulas [21]. In contrast, we manually translate the rules in the driver handbook to first-order formulas and do not restrict ourselves to temporal logic.

III. Methodology

Since our focus in this paper is formally modeling traffic rules that are applicable at unprotected intersections, we first list the relevant sentences in the drivers' manual and explain their semantics. We then mathematically rewrite these sentences in first order logic and assign rigorous semantics to all the relevant predicates. Finally, we provide some details on converting these first order sentences into statements in Clingo.

Traffic is a multi-agent system. Therefore, each traffic rule determines the *correct* behaviour of only one vehicle, the *ego vehicle*, with respect to other agents. Each vehicle has complete access to its own state but has limited information about its environment. Nevertheless, the rules assume that the ego vehicle has some minimum knowledge about its environment. For example, a vehicle is supposed to know whether it was the first one reaching a four-way-stop or not. However, it may not be supposed to know whether another vehicle's left-turn signal means a left-turn or a U-turn.

In each traffic scenario, the correct behavior (of the ego vehicle) is dependent on three factors: the static features (e.g., lane shapes, lane types, lane overlaps, posted signs, etc), the dynamic features (e.g., arrival at an intersection, turn signals, etc.), and the relevant traffic rules. A traffic rule is a statement about the static and dynamic features of a scenario.

Our formal model of traffic rules hence has three components. The logical form of a traffic rule is modeled as a first-order-logic formula. The predicates in the formula refer to the static and dynamic features of the traffic scenario. The static features are modeled by mathematical concepts such as sets, functions, relations, curves, etc. The dynamic features are modeled by *events*. An *event* indicates that the truth of a mathematical statement transitions from false to true at a specified time. Therefore, each predicate representing an event has a time parameter.

We follow a top-down approach to model the three components. That is, for each rule, first we determine its high-level logical form and the corresponding predicates, then we define each predicate. In the following subsection, we study the California driver handbook and identify the concepts that should be included in our model.

A. The Analysis of Driver Handbook

Before we delve into the traffic rules examples, we note that each phrase in the driver handbook plays a role. These roles fall into three main categories:

1) definition of terms,
2) rule contexts,
3) rules.

A *definition* specifies the meaning of an informal (natural) language term used in the document. A rule *context* specifies the circumstances in which a traffic rule applies. A traffic *rule* specifies the correct behaviors of each vehicle in applicable contexts.

Now we look at a few sentences from the California driver handbook. We focus on the rules that concern the right-of-way at an *uncontrolled intersection*, i.e. an intersection without 'STOP' or 'YIELD' signs [5].

1) *"An intersection is any place where one line of roadway meets another roadway. Intersections include cross streets, side streets, alleys, freeway entrances, and any other location where vehicles traveling on different highways or roads join each other." [6, p. 35]*

   Here we see the semantic definition of the term 'intersection' which is used in the rest of the driver handbook. The definition is followed by some instances of the definition, such as 'cross streets' and 'freeway entrances'.

2) *"At intersections without 'STOP' or 'YIELD' signs, yield to traffic and pedestrians already in the intersection or just entering the intersection." [6, p. 36]*

   First we observe that the phrase "at intersections without 'STOP' or 'YIELD' signs" specifies the context in which the rule applies. The context here is whenever the ego vehicle is *at* an *uncontrolled intersection*. We rewrite the rule (without the context specifier) in the following form which makes the logical form of the sentence more transparent:

   'If vehicle $V1$ is at the intersection and vehicle $V2$ is in the intersection, then $V1$ must yield to $V2$.'

   The rule above introduces the predicates *must yield to*, *at the intersection* and *in the intersection*. Note that for this rule, we can abstract 'in the intersection' and 'just entering the intersection' as the same. (That is, it does not matter whether the whole vehicle or only part of it is in the intersection.) In this rule, $V1$ is the ego vehicle, since the rule requires $V1$ (and only $V1$) to take an action.

3) *"At intersections without 'STOP' or 'YIELD' signs, yield to the vehicle or bicycle that arrives first." [6, p. 36]*

   We rewrite the rule in the following form:

   'If vehicles $V1, V2$ are at the intersection and $V1$ arrived earlier than $V2$, then $V2$ must yield to $V1$.'

   The rule above introduces the temporal event of *arrival at an intersection*. The relation 'arrived earlier than' suggests that the events are sorted by a notion of global

clock. The original form of the rule explicitly presumes that the ego vehicle (i.e. $V2$ in our formulation) is *at* the intersection. However, we can assume that $V1$ is also *at* the intersection since the case where $V1$ is *in* the intersection is covered by Rule (2).

4) *"At intersections without 'STOP' or 'YIELD' signs, yield to the vehicle or bicycle on your right if it reaches the intersection at the same time as you." [6, p. 36]*

   First, we rewrite the rule in the following form:

   "If vehicles $V1, V2$ are at the intersection, $V1$ arrived at the same time as $V2$, and $V1$ is on the right of $V2$, then $V2$ must yield to $V1$."

   The relation *arrived at the same time* is again referring to the global time of the arrival event. This rule introduces '*is on the right of*' as a temporal relation between two vehicles.

5) *"At 'T' intersections without 'STOP' or 'YIELD' signs, yield to traffic and pedestrians on the through road. They have the right-of-way." [6, p. 35]*

   The context of this rule is a T-intersection. The rule introduces the 'through road' type. The *through road* is sometimes referred to as the *major road*. The road attaching to the major road is called the *minor road*. We rewrite the rule as follows:

   "If vehicles $V1, V2$ are at the intersection, $V1$ is on the through road, and $V2$ is not on the through road, then $V2$ must yield to $V1$."

6) *"When crossing or entering city or highway traffic from a full stop, signal, and leave a large enough gap to get up to the speed of other vehicles. You need a gap that is about: Half a block on city streets; A full block on the highway." [6, p. 67]*

   This rule is applicable to uncontrolled T-intersections since a vehicle on the minor road must cross and/or enter the traffic on the major road.[4] We included this rule since it sheds light on the term 'on the through road' in Rule 5. That is, we consider a vehicle *on the through road* only if it is within a certain distance (about half or a full block) from the intersection.

7) The rules 2, 3, 4, and 5, define when a vehicle has a *yield obligation*. However, we still have to define what it means *to yield* as an action. First note that 'yield' has different meanings in different rules. For example,

   - *"A 3-sided red YIELD sign indicates that you must slow down and be ready to stop, if necessary, to let any vehicle, bicyclist, or pedestrian pass before you proceed." [6, p. 31]*
   - *"Do not block the passing lane. Stay out of the far left lane if other traffic wants to drive faster, and yield to the right for any vehicle that wants to pass." [6, p. 84]*
   - *"When 2 vehicles meet on a steep road where neither vehicle can pass, the vehicle facing downhill must yield the right-of-way by backing up until the vehicle going uphill can pass." [6, p. 37]*

   In the case of uncontrolled intersections, the handbook does not explicitly define the yield action. Following the meaning of 'YIELD' sign, we interpret the *yield action* either as *slow down* or *stop*, depending on the context.

We argue that for each context, we must formalize the handbook separately:

First, a term may have different meanings in different contexts. For example, in Item 7, we observed that 'yield' has several meanings. Therefore, one cannot first decide the meaning of a term and then use it in irrelevant contexts.

Second, whether a rule applies to a context depends on the other rules for that context. This is because some rules override others. In particular, if a rule for a special type of intersection is inconsistent with a rule for all intersections, then the specialized rule overrides the generic one. For example, Rule 5 overrides Rule 4: In a T-intersection, the minor road is on the right of (one side of) the major road, but the traffic on the major road has the right of way. Therefore, one should first pick a context, then formalize the applicable rules.

To formalize the whole handbook for a context, one may try an iterative approach. At each iteration, more rules are added to the context while maintaining consistency. Consistency may require to redefine a previous rule. In this paper, we take the first step by formalizing a few rules for two contexts, a generic uncontrolled intersection, and an uncontrolled T-intersection.

IV. Modeling uncontrolled intersections

Each rule is modeled with a first-order formula. A formula is a combination of atomic formulas, logical connectives and quantifiers. We identify some subformulas that have an intuitive intended meaning as *auxiliary predicates*. This is to simplify the expression of long formulas.

Atomic formulas represent either *events* or *static facts*. The events and static facts are defined in terms of mathematical concepts, such as shapes, curves, sets, intersection, union, collision (the first time two sets intersect), etc. An *event* is represented by a predicate with time as one of the arguments. A *static fact* is represented by a predicate without time as an argument.

We use Herbrand semantics to interpret the formulas [22]. A traffic scenario is described by a finite collection of ground atomic formulas which represent static facts and events. Therefor, traffic rules are grounded on traffic scenarios. This choice of semantics provides a more intuitive interpretation (than a Tarskian semantics.) Furthermore, implementation becomes straightforward using Answer Set Programming.

In modeling traffic rules, we avoid using function symbols. Therefore, each Herbrand model (of some traffic rules and a traffic scenario) is finite. For example, the domain of a

[4] To turn right, the vehicle must *enter* the traffic approaching from left. To turn left, the vehicle must *cross* the traffic approaching from left, and *enter* the traffic approaching from right.

quantifier on a time variable T is the set of all timestamps observed in the traffic scenario.

A. Traffic Rules

In this section, we model the logical form of the traffic rules. In the fragment of handbook that we analyzed, we identified three rules: Rule 2, 3 and 4. Here we present the formula for each rule using the predicates that were introduced in the analysis.

1) Rule (2):

$$mustYieldToForRule(V1, V2, yieldToInside) \leftrightarrow \Big(atTheIntersection(V1) \wedge inTheIntersection(V2)\Big)$$

In our model of the yield obligation, we keep track of who must yield to whom, and an identifier for the corresponding rule (e.g. $yieldToInside$). The identifier accounts for the fact that the mapping from the yield obligations to the actions depends on the rule corresponding to the obligation. This will become clear in the definition of $mustStopToYield$.

2) Rule (3):

$$mustYieldToForRule(V2, V1, firstInFirstOut) \leftrightarrow \Big(atTheIntersection(V1) \wedge atTheIntersection(V2) \wedge arrivedEarlierThan(V1, V2)\Big)$$

3) Rule (4):

$$mustYieldToForRule(V1, V2, yieldToRight) \leftrightarrow \Big(atTheIntersection(V1) \wedge atTheIntersection(V2) \wedge arrivedSameTime(V1, V2) \wedge isOnRightOf(V2, V1)\Big)$$

4) The yield action: In the case of an uncontrolled intersection, there is no explicit definition of the yield action in the driver handbook. We interpret the *yield* action to be *stopping and waiting for a certain amount of time*. The duration of wait depends on the rule corresponding to the yield obligation. For rules $firstInFirstOut$ and $yieldToRight$, we define the duration to be until the other vehicle enters the intersection; thus the duration is the same as of the yield obligation. However, for rule $yieldToInside$, the duration depends on a notion of lane *reservation*. The predicate $mustStopToYield$ defines the yield action:

$$mustStopToYield(V1) \leftrightarrow \exists V2\Bigg(\exists L\Big(mustYieldToForRule(V1, V2, yieldToInside) \wedge requestedLane(V1, L) \wedge reservedLane(V2, L)\Big) \vee mustYieldToForRule(V1, V2, firstInFirstOut) \vee mustYieldToForRule(V1, V2, yieldToRight)\Bigg).$$

B. Auxiliary Predicates

1) At the intersection:

$$atTheIntersection(V) \leftrightarrow \Big(arrived(V) \wedge \neg entered(V)\Big)$$

where

$$arrived(V) \leftrightarrow (\exists F)(\exists T) arrivedAtForkAtTime(V, F, T)$$

and

$$entered(V) \leftrightarrow (\exists F)(\exists T) enteredForkAtTime(V, F, T).$$

The predicates $arrivedAtForkAtTime$ and $enteredForkAtTime$ represent the events *arrival at an intersection* and *entering an intersection*, respectively. The time parameter T in these predicates is called a *time stamp*.

2) In the intersection:

$$inTheIntersection(V) \leftrightarrow \Big(entered(V) \wedge \neg exited(V)\Big)$$

where

$$exited(V) \leftrightarrow (\exists E)(\exists T) exitedFromAtTime(V, E, T).$$

3) Arrival precedence:

$$arrivedEarlierThan(V1, V2) \leftrightarrow (\exists T1)(\exists T2)\Big(arrivedAtTime(V1, T1) \wedge arrivedAtTime(V2, T2) \wedge T1 < T2\Big).$$

where

$$arrivedAtTime(V, T) \leftrightarrow (\exists F) arrivedAtForkAtTime(V, F, T).$$

The $arrivedSameTime$ is similar, except '$=$' instead of '$<$'.

4) On-the-right-of relation for vehicles:

$$isOnRightOf(V1, V2) \leftrightarrow (\exists F1)(\exists F2)(\exists T1)(\exists T2)\Big(arrivedAtForkAtTime(V1, F1, T1) \wedge arrivedAtForkAtTime(V2, F2, T2) \wedge isOnRightOf(F1, F2)\Big).$$

The latter $isOnRightOf$ is a static fact between two forks.

5) Lane request: At arriving at an intersection, a vehicle must communicate (to other vehicles) its path through the intersection.[5] The turn signal is used to communicate that. We say a vehicle *requests* an intersection-lane using its turn signal.[6]

An *intersection lane* is a lane that connects an incoming lane of the intersection to an outgoing lane. A *fork* is the collection of all intersection lanes from an incoming lane. Therefore, an incoming lane uniquely identifies a fork, and vice versa.

Each intersection-lane determines which turn signal must be used if a vehicle wants to pass through the intersection using that lane. That signal is called the *correct signal* for the lane.

[5] In fact, a vehicle must signal starting at 100 feet from the intersection [6, p. 61]

[6] Note that due to the limited communication capacity of a turn signal, more than one lane may be requested. For example, a left-turn signal could mean a U-turn, or a left-turn to a crossing street.

Therefore, a lane L is requested by a vehicle V, if V signaled the correct signal for L when arriving at L's fork F:

$$requestedLane(V, L) \leftrightarrow (\exists S)(\exists F)\Big(signaledAtFork(V, S, F) \wedge branchOf(L, F) \wedge laneCorrectSignal(L, S)\Big).$$

where

$$signaledAtFork(V, S, F) \leftrightarrow (\exists T)signaledAtForkAtTime(V, S, F, T)$$

and

$$branchOf(L, F) \leftrightarrow (\exists E)laneFromTo(L, F, E).$$

6) Lane reservation: We say that a vehicle inside the intersection *reserves* a lane L, to indicate that it started passing through the intersection and its path (may) pass through (portions of) L, so it is not safe for other vehicles to use L. We assume that the reserving vehicle's path through the intersection will be consistent with their requested lane(s).[7] A vehicle V *reserves* a lane $L1$ if

1) V requested $L1$ and V is on $L1$, or if
2) V requested $L2$, V is on $L2$, the lanes overlap (i.e. intersect), and V has not left the overlapping lane $L1$ yet.

Formally:

$$reservedLane(V, L1) \leftrightarrow \Bigg(\Big(requestedLane(V, L1) \wedge isOnLane(V, L1)\Big) \vee \exists L2\Big(requestedLane(V, L2) \wedge isOnLane(V, L2) \wedge overlaps(L2, L1) \wedge \neg leftTheLane(V, L1)\Big)\Bigg).$$

where

$$leftTheLane(V, L) \leftrightarrow (\exists T)leftLaneAtTime(V, L, T).$$

The predicate $leftTheLane(V, L)$ indicates that vehicle V entered lane L and then left L. For most intersection geometries, this implies that V will not enter L again. For example, observe the pair of intersecting lanes in Figure 1. If a vehicle passes the intersection through one of these lanes, it will enter and leave the other lane only once.[7]

C. Events

1) Time stamp: A *time stamp* is a numerical representation of the global clock. In real time, one event may strictly precede another but the time difference be imperceptible to humans. Hence, it is reasonable to assume that real time events that are closer (in time) than a threshold, happened at the same time. Therefore, we adopt a discrete view of time. That is, a timestamp is simply the number of time steps since the beginning of a scenario. The length of a time step is a hyperparameter to our model.

[7]See §VIII for more discussion on this.

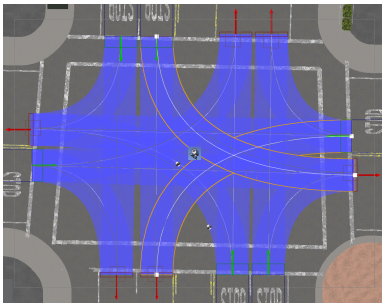


Fig. 1: Intersecting lanes.

2) Arrival event: The predicate $arrivedAtForkAtTime(V, F, T)$ represents the event of vehicle V arriving at the intersection from fork F at time T. The *arrival time* is when the vehicle intersects with the *arrival box*.

3) Turn signal event: The predicate $signaledAtForkAtTime(V, S, F, T)$ represents the event of vehicle V using turn signal S when it arrives at fork F at time T.

4) Entrance event: The predicate $enteredForkAtTime(V, F, T)$ represents the event of vehicle V entering (branches of) fork F at time T. An *entrance box* represents the border between the incoming lane and the branches of F.

5) Lane leave event: The predicate $leftLaneAtTime(V, L, T)$ represents the event of vehicle V leaving lane L completely at time T. That is, at time T, intersection of V and L transitions from nonempty to empty.

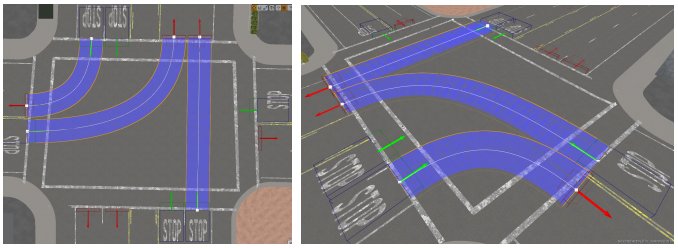
6) Exit event: The predicate $exitedFromAtTime(V, E, T)$ represents the event of vehicle V leaving the intersection from the *exit* E at time T. The time T is when the vehicle's body stops intersecting the exit E's extent box. The *extent box* is simply a box-shaped subset of the 3D space.

D. Static predicates

In this section we define the static predicates identified in the traffic rules. The arguments of the predicates are traffic-related objects such as lane, turn signal, etc.

1) Intersection lane: An *intersection lane* is a lane that connects an incoming lane to an outgoing lane of the intersection. A *lane* is a tube-shaped volume that is curved along its length and has a rectangular cross section. The *center* of the lane is a spline curve along the length of the tube. The left and right boundaries of an intersection lane are offsets of the center by half of the lane width. The *width* of a lane changes from the width of the incoming lane to the width of the outgoing lane. This change is linear with respect to the length of the center curve from the incoming lane. The bottom of the lane aligns on the pavement. In Figure 2 we see an example of left-turn, right-turn, and no-turn intersection lanes.

2) Fork: A *fork* is the set of all intersection-lanes that start from the same incoming lane. Each intersection-lane is called a *branch* of its fork. Since there is a one-to-one correspondence



(a) Top view (b) Perspective view

Fig. 2: Examples of intersection-lane geometry.

between forks and incoming lanes, we can identify a fork with an incoming lane or vice versa.

3) Lane overlaps: The predicate $overlaps$ holds for a pair of intersection lanes if their regions intersect. Therefore, it defines a symmetric relation.

4) Is-on-right-of relation for forks: We define $isOnRightOf(F2, F1)$ for two forks, based on their corresponding incoming lanes. The relation $isOnRightOf$ must be irreflexive and anti-symmetric.[8] The driver handbook does not give a clear definition. We define this relation based on angles between vectors. The direction of an incoming lane defines a 2D vector on the plane (i.e. the pavement). We say $F2$ is *on the right of* $F1$ if the angle of $F2$ relative to $F1$, measured counterclockwise, is more than 30 and less than 150 degrees.[9] The direction *counterclockwise* is with respect to a downward view of the intersection from above.

For example, consider Figure 3 (b).[10] It shows the incoming lanes and the angles between consecutive lanes. Then, as expected intuitively, the predicate $isOnRightOf$ holds for the following pairs: $(East, South)$, $(North, East)$, $(West, North)$, and $(South, West)$. Under our definition, more than one leg of an intersection may be on the right of another leg. For example, consider Figure 3 (c).[11] The F Street and Pico Way are 50 degrees and 140 degrees from south side of 46th Street, counterclockwise. Hence, both of them are on the right of (the south side of) 46th Street.

If the angle from $F1$ to $F2$ is in the interval $[0, 30]$ or $[330, 360)$, then the incoming lanes are considered heading the same direction, i.e. on the same street. If the angle is in $[150, 210]$ we say $F2$ is *in front of* $F1$ i.e. vehicles on $F2$ are *oncoming traffic* relative to $F1$. If the angle is in $(210, 330)$, then $F2$ is on the left of $F1$, i.e. $F1$ is on the right of $F2$.

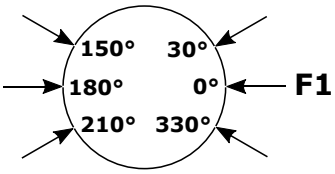
5) Arrival box: The *arrival box* is a box on an incoming lane that starts at a pre-specified distance from the intersection

[8]That is, no fork is to the right of itself; and if $F2$ is on the right of $F1$ then $F1$ is not on the right of $F2$.

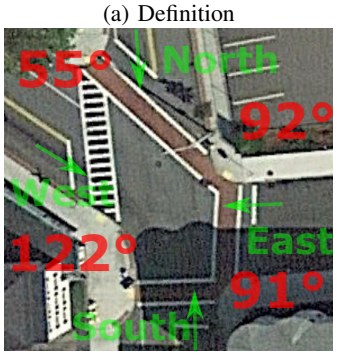
[9]The Federal Highway Administration recommends that "in the design of new facilities or redesign of existing facilities where right-of-way is restricted, intersecting roadways should meet at an angle of not less than 75 degrees." [23] However, as we see in Figure 3 (c), the north side of 46th St is 40 degrees to the right of Pico Way. Therefore, we choose a conservative infimum of 30 degrees.

[10]Intersection of Harrison Street and Providence Street, Worcester, Massachusetts.

[11]Intersection of 46sth St, F St, and Pico Way, Sacramento, California.



(a) Definition



(b) Example.

(c) Multiplicity

Fig. 3: Is-on-right-of relation for forks.

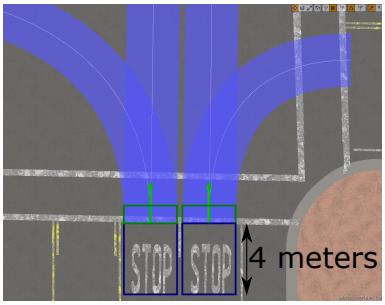


Fig. 4: Arrival box.

(i.e. before the end of the incoming lane) and ends at the border between the incoming lane and the intersection. This distance is the *length* of the box. The *width* of the box is the width of the arriving lane. The length of the box is a hyperparameter. For example, in Figure 4, the dark-blue boxes at the crosswalk lines, designate the boundaries of the arrival box. In this example, the length of the box is 4 meters. When a vehicle overlaps an arrival box, an arrival event is generated.

6) Entrance box: This box is used to generate the entrance event. It marks the boundary between an incoming lane and the intersection area. Each fork has an entrance box.

7) Lane-from-to: The predicate $laneFromTo(L, F, E)$ indicates that lane L starts at fork F and ends at exit E. Therefore, this predicate specifies the possible routes through the intersection.

V. Modeling T-intersections

Most of our modeling for an uncontrolled intersection can be used for an uncontrolled T-intersection. The additions and alterations needed are as follows.

A. Traffic Rules

1) Rule (5):

$$mustYieldToForRule(V1, V2, throughRoadFirst) \leftrightarrow$$

$$\Big(atTheIntersection(V1) \wedge \\ vehicleOnThroughRoad(V2) \wedge \\ \neg vehicleOnThroughRoad(V1)\Big).$$

2) Yield action for traffic on minor road:

$$mustStopToYield(V1) \leftrightarrow (\exists V2)(\exists L1)(\exists L2)\Big(\\ mustYieldToForRule(V1, V2, throughRoadFirst) \wedge \\ requestedLane(V1, L1) \wedge requestedLane(V2, L2) \wedge \\ overlaps(L1, L2) \wedge \neg leftTheLane(V2, L1)\Big).$$

3) Rule (2): For T-intersections, we need a different model of the yield action for this rule. This is because a vehicle that arrives from the through road and is not turning, need not stop for a vehicle in front of it that is going the same direction. We choose to skip presenting this nuance and refer the reader to the implementation file provided on the web.[12]

B. Auxiliary Predicates

1) Traffic on through road: Recall that we consider a vehicle *on through road* only if it has reached within a certain distance from the intersection. This is similar to the arrival event for generic uncontrolled intersections which is generated when a vehicle reaches to a distance from the intersection. Therefore, we reuse the arrival event to determine whether a vehicle is on the through road:

$$vehicleOnThroughRoad(V) \leftrightarrow \\ (\exists F)(\exists T)\Big(arrivedAtForkAtTime(V, F, T) \wedge \\ forkOnThroughRoad(F) \wedge \neg exited(V)\Big).$$

where

$$forkOnThroughRoad(F) \leftrightarrow (\exists L)(\exists E)\Big(\\ laneFromTo(L, F, E) \wedge laneCorrectSignal(L, off)\Big).$$

VI. Implementation

In this section we briefly explain how to implement our model of the rules as an ASP program in Clingo. The logic programs are made available on the web[12]. It is interesting to note that the relevant rules for unprotected intersections can be specified in approximately 150 lines of code.

An ASP program is a collection of rules of the form:

```
h :- p_1, ..., p_n, not p_{n+1}, ..., not p_{n+m}.
```

The intended meaning of the above sentence is

$$p_1 \wedge \cdots \wedge p_n \wedge \neg p_{n+1} \wedge \cdots \wedge \neg p_{n+m} \rightarrow h$$

A. Conjunctive condition

1) Positive conjuncts: Rule (2) is straightforwardly implemented as:

```
mustYieldToForRule(V1, V2, yieldToInside):-
  atTheIntersection(V1),
  inTheIntersection(V2).
```

[12]https://tinyurl.com/y4lkrn6g

2) Positive and negative conjuncts:

```
atTheIntersection(V):-
  arrived(V),
  not entered(V).
```

B. Existential Quantifier

If we have an existential quantifier over variables ($T1$ and $T2$ here) that only appear in positive conjuncts:

```
arrivedEarlierThan(V1, V2):-
  arrivedAtTime(V1, T1),
  arrivedAtTime(V2, T2),
  T1 < T2.
```

If the existentially quantified variable is referenced only once, we do not need to name it. An underscore represents a nameless variable. For example,

```
arrived(V):-
  arrivedAtForkAtTime(V, _, _).
```

C. Disjunctive condition

If the definition of a predicate is a disjunction of two subformulas, then the definition can be broken down into two definitions where each subformula is the definition.

For example, consider the definition of $mustStopToYield$ for uncontrolled intersections. The condition can be written in a disjunctive form, by distributing the existential quantifier over disjunctions:

$$mustStopToYield(V1) \leftrightarrow \Bigg(\\ (\exists V2)(\exists L)\Big(mustYieldToForRule(V1, V2, yieldToInside) \\ \wedge requestedLane(V1, L) \wedge reservedLane(V2, L)\Big) \\ \vee \\ \exists V2\Big(mustYieldToForRule(V1, V2, firstInFirstOut)\Big) \\ \vee \\ \exists V2\Big(mustYieldToForRule(V1, V2, yieldToRight)\Big)\Bigg).$$

Then we can implement $mustStopToYield$ by the following three rules:

```
mustStopToYield(V1):-
  mustYieldToForRule(V1, V2, yieldToInside),
  requestedLane(V1, L),
  reservedLane(V2, L).

mustStopToYield(V):-
  mustYieldToForRule(V, _, firstInFirstOut).

mustStopToYield(V):-
  mustYieldToForRule(V, _, yieldToRight).
```

The traffic objects, i.e. vehicles, lanes, arrival boxes, entrance boxes and exit boxes, are implemented as polygon meshes in Unreal Engine. We use Unreal Engine's built-in collision checking to determine whether two objects overlap. This enables an efficient and scalable computation of atomic predicates. If both objects are static, we query the collision checking only once at the beginning of a scenario. Otherwise, we ask Unreal Engine to generate collision events when a dynamic object begins or ends overlapping with other traffic

objects. Unreal Engine's game time is used as the global time. We discretize this time to timestamp the events.

VII. Applications

A. Detecting violations of traffic rules

In the previous section, we used the formalized rules to determine the correct action obligated by the right-of-way rules. A *violation* happens when a vehicle takes an action inconsistent with the rules. Similar to traffic rules, there can be several ways to formalize the notion of violation. One approach is to use the same predicates (used in traffic rules) to define a violation.

For example, in our case study of uncontrolled intersections, $mustStopToYield$ at the intersection is the obligated action. Therefore, we formulate violations as entering the intersection while a $mustStopToYield$ is required: Vehicle V violates a rule at time $T+1$ if and only if

$$P_T \models mustStopToYield(V) \ \wedge \ P_{T+1} \models entered(V)$$

where P_T is the logic program at time T.

B. Controlling CARLA's Autopilot vehicles

Using the predicate $mustSlowToYield$ and $mustStopToYield$ we can determine the correct action for an AV at a given point in time. We only need to evaluate the ASP when a new event happens. Each intersection has a *monitor* that tracks the events. When a vehicle arrives at the intersection, the monitor starts tracking the events of that vehicle until the vehicle exits the intersection. When a vehicle exits the intersection, all of its events are removed from the monitor.[13] Since each intersection has finite space, each monitor has a bound on the number of vehicles in its event list. Assuming that each vehicle only moves forward along its path, each vehicle generates a bounded number of events from arrival to exit. Therefore, for each intersection, there is a bound on the size of the event list, hence a bound on the size of the logic program.

To test our model, we used CARLA (version 0.9.6) on an Ubuntu 18.04 on a laptop with Intel Core i7-8750H processor, 32GB RAM, and NVIDIA GeForce GTX 1070 Max-Q graphics card.

We deployed our traffic controller on an uncontrolled multi-lane four-way intersection in a virtual town called Town05 in CARLA. With 120 vehicles (sedans, trucks and motorcycles) on the map, the simulator runs around 40 frames per second. We also deployed our traffic control on various uncontrolled intersections of other types in different maps: single-lane and multi-lane three-way intersection, single-lane four-way intersection, and single-lane and multi-lane T-intersection. A few videos of the simulations are made available on the web.[12]

Here we point out a few of the scenarios that are simulated and are in the videos:

[13]If a vehicle is not present in the context of a rule, then it is irrelevant to the rule. For example, if the ego vehicle is at an intersection, vehicles that have not arrived at that intersection nor are in the intersection are irrelevant.

1) At a four-way intersection, $V1$ arrive at the same time as $V2$ from two different lanes of the same street. According to our definition of $isOnRightOf$, none of the vehicles is on-the-right-of the other, hence neither need to yield to the other (based on the implemented rules.) See the beginning of (the video of) Scenario 1. Here, $V1$ and $V2$ are the vehicles from the south side of the intersection.
2) At a four-way intersection, $V1$ arrives at the intersection when $V2$ is in the intersection but their intended paths do not intersect. According to Rule 2, $V1$ must yield to $V2$. However $V1$ need not stop for $V2$, based on our definition of the yield action. See the beginning of Scenario 1. Here $V1$ is the vehicle from north, and $V2$ is either of the vehicles from south.
3) At a four-way intersection, $V1$ arrives at the intersection when $V2$ is in the intersection and the path of $V2$ intersects $V1$'s requested lane. Hence $V1$ must wait until $V2$ leaves $V1$'s requested lane. See the beginning of Scenario 1. Here, $V1$ is the vehicle on the east side of the intersection, and $V2$ is the vehicle from left lane of the south side.
4) In Scenario 4, we have a single-lane three-way intersection so each incoming lane is either on the right or on the left of another.
5) In a T-intersection, $V1$ arrives from the minor road while $V2$ is on the through road. Hence $V1$ must stop for $V2$, according to Rule 5. Furthermore, before $V2$ exits, $V3$ arrives from the through road. Then $V1$ must wait for $V3$ as well. Therefore, $V1$ waits until there is no vehicle on the through road. See Scenario 5.
6) In Scenario 6, we have a T-intersection where the minor road is not perpendicular to the major road.

The behavior of all of the vehicles in the videos is completely decided by the traffic controller that respects all the traffic rules and is generated by ASP. For each of these videos, the events generated by the vehicles entering and exiting the intersection, and the reasoning being performed for arriving at the correct action for each vehicle is being displayed on the right side and the corresponding actions of each of the autonomous vehicle can be seen on the left side of the video. *To the best of authors knowledge, this is the first instance where the actions of autonomous vehicles in a simulated environment are synthesized based on the rules that they need to obey.*

VIII. Limitations

A. Traffic flow assumptions

In our model, we presupposed some regularity assumptions which make the definition of rules simpler. Therefore, our model is applicable to a traffic scenario only if the assumptions are satisfied. A behaviour that violates such regularity assumptions is more complicated or unpredictable.

For example, we assumed that vehicles' paths through the intersection will be consistent with their turn signal. This

assumption is true for CARLA's autopilot vehicles.[14] However, AI or manually driven vehicles may violate this assumption, say by not using their signal when turning. Another example is the assumption that vehicles move only forward along a requested lane and do not backup in the intersection.

To tame the complexity of irregular behaviours, the driver handbook has the following provision:

> "Never assume other drivers will give you the right-of-way. Yield your right-of-way when it helps to prevent collisions." [6, p. 34]

CARLA's autopilot has a basic collision-avoidance system based on forward free distance.

Another remedy would be to encode the assumptions as a logic program and use the standard definitions only if the program has a solution. That is, the logic program defines a guard on validity of the standard definitions.

B. Nontrivial lane intersections

In §IV, we assumed that a vehicle enters and leaves an overlapping lane at most once. If this assumption does not hold for a particular intersection, one has to replace that predicate with a predicate that is true only when the vehicle has left the last intersecting part of the intersecting lane. The latter predicate would be slightly more complicated to implement in Unreal Engine, since one has to keep track of all the pieces of the overlap between two lanes.

IX. Future work

In this work, we evaluated our model by developing a traffic controller and manually observing the intuitive quality of several traffic flows that follow the rules. However, we need a more rigorous and automated method to evaluate a formal model of traffic rules.

One framework to quantify *accuracy* is with respect to *false positives* and *false negatives*. For our model, a *false positive* is a scenario in which our model deduces a $mustStopToYield$ while the driver handbook did not intend so. A *false negative* is a scenario in which our model does not deduce a $mustStopToYield$ while the driver handbook did intend so. The challenge here is to determine the *true intention* of the driver handbook in each traffic scenario. The legal authorities (legislature, attorneys, police, transportation engineers, etc.) are the reference for the *true* meaning of the rules.

Acknowledgment

We thank anonymous reviews who provided valuable feedback on earlier drafts of this paper. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-19-1-0288 and National Science Foundation (NSF) under grant numbers CNS 1739936, 1935724. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force or National Science Foundation.

[14] For autopilot vehicles, upon approaching an intersection, a predefined route is randomly assigned and the corresponding turn signal and waypoints (for the vehicle controller) are determined.

References

[1] J. Hedlund, "Autonomous vehicles meet human drivers: Traffic safety issues for states," 2017, https://www.ghsa.org/sites/default/files/2017-01/AV%202017%20-%20FINAL.pdf.

[2] NHTSA, "Traffic Safety Facts 2015," *Report No. DOT HS 812 384*, 2017, https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812384.

[3] "Report of traffic accident involving an autonomous vehicle (OL 316)," 2017, https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/autonomousveh_ol316 Accessed: July 2017.

[4] "National highway traffic safety administration: Investigation pe 16-007 report," 2017, https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF Accessed: July 2017.

[5] Federal Highway Administration, "Unsignalized Intersections," 2019, https://safety.fhwa.dot.gov/intersection/conventional/unsignalized/.

[6] California Department of Motor Vehicles, "Driver handbook," 2019, https://www.dmv.ca.gov/web/eng_pdf/dl600.pdf.

[7] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[8] CARLA, "CARLA Autonomous Driving Challenge," 2019, https://carlachallenge.org/.

[9] M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. Schneider, "Potassco: The Potsdam Answer Set Solving Collection," *AI Communications*, vol. 24, no. 2, pp. 107–124, 2011.

[10] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow, "Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL," in *International Conference on Integrated Formal Methods*. Springer, 2017, pp. 50–66.

[11] M. Hilscher and M. Schwammberger, "An abstract model for proving safety of autonomous urban traffic," in *International Colloquium on Theoretical Aspects of Computing*. Springer, 2016, pp. 274–292.

[12] S. R. Azimi, G. Bhatia, R. R. Rajkumar, and P. Mudalige, "Vehicular networks for collision avoidance at intersections," *SAE International Journal of Passenger Cars-Mechanical Systems*, vol. 4, no. 2011-01-0573, pp. 406–416, 2011.

[13] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1162–1175, 2013.

[14] S. M. Loos and A. Platzer, "Safe intersections: At the crossing of hybrid systems and verification," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 1181–1186.

[15] ASAM OpenDRIVE, 2019, https://www.asam.net/standards/detail/opendrive/.

[16] ASAM OpenScenario, 2019, https://www.asam.net/standards/detail/openscenario/.

[17] W. Damm, S. Kemper, E. Möhlmann, T. Peikenkamp, and A. Rakow, "Using traffic sequence charts for the development of HAVs," *Embedded Real Time Software and Systems*, 2018.

[18] Voyage, 2018, https://oas.voyage.auto/.

[19] S. Ghosh, N. Shankar, P. Lincoln, D. Elenius, W. Li, and W. Steiener, "Automatic requirements specification extraction from natural language (ARSENAL)," SRI INTERNATIONAL MENLO PARK CA, Tech. Rep., 2014.

[20] J. Badger, D. Throop, and C. Claunch, "VARED: Verification and Analysis of Requirements and Early Designs," in *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*. IEEE, 2014, pp. 325–326.

[21] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE, 1977, pp. 46–57.

[22] Michael Genesereth and Eric Kao, "Herbrand Semantics," 2019, http://logic.stanford.edu/herbrand/herbrand.html.

[23] Federal Highway Administration Research and Technology, "Highway Design Handbook for Older Drivers and Pedestrians," 2001, https://www.fhwa.dot.gov/publications/research/safety/humanfac/01103/chp1rec.cfm#a.