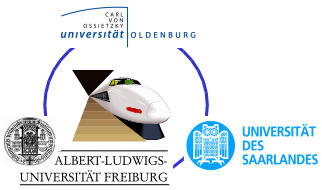


AVACS – Automatic Verification and Analysis of Complex Systems

REPORTS

of SFB/TR 14 AVACS

Editors: Board of SFB/TR 14 AVACS

A Lane Change Assistance System: Cooperation and Hybrid Control

by

Boris Wirtz Tim Strazny Astrid Rakow Jan Rakow

AVACS Technical Report No. 78
July 2011
ISSN: 1860-9821

Publisher: Sonderforschungsbereich/Transregio 14 AVACS
(Automatic Verification and Analysis of Complex Systems)
Editors: Bernd Becker, Werner Damm, Bernd Finkbeiner, Martin Fränzle,
Ernst-Rüdiger Olderog, Andreas Podelski
ATRs (AVACS Technical Reports) are freely downloadable from `www.avacs.org`

Author(s) contact: Boris Wirtz (`boris.wirtz@informatik.uni-oldenburg.de`).

A Lane Change Assistance System: Cooperation and Hybrid Control

Boris Wirtz Tim Strazny Astrid Rakow Jan Rakow

July 21, 2011

Abstract

Automated Highway Systems (AHS's) are considered as a key technology that promises increased safety, reduced energy consumption and optimized traffic flow. Safe and dependable operation of AHS's is of paramount importance and requires the application of rigid formal methods at design time. In this report we present a model for a lane change assistance system which is meant to serve as a foundation for benchmarks boosting theoretic and algorithmic advances in formal verification of the challenging class of cyber-physical systems. The assistance system implements an autonomous lane change manoeuvre conducted in cooperation with other communicating agents. The model implements a layered design for traffic agents where aspects of communication and autonomous control are described as real-time and hybrid systems, respectively, which are intertwined by synchronous message passing.

Contents

1 Introduction

Automated Highway Systems (AHS's) are considered a key technology that promises increased safety, reduced energy consumption and optimized traffic flow [9]. Consider for example the prominent application of car platooning [4, 9], where vehicles drive organized in densely spaced platoons of limited length. By the deployment of autonomously driving communicating vehicles sensor-actuator response times can be reduced dramatically so that the slipstream of vehicles driving in front can be utilized. Low relative velocities and maintenance of safe distances within platoons attenuate the impact of inner-platoon accidents and together yield an increase in safety. Routing and formation of platoons take into account events such as obstacles, road works or rush hours, thus ensuring an optimized traffic flow.

Intelligent transportation systems in general are large scale distributed systems that expose characteristics of both hybrid systems and dynamic communication systems. The design of such typically safety critical systems is challenging for several reasons: The overall system is composed of an unbounded and varying number of autonomous traffic agents that dynamically detect other nearby agents or roadside equipment, respectively. By this, not only the number of agents is variable but also the communication topology is changing over time.
Autonomous manoeuvres such as lane changes (which may be either conducted as an outcome of negotiations with other agents or roadside equipment), and persistent services such as following a car or staying on a lane, necessitate sophisticated controllers that not only ensure safe travel but also comfort to the passengers. Fully operational AHS's are widely considered to become the final result of an evolutionary process integrating assistance systems to more and more complex Advanced Driver Assistance Systems (ADAS's). Since these systems are intricate and safety critical, the application of formal methods becomes indispensable.

In this report we present a formal model of a *Lane Change Assistance System* (LCAS) that obeys layered design principles [9, 5]. Our modelling efforts aim to build a multi-layered hybrid system model of a relevant domain, to study the characteristics of industrial hybrid systems and thereby to provide a realistic model for benchmarks of formal verification techniques.

Model Overview The basic structure of the model presented in this report is depicted in Fig. 1. The inherent complexity of AHS system development necessitates layered design approaches to hide complexity of lower level layers by providing abstract interfaces to the higher layers [9]. So a vehicle controller is conceptually divided up into an *autonomous layer* (`AUT`) and a *protocol layer* (`PROT`) and a *communication layer* (`COM`). The autonomous layer (`AUT`) provides basic vehicle control and sensor access and by this the ability to actually *conduct* manoeuvres such as following another car in safe distance or approaching a reference line. The protocol layer coordinates manoeuvres—such as a lane change—among several agents. The communication layer provides services for inter-agent communication.

In systems performing more than one manoeuvre to accomplish a targeted goal, a further layer, sometimes called *deliberative* [5] layer, is added to the system architecture to coordinate the different manoeuvres.

The key idea for the lane change protocol is that a car which intends to change into a neighbouring lane, asks vehicles on the target lane to help, i.e. establish and maintain a safety distance to it for a certain amount of time. Then the initiator will enter the lane in-between the helper and the helper's car ahead. While the helper ensures a safety distance to the initiator, the initiator itself

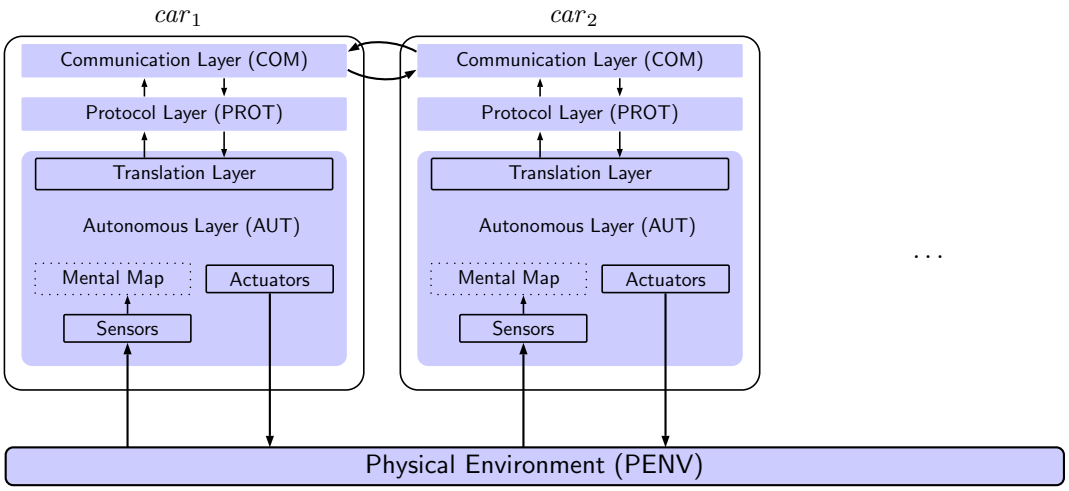


Figure 1: Model Architecture.

will maintain a safety distance to its car ahead—if any—and it will establish and maintain a safety distance to the helpers car ahead—if any.

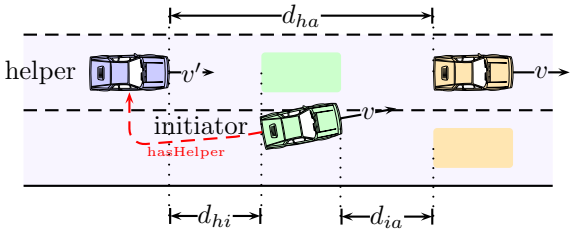


Figure 2: The Lane Change Manoeuvre. The initiator will enter the target lane between its helper and the helper's car ahead. The lane change can proceed when the distances d_{ha}, d_{hi} and d_{ia} at least equals the respective safety distances.

Autonomous Layer The Autonomous Layer (`AUT`) provides a library of reactive skills and encapsulates access to actuators and sensors. The controllers implement longitudinal and lateral vehicle control based on a vehicle model as in Fig. 3. A vehicle on the track is characterized by orientation angle, β_{ori}, its lateral and longitudinal position, x and y, vehicle dimensions, width, length and vehicle velocity v. The controllers regulate the vehicle velocity by discretely adjusting its acceleration. The steering controllers determine paths of line segments connected with tangential circular arcs. Vehicle dynamics are constrained by constants such as maximal and minimal velocity or acceleration, a maximal centrifugal force that may occur or a vehicle's turning radius. The vehicle's own position and those of the surrounding vehicles are determined by on-board sensors and are accessible via `AUT`.

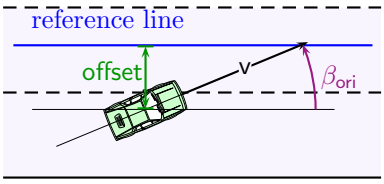


Figure 3: Car dimensions. Controllers of the AUT layer refer to offset, orientation β_{ori} and velocity v and distances to other cars.

Protocol Layer The Protocol Layer (PROT) coordinates the vehicle's behaviour with respect to other vehicles. When a car wants to perform a lane change, it becomes an initiator. It then may ask other agents to build a gap in order to facilitate a successful lane change. The three main steps of the initiator protocol are (1) determine feasible helpers (2) request help (3) perform lane change / abort. An agent asked for help obeys a helper protocol. The protocols are conditioned sequences of communications with other agents or AUT service calls. A snapshot of a lane change is depicted in Fig. 2. As soon as the initiator has perceived the sufficiency of size of the gap between its helper and the car directly ahead of the helper, it starts changing lane while continuously maintaining safety distances to the vehicles in front. The helper has agreed to establish and maintain the gap to the initiator. Note that the initiator vehicle maintains a *link* to the helper. This logical topology is a result of negotiation performed by PROT.

Physical Environment The physical environment (PENV) describes the time-continuous evolution of the vehicles on the road. Inputs of the physical environment are the actuator values of the controllers as defined in the AUT layer. Discrete jumps are performed for instance to represent which car is currently on which lane.

Translation Controllers and Sensors Translation controllers act as a glue between the protocol layer and the autonomous layer. They translate the messages of the protocol layer for the autonomous layer and generate messages for the protocol triggered by the autonomous layer. Sensors read the current state of a vehicle from the environment. They have exclusive access to the real world status of a vehicle. The sensors' readings constitute the vehicle's internal model of its environment, its *mental map*.

Prediction As a crucial part of the LCAS proposed, the vehicle has to decide which vehicle to cooperate with before a lane change manoeuvre is being initiated. To this end, we developed a prediction method that for a given traffic situation evaluates the feasibility of a lane change manoeuvre and finds the best suitable helpers. The predictions are based on approximate relative velocities and distances with respect to the vehicle's state and dynamics.

Outline Section 2 introduces the autonomous layer in detail. In Sect. 3 a coordination protocol for a lane change is presented. The translation controllers are described in Sect. 4 and sensors in Sect. 5. Section 6 describes the physical environment. Model extensions are discussed in Sect. 7.

2 Autonomous Layer

The autonomous layer (`AUT`) provides reactive control mechanisms based on continuous space and time representations. Vehicle behaviour is defined by a set of basic skills that are called from higher level layers. Such basic skills are for example *keeping a certain velocity*, *keeping a certain distance to some vehicle* or *steering into a neighbouring lane*. In this section we present the controllers of the `AUT` layer providing all reactive skills required for an LCAS. In Sect. 3 we will show how these skills are used by the protocol layer `PROT`.

Assumptions Following assumptions have been made. We assume that all cars on a highway are identical, which means they have the same physical dimensions and also the same abilities to accelerate and to decelerate. We model simplified car dynamics. The controllers set the acceleration and the motion of a car directly. A car's motion is set to be either a straight line or a circular arc, so that paths are made up of line segments and circular arcs (cf. Fig. 4). Transitions between straight lines and different circular arcs are smooth, i.e. without sharp bends. When a car follows its current movement circle, its centreline is tangent of that movement circle.

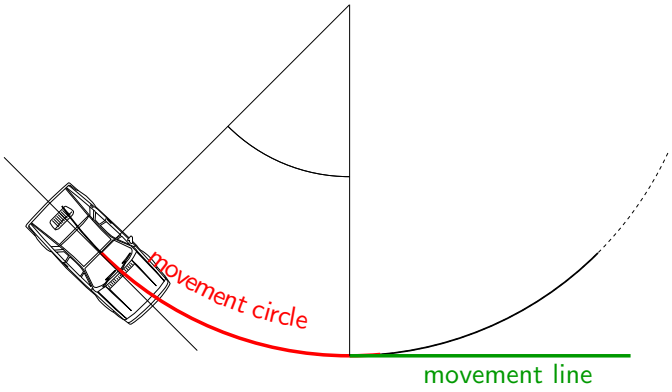


Figure 4: A vehicle's line of motion.

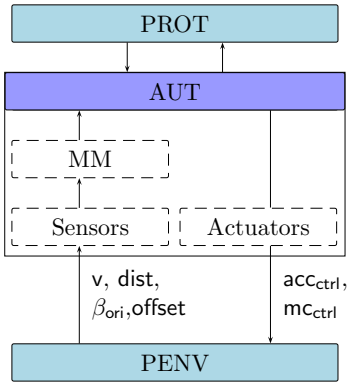


Figure 5: Interface of `AUT` to `PENV`.

AUT and PENV Figure 5 illustrates the structure of `AUT`. The sensors perceive the vehicle's velocity, v, its distance to some reference object, dist, the distance to the mid of a reference lane, offset, and its orientation, β_{ori}. Based on these readings `AUT` determines which values to set as acceleration, $\mathsf{acc}_{\mathsf{ctrl}}$, and as next movement, $\mathsf{mc}_{\mathsf{ctrl}}$. Since sensor data is provided only periodically, the mental map reflects a blurred perception of the real world traffic situation only. In this document we do not consider this phenomena or problems arising from inaccuracies of sensor readings or disturbances and instead assume continuous sensor data and that the vehicle behaves as specified by the controller. In Sect. 7 we discuss variants and extensions to the sensor modelling.

2.1 Controller

The fundamental controller architecture is depicted in Fig. 6. The `Error` mode indicates uncontrollable situations such as violation of a safety distance. In such cases the controller exits its current

mode and enters the `Error` mode where driver interaction is necessary to resolve such a situation. In *normal* situations the vehicle is controlled by a Steering Controller and a Velocity Controller—both are loosely coupled and operate in parallel. The Velocity Controller itself is composed of a Keep Velocity Controller that is active only if no other car is in front of the vehicle and the Approach Velocity Controller, which is active if the car has to adapt its speed to some other car in order to maintain a required distance to it.

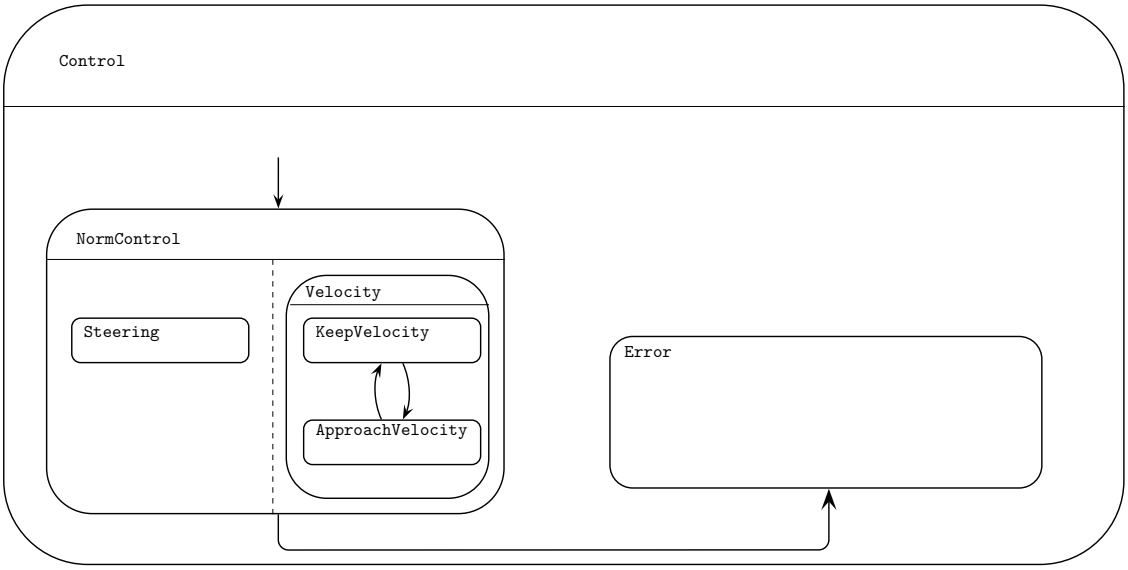


Figure 6: Controller of the Autonomous Layer.

2.1.1 Velocity Controller

The Velocity Controller (VC) adjusts the vehicle's velocity. As depicted in Fig. 6 the VC is composed of the Keep Velocity Controller (KVC) and the Approach Velocity Controller (AVC).

The KVC is designed to reach and keep a given speed, $\mathsf{v}_{\mathsf{goal}}$. The AVC has a two dimensional set-point, $(\mathsf{v}_{\mathsf{goal}}, \mathsf{dist}_{\mathsf{goal}})$, consisting of speed and distance values. It adjusts the acceleration to reach and keep a certain speed and a certain distance to a (possibly moving) reference object.

The KVC is hence activated when the agent car does not need to care about other cars. When the agent approaches some other car, the AVC is activated and ensures that safety distances are respected. The AVC also plays an important role for the lane change manoeuvre even if the agent has no car in front, since it may be necessary to accelerate or decelerate to reach a targeted gap.

Keep Velocity Controller (KVC) The KVC, depicted in Fig. 7, is responsible for keeping the velocity at approximately $\mathsf{v}_{\mathsf{goal}}$. To reach the desired velocity $\mathsf{v}_{\mathsf{goal}}$ with an allowed deviation ε_{v}, it accelerates or decelerates depending on whether the agent is slower than $\mathsf{v}_{\mathsf{goal}} - \varepsilon_{\mathsf{v}}$ or faster than $\mathsf{v}_{\mathsf{goal}} + \varepsilon_{\mathsf{v}}$. When a car accelerates it has to check whether it is feasible to steer in line with the lane again, otherwise the car has to keep its velocity. The function $\mathsf{accIsFeasible}(\mathsf{v},\mathsf{acc},\beta,\mathsf{offset})$ evaluates to true iff the agent can *comfortably* follow a movement circle to get in the direction of the traffic

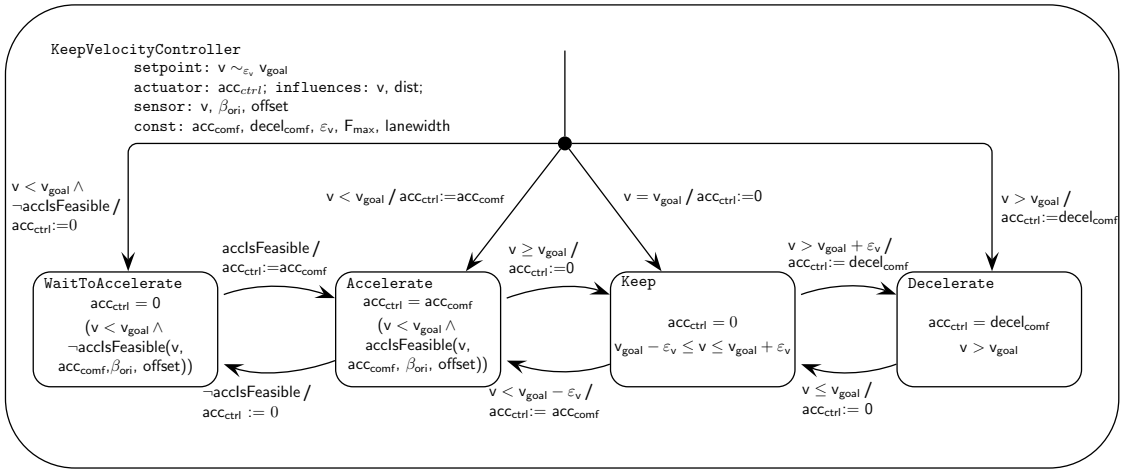


Figure 7: Keep Velocity Controller.

flow without leaving its lane even when the agent accelerates for time t_{acc} with acceleration acc. An agent can comfortably follow a movement circle if the centrifugal force is less than $\mathsf{F}_{\mathsf{comf}}$. Hence the faster an agent is, the wider its movement circles have to be. The mathematical equations are given on p. 15.

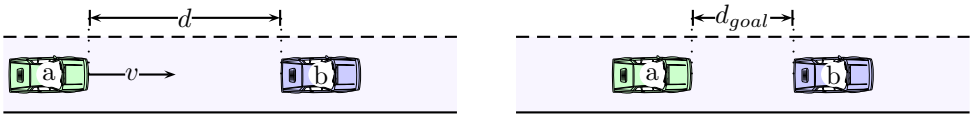
In Fig. 7 the interface of the KVC is described. The set-point is a goal velocity, which is reached by adjusting the acceleration of the vehicle via $\mathsf{acc}_{\mathsf{ctrl}}$. This implies changes of the vehicle's velocity, its distance to other objects and its offset to the reference line. Although the set-point is just a goal velocity, the controller monitors offset and orientation, since they are necessary to determine whether accIsFeasible(v,acc,β,offset) equals true, i.e. the agent can accelerate without leaving its lane (cf. Sect. 2.1.2). The interface documentation does not specify the value of $\mathsf{v}_{\mathsf{goal}}$. Depending on the context, $\mathsf{v}_{\mathsf{goal}}$ may be for instance $\mathsf{v}_{\mathsf{cruise}}$, a velocity the agent likes as default travel speed or it may be $\mathsf{v}_{\mathsf{max}}$, the maximal velocity. We will see in Sect. 4 that so called *translation controllers*, triggered by the protocol layer, determine the values for the controllers' set-points.

Approach Velocity Controller (AVC) The AVC is activated in case the agent has to take care of a reference object. The AVC controls the acceleration, so that the agent reaches and keeps the desired distance to a (possibly moving) reference object at the desired speed. The AVC is deactivated if there is no longer a reference object to take care of.

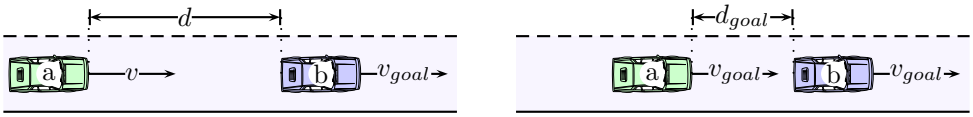
The AVC thus implements a very powerful control law. Figure 8 lists some manoeuvres the AVC is capable to perform:

(a) Approach a standing object.
A translation controller then sets v_{goal} *to zero and* dist_{goal} *to the desired distance between the agent and standing object. The AVC chooses an acceleration so that the agent comes to halt, i.e. reaches* $\mathsf{v} = 0$*, in distance* dist_{goal} *from the standing object.*

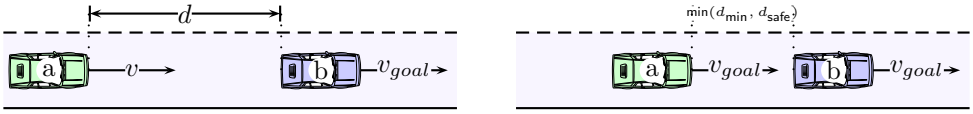
(b) Follow a moving object while maintaining the safety distance.
v_{goal} *is set to the velocity of the reference object and* dist_{goal} *is the safety distance. Then AVC adjusts the acceleration so that the agent reaches* v_{goal} *in the distance* dist_{goal} *to this object.*



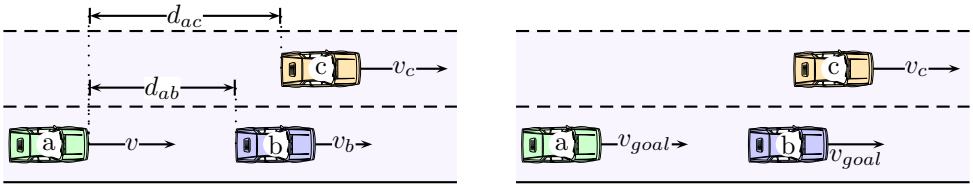
(a) Approach a standing object.



(b) Follow a moving object while maintaining the safety distance.



(c) Travel with $\mathsf{v}_{\mathsf{cruise}}$ but keep at least distance $\mathsf{d}_{\mathsf{min}}$ to a moving object.



(d) Stay behind two moving objects while respecting safety distances.

Figure 8: Sample scenarios (initial and goal) the AVC controls.

When the set-point is reached and the reference object does not change its velocity, dist_{goal} *is kept and AVC sets* acc_{ctrl} *to zero; otherwise AVC adjusts the acceleration accordingly.*

(c) Travel at most with $\mathsf{v}_{\mathsf{cruise}}$ but keep at least distance $\mathsf{d}_{\mathsf{min}}$ to a moving object.
v_{goal} *is set to the minimum of desired travel speed* v_{cruise} *and the velocity of the reference object.* dist_{goal} *is set to the maximum of the safety distance and* $\mathsf{d}_{\mathsf{min}}$*. Hence AVC adjusts the acceleration so that the agent does not fall below neither the safety distance nor* $\mathsf{d}_{\mathsf{min}}$*.*

(d) Stay behind two moving objects while respecting safety distances.
Suppose the agent has a predecessor on its lane and wants to change onto a neighbouring lane behind some car. The AVC can now be used to make the agent maintain the safety distances to its predecessor and the intended new predecessor. v_{goal} *is set to the minimum velocity of both these reference cars and* dist_{goal} *to a distance that implies that both safety distances are respected.*

The above list gives an impression of what the AVC is capable of. In order to bring the agent behind the reference car with the desired distance, AVC controls the acceleration only. Based on the current distance and velocity it determines whether it can find an acceleration—the so called *target acceleration*—that makes the agent reach the goal velocity and goal distance to the reference object. The basic design principles for the AVC can be summarized as:

- The AVC is built to keep an acceleration (may it be positive or negative) as long as possible.
- A comfortable acceleration is chosen if either the target acceleration is too large to be comfortable or if the target acceleration is too small to reach the set-point in reasonable time.

  Whereas the first aspect makes a manoeuvre comfortable, the second aspect yields an optimization on the overall manoeuvre time.

The controller automaton is given in Fig. 9. In each state a part of the invariant is given defining a constraint on the controller's output. The complete state invariant then is the conjunction with the state invariant given in Table 1. Edge labels, that is pairs of a guard and a discrete update action, are omitted here. The update action corresponds to the invariant inscribed into the respective target state in Fig. 9. The guards correspond to the invariants of the target state as given in the Table 1.

In the following we explain the control laws of the Approach Velocity Controller based on the system dynamics as illustrated in Fig. 10. For a rough orientation, we added the mode names of the Approach Velocity Controller of Fig. 9 to Fig. 10, indicating when the respective mode is active.

ChooseDecel Consider the upper right quadrant of Fig. 10. In this situation the agent is too fast ($\mathsf{v} > \mathsf{v}_{\mathsf{goal}}$) and the distance to the reference object is too large ($\mathsf{dist} > \mathsf{dist}_{\mathsf{goal}}$). Below the curve $\mathsf{decel}_{\mathsf{comf}}$ there is one deceleration $\mathsf{decel}_{\mathsf{target}}$ to bring the agent as close as desired to the reference object, and when the distance is reached also the goal velocity is reached ($\mathsf{dist} = \mathsf{dist}_{\mathsf{goal}}$ and $\mathsf{v} = \mathsf{v}_{\mathsf{goal}}$). The agent chooses $\mathsf{decel}_{\mathsf{target}}$ as acceleration except in the following three cases:

1. If the time to reach $\mathsf{dist}_{\mathsf{goal}}$ and $\mathsf{v}_{\mathsf{goal}}$ with $\mathsf{decel}_{\mathsf{target}}$ is greater than reasonable, the agent accelerate comfortably to then decelerate more vehemently. $\mathsf{t}_{\mathsf{reason}}$ specifies the maximum time the agent is willing to keep its acceleration (or deceleration) in order to reach the goal (velocity and distance). We thereby reduce the overall manoeuvre time.

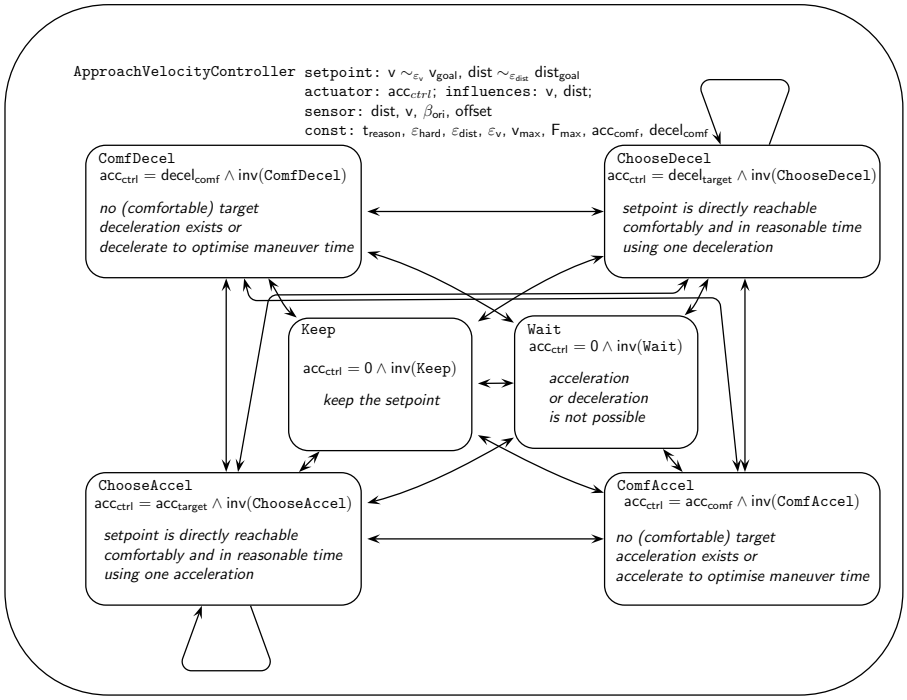


Figure 9: Approach Velocity Controller. $\mathsf{t}^{\mathsf{acc}}_{\mathsf{target}}(\mathsf{t}^{\mathsf{dec}}_{\mathsf{target}})$ is an abbreviation of $\frac{\mathsf{v}_{\mathsf{goal}}-\mathsf{v}}{\mathsf{acc}_{\mathsf{target}}}(\frac{\mathsf{v}_{\mathsf{goal}}-\mathsf{v}}{\mathsf{decel}_{\mathsf{target}}})$, the time spend to accelerate/decelerate from v to $\mathsf{v}_{\mathsf{goal}}$ with acceleration $\mathsf{acc}_{\mathsf{target}}(\mathsf{decel}_{\mathsf{target}})$. For the invariants cf. Table 1.

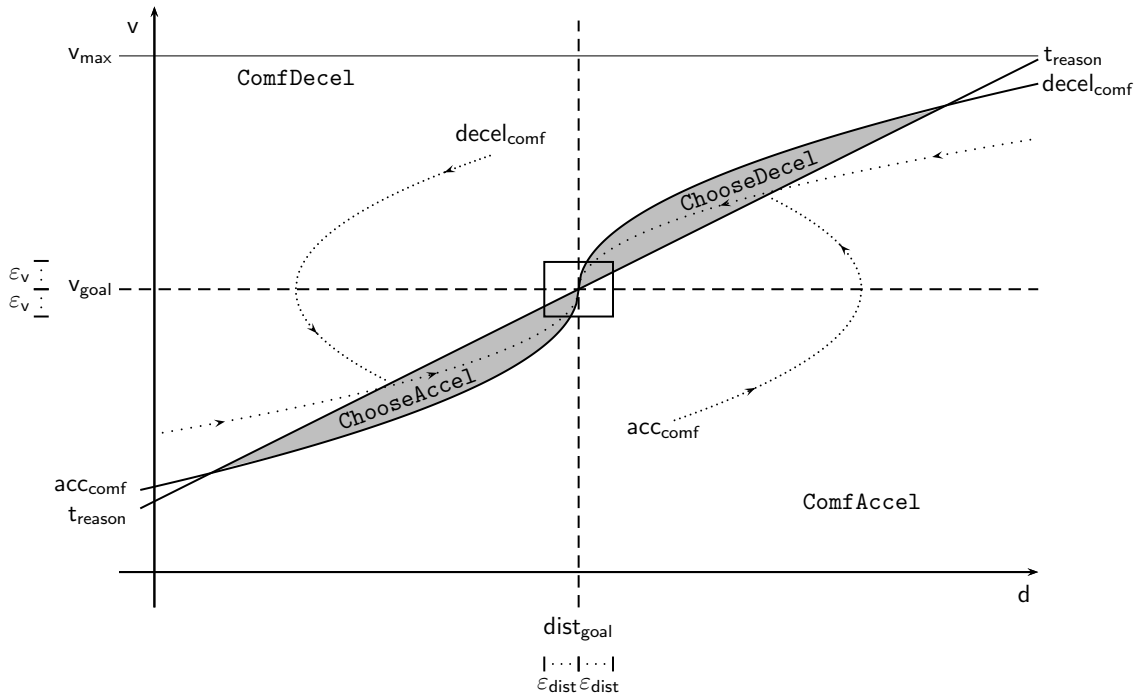


Figure 10: Key idea for the AVC design I.

state	invariant	
`ComfDecel`	$v > 0 \wedge ($	
	$(dist \geq dist_{goal} \wedge v > v_{goal} \wedge \lvert decel_{target}\rvert \geq \lvert decel_{comf}\rvert)$	(CoD1)
	$\vee\, (dist < dist_{goal} \wedge v \geq v_{goal})$	(CoD2)
	$\vee\, (dist < dist_{goal} \wedge v < v_{goal} \wedge acc_{target} < acc_{comf} \wedge t_{reason} \leq t^{acc}_{comf})$	(CoD3)
	$\vee\, (dist < dist_{goal} \wedge v < v_{goal} \wedge acc_{target} < acc_{comf} \wedge t_{reason} < t^{acc}_{target}))$	(CoD4)
`ComfAccel`	$v < v_{max} \wedge accIsFeasible(v, acc_{comf}, \beta_{ori}, offset) \wedge$	
	$((dist \leq dist_{goal} \wedge dist > dist_{hard} + \varepsilon_{hard} \wedge v < v_{goal} \wedge acc_{target} \geq acc_{comf})$	(CoA1)
	$\vee\, (dist > dist_{goal} \wedge v \leq v_{goal})$	(CoA2)
	$\vee\, (dist > dist_{goal} \wedge v > v_{goal} \wedge \lvert decel_{target}\rvert < \lvert decel_{comf}\rvert \wedge t_{reason} \leq t^{dec}_{comf})$	(CoA3)
	$\vee\, (dist > dist_{goal} \wedge v > v_{goal} \wedge \lvert decel_{target}\rvert < \lvert decel_{comf}\rvert \wedge t_{reason} < t^{dec}_{target}))$	(CoA4)
`ChooseDecel`	$v > 0 \wedge$	
	$dist > dist_{goal} \wedge v > v_{goal} \wedge \lvert decel_{target}\rvert < \lvert decel_{comf}\rvert \wedge (t^{dec}_{target} \leq t_{reason})$	(ChD)
`ChooseAccel`	$dist > dist_{hard} + \varepsilon_{hard} \wedge v < v_{max} \wedge accIsFeasible(v, acc_{target}, \beta_{ori}, offset)$	
	$\wedge\, dist < dist_{goal} \wedge v < v_{goal} \wedge acc_{target} < acc_{comf} \wedge t^{acc}_{target} \leq t_{reason}$	(ChA)
`Keep`	$dist_{goal} - \varepsilon_{dist} \leq dist \leq dist_{goal} + \varepsilon_{dist} \wedge v_{goal} - \varepsilon_v \leq v \leq v_{goal} + \varepsilon_v$	
`Wait`	$((dist \leq dist_{hard} + \varepsilon_{hard} \vee v \geq v_{max} \vee \neg accIsFeasible(v, acc_{target}, \beta_{ori}, offset))$	
	$\wedge$ (CoA1 $\vee$ CoA2 $\vee$ CoA3 $\vee$ ChA)) $\vee$	
	$((v \leq 0) \wedge$ (CoD1 $\vee$ CoD2 $\vee$ CoD3 $\vee$ ChD))	

Table 1: Invariants of the Approach Velocity Controller.

2. If $decel_{target}$ is too vehement to be comfortable, the agent chooses a lesser deceleration, that is it decelerates with $decel_{comf}$.

3. The agent is already at a standstill and hence cannot further decelerate.

ComfDecel In the upper left quadrant of Fig. 10 the agent's velocity is greater than the goal velocity and the agent is too close to the car ahead. The agent decelerates comfortably. The agent also uses comfortable deceleration, in case the target deceleration is greater than comfortable (upper right quadrant, left of the grey area) or in case the agent is too close and too fast (upper left quadrant) or in case the agent is slower than the target velocity and too far close, but acceleration is not yet reasonable (lower left quadrant above the grey area).

ChooseAccel The mode ChooseAccel corresponds basically to the grey area in the lower left quadrant of Fig. 10. That means, a target acceleration is exists, if the agent's velocity is less than the goal velocity and the distance is less than the goal distance. There are four cases in which acc_{target} is not chosen (even if it exists):

1. If acceleration with acc_{target} would take unreasonable time, the agent decelerates comfortably (i.e. as vehemently as comfortable) and then uses a greater acceleration.

2. If the acceleration is too large to be comfortable the agent chooses a less vehement acceleration that is it accelerates with acc_{comf}.

3. The agent car already drives with maximum speed and is not allowed to further accelerate.

4. The agent is not directly following the traffic flow. In this case the agent is allowed to accelerate only if the agent is able to follow a movement circle that gets it heading in direction of the traffic.

ComfAccel The agent uses comfortable acceleration, in case the target acceleration is greater than the comfortable acceleration (lower left quadrant, right of the grey area) or in case the agent is too far away and too slow (lower right quadrant) or in case the agent is faster than the target velocity and too far away, but deceleration makes the vehicle reaching its set point after more than reasonable time (upper right quadrant below the grey area).

Wait If the agent should accelerate, but is not able to, because it either cannot keep its lane or is too close or too fast, it keeps its current velocity. Likewise if the agent should decelerate but is not able to because it is at a standstill, it keeps its velocity.

Figure 11 exemplifies the controller's behaviour by the blue (where $\mathsf{acc} = \mathsf{acc}_{\mathsf{comf}}$), green (where $\mathsf{acc} = 0$) and finally red trajectory (where $\mathsf{acc} = \mathsf{decel}_{\mathsf{target}}$). During the blue part of the trajectory, the agent is slower than the car ahead and too far away, so it accelerates with maximal acceleration $\mathsf{acc}_{\mathsf{comf}}$ to catch up with the reference object and gets even faster. During the green part of the trajectory, the agent keeps the velocity when its maximum speed is reached and at the red part of the trajectory the agent constantly uses the target deceleration. It starts to brake as soon as it is close enough to reach the goal distance and velocity in reasonable time.

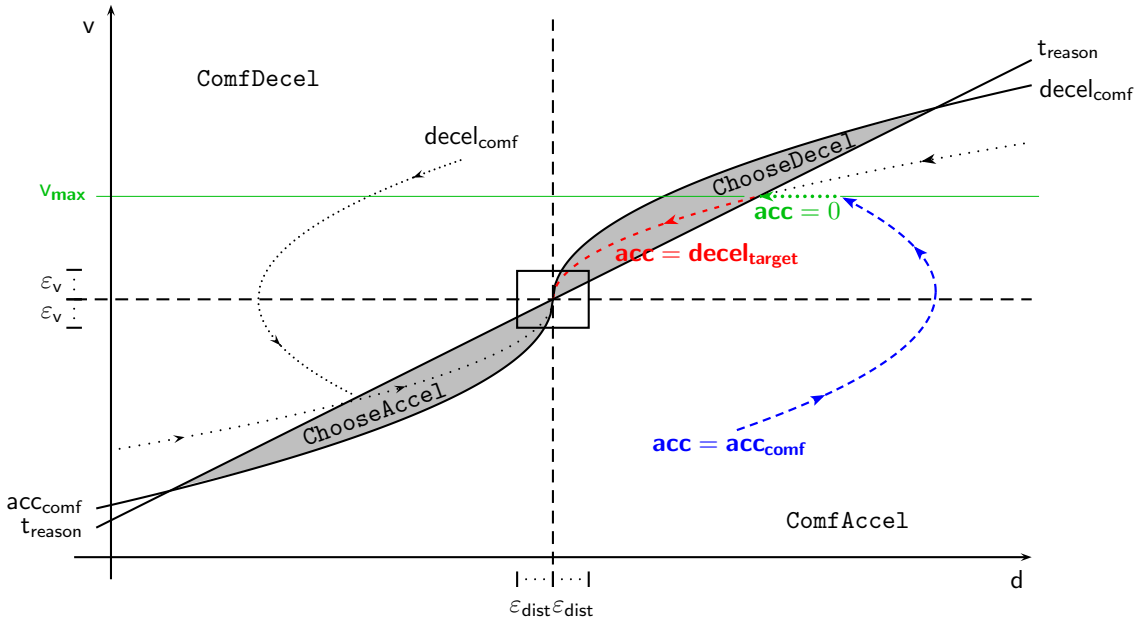


Figure 11: Key idea for the AVC design II.

Determining the Target Acceleration We stated above that there is (i) a target acceleration $\mathsf{acc}_{\mathsf{target}}$ that makes a vehicle reach the set-point's velocity and distance, if the agent is slower than its reference object and too close to it, and that there is (ii) a target deceleration $\mathsf{decel}_{\mathsf{target}}$, if the agent is faster than the reference object and too far away from it. In the following we will derive a formula for determining $\mathsf{acc}_{\mathsf{target}}$. Consider the scenario as illustrated in Fig. 12.

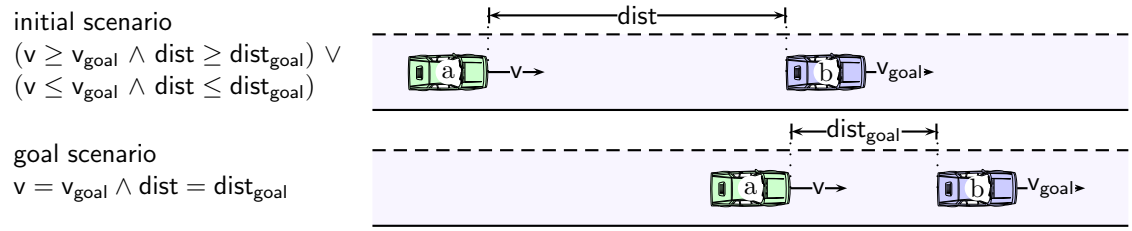


Figure 12: Determining $\mathsf{acc}_{\mathsf{target}}$ and $\mathsf{decel}_{\mathsf{target}}$. The controlled car is a and car b is the reference object.

Let us denote the initial positions of the agent and its reference object as p_a and p_b, respectively. We denote their respective positions in the goal scenario as p'_a and p'_b. For this scenario we use as $\mathsf{v}_{\mathsf{goal}}$ the velocity of the reference object and denote the agent's velocity simply with v. We denote the (goal) displacement as dist ($\mathsf{dist}_{\mathsf{goal}}$). We assume that the car ahead drives with constant speed. This gives rise to the following system of equations:

$$\begin{aligned} p'_a &= p_a + \mathsf{v} \cdot t + \frac{1}{2}\,\mathsf{acc} \cdot t^2 \\ p'_b &= \mathsf{v}_{\mathsf{goal}} \cdot t + p_b \\ \mathsf{dist}_{\mathsf{goal}} &= p'_b - p'_a \\ \mathsf{v}_{\mathsf{goal}} &= \mathsf{v} + \mathsf{acc} \cdot t \end{aligned} \tag{1}$$

Hence the vehicle a reaches the distance $\mathsf{dist}_{\mathsf{goal}}$ when taking the following acceleration/deceleration acc:

$$\mathsf{acc} = \frac{(\mathsf{v}_{\mathsf{goal}} - \mathsf{v})^2}{2(\mathsf{dist}_{\mathsf{goal}} - (p_b - p_a))}$$

If we denote the initial distance of the two cars, $p_b - p_a$, with dist, we can express the above equation as:

$$\mathsf{acc} = \frac{(\mathsf{v}_{\mathsf{goal}} - \mathsf{v})^2}{2(\mathsf{dist}_{\mathsf{goal}} - \mathsf{dist})}, \tag{2}$$

Equation 2 on page 15 describes the case that the reference object is slower than the agent and $\mathsf{dist}_{\mathsf{goal}}$ is greater than dist, gives $\mathsf{acc}_{\mathsf{target}}$.

Formula Eq. 2 determines exactly the necessary acceleration $\mathsf{acc}_{\mathsf{target}}$ (deceleration $\mathsf{decel}_{\mathsf{target}}$) assuming that the agent drives on a straight line and the reference object drives with constant speed, that is the displacement equals the distance covered by the agent. Since the distance covered by the agent is greater than the displacement, $\mathsf{decel}_{\mathsf{target}}$ may thus be too large and $\mathsf{acc}_{\mathsf{target}}$ may thus be too small, if the agent is currently on a movement circle. Thus using this target deceleration/acceleration will not be safety critical. Since the target acceleration/deceleration is only used when it brings the agent to the set-point within reasonable time, there is an upper bound on the difference between displacement and distance.

Switching Between Approach Velocity Controller and Keep Velocity Controller Up to now, we have introduced the two controllers, AVC and KVC, that together make up the velocity control. Figure 6 illustrates that control is switched between these two controllers; they are composed sequentially. As the AVC is used to ensure safety distances between cars, the activation of AVC is safety critical and a belated activation may lead to a violation of safety distances. Roughly put, control passes from the AVC to the KVC when the agent has no way to follow its reference object and the distance exceeds the goal distance, and control passes from the KVC to the AVC if the distance is less than the goal distance or if the agent has to decelerate soon.

Table 2 gives the guards for switching between the KVC and the AVC. Control passes from the AVC to the KVC when the agent's velocity is lower than the goal velocity and the current velocity is bigger than the goal velocity, so that the agent would have to accelerate with *maximal (not comfortable!)* acceleration to merely reach the goal velocity. The AVC gets activated when the distance between the agent and its reference car is too small but also when the distance is too big but the agent would need to decelerate more than comfortable to reach the goal or the target deceleration is to small to reach the goal within reasonable time but it would reach the goal in thrice the reasonable time.

state	invariant	
AVC $\rightarrow$ KVC	$\mathsf{dist} > \mathsf{dist}_{\mathsf{goal}} + \varepsilon_{\mathsf{dist}} \wedge \mathsf{v} + \mathsf{acc}_{\mathsf{max}} \cdot 3 \cdot \mathsf{t}_{\mathsf{reason}} < \mathsf{v}_{\mathsf{goal}} - \varepsilon_{\mathsf{v}}$	
KVC $\rightarrow$ AVC	$(\mathsf{dist} \leq \mathsf{dist}_{\mathsf{goal}} + \varepsilon_{\mathsf{dist}}) \vee (\mathsf{dist} > \mathsf{dist}_{\mathsf{goal}} - \varepsilon_{\mathsf{dist}} \wedge \mathsf{v} > \mathsf{v}_{\mathsf{goal}} - \varepsilon_{\mathsf{v}} \wedge$	
	$(\lvert\mathsf{decel}_{\mathsf{target}}\rvert \geq \lvert\mathsf{decel}_{\mathsf{comf}}\rvert$	(CoD1')
	$\vee\, \mathsf{t}^{\mathsf{dec}}_{\mathsf{comf}} \leq \mathsf{t}^{\mathsf{dec}}_{\mathsf{target}} \leq 3 \cdot \mathsf{t}_{\mathsf{reason}})$	(CoA3')

Table 2: Guards for switching between AVC and KVC

Steering Influences Velocity When the agent is able to reach the set-point by a constant acceleration $\mathsf{acc}_{\mathsf{target}}$ (ChooseAccel or ChooseDecel), the chosen acceleration brings the agent in distance $\mathsf{dist}_{\mathsf{goal}}$ to the car ahead with speed $\mathsf{v}_{\mathsf{goal}}$. If the agent's movement circle changes and the agent is on a narrower curve, the agent may not continue using $\mathsf{acc}_{\mathsf{target}}$, because it is not feasible ($\mathsf{accIsFeasible}$ becomes false) any more. Likewise it may still be possible for the agent to use a constant acceleration to reach its set-point. Therefore the automaton (cf. Fig. 9) contains self loops on states ChooseAccel and ChooseDecel. They are triggered by a change on the movement circle.

2.1.2 Steering Controller

The *Steering Controller* (SC) is responsible for bringing and keeping the vehicle on a reference line. Usually *the reference line* is the centreline of the agent's lane, but for instance during a lane change, the reference line becomes the mid of the target lane so that the SC makes the agent change lane.

Figure 14 gives the Steering Controller and describes its interface. The Steering Controller has a two dimensional set-point of offset offset and orientation β_{ori}. Its goal is to bring the car in $\beta_{\mathsf{ori}} \sim_{\varepsilon_\beta} 0$ and $\mathsf{offset} \sim_{\varepsilon_{\mathsf{off}}} 0$, which means the car has to follow the traffic flow and be on the reference line, for both a slight deviation is tolerated. The two dimensions of the Steering Controller are illustrated in Fig. 13.

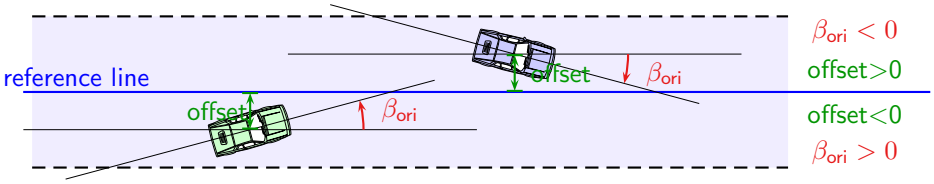


Figure 13: Set-point dimensions of the Steering Controller. The orientation equals zero if the car is parallel to the road. Its offset equals zero when it is on the reference line.

Recall that the agent is either on a straight line or follows some movement circle. A *movement circle* is given by a radius and a direction. A vehicle is on a movement circle if its centreline is tangent of the movement circle (cf. Fig. 15). The transitions between different movement circles and lines are smooth. If the reference line can be reached by following a certain circle, the circle can be chosen as so called *target movement circle*. In general, there may be several such circles, if the car is facing towards the reference line, or none, in case the car is facing away from the reference line.

The basic design principles of the SC are similar to those of AVC:

- The SC is built to keep a movement as long as possible.
- A comfortable movement circle is chosen if either the target movement circle is to narrow—centrifugal forces become uncomfortably strong—or if the target movement circle is yet too wide to get the agent on the reference line in reasonable time.

In the states `ChooseRight` and `ChooseLeft` the agent drives at a speed such that the reference line is reached by following a target movement circle comfortably and in reasonable time. A movement is *reasonable* ($\mathsf{isReasnble}(\beta, \mathsf{mc}, v)$ evaluates to true), if it takes at most time $\mathsf{t}_{\mathsf{reason}}$ for the agent to reach the reference line while driving with the current speed v. A movement circle is *comfortable* at speed v ($\mathsf{isComf}(\mathsf{mc}, v)$ evaluates to true), if the agent experiences a centrifugal force less than $\mathsf{F}_{\mathsf{max}}$ when following the movement circle mc with speed v. There may be several movement circles that the agent could follow satisfying these constraints. We choose the widest such movement circle.

The states `ComfLeft` and `ComfRight` represent situations where the agent cannot directly follow a movement circle to reach the mid of lane because either

1. the agent is facing away from the reference line (CoL2,CoR2), or
2. the agent is too close to the reference line to reach it on a comfortable movement circle (CoR1,CoL1), or
3. the agent is too far away *(the offset is too large)* to reach the reference line in a reasonable time (CoL3,CoR3).

In the first two cases the agent takes the narrowest possible movement circle, $\mathsf{mc}_{\mathsf{comf}}$, that is still comfortable. In the last case the agent steers with the narrowest possible movement circle towards the reference line to reduce its offset, to then use the target movement circle—which is in the opposite direction—to get onto the reference line.

The loops on states `ChooseRight`, `ChooseLeft`, `ComfRight` and respectively `ComfLeft` are taken whenever the velocity changes, so that isReasnble and isComf are evaluated with the up to date speed. Again part of the invariants of the automaton in Fig. 14 are given in an extra table, that is Table 3 and edge labels can be derived accordingly.

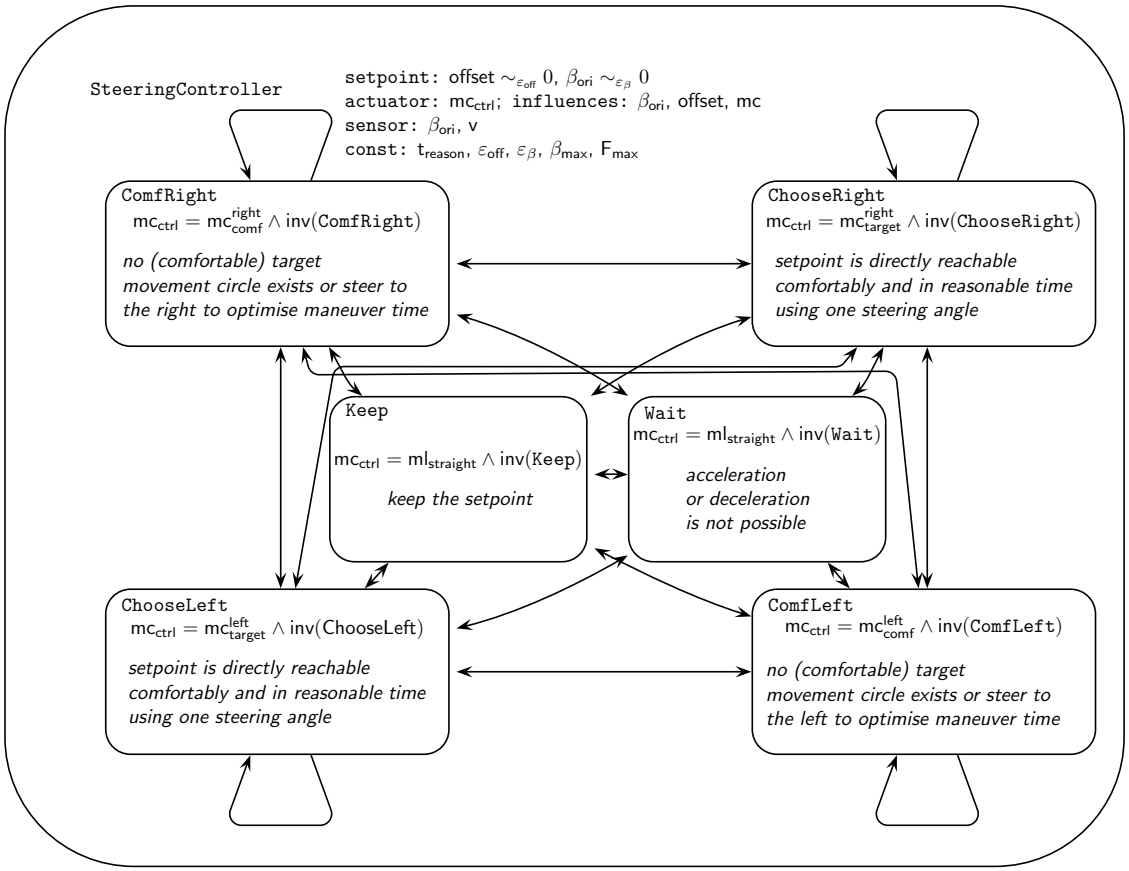


Figure 14: Steering Controller. mc_{target} is a movement circle that makes the vehicle reach the reference line tangentially. A movement circle is defined as a pair (radius, side). For the invariants cf. Table 3.

Determining the Movement Circle mc_{ctrl} The SC sets the movement circle of its vehicle to $\pm mc_{comf}$ or to mc_{target}. In the following we describe how these movement circles are determined. We first show how to determine the least radius the agent can comfortably follow. The faster an agent is, the wider such a circle has to be, since otherwise uncomfortable centrifugal forces are experienced. We assume a global coordinate system to some roadside origin with a horizontal x-axis. Lanes are parallel to the x-axis and cars are heading in direction of increasing x.

state	invariant	
`ComfRight`	$-\beta_{\mathsf{ori}} < \beta_{\mathsf{max}} - \varepsilon_\beta \wedge$	
	$((\mathsf{offset} \leq 0 \wedge \beta_{\mathsf{ori}} > 0 \wedge \neg \mathsf{isComf}(\mathsf{mc}^{\mathsf{right}}_{\mathsf{target}}, \mathsf{v}))$	(CoR1a)
	$\vee ((\mathsf{offset} \leq 0 \wedge \beta_{\mathsf{ori}} > 0 \wedge \mathsf{isComf}(\mathsf{mc}^{\mathsf{right}}_{\mathsf{target}}, \mathsf{v}) \wedge \neg \mathsf{isReasnble}(\beta_{\mathsf{ori}}, \mathsf{mc}^{\mathsf{right}}_{\mathsf{target}}, \mathsf{v}))$	(CoR1b)
	$\vee (\mathsf{offset} < 0 \wedge \beta_{\mathsf{ori}} \geq 0)$	(CoR2)
	$\vee (\mathsf{offset} > 0 \wedge \beta_{\mathsf{ori}} < 0 \wedge \neg \mathsf{isReasnble}(\beta_{\mathsf{ori}}, \mathsf{mc}^{\mathsf{left}}_{\mathsf{target}}, \mathsf{v})))$	(CoR3)
`ComfLeft`	$\beta_{\mathsf{ori}} < \beta_{\mathsf{max}} - \varepsilon_\beta \wedge$	
	$((\mathsf{offset} \geq 0 \wedge \beta_{\mathsf{ori}} < 0 \wedge \neg \mathsf{isComf}(\mathsf{mc}^{\mathsf{left}}_{\mathsf{target}}, \mathsf{v}))$	(CoL1a)
	$\vee ((\mathsf{offset} \geq 0 \wedge \beta_{\mathsf{ori}} < 0 \wedge \mathsf{isComf}(\mathsf{mc}^{\mathsf{left}}_{\mathsf{target}}, \mathsf{v}) \wedge \neg \mathsf{isReasnble}(\beta_{\mathsf{ori}}, \mathsf{mc}^{\mathsf{left}}_{\mathsf{target}}, \mathsf{v}))$	(CoL1b)
	$\vee (\mathsf{offset} < 0 \wedge \beta_{\mathsf{ori}} \leq 0)$	(CoL2)
	$\vee (\mathsf{offset} < 0 \wedge \beta_{\mathsf{ori}} > 0 \wedge \neg \mathsf{isReasnble}(\beta_{\mathsf{ori}}, \mathsf{mc}^{\mathsf{right}}_{\mathsf{target}}, \mathsf{v})))$	(CoL3)
`ChooseRight`	$\mathsf{offset} < 0 \wedge \beta_{\mathsf{ori}} > 0 \wedge \mathsf{isComf}(\mathsf{mc}^{\mathsf{right}}_{\mathsf{target}}, \mathsf{v}) \wedge \mathsf{isReasnble}(\beta_{\mathsf{ori}}, \mathsf{mc}^{\mathsf{right}}_{\mathsf{target}}, \mathsf{v})$	(ChR)
	$\wedge -\beta_{\mathsf{ori}} < \beta_{\mathsf{max}} - \varepsilon_\beta$	
`ChooseLeft`	$\mathsf{offset} > 0 \wedge \beta_{\mathsf{ori}} < 0 \wedge \mathsf{isComf}(\mathsf{mc}^{\mathsf{left}}_{\mathsf{target}}, \mathsf{v}) \wedge \mathsf{isReasnble}(\beta_{\mathsf{ori}}, \mathsf{mc}^{\mathsf{left}}_{\mathsf{target}}, \mathsf{v})$	(ChL)
	$\wedge \beta_{\mathsf{ori}} < \beta_{\mathsf{max}} - \varepsilon_\beta$	
`Keep`	$-\varepsilon_{\mathsf{off}} \leq \mathsf{offset} \leq \varepsilon_{\mathsf{off}} \wedge -\varepsilon_\beta \leq \beta_{\mathsf{ori}} \leq \varepsilon_\beta$	
`Wait`	$(-\beta_{\mathsf{ori}} \geq \beta_{\mathsf{max}} - \varepsilon_\beta \wedge (\mathrm{CoR1a} \vee \mathrm{CoR1b} \vee \mathrm{CoR2} \vee \mathrm{CoR3} \vee \mathrm{ChR})) \vee$	
	$(\ \beta_{\mathsf{ori}} \geq \beta_{\mathsf{max}} - \varepsilon_\beta \wedge (\mathrm{CoL1a} \vee \mathrm{CoL1b} \vee \mathrm{CoL2} \vee \mathrm{CoL3} \vee \mathrm{ChL}))$	

Table 3: Invariants of the Steering Controller.

Determining a Comfortable Movement Circle The SC chooses in states `ComfLeft` as movement circle $\mathsf{mc}_{\mathsf{comf}}$ and in state `ComfRight` -$\mathsf{mc}_{\mathsf{comf}}$. Intuitively, a movement circle is comfortable if the centrifugal force a driver experiences following the movement circle with the current speed is comfortable. Given the maximal centrifugal force $\mathsf{F}_{\mathsf{comf}}$ that a passenger with assumed weight $\mathsf{mass}_{\mathsf{passenger}}$ experiences as comfortable, then

$$\mathsf{r}_{\mathsf{comf}} = \frac{\mathsf{mass}_{\mathsf{passenger}} \cdot \mathsf{v}^2}{\mathsf{F}_{\mathsf{comf}}}. \tag{3}$$

is the least radius of a circle that the agent can drive comfortably. So we say that $\mathsf{isComf}(\mathsf{mc}, \mathsf{v}) = \mathsf{true}$ iff mc.r¿$\mathsf{r}_{\mathsf{comf}}$.

Determining a Reasonable Time Movement Circle In the following we will determine a movement circle so that the agent reaches the reference line in time $\mathsf{t}_{\mathsf{reason}}$. We assume that the agent currently has an orientation of β_{ori} and that it has (and keeps) a velocity of v.

In time $\mathsf{t}_{\mathsf{reason}}$ the car drives

$$\mathsf{s}_{\mathsf{reason}} = \mathsf{v} \cdot \mathsf{t}_{\mathsf{reason}}. \tag{4}$$

The distance $\mathsf{s}_{\mathsf{reason}}$ is regarded to be the distance the car covers to reach the reference line tangentially following the reasonable time movement circle. So a sector is spanned by the reference line and the agent's perpendicular as illustrated in Fig. 15.

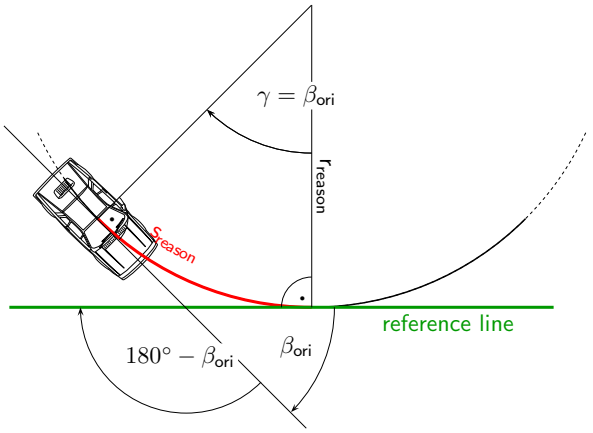


Figure 15: Reaching the reference line in time $\mathsf{t}_{\text{reason}}$.

As denoted in Fig. 15 the central angle of the sector equals the car's orientation, β_{ori}. Hence we can derive that

$$\mathsf{r}_{\text{reason}} = \frac{\mathsf{s}_{\text{reason}} \cdot 180°}{\pi \cdot \beta_{\text{ori}}}. \tag{5}$$

by using the well known formula for the arc length of a sector. We say that $\mathsf{isReasnble}(\beta_{\text{ori}}, \mathsf{mc}, \mathsf{v}) = \mathsf{true}$ iff $\mathsf{mc.r} \leq \mathsf{r}_{\text{reason}}$.

Choosing a Target Movement Circle In modes `ChooseRight` and `ChooseLeft` the agent is allowed to choose any radius that is comfortable and timely reasonable, $\mathsf{isReasnble}(\beta_{\text{ori}}, \mathsf{mc}, \mathsf{v}) \wedge \mathsf{isComf}(\mathsf{mc}, \mathsf{v})$. The agent's centreline and the reference line are both tangents of the movement circle. The former implies that the circle centre is on its perpendicular (cf. first line of Eq. 6). The latter implies that the circle centre's y-position is $\mathsf{r}_{\text{target}}$ above or below the y-position of the reference line (cf. second line of Eq. 6).

$$\begin{aligned}
\mathsf{y}_{\text{circ}} &= -\frac{\mathsf{x}_{\text{circ}}}{\tan \beta_{\text{ori}}} + \mathsf{y}_{\text{centre}} + \frac{\mathsf{x}_{\text{centre}}}{\tan \beta_{\text{ori}}} \\
\mathsf{y}_{\text{circ}} &= \mathsf{reference_lane} \pm \mathsf{r}_{\text{target}} \\
\mathsf{r}_{\text{target}} &= \sqrt{(\mathsf{x}_{\text{centre}} - \mathsf{x}_{\text{circ}})^2 + (\mathsf{y}_{\text{centre}} - \mathsf{y}_{\text{circ}})^2} \\
\mathsf{r}_{\text{target}} &\leq \mathsf{r}_{\text{reason}} \\
\mathsf{r}_{\text{target}} &\geq \mathsf{r}_{\text{comf}}
\end{aligned} \tag{6}$$

Hence we have to find a movement circle satisfying the above equations. In mode `ChooseLeft` the second line of Eq. 6 has to be $\mathsf{y}_{\text{circ}} = \mathsf{reference\_lane} - \mathsf{r}_{\text{target}}$ whereas in mode `ChooseRight` it has to be $\mathsf{y}_{\text{circ}} = \mathsf{reference\_lane} + \mathsf{r}_{\text{target}}$.

Note, that a target movement circle is only chosen if the car is not parallel to the lane ($0° \neq \beta_{\text{ori}} \neq 180°$) and that a car has a maximal orientation β_{max} which we assume to be less than 90°. So that the above system of equations is well defined.

Determining Whether a Movement Circle is Feasible The SC influences the VC by changing the agent's orientation. When the the agent's orientation is not equals zero (i.e. the agent is not following the traffic flow) but wants to accelerate, it has to check whether it can still keep its lane. The VC therefore evaluates whether accIsFeasible(v,acc,β,offset) equals true. Intuitively, we say that accIsFeasible(v,acc,β,offset) equals true, if the agent can accelerate for a given constant time $\mathsf{t}_{\mathsf{accelerate}}$ reaching a velocity v_2 and with this velocity it can still find a comfortable movement circle that keeps it on its lane. Thus accIsFeasible(v,acc,β,offset) equals true, if the following system of equations has a solution.

$$
\begin{aligned}
\mathsf{y}_{\mathsf{circ}} &= -\frac{\mathsf{x}_{\mathsf{circ}}}{\tan\beta_{\mathsf{ori}}} + \mathsf{y}_{\mathsf{centre}} + \frac{\mathsf{x}_{\mathsf{centre}}}{\tan\beta_{\mathsf{ori}}} \\
\mathsf{y}_{\mathsf{circ}} &= \mathsf{horizontal\_line} \pm \mathsf{r}_{\mathsf{target}} \\
\mathsf{r}_{\mathsf{target}} &= \sqrt{(\mathsf{x}_{\mathsf{centre}} - \mathsf{x}_{\mathsf{circ}})^2 + (\mathsf{y}_{\mathsf{centre}} - \mathsf{y}_{\mathsf{circ}})^2} \\
\mathsf{r}_{\mathsf{target}} &< \frac{\mathsf{mass}_{\mathsf{passenger}} \cdot (\mathsf{v} + \mathsf{t}_{\mathsf{accelerate}} \cdot \mathsf{acc})^2}{\mathsf{F}_{\mathsf{comf}}} \\
\mathsf{horizontal\_line} &\geq \mathsf{lane} \cdot \mathsf{lanewidth} + \mathsf{margin}_{\mathsf{safety}} \\
\mathsf{horizontal\_line} &< (\mathsf{lane}+1) \cdot \mathsf{lanewidth} - \mathsf{margin}_{\mathsf{safety}}
\end{aligned}
\tag{7}
$$

Here $\mathsf{margin}_{\mathsf{safety}}$ over-approximates the difference between the vertical position of the agent's centre and its outer front/rear. Line 1-3 express that there is a movement circle for the agent that brings it onto a horizontal line. According to line 4 this circle can be driven comfortably using the speed reached after accelerating for time $\mathsf{t}_{\mathsf{accelerate}}$ (cf. 2.1.2). The reached horizontal line is within the current lane outside the safety margin according to lines 5-6.

2.2 Summary

In this section we presented the controllers of the `AUT` layer. These controllers together specify the autonomous behaviour of an agent and thereby provide a set of reactive skills. The control of a vehicle is determined by two loosely coupled controllers, one for steering and one for velocity control. Both controllers implement a policy to reduce the manoeuvre time and changes in its control outputs, i.e. changes in the vehicle's movement and acceleration/deceleration, respectively. Further the steering and velocity controllers respect comfort constraints: There is a fixed maximum for comfortable acceleration/deceleration and the centrifugal force a vehicle experiences is at most equals a fixed maximum that is still experienced as comfortable.

In the next section we will see how the reactive skills provided by `AUT` are used by the `PROT` layer to build up a lane change manoeuvre.

3 Protocol Layer

While AUT implements simple autonomously controlled reactive skills based on sensor readings, PROT uses these skills to build manoeuvres where agents are coordinated by communication. In other words PROT specifies protocols by which the agents can perform cooperative manoeuvres. Agents coordinate by passing messages and the protocols specify how this is done to realise a complex manoeuvre The current model uses synchronous message passing in the protocols. We assume that all received messages belong to the current manoeuvre run and all messages are to be trusted. We further assume that no messages are lost.

In the following, we will show how PROT builds a lane change manoeuvre from the skills provided by the AUT layer as described in the previous section.

3.1 An Overview of the Lane Change Manoeuvre Protocol

We start by informally describing how the lane change manoeuvre proceeds.

The manoeuvre starts when a car wants to change onto a neighbouring lane. It then assesses the traffic situation on the target lane. If the target lane is not occupied, it performs a lane change on its own. But if the lane is occupied the initiator dynamically determines feasible helpers and negotiates with them to find its actual helper. Only when a helper has agreed, the initiator performs its lane change.

So the protocol considers the following roles: An agent acts as an *initiator* if it intends to change to one of its neighbouring lanes. It then asks certain other agents on the target lane for help, the *feasible helpers*. The one feasible helper that agrees to help, takes on the role of a *helper* and is then contracted to a certain behaviour.

Fig. 16 illustrates the two different traffic situations at which a lane change can be performed: either the target lane is free or the target lane is occupied.

Free Target Lane Certainly the lane change is much simpler if the target lane is not occupied. The agent car has simply to check via its sensors whether the lane is free and it has to make sure that no other car intends to change lane and may thereby drive into its way. Then the agent may steer onto its target lane performing the actual lane change.

Occupied Target Lane When the target lane is occupied, the lane change manoeuvre is more complex. The initiator first makes an estimation about which cars on its target lane are feasible helpers. It then asks these agents to be its helper. When an agent agrees to be a helper, it is obliged to keep the safety distance to the initiator for the amount of time agreed upon. The initiator will change lane by positioning itself in front of its helper. If there is a car ahead of the helper, the initiator adjusts its velocity to establish the safety distance to this car, which will be its new car ahead. The initiator changes lane when it has established the safety distance to its new car ahead—if it exists—and the helper established the safety distance to the initiator.

Fig. 17 sketches the case of the lane change manoeuvre where a helper and the helper's ahead are present.

An agent is always obliged to keep the safety distance to its ahead. Thus, the safety distances between cars on one lane are always respected. A helper additionally establishes a safety distance to the initiator as if it was on the helper's lane and likewise the initiator establishes a safety distance to the helper's car ahead (if present). It is easy to see that the Approach Velocity Controller can

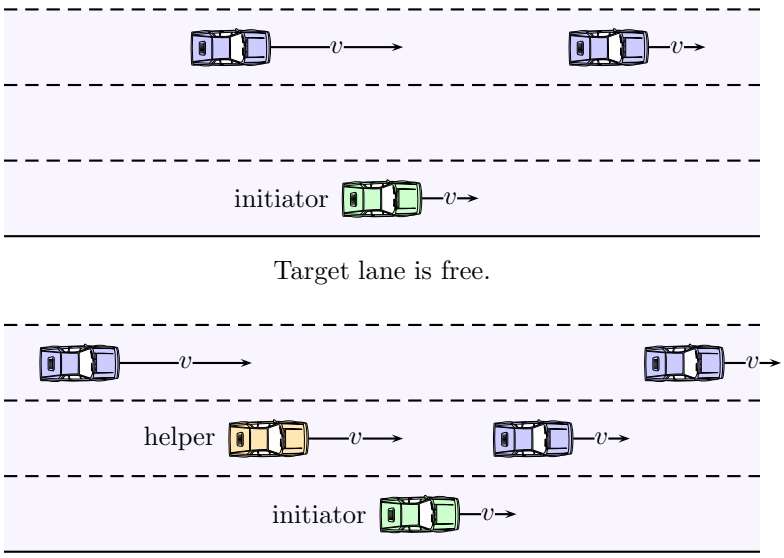


Figure 16: Typical traffic situations the protocol has to handle.

be employed to ensure the safety distances (cf. Sect. 2.1.1 on p. 8). In the following we introduce the lane change protocol more formally. We will see that the protocol invokes services of `AUT`. The implementation of the service invocation and also the details of estimations which agents are feasible are given later in Sect. 4 and Sect. 3.4, respectively.

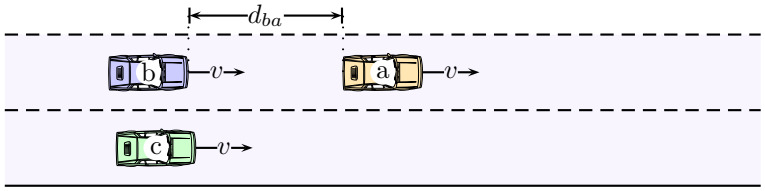
Automata After having informally described how a lane change manoeuvre proceeds according to our lane change protocol, we now introduce a formal specification of the protocol via timed, synchronously communicating automata. The automatons' edges are labelled with a pair [guard/action] where the guard specifies a condition that has to be satisfied to transit from the source to the target state. When a transition is executed its action is performed. We use events of receiving a message also as guards in the sense of *the receiving event occurs*. Borrowed from the world of hybrid automata, we use real valued variables to model clocks: States may specify a derivative d, d$\in \{1, -1\}$ which describes whether the clock increases or—like a stop watch—decreases with time. Clock variables can be discretely updated by an transition's action and continuously evolve according to the flow of the current state.

We have already seen that an agent can take on different roles. The protocol specification reflects this by defining an automaton for the initiator and an automaton for (feasible) helpers.

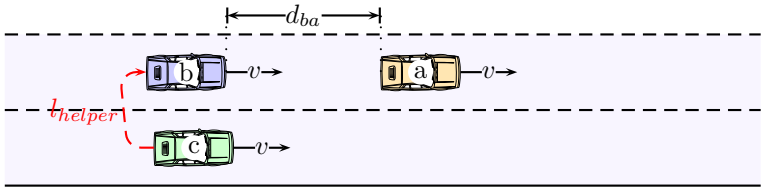
3.2 Initiator Role

In Fig. 18 the timed automaton for the initiator protocol is given. The derivatives are inscribed into its states and states are descriptively labelled. The edge labels are given in Table 4.

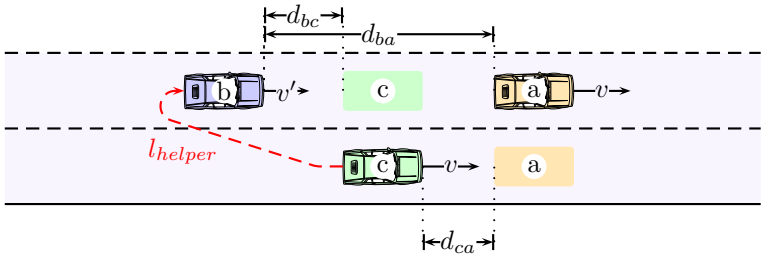
As an overview, we give an informal walk-through of the automaton.



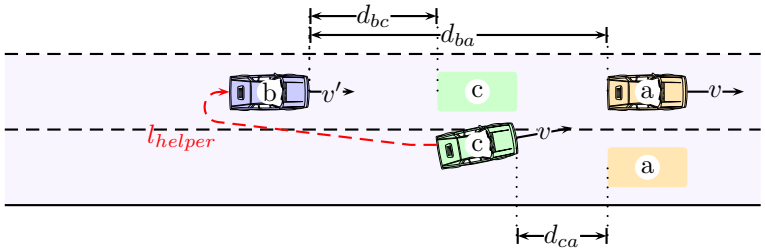
Agent c makes a lane change request.



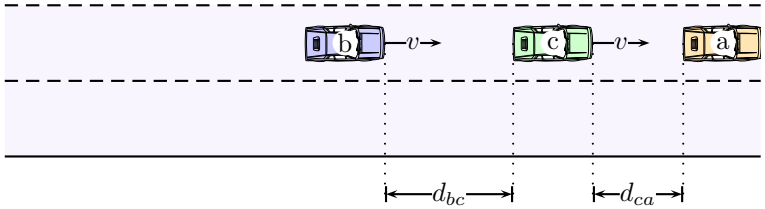
On request of c agent b agreed to act as a "helper".



Agent b controls actuators to establish safety distances to a and c.
Agent c establishes a safety distance to a.



Safety distances are maintained continuosly, while c changes lanes.



The helper contract is released.

Figure 17: Overview of the Lane Change Manoeuvre.

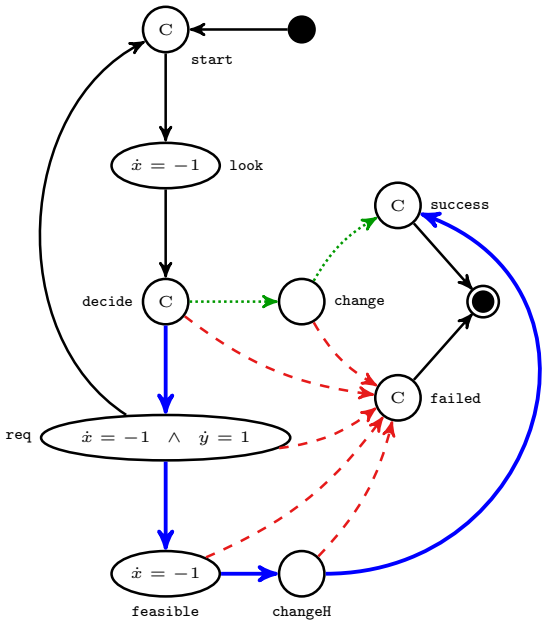


Figure 18: Initiator Protocol. Red (dashed) arcs lead to failure of the lane change manoeuvre. The lane change can succeed without helper (marked dotted in green) or with helper (thick in blue).

`start`→ `look` **Request a set of best feasible helpers.**
The initiator starts by assessing the traffic situation on its target lane. The autonomous layer is asked to use its sensors and estimate which of the cars on the target lane are feasible helpers, taking into account the agents which have already declined. The autonomous layer can either answer (1) there are no agents on the target lane ($helpers = \{\bot\}$) or (2) there are no feasible helpers ($helpers = \emptyset$) or (3) these are the feasible helpers ($\emptyset \neq helpers \neq \{\bot\}$).

`look`→ `decide` **AUT provides feasible helpers.**
The autonomous layer did assess whether there are feasible helpers on the target lane and informs the initiator's protocol layer.

`decide`→ `change` **If there are no agents, change lane.**
The initiator decides to undertake a lane change on its own, if the autonomous layer reports that there are no other agents on the target lane.

`decide`→ `req` **If there is a feasible helper, choose a helper h.**
The initiator asks a feasible helper to be its helper.

`decide`→ `failed` **If there are no feasible helpers, abort the lane change manoeuvre.**
In case the target lane is occupied and all feasible helpers declined to be helpers, the lane change manoeuvre is aborted.

`req`→ `start` **If a help request was declined, assess anew.**
A feasible helper, when asked to be a helper, may not agree. In this case the initiator assesses the situation on the target lane anew.

edge	guard	action
`init`→ `start`	true /	$declined' := \emptyset$, $x' := timeout_{lc}$
`start`→ `look`	true /	aut!(`findHelpers`, $lane$, $declined$)
`look`→ `decide`	aut?($helpers$) /	
`decide`→ `req`	$\emptyset \neq helpers \neq \{\perp\}$ /	$h' :\in helpers$, h'!(`lcReq`), $y' := 0$
`decide`→ `failed`	$helpers = \emptyset$ /	
`decide`→ `change`	$helpers = \{\perp\}$ /	prot!(`doLC`, $lane$, x)
`req`→ `feasible`	h?(`yes`, $timeout_{help}$) /	$timeout' := \min\{x, timeout_{help} - y\}$, aut!(`feasible`, h, $timeout$)
`req`→ `start`	h?(`no`) $\vee$ $y \geq timeout_{com}$ /	h!(`abort`), $declined' := declined \cup \{h\}$
`req`→ `failed`	$x \geq 0$ /	h!(`abort`)
`feasible`→ `changeH`	aut?(`yes`) /	h!(`yes`), prot!(`doLC`, h, $timeout$)
`feasible`→ `failed`	prot?(`no`) /	h!(`no`)
`changeH`→ `failed`	prot?(`abort`) /	h!(`abort`)
`changeH`→ `success`	prot?(`done`) /	h!(`done`)
`change`→ `failed`	prot?(`abort`) /	
`change`→ `success`	prot?(`done`) /	
`success`→ `final`	true /	
`failed`→ `final`	true /	

Table 4: Edge Labels for the Initiator Automaton of Fig. 18.

`req`→ `feasible` **If a helper agreed, check if the offered time-out is feasible.**
The helper agrees by letting the initiator know about the duration it is willing to help. The autonomous layer of the initiator evaluates whether the offered time frame is feasible.

`feasible`→ `changeH` **If the time frame is feasible, start the lane change.**
In case the time frame offered by the helper is feasible, the lane change is started.

`changeH`→ `failed`, `change`→ `failed` **Lane change did not succeed.**
From the time on, that the initiator decides to actually try to perform a lane change, the protocol specifies a couple of safety measurements for a safe lane change. If one of them fails or the time for lane change runs out, the manoeuvre is aborted. The safety measures are specified by the automaton in Fig. 19.

`req`→ `failed` **Time is out.**
The initiator is given a maximal manoeuvre time. When the attempt of finding a feasible helper takes longer, the manoeuvre is aborted.

The walk-through of the automaton of Fig. 18 left open how the lane change is actually realized. Table 4 states that the protocol layer sends the message *prot*!(`doLC`, $lane$, x) when a lane change is attempted without helper and *prot*!(`doLC`, h, *time-out*) for a lane change with helper h, meaning that a message is send to the protocol layer where it is told to do a lane change within the time frame *time-out*. If the lane change is attempted without helper the target lane is specified as $lane$ otherwise the helper is given and the initiator has to change onto the helper's lane. Both of these two messages trigger the automaton of Fig. 19. This automaton specifies the further proceeding (and the safety measures to be taken) for both scenarios. Although the automaton of Fig. 19 could as well be directly integrated into that of Fig. 18, we specified it separately to avoid duplication of this protocol procedure and to underline that it is the same for both scenarios.

state	$\frac{\mathsf{d}}{\mathsf{dt}}$
`ready`	$\frac{\mathsf{d}}{\mathsf{dt}} t_{finished} = 0,\ \frac{\mathsf{d}}{\mathsf{dt}} t_{signal} = 0$
`await gap`	$\frac{\mathsf{d}}{\mathsf{dt}} t_{finished} = 1,\ \frac{\mathsf{d}}{\mathsf{dt}} t_{signal} = 0$
`look out`	$\frac{\mathsf{d}}{\mathsf{dt}} t_{finished} = 1,\ \frac{\mathsf{d}}{\mathsf{dt}} t_{signal} = 1$
`signal`	$\frac{\mathsf{d}}{\mathsf{dt}} t_{finished} = 1,\ \frac{\mathsf{d}}{\mathsf{dt}} t_{signal} = 1$
`lane change`	$\frac{\mathsf{d}}{\mathsf{dt}} t_{finished} = 1,\ \frac{\mathsf{d}}{\mathsf{dt}} t_{signal} = 0$

Table 5: Derivatives in the Automaton in Fig. 19.

Again the states of the automaton are descriptively labelled. The state derivatives are given in Table 5 and the edge labels are given in Table 6.

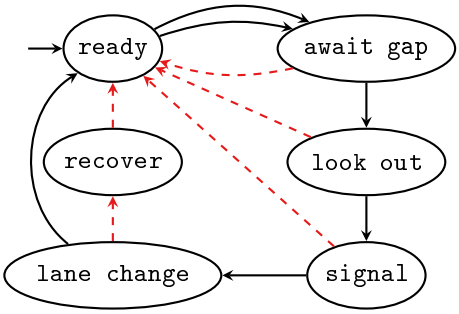


Figure 19: Protocol for Changing Lane. The lane change succeeds along black arcs. Red, dashed arcs signal some failure.

`ready`→ `await gap` **Ready to attempt a lane change.**
The control is switched from the automaton of Fig. 18 to that of Fig. 19 by sending the message *prot*!(`doLC`) and the automaton of Fig. 18 will only proceed when told to do so by the automaton of Fig. 19. The message *prot*!(`doLC`) means that now a lane change is attempted. We next give a quick walk-through of the signalling automaton of Fig. 19.

`await gap`→ `look out` **When there is a sufficient gap, check for signalling competitors.**
When the sensors report that a sufficiently large gap on the target lane has been established, the initiator checks whether it risks to collide with other cars simultaneously attempting to change lanes, i.e. it checks whether there is another car signalling its intend to change lane into the initiator's target gap.

`look out`→ `signal` **If no competitors signalled, set signals.**
After checking whether other cars want to change lane, the initiator sets its own signals to inform other agents of its intended lane change.

`signal`→ `lane change` **Change onto target lane.**
After having set the signals for a long enough time for other agents to react, the autonomous layer is now told to perform the change of lanes.

edge	guard	action
`ready`→ `await gap`	prot?(doLC,lane, *timeout*) /	$t'_{finished} = 0$, aut!(ensureGap,lane)
`ready`→ `await gap`	prot?(doLC,helper, *timeout*) /	$t'_{finished} = 0$, aut!(ensureGap,helper)
`await gap`→ `look out`	sens?gapOK $\wedge\ t_{finished} < 0$ /	$t'_{signal} = 0$, sens!watchSignals
`look out`→ `signal`	$t_{signal} \geq duration_{watch} \wedge t_{finished} < timeout$ /	aut!setSignals(lane), $t'_{signal} = 0$
`signal`→ `lane change`	$t_{finished} < timeout \wedge t_{signal} \geq duration_{signal}$ /	aut!(changeLane, lane)
`lane change`→ `ready`	aut?laneChanged/	aut!dismissGap, sens!ignoreSignals, prot!success
`lane change`→ `recover`	sens?alert /	aut!recover
`recover`→ `ready`	aut?recovered /	aut!dismissGap, aut!signalOff, prot!failed
`await gap`→ `ready`	$t_{finished} \geq timeout$ /	aut!dismissGap, prot!failed
`look out`→ `ready`	$t_{finished} \geq timeout \vee$ sens?signalAlert /	aut!dismissGap, sens!ignoreSignals, prot!failed
`signal`→ `ready`	$t_{finished} \geq timeout \vee$ sens?alert /	aut!dismissGap, act!signalOff,sens!ignoreSignals, prot!failed

Table 6: Edge Labels for the Lane Change Automaton of Fig. 19.

`lane change`→ `ready` **Lane change is done.**
When the autonomous layer reports that the lane change succeeded, concurrently running sensor and autonomous services are signalled to terminate and the control is switched back to the automaton in Fig. 18.

`lane change`→ `recover` **Driving onto the target lane failed.**
The sensors report that safety distances are violated, so that the lane change is aborted and recovery measures are triggered.

`recover`→ `ready` `AUT` **gets ready for a new attempt.**
When the autonomous layer reports a successful recovery, other services of the autonomous layer still running are told to terminate, then the control is switched back to the automaton of Fig. 18 by sending the message prot!failed.

`await gap`→ `ready`, `look out`→ `ready`, `signal`→ `ready` **Time is out.**
In case the maximal manoeuvre time is exceeded, the manoeuvre is ended.

Timing The protocol imposes several timing constraints on the initiator's behaviour during a lane change manoeuvre. In the sequel these constraints are discussed:

- *time-out*$_{lc}$
  For a lane change manoeuvre a maximum time duration, *time-out*$_{lc}$, is given. The protocol monitors that the lane change manoeuvre does not exceed this time limit. Therefore the clock x is started in state `start` and takes the remaining manoeuvre time.

- *time-out*$_{com}$
  time-out$_{com}$ specifies the maximum time the negotiation with a feasible helper may take. When this time exceeds, the transition from `req`→ `start` is taken.

- *duration*$_{watch}$
  The initiator has to watch whether other cars signal their intend to change lane for at least *duration*$_{watch}$.

- $duration_{signal}$
  Right before the initiator steers onto the target lane, it has to set signals for at least $duration_{signal}$.

3.3 Helper Role

So far we have seen the protocol specification for the initiator of a lane change and know that the initiator asks other agents, that he thinks are feasible helpers, for help. An agent, when asked for help, offers a time frame during which it is willing (or able) to help. When the initiator thinks that a lane change within this time frame is feasible, the agent is accepted as a helper and thus contracted to keep the safety distance to the initiator. The timed automaton in Fig. 20 specifies the protocol for an agent in the role of a feasible helper. The derivatives are inscribed in the states and the edge labels are given in Table 7.

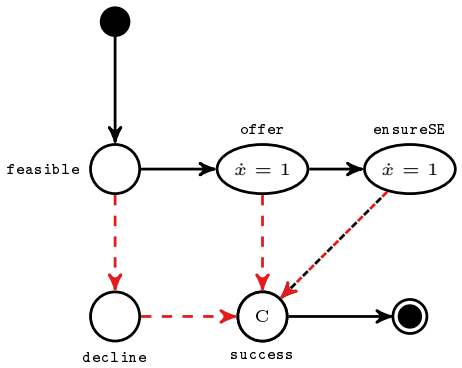


Figure 20: Helper Protocol. Marked dashed in red is the behaviour when the feasible helper does not become a helper.

edge	guard	action
`init`→ `feasible`	true /	aut!(`hFeasible`, ag)
`feasible`→ `offer`	aut?($timeout_{help}$) /	ag!(`yes`, $timeout_{help}$), $x' := 0$
`feasible`→ `decline`	aut?(`no`) /	ag!(`no`)
`offer`→ `ensureSE`	ag?(`yes`) /	aut!((respect,ag))
`offer`→ `success`	ag?(`no`) $\vee$ $x \geq timeout_{help}$ /	
`ensureSE`→ `success`	ag?(`done`) $\vee$ ag?(`abort`) $\vee$ $x \geq timeout_{help}$ /	aut!((forget,ag))
`decline`→ `success`	ag?(`abort`) /	
`success`→ `final`	true /	$result$!(`success`)

Table 7: Edge Labels for the Helper Protocol Automaton in Fig. 20.

Again we give an informal walk-through of the automaton.

`init`→ `feasible` **Assessing feasible time frame for helping.**
As first step a potential helper tells its autonomous layer to assess in which time frame it is feasible for him to help.

`feasible`→ `decline` **Helping is not possible.**
The autonomous layer reports that it is not feasible to help the initiator, so the helper rejects the initiator's request.

`feasible`→ `offer` **Offer feasible time frame for helping.**
The autonomous layer determines a time frame for which helping the initiator is feasible. This time frame is communicated to the initiator.

`offer`→ `success` **Initiator rejects offered time frame.**
The transition `offer`→ `success` is taken in two scenarios: (i) The initiator may assess itself that a time frame offered by the potential helper is not feasible for it and rejects the offer. (ii) The time for helping runs out. In both cases, the cooperation between the two agents is aborted.

`offer`→ `ensureSE` **Initiator accepts offered time frame.**
The time frame offered by the helper is accepted. Now the helper is obliged to respect the safety distance to the initiator.

`ensureSE`→ `success` **Helper stops helping.**
The helper stops respecting the safety distance, when the initiator dismisses the helper or the time is up.

`decline`→ `success` **Initiator dismisses unsuitable helper.**
The helper told the initiator that helping is not feasible for him. The initiator then dismisses the helper.

Timing The protocol imposes just one timing constraint on the helper, $\textit{time-out}_{help}$. When this time is up, the contract between helper and initiator is automatically terminated.

3.4 Predictions by Helper and Initiator

During evolution of AHS, situations arise where vehicles have to choose other vehicles as cooperation partners. In our model of the lane change manoeuvre a car has to make predictions on the future evolution of the traffic situation around it in order to predict

- as an initiator
  - which cars on the target lane are feasible helpers
    The protocol triggers such a service of the autonomous layer by sending message aut!(`findHelpers`, *lane*) *(cf. Table 4).*
  - which feasible helper is asked first.
  - whether a lane change is feasible within the time frame offered by a helper
    The protocol sends its autonomous layer aut!(`feasible`, h, *time-out*) *(cf. Table 4).*
- as a helper whether helping is feasible (cf. aut!(`hFeasible`, ag) in Table 7) and which time is to be offered.

Feasibility Predictions of an Initiator We call a $\mathsf{car_A}$ a feasible helper of an initiator $\mathsf{car_B}$, if we can predict a *successful* lane change for $\mathsf{car_B}$ assuming that $\mathsf{car_A}$ cooperates with $\mathsf{car_B}$ and the uncontrollable system part shows nominal behaviour. A lane change is performed successfully if it is performed in a safe and comfortable manner respecting timing constraints, i.e. the lane change manoeuvre satisfies the following three constraints:

1. Safety: The controllable system part stays safe (safety envelopes are respected).
2. Timing: The manoeuvre is completed within a certain time window.
3. Comfort: The cars' controls stay within comfort ranges for accelerations and velocities.

These constraints impose a relation between comfortability of a manoeuvre and the time to complete it.

Description of Solution Space Whether a manoeuvre is feasible, can be described by a system of constraints that restricts the possible future system evolution to those that result in situations where a lane change may be possible. Note, that it is not necessary to guarantee that the constraints imply that a lane change is actually possible, since the `AUT` layer ensures that safety distances are always respected irrespective of feasibility predictions. In case a manoeuvre is considered dangerous, the `AUT` layer informs the `PROT` layer, which in turn triggers recovery measures of the `AUT` layer (cf. Fig. 19 on p. 27).

Let $\mathsf{car_A}$ be a candidate feasible helper of the initiator $\mathsf{car_B}$. Assume, we have access to an oracle Ω which provides information on the behaviour of cars in the surrounding of $\mathsf{car_B}$. Given an (uncontrollable) car $\mathsf{car_C}$, we denote by $\mathsf{car_C}^{\Omega(t)}$ its state (acceleration, velocity, displacement) after t time units. Analogously, for the controllable agents of the manoeuvre, we introduce $\mathsf{car}_*^{\Lambda(t)}$, the state of car_* after t time units where $* \in \{\mathsf{A}, \mathsf{B}\}$. With $\mathsf{SD}(\mathsf{car}_*, \mathsf{car}_\#)$ we denote the formula for the safety distance between car_* following $\mathsf{car}_\#$ as given in Sect. A.1 and $\mathsf{dist}(\mathsf{car}_*, \mathsf{car}_\#)$ describes the distance between car_* and $\mathsf{car}_\#$ in terms of their expected dynamics starting from a initial position. By $\mathsf{ahead}(\mathsf{car}_*)$ we denote the car ahead of car_*—w.l.o.g. we assume there is always a car ahead. We assume that the cars all stay in their lanes.

The following system of inequalities expresses that after $\mathsf{t_{gap}}$ time units there is a sufficiently large gap in front of $\mathsf{car_A}$ until $\mathsf{t_{done}}$ and $\mathsf{car_B}$ is aside the gap (cf. lines 1 and 2 of Eq. 8), while safety distances are respected for the whole time (cf. lines 3 and 4 of Eq. 8).

$$\begin{array}{lll} & \forall \mathsf{t_{gap}} \leq t \leq \mathsf{t_{done}}: & \mathsf{SD}(\mathsf{car_A}^{\Lambda(t)}, \mathsf{car_B}^{\Lambda(t)}) \leq \mathsf{dist}(\mathsf{car_A}^{\Lambda(t)}, \mathsf{car_B}^{\Lambda(t)}) \\ \wedge & \forall \mathsf{t_{gap}} \leq t \leq \mathsf{t_{done}}: & \mathsf{SD}(\mathsf{car_B}^{\Lambda(t)}, \mathsf{ahead}(\mathsf{car_A})^{\Omega(t)}) \leq \mathsf{dist}(\mathsf{car_B}^{\Lambda(t)}, \mathsf{ahead}(\mathsf{car_A})^{\Omega(t)}) \\ \wedge & \forall 0 \leq t \leq \mathsf{t_{done}}: & \mathsf{SD}(\mathsf{car_A}^{\Lambda(t)}, \mathsf{ahead}(\mathsf{car_A})^{\Omega(t)}) \leq \mathsf{dist}(\mathsf{car_A}^{\Lambda(t)}, \mathsf{ahead}(\mathsf{car_A})^{\Omega(t)}) \\ \wedge & \forall 0 \leq t \leq \mathsf{t_{done}}: & \mathsf{SD}(\mathsf{car_B}^{\Lambda(t)}, \mathsf{ahead}(\mathsf{car_B})^{\Omega(t)}) \leq \mathsf{dist}(\mathsf{car_B}^{\Lambda(t)}, \mathsf{ahead}(\mathsf{car_B})^{\Omega(t)}) \end{array} \tag{8}$$

We instantiate Eq. 8 by fixing values for $\mathsf{t_{done}}$ and $\mathsf{t_{gap}}$ and by restricting the accelerations to be in certain intervals. We use an oracle Ω that predicts uninvolved agents to simply keep their initial acceleration and further assume that a controlled agent chooses an arbitrary but fixed acceleration. We consider a minimal $\mathsf{t_{gap}}$ and use as $\mathsf{t_{done}}$ a value satisfying

$$\mathsf{t_{done}} = \mathsf{t_{gap}} + \text{“duration to change lane at given velocity”}.$$

The acceleration interval is handled by examining the lower and upper bounds for acceleration. With these instantiations, the system of inequalities can be directly solved.

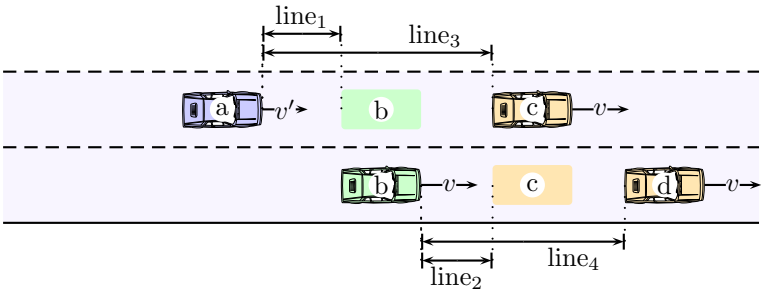


Figure 21: The constraints of Eq. 8 imply that a sufficiently large gap exists in front of helper a for time $\mathsf{t}_{\mathsf{gap}}$ to $\mathsf{t}_{\mathsf{done}}$.

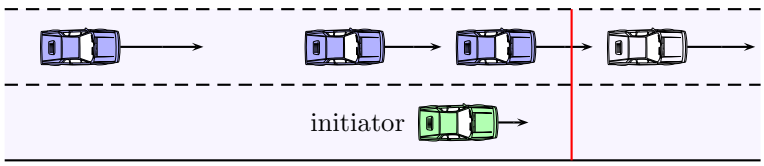


Figure 22: Frontal bound for feasible helper candidates. With comfortable acceleration the initiator can only reach certain gaps ahead of it on the target lane. The bound is determined based on the speed difference of the initiator and the estimated speed on the target lane.

Determining Feasible Helpers The initiator protocol specifies that in each iteration of the protocol some candidate is chosen from the set of feasible helpers which is then asked to take on the helper role.

The equation Eq. 8 can now be used to determine feasible helpers. Feasible helpers have to satisfy the constraint system. To identify feasible helpers, for each car on the target lane in proximity of the initiator, the set of vehicles is determined for which Eq. 8 has a non-empty solution space. Every vehicle of this set is considered a feasible helper. We assume a uniform evolution of a benign system where the absolute accelerations of uninvolved cars are relatively small. As an estimate of the velocity of the traffic flow on the target lane, we build the average velocity of several cars nearby. We refrain from considering exact values for each car individually, as acquiring this knowledge either implies additional communications between agents or elaborate adoption of sensors.

The constraints on timing and comfort allow to derive an upper bound on how far ahead of the initiator a feasible helper can be, such that no agent further ahead is a feasible helper (cf. Fig. 22).

We do not infer such a bound to restrict the area behind the initiator, since every car behind the initiator could be a feasible helper if the gap in front of it expands up to the initiator. However, if a car some certain distance behind the initiator declines to be a helper, no car further behind is asked to be a helper, as the declining car is uninvolved and limits the gaps and the movement capabilities of the following cars. At a certain distance the gap would not be reachable while satisfying the timing constraints.

Further, coming from the idea "if there is enough time, go comfortably" we propose to weight comfort against minimal completion time in order to choose the best feasible helper to be asked. Therefore we utilize a function $urgency(i, m) : \mathbb{N}_{>0} \times \mathbb{N}_{>0} \to [0, 1] \subset \mathbb{R}$ which maps each iteration of the protocol (up to a threshold iteration m—the expected number of iterations) to a real value

between 0 and 1. Here, $urgency(i, m) = 0$ means the "comfortable" values for acceleration and braking are used and $urgency(i, m) = 1$ stands for the "urgent" values of acceleration and braking. A naive instance of the urgency function is

$$urgency(i, m) := \begin{cases} 1, & \text{if } i \geq m \\ (i-1)/m, & \text{else} \end{cases},$$

which increases urgency in equidistant steps from 0, for iteration 1, to 1, for iterations m and above. For example, to predict the future state(s) of car_B we use as an upper bound on the acceleration acc_B in Eq. 8 where acc_B is

$$acc_\mathsf{B} := acc_\mathsf{low} + urgency(i, m) \cdot (acc_\mathsf{high} - acc_\mathsf{low}),$$

with adequate values for acc_low and acc_high, e.g. $\frac{1}{2} \cdot acc_{comfort}$ and $acc_{comfort}$. Using the above naive urgency function the difference between $acc_{comfort}$ and acc_max is divided in m equidistant steps. For the first iteration of the protocol, $acc_{comfort}$ is chosen and from the m-th on, acc_max is chosen.

So we consider vehicles as feasible helper if they can generate a sufficiently large gap using acc_i and decel_i, the acceleration accepted in the protocol iteration i.

Feasibility Predictions of a Helper During a run of the protocol, a feasible helper may be requested by an initiator to take on the helper role for a certain amount of time. In contrast to the partner-advice for initiator, upon reception of an aut!(`hFeasible`, ag) message, the feasible helper has to solve the system of inequalities only for itself as helper, to check if the request is sane. It is then able to answer with the respective amount of time it is willing participate in the manoeuvre. In particular, the feasible helper may solve the inequalities for minimal time analogously to the initiator and estimate its amount of time to participate based on the expected minimal time for the manoeuvre.

Determining whether a Manoeuvre is Feasible within a Given Time Frame We consider a lane change manoeuvre as feasible, if a lane change manoeuvre for the initiator with the current helper satisfies Eq. 8.

Conclusion The proposed feasibility predictions rely on assumptions concerning the future evolution of the traffic situation w.r.t. cars which are uninvolved. When quantifying the effectiveness and efficiency of the feasibility functions for the lane change manoeuvre, validity of the predictions has to be taken into account. If the system evolves as predicted and the determined feasible helper accepts its helper role, then a situation can be enforced within the given time frame, such that the initiator arrives aside a gap in front of feasible helper which is large enough to allow a safe lane change.

The feasibility analysis presented is not safety critical, as the car's control is responsible for maintaining a safe state. However, the feasibility analysis has to be adequate in the sense that, on one hand, if it is too optimistic, feasible helpers are identified for which the manoeuvre is unlikely to be completed successfully and on the other hand, if it is too pessimistic, cars which would make good feasible helpers are overlooked.

4 Translation Controller

To keep the autonomous layer as simple (and general) as possible, to allow component reuse and separate concerns, we introduced *translation controllers*. Translation controllers translate between different layers, they map the identity centred messages of the protocol to the world of variable values monitoring controllers of the autonomous layer. The autonomous layer receives from the protocol messages like `prot?(ensureGap,car)`, `prot?(respect,car)` or `prot?(changeLane, lane)`, whereas the controllers of the autonomous layer are influenced by the values of variables dist, β_{ori}, $\text{dist}_{\text{goal}}$, v_{goal} and offset. These values are determined w.r.t. the reference objects new_ahead, car_ahead, `referenceline`. Translation controllers update references to entities new_ahead, car_ahead, `referenceline` and generate messages while monitoring the state of these entities. We denote the case that there is no new_ahead as new_ahead $=\perp$ and analogously we write car_ahead $=\perp$ meaning there is no car_ahead.

$$\text{v}_{\text{goal}} = \begin{cases} \min(\text{vel}(\text{car\_ahead}), \text{vel}(\text{new\_ahead})) & \text{if new\_ahead} \neq \perp \neq \text{car\_ahead} \\ \text{vel}(\text{car\_ahead}) & \text{if new\_ahead} = \perp \neq \text{car\_ahead} \\ \text{vel}(\text{new\_ahead}) & \text{if car\_ahead} \neq \perp = \text{car\_ahead} \\ \text{vel}_{\text{cruise}} & \text{else} \end{cases} \tag{9}$$

$$\text{dist} = \begin{cases} \text{dist}(\text{new\_ahead}, \mathit{self}) & \text{if } (\text{new\_ahead} \neq \perp = \text{car\_ahead} \vee \\ & \quad \text{new\_ahead} \neq \perp \neq \text{car\_ahead} \wedge \\ & \quad \text{new\_ahead}.x < \text{car\_ahead}.x) \\ \text{dist}(\text{car\_ahead}, \mathit{self}) & \text{if } (\text{new\_ahead} = \perp \neq \text{car\_ahead} \vee \\ & \quad \text{new\_ahead} \neq \perp \neq \text{car\_ahead} \wedge \\ & \quad \text{car\_ahead}.x < \text{new\_ahead}.x) \\ \infty & \text{else.} \end{cases} \tag{10}$$

$$\text{dist}_{\text{goal}} = \begin{cases} \text{SD}(\mathit{self}, \text{car\_ahead}) & \text{if } (\text{new\_ahead} = \perp \neq \text{car\_ahead} \vee \\ & \quad \text{new\_ahead} \neq \perp \neq \text{car\_ahead} \wedge \\ & \quad \text{new\_ahead}.x - \text{SD}(\mathit{self}, \text{new\_ahead}) < \\ & \quad\quad \text{car\_ahead}.x - \text{SD}(\mathit{self}, \text{car\_ahead})) \\ \text{SD}(\mathit{self}, \text{new\_ahead}) & \text{if } (\text{car\_ahead} = \perp \neq \text{new\_ahead} \\ & \quad \text{new\_ahead} \neq \perp \neq \text{car\_ahead} \wedge \\ & \quad \text{new\_ahead}.x - \text{SD}(\mathit{self}, \text{new\_ahead}) < \\ & \quad\quad \text{car\_ahead}.x - \text{SD}(\mathit{self}, \text{car\_ahead})) \\ \infty & \text{else.} \end{cases} \tag{11}$$

$$\text{offset} = \mathit{self}.y - \mathit{self}.\text{referenceline} \tag{12}$$

Equation 9 makes the controller of Sect. 2 adjust its velocity to match the minimum velocity of the cars car_ahead and new_ahead. We will see that car_ahead is the car ahead on the same lane and new_ahead is the car behind which a lane changer will enter its target lane.

Eq. 10 together with Eq. 11 allows an agent to simultaneously respect the safety distances to car_ahead and new_ahead. dist will be the distance to the object whose safety envelope is closer to the agent. Eq. 11 accordingly sets $\text{dist}_{\text{goal}}$ to the respective safety distance.

Eq. 12 defines the offset to be the distance to a reference line. A translation controller sets the reference line to the mid of the target lane during a lane change manoeuvre otherwise the reference line is the mid of the current lane. The controller of Sect. 2 then makes the agent change onto the target lane.

In the following we describe the set of translation controllers needed as glue between the protocol given in Sect. 3 and the controllers as given in Sect. 2. Note that some reference values of `AUT` are updated by translation controllers while others are determined by sensor readings, as we will see in Sect. 5.

4.1 Signalling

The initiator protocol (cf. Fig. 18) specifies that a car has to signal when it intends to change lane. When the protocol tells the autonomous layer to set signals during an intended change onto the lane *lane*, the following translation controller sets signals appropriately to the right or to the left and switches the signals off when told so by the protocol.

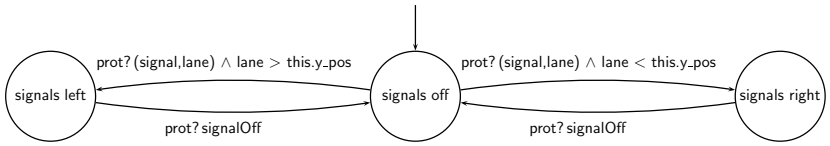


Figure 23: Setting of Signals.

4.2 Lane Change Monitor

The lane change monitor (cf. Fig. 24) sets the target lane as new reference line, to which the steering controller directs the agent car (cf. Sect. 2.1.2 on p. 16). In case the autonomous layer is told by the protocol to recover (prot?recover), the original reference_line is restored, so that the agent returns to the lane it was on when the manoeuvre was initiated. When the target lane is reached with accepted deviation in terms of offset and orientation, the protocol is informed about this (prot!recovered).

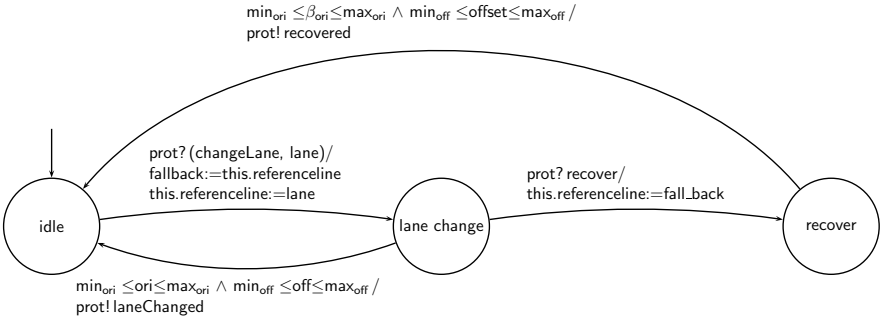


Figure 24: Lane Change Monitor.

4.3 Distance Monitoring During a Lane Change

When driving on a road a car has always to maintain safety distances to other cars. During a lane change manoeuvre an agent also has to establish and simultaneously maintain safety distances to its ahead and its intended new ahead. Similarly, a helper has to establish and simultaneously maintain safety distances to its ahead and the initiator of the lane change. In the following we introduce the translation controller concerned with distance monitoring. We distinguish three cases:

1. The initiator monitors distances during a lane change that is performed without a helper.
2. The initiator monitors distances during a lane change with helper.
3. The helper monitors distances during a lane change.

Lane Change With Helper (cf. Fig. 25) During a lane change manoeuvre the protocol layer asks the autonomous layer to ensure a gap to the car ahead of the helper. For the autonomous layer this means that the agent not only has to keep out of the safety envelope of its car ahead but it also should adjust its velocity and distance so that it eventually respects the safety envelope of the helper's car ahead. The translation controller sets and updates a reference to the car ahead of the helper (= new_ahead) to which the agent is supposed to ensure a gap and generates a message to the protocol layer as soon as the sensors signal that the gap is sufficiently large. For a lane change with helper we consider a gap as large enough, if the agent respects the safety distance to the helpers car ahead (i.e. its new car ahead) and also to its helper.

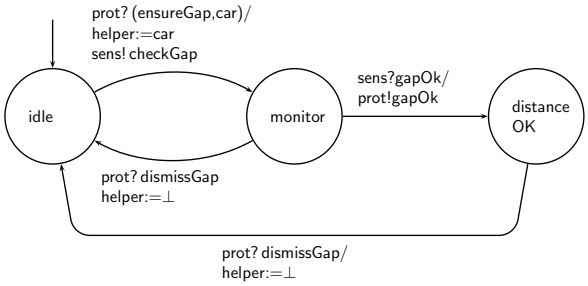


Figure 25: Gap Monitor I.

Lane Change Without Helper (cf. Fig. 26) Of course, a lane change can also be performed if the target lane is not occupied. In this case the agent tells its sensors to check whether its target lane is actually not occupied. If the sensors confirm, the protocol is told that the gap is OK.

Distance Monitoring as Helper (cf. Fig. 27) If the agent agrees to be a helper of an initiator, the translation controller sets the new_ahead reference to the initiator car. Hence its velocity controller will adjust the velocity in order to also respect the safety distance to the initiator. In case the speed difference to the initiator is so big that it gets out of the agent's sight, the agent gives up to get exactly the safety distance to the initiator (new_ahead := ⊥), i.e. it does not adapt its speed any more. But as soon as the initiator reappears the agent adjusts its speed again (new_ahead :=

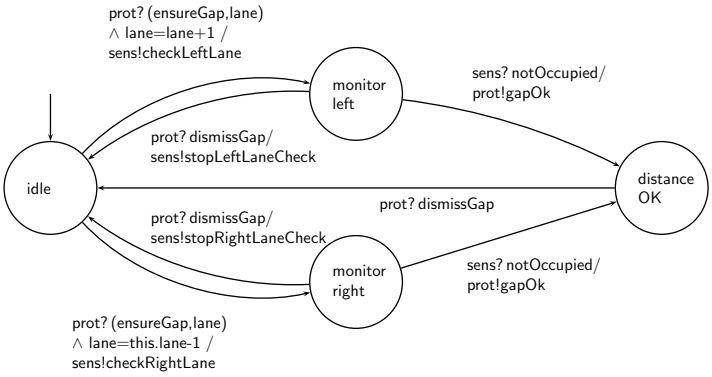


Figure 26: Gap Monitor II.

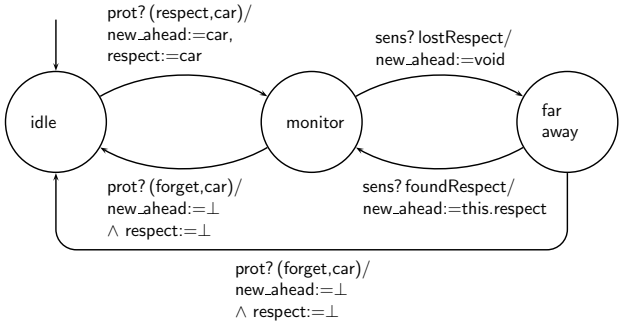


Figure 27: An agent as Helper.

car). The variable `respect` stores a reference to the initiator for the case it temporarily disappears from the agent's sensors. This reference is erased only by the protocol message `(forget,car)`.

5 Sensors

Sensors have the exclusive access to the real world status of a vehicle and its environment. The sensors' readings constitute the vehicle's internal model of its environment, the mental map. For instance, sensors identify which car is currently directly ahead.

We model sensors that are limited to a certain range. When they discover a change, i.e. an inconsistency between their readings and the mental map, appropriate events are triggered. We abstract from specific types of sensors and encapsulate these specifics in predicates like e.g. `isSensible`(car), which evaluates to true when car is currently sensible in the environment.

5.1 Monitoring the car_ahead

Figure 28 describes a sensor monitoring the car directly ahead. The predicate `isSensible(car)` evaluates to true, if `car` is in front of the agent and close enough to be noticed by the sensors of the agent. Expression env.isAhead(self) refers to the car actually in front of the considered agent, that is the predicate is evaluated on the values evolving in the physical environment of the car (cf. Sect. 6). In contrast, this.car_ahead refers to the car that an agent *thinks* is currently directly ahead.

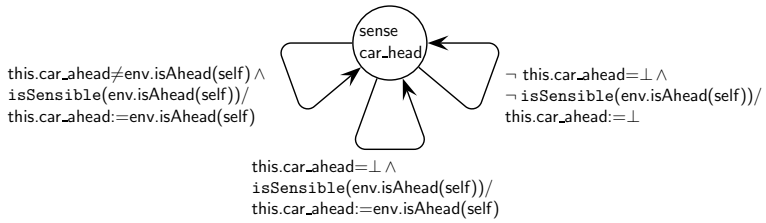


Figure 28: Sensor for car_ahead.

5.2 Monitoring the new_ahead

Figure 29 is the analogue of Fig. 28 but monitoring the helper's car ahead (that is the new car_ahead of an initiator after successful completion of a lane change manoeuvre). If this car is close enough to be noticed by the sensors the variable new_ahead is set, so that the agent is made to respect this car also. If the new_ahead is detected as not sensible, the new_ahead variable is set to be void, such that the agent does not need to adjust its velocity.

5.3 Gap Sensor

This sensor reports whether the gap between the agent and its new_ahead is sufficiently large. When triggered by the protocol the sensor checks instantaneously whether the gap is sufficient, i.e. whether safety distances are respected. In case this initial check fails, the sensors observes the gap until either the gap can be reported to be sufficient or the sensor receives a `dismissGap`.

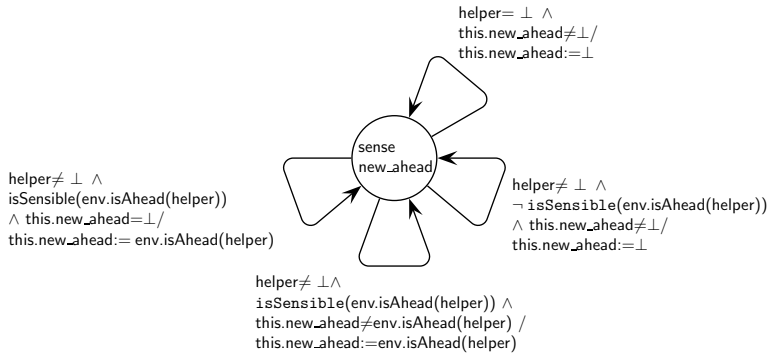


Figure 29: Sensor for new_ahead.

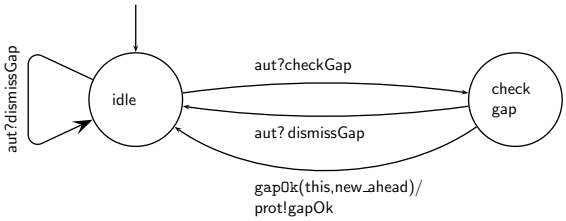


Figure 30: Gap Sensor.

5.4 Signal Sensor

The protocol specifies that before the agent actually changes lane, a certain time span has to pass without any other (relevant) car signalling its intend to change lane. The sensor given in Fig. 31 monitors car signals and reports whether a relevant car signals. Only when triggered by the protocol, signals from other cars are considered. If a relevant signal is perceived, an alert notice is sent to the protocol. A signal is considered as relevant if the signalling car could get in the way of the agent.

Determining when signals are relevant We safely over-approximate potential collision situations by the following considerations: In time t a car can cover a maximal distance of $\mathsf{v} \cdot t + \frac{\mathsf{acc}_{\mathsf{max}}}{2} \cdot t^2$ (and did at least cover $\mathsf{v} \cdot t - \frac{\mathsf{decel}_{\mathsf{max}}}{2} \cdot t^2$). A residence area describes an interval of potential positions—or rather x-coordinates—the car may be at in the given time interval. So if two cars A and B are the same position at the same time then their residence areas intersect. With other words, two cars can only be colliding if their areas intersect.

$$\mathsf{area}(\mathsf{car}, t) = [\mathsf{car}.x, \mathsf{car}.x + t \cdot \mathsf{v} + \frac{\mathsf{acc}_{\mathsf{max}}}{2} \cdot t^2]$$

We want not only that two cars do not collide but also that they keep their safety distances at all times. We take care of this by extending the residence area by the safety distance. To this end we define the function $\mathsf{area}_{\mathsf{SD}}(\mathsf{car}_{\mathsf{A}}, \mathsf{car}_{\mathsf{B}}, t)$ which gives us the residence area of car A extended by

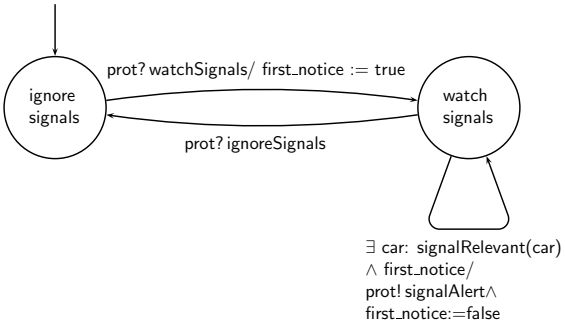


Figure 31: Signal Sensor.

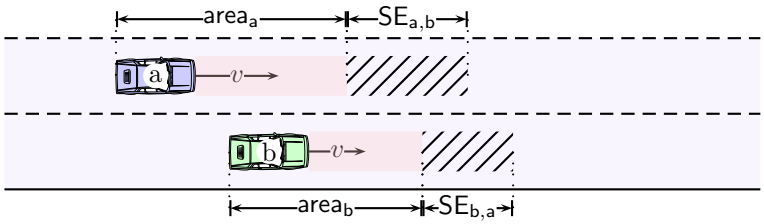


Figure 32: A collision situation: Areas extended by safety envelopes intersect.

the safety distance to $\mathsf{car_B}$:

$$\mathsf{area_{SD}}(\mathsf{car_A}, \mathsf{car_B}, t) = [\mathsf{car_A}.x, \mathsf{car_A}.x + t \cdot \mathsf{car_A.v} + \frac{\mathsf{acc_{max}}}{2} \cdot t^2 + \mathsf{SD_{max}}(\mathsf{car_A}, \mathsf{car_B}, t)]$$

The values of $\mathsf{SD_{max}}(\mathsf{car_A}, \mathsf{car_B}, t)$ is again an upper approximation of the necessary safety distance. We use

$\mathsf{SD_{max}}(\mathsf{car_A}, \mathsf{car_B}, t) :=$

$\max\{\mathsf{SD}(\mathsf{car_A.v} + \mathsf{acc_{max}} \cdot t, \mathsf{car_B.v} + \mathsf{acc_{max}} \cdot t), \mathsf{SD}(\mathsf{car_B.v} + \mathsf{acc_{max}} \cdot t, \mathsf{car_A.v} + \mathsf{acc_{max}} \cdot t)\}$,

i.e. we neglect the order of $\mathsf{car_A}$ and $\mathsf{car_B}$ and consider their maximal reachable speed.

Now we can consider a signal as irrelevant for an agent A, if during the manoeuvre time $\mathsf{t_{maneuver}}$ the extended areas of the agent A and a signalling car are distinct, that is

$$\mathsf{area_{SD}}(\mathsf{signalling\_car}, A, \mathsf{t_{maneuver}}) \cap \mathsf{area_{SD}}(A, \mathsf{signalling\_car}, \mathsf{t_{maneuver}}) = \emptyset.$$

This is a very pessimistic approximation. A finer approximation of collision free situations can be done by three simple refinements of the above approach: (1) We consider the direction of lane change, (2) we take into account that there is a maximal and minimal velocity, (3) we introduce additional observation points, since the above estimation gets more pessimistic the longer the manoeuvre takes. The reason is that a residence area collects every position from time $t = 0$ to the manoeuvre end. We get a less pessimistic estimation by scheduling more observation points, as described in

the following.

$$\begin{aligned}\mathsf{area}_{\mathsf{SD}\,i}(\mathsf{car_A}, \mathsf{car_B}, \mathsf{t_{maneuver}}) = [&\mathsf{car_A}.x + ((i-1)\cdot \mathsf{t_{sample}})\cdot \mathsf{v_{slow}}(\mathsf{car_A}, i-1),\\ &\mathsf{car_A}.x + (i\cdot \mathsf{t_{sample}})\cdot \mathsf{v_{fast}}(\mathsf{car_A}, i-1) +\\ &\mathsf{SD_{max}}(\mathsf{car_A}, \mathsf{car_B}, \mathsf{t_{sample}})]\end{aligned}$$

with

$$\mathsf{v_{slow}}(\mathsf{car_A}, j) = \begin{cases} 0, & \text{if } (j\cdot \mathsf{t_{sample}})\cdot \mathsf{car_A}.\mathsf{v} - \frac{\mathsf{decel_{max}}}{2}\cdot (j\cdot t)^2 \leq 0\\ (j\cdot \mathsf{t_{sample}})\cdot \mathsf{car_A}.\mathsf{v} - \frac{\mathsf{decel_{max}}}{2}\cdot (j\cdot t)^2, & \text{else}\end{cases}$$

$$\mathsf{v_{fast}}(\mathsf{car_A}, j) = \begin{cases} \mathsf{v_{max}}, & \text{if } (j\cdot \mathsf{t_{sample}})\cdot \mathsf{car_A}.\mathsf{v} + \frac{\mathsf{acc_{max}}}{2}\cdot (j\cdot t)^2 \geq \mathsf{v_{max}}\\ (j\cdot \mathsf{t_{sample}})\cdot \mathsf{car_A}.\mathsf{v} + \frac{\mathsf{acc_{max}}}{2}\cdot (j\cdot t)^2, & \text{else}\end{cases}$$

for all $i \geq 1$ with $i\cdot \mathsf{t_{sample}} \leq \mathsf{t_{maneuver}}$. For the $i \in \mathbb{N}, i \geq 1$ with $(i-1)\cdot \mathsf{t_{sample}} \leq \mathsf{t_{maneuver}} \leq i\cdot \mathsf{t_{sample}}$, we use

$$\begin{aligned}\mathsf{area}_{\mathsf{SD}\,i}(\mathsf{car_A}, \mathsf{car_B}, \mathsf{t_{maneuver}}) = [&\mathsf{car_A}.x + ((i-1)\cdot \mathsf{t_{sample}})\cdot \mathsf{v_{slow}}(i-1),\\ &\mathsf{car_A}.x + \mathsf{t_{maneuver}}\cdot \mathsf{v_{fast}}(i-1) + \mathsf{SD_{max}}(\mathsf{car_A}, \mathsf{car_B})].\end{aligned}$$

The $\mathsf{area}_{\mathsf{SD}\,i}$ gives the (extended) residence area, the car has been in at times t, $(i-1)\cdot \mathsf{t_{sample}} \leq t \leq i\cdot \mathsf{t_{sample}}$, $i \geq 1$, where the lower bound approximates the position of the car assuming maximal deceleration up to at most $\mathsf{v} = 0$ and the upper bound assumes maximal acceleration up to at most $\mathsf{v} = \mathsf{v_{max}}$.

Based on this we define the predicate $\text{signalRelevant}_{\mathsf{car_B}}(\mathsf{car_A}, \mathsf{t_{maneuver}})$ that evaluates to true, when $\mathsf{car_A}$ is considered as relevant for $\mathsf{car_B}$ during the next $\mathsf{t_{maneuver}}$ time units. Let $\mathsf{t_{maneuver}}$ be the time a manoeuvre is allowed to take at most. Let $\mathsf{t_{sample}}$ be the sampling rate and l the smallest number with $\mathsf{t_{maneuver}} \leq l\cdot \mathsf{t_{sample}}$.

$$\begin{aligned}&\text{signalRelevant}_{\mathsf{car_B}}(\mathsf{car_A}, \mathsf{t_{maneuver}}) :=\\ &\quad \exists i : 1 \leq i \leq l : \mathsf{area}_{\mathsf{SD}\,i}(\mathsf{car_B}, \mathsf{car_A}, \mathsf{t_{maneuver}}) \cap \mathsf{area}_{\mathsf{SD}\,i}(\mathsf{car_A}, \mathsf{car_B}, \mathsf{t_{maneuver}}) \neq \emptyset\\ &\quad \wedge (\quad (\mathsf{car_A}.\mathsf{lane} - 2 = \mathsf{car_B}.\mathsf{lane} \wedge \mathsf{signalsRight}(\mathsf{car_A}) \wedge \mathsf{car_B}.\mathsf{signalsLeft})\\ &\qquad \vee (\mathsf{car_A}.\mathsf{lane} + 2 = \mathsf{car_B}.\mathsf{lane} \wedge \mathsf{signalsLeft}(\mathsf{car_A}) \wedge \mathsf{car_B}.\mathsf{signalsRight})\ )\end{aligned}$$

where $\mathsf{car_B}.\mathsf{signalsLeft}$ and $\mathsf{car_B}.\mathsf{signalsRight}$ are attributes of the current state of $\mathsf{car_B}$, whereas $\mathsf{car_B}.\mathsf{signalsLeft}(\mathsf{car_A})$ is a predicate that evaluates to true if the sensors of $\mathsf{car_B}$ read that $\mathsf{car_A}$ is signalling to its left. $\mathsf{car_B}.\mathsf{signalsRight}(\mathsf{car_A})$ is analogously defined.

That means, we consider the signal of a $\mathsf{car_A}$ as relevant for $\mathsf{car_B}$ if the residence areas of $\mathsf{car_A}$ and $\mathsf{car_B}$ for the same time interval have an intersection—it is a necessary condition for a collision of $\mathsf{car_A}$ and $\mathsf{car_B}$ that they can be at the same place at the same time (interval)—and if also $\mathsf{car_A}$ is two lanes left of $\mathsf{car_B}$ signalling to its right while $\mathsf{car_B}$ intends to go to the left, or vice versa $\mathsf{car_A}$ is two lanes right of $\mathsf{car_B}$ and signals to the left while $\mathsf{car_B}$ intends to change to its right. We do not observe cars on (left or right) neighbouring lanes, because the protocol uses other means to make sure that they do not get in the way: An agent has either a helper or checks via sensors whether the target lane is occupied.

Figure 33 shows a situation in which the two cars do not collide and the refined approach could be used be detect this given an appropriate sampling, while the original approach will classify the situation as collision situation: Both cars drive with approximately the same speed and initially the distance between $\mathsf{car_A}$ and $\mathsf{car_B}$ is greater than the safety distance. When $\mathsf{car_B}$ reaches a potential collision position ($\mathsf{position}(\mathsf{car_B}) \in \mathsf{area_{SD}}(\mathsf{car_A}) \cap \mathsf{area_{SD}}(\mathsf{car_B})$), $\mathsf{car_A}$ will have reached a position at the end of $\mathsf{area}(\mathsf{car_A})$, so that again $\mathsf{car_A}$ and $\mathsf{car_B}$ are in a distance greater than the safety distance.

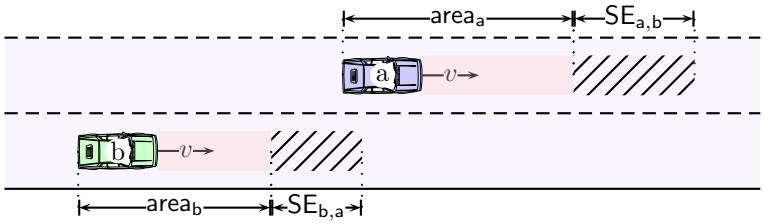


Figure 33: No collision situation.

5.5 Monitoring Neighbour Lanes

If the autonomous layer tells the sensors to check whether the lane to the agent's left/right is occupied, the sensors are started to perform this check. Either the sensor immediately returns that the respective lane is free or the sensor continuously observes the neighbour lane until told to stop by the autonomous layer. The predicate `rightLaneFree` is true iff any car that is right of the agent

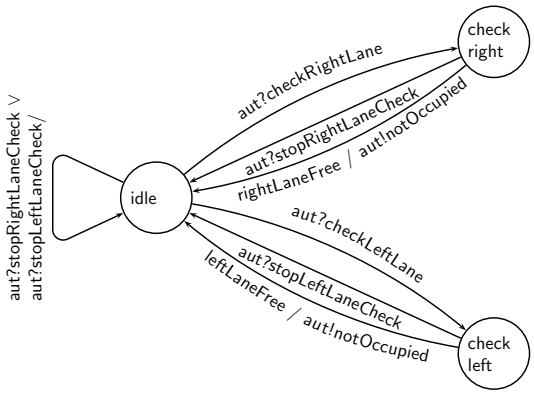


Figure 34: Next Lane Sensor.

is so far away that its possible area of residence is disjoint from the agent's.

> `rightLaneFree`:=$\forall$car$\in$Cars: $\mathrm{area_{SD}}(\mathit{self}, t) \cap \mathrm{area}(\mathrm{car}, t) \neq \emptyset$ where t is an estimate time the agent needs to change lane.

6 Environment

The continuous system evolution takes place within what we call *environment*. There, the orientation of cars, their positions, velocities and accelerations evolve over time. Their values represent the real world situation at a given time, and cars learn about its environment via sensors, constructing a mental map of the relevant aspects. The value $\mathsf{x}_{\mathsf{centre}}$ of a car represents its longitudinal position whereas the $\mathsf{y}_{\mathsf{centre}}$ value represents the lateral position. Every car has a certain orientation, β_{ori}, and a certain speed, v. The car's orientation and the speed determine the evolution of its position.

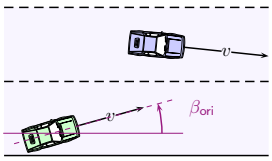


Figure 35: Evolution of Car Coordinates. Orientation and speed determine future $\mathsf{x}_{\mathsf{centre}}$ and $\mathsf{y}_{\mathsf{centre}}$ coordinates of a car.

The speed of a car changes according to its acceleration acc. When a car is driving on a straight line its orientation stays constant but when following a movement circle, its orientation changes over time with angular velocity ω.

The environment also knows the current mode of movement, that is whether the car drives on a straight line or follows a movement circle. The movement circle is changed discretely by the Steering Controller of the autonomous layer. Also the acceleration is changed discretely on the controllers' demands (cf. Sect. 2.1.1 on p. 7).

Discrete and Continuous Evolution w.r.t. a Single Vehicle The environment's evolution for a single arbitrary car is given in Fig. 36. As described above, a car's position, velocity and orientation change continuously. Discrete updates are implemented by single-state state machines composed in parallel.

`Environment`

$$\begin{aligned} \frac{\mathsf{d}}{\mathsf{dt}}\mathsf{x} &= \cos\beta_{\mathsf{ori}} \cdot \mathsf{v} \\ \frac{\mathsf{d}}{\mathsf{dt}}\mathsf{y} &= \sin\beta_{\mathsf{ori}} \cdot \mathsf{v} \\ \frac{\mathsf{d}}{\mathsf{dt}}\mathsf{v} &= \mathsf{acc} \\ \frac{\mathsf{d}}{\mathsf{dt}}\beta_{\mathsf{ori}} &= \frac{v}{r} \cdot \mathsf{side} \cdot \mathsf{mode} \end{aligned}$$

Figure 36: The Environment.

Discrete updates monitor the following values and ensure that

- $\mathsf{car\_ahead}$ is always the directly preceding car.
  The following transition changes $\mathsf{car\_ahead}$ appropriately.

$\exists$ car $\in$ Cars: AheadOf(car, self) $\wedge$ dist(self,car) $<$ dist(self, car_ahead) $\rightarrow$ car_ahead' $\in$ Cars $\wedge$ AheadOf(car_ahead', self) $\wedge$ $\forall$ car $\in$ Cars: AheadOf(car, self) $\Rightarrow$ dist(self, car_ahead')$\leq$ dist(self,car)

The predicate `AheadOf(car1,car2)` is true if `car1` is ahead of `car2` on the same lane. So the above transition, updates the car_ahead reference when the agent directly ahead changes.

- lane represents the lane number the car is currently on.

  $\neg$(this.lane $\cdot$ lanewidth $\leq$ this.y $<$ (this.lane+1) $\cdot$ lanewidth) $\rightarrow$ this.lane' $\cdot$ lanewidth $\leq$ this.y $<$ (this.lane'+1) $\cdot$ lanewidth

  This transition is triggered, when a vehicle's y-position comes into the range of another lane. The lane number is then accordingly updated.

- mc is the current movement circle and mode is true (=1) iff the car follows a circle with angular speed, ω and false (=0), otherwise.

  aut? isMC(r,side) $\rightarrow$ mc'=(r,side) $\wedge$ mode'=(r $< \infty$)

- acc is the current acceleration.

  aut? isAcc(acc_{ctrl}) / this.acc' = acc_{ctrl}

  The above means that the controller's acceleration directly and unperturbedly affects the velocity of the vehicle.

The primed variables refer to next state values.

Initial States On system initialization, the number of lanes is specified and kept fixed during the system's evolution. Fixed model values also include the lane width (lanewidth), the width and length of cars. For each car a position on the road, a velocity, acceleration and orientation is chosen. We assume that when the system's evolution starts, safety distances between cars are respected, i.e. any car has a certain velocity and a sufficient safety distance to its car ahead. Further the orientation of any car is such that the car still can choose a comfortable movement circle to bring it parallel to the lane at its current velocity and accelerating with its current acceleration for t_{acc} (cf. accIsFeasible(v,acc,β,offset) on p. 7).

7 Model Extensions

In the preceding we presented a coherent model on a carefully chosen abstraction level that features many aspects of a fully automatised system, exposing challenging verification tasks for hybrid system verification. Some design decisions of our model are discussed in the following, outlining the resulting consequences for possible extensions.

7.1 Track Generation / Lane Segmentation

In this report, we consider cars travelling on a straight road with a fixed number of lanes that are a priori unbounded in length. We investigated segmentation to overcome these limitations. Segments spatially partition the highway. This conveniently allows to define a varying number of lanes, a curved lane course and lane sections of varying road conditions. However, a curved lane course would require more adaptations. For one, in the current model, predicates like $\texttt{aheadOf(}\mathsf{car_A}\texttt{,}\ \mathsf{car_B}\texttt{)}$ can easily be evaluated by examining the car's coordinates: The two cars have to be on the same lane, i.e. $\mathsf{car_A}.\mathrm{y} \in]n-1 \cdot \mathsf{lanewidth}, n \cdot \mathsf{lanewidth}] \Leftrightarrow \mathsf{car_B}.y \in]n-1 \cdot \mathsf{lanewidth}, n \cdot \mathsf{lanewidth}]$ for all lane numbers n. When lanes are curved, it is necessary to know the course of lane, to evaluate whether two cars are on the same lane. On top of the adaptation of such basic notions, the Steering Controller should implement a strategy with a certain look-ahead to get onto its reference line.

7.2 Sensors

Our model specifies sensors continuously observing the changes in the environment. Certainly other sensor variants can easily be implemented, for example, time triggered sensors that periodically read data.

7.3 Disturbances

When the controllers of the autonomous layer set acceleration or movement line of a car, we assume that the car behaves accordingly. We do not model disturbances that could represent for instance different road conditions such as a slippery road or slopes. The standard way to model such disturbances is to extend the differential equations of the environment by an input representing these disturbances. One possible way to ensure collision freedom for different road conditions is then to adjust the safety distance.

7.4 Car Characteristics

In our model all vehicles follow the same blueprint and hence exhibit the same capabilities and physical shape. In particular they have the same maximal acceleration and the same maximal deceleration. The feasibility predictions (cf. Sect. 3.4) and the formula for safety distances (cf. Sect. A.1) are based on this assumption. So if we would allow vehicles to have individual maximal acceleration and deceleration, the computation of the safety distances needs to be adjusted. An easily implemented solution is to then fix a global maximal deceleration and determine the safety distance w.r.t. to these values. Downside of this solution is that safety distances now over-approximate the actually necessary safety distance. Another way would be to be determine the safety distance w.r.t. the actual deceleration of the respective cars. But then it has to be decided upon how a car gets to know about the maximal deceleration of its car ahead. Although individual maximal acceleration

and deceleration would also make the current feasibility predictions less accurate, safety of the lane change manoeuvre should not be influenced by this change, as feasibility predictions are not safety critical. A lane change is always monitored to respect the necessary safety distances.

7.5 Steering

We assumed that a vehicle is able to follow paths made up of line segments connected with tangential circular arcs. Based on time predictions and feasibility requirements, the steering controller decides on the most suitable next movement that is either straight or some kind of arc.

Paths of this type are considered by most planning techniques for the Reeds and Shepp car, which is by far the most widely used car model in optimal path planning [5]. However the curvature of the paths is discontinuous at the transition between segment and arcs, which means that at each discontinuity the car has to instantaneously change its wheel angle.

An interesting alternative steering controller may be based on the Ackermann steering, which can be expressed by three differential equations. But with such a steering model, path planning becomes more complicated.

7.6 Communication

In our model we refrained from giving an explicit implementation of the communication layer, but assumed synchronous—that is instantaneous and reliable—message passing between the agents' protocol layers directly. The protocol in Sect. 3 is based on these assumptions. We remark that such synchronous message passing can be used to explicitly model asynchronous communication, for example, a communication layer might explicitly model a bounded FIFO queue by means of synchronous message passing. A more sophisticated communication layer model could take further aspects into account such as communication delays, corrupted messages and message losses. The communication layer in isolation thus might be a simple communicating finite state machine, possibly enriched by clocks or by probabilistic choices. Note, that in any case the autonomous layer ensures that manoeuvres are safe at all times, thus also in case of unreliable communication.

8 Summary and Future Work

Intelligent transportation systems promise a myriad of merits [6], but at the same time demand extraordinary measures to ensure that safety requirements are met [7]. The application of formal methods becomes indispensable.

Since intelligent transportation systems often expose aspects of both, hybrid systems and dynamic communication systems, verification of these systems is especially challenging.

We presented a coherent hybrid system model of dynamically communicating vehicles that coordinate cooperative lane change manoeuvres in a decentralized fashion. Its comprehensive formal treatment is rare in literature but poses a prerequisite to push forward results on the application of formal methods. The presented model may serve as a blueprint to derive various sub-models posing interesting verification challenges.

To summarize, we described an extensible model of an advanced driver assistance system, with a layered system architecture—as is state of the art—focussing on the core model of a lane change assistance system.

First verification results have been achieved. Further, an implementation of a simulation model is under construction as preparatory step to a formal model for hybrid system verification. We are confident to cope with new challenging verification tasks derived from our model following ideas of [2, 8, 10, 1] that provide a whole verification methodology for cooperating traffic agents and abstractions tackling the unboundedness of the spatial environment [8] and the number of agents [10, 1], respectively. An important contribution in this direction is presented in [3].

A Appendix

A.1 Safety Distances

A safety distance describes the minimum distance between an agent $\mathsf{car_A}$ and an agent $\mathsf{car_B}$ for which $\mathsf{car_A}$ can guarantee to stop $dist_{\min}$ behind $\mathsf{car_B}$. We distinguish between *hard safety distances* and *comfortable safety distances*. A safety distance is hard when we allow the agent $\mathsf{car_A}$ to brake hard and a safety distance is called comfortable if $\mathsf{car_A}$ may only decelerate comfortably. The comfortable safety distance is expected to be maintained in standard situations. Hence when speaking of safety distances (SD) we refer to the comfortable safety distances. The Approach Velocity Controller (cf. Sect. 2.1.1) may bring a car closer to its agent directly ahead than comfortable safety distances would allow but it always respects hard safety distances.

In the sequel we derive formulas for the safety distances.

Braking Distance The following formula evaluates the distance an agent needs to brake from velocity v to a complete halt, when it accelerates with a for another t time units before the braking force b is entirely available.

$$\mathsf{dist_{stop}}(v, b, a, t) = \frac{v^2}{2b} + (\frac{a}{b} + 1) \cdot (v \cdot t + \frac{a}{2} \cdot t^2) \tag{13}$$

Safety Distance ($\mathsf{SD(car_A, car_B}, b)$) Given an agent $\mathsf{car_A}$ following with velocity v_1 an agent $\mathsf{car_B}$ which is travelling at velocity v_2. The safety distance $\mathsf{SD}(\mathsf{car_A}, \mathsf{car_B}, b)$ is the minimum distance between $\mathsf{car_A}$ and $\mathsf{car_B}$ for which $\mathsf{car_A}$ can guarantee to stop $dist_{\min}$ length units behind $\mathsf{car_B}$, if $\mathsf{car_B}$ suddenly brakes with b_{max} while $\mathsf{car_A}$ might accelerate with acc_{max} for another t_{brake} time units before it brakes with b.

$$\mathsf{SD}(\mathsf{car_A}, \mathsf{car_B}, b) = \mathsf{dist_{stop}}(v_1, b, acc_{\mathsf{max}}, t_{brake}) - \frac{v_2^2}{2b_{\mathsf{max}}} + dist_{\min} \tag{14}$$

Safety Distance ($\mathsf{SD(car_A, car_B})$) The safety distance $\mathsf{SD}(\mathsf{car_A}, \mathsf{car_B})$ is the minimum distance between agents $\mathsf{car_A}$ and $\mathsf{car_B}$ for which $\mathsf{car_A}$ can guarantee to comfortably stop $dist_{\min}$ length units behind $\mathsf{car_B}$, even if $\mathsf{car_B}$ brakes with maximal braking force.

$$\mathsf{SD}(\mathsf{car_A}, \mathsf{car_B}) = \mathsf{SD}(\mathsf{car_A}, \mathsf{car_B}, b_{comfort}) \tag{15}$$

A.2 Target Acceleration

In Sect. 2 we give a formula for the acceleration that constantly kept makes a car reach the set-point velocity and distance. In the following we give a step by step derivation of the formula given on page 15. The formula is based on the scenario depicted in figure 37.

Consider the scenario as illustrated in Fig. 37. Let us denote the initial positions of the agent and its reference object as p_a and p_b, respectively. We denote their respective positions in the goal scenario as p'_a and p'_b. For this scenario we use as $\mathsf{v_{goal}}$ the velocity of the reference object and denote

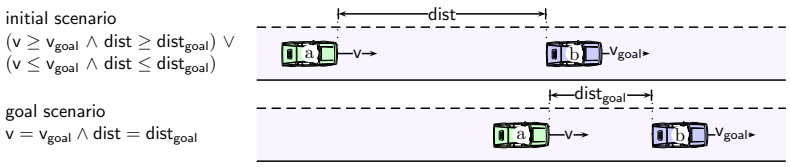


Figure 37: Determining $\mathsf{acc}_{\mathsf{target}}$ and $\mathsf{decel}_{\mathsf{target}}$. The controlled car is a and car b is the reference object.

the agent's velocity simply with v. We denote the (goal) displacement as dist ($\mathsf{dist}_{\mathsf{goal}}$). We assume that the car ahead drives with constant speed. This gives rise to the following system of equations:

$$\begin{aligned} p'_a &= p_a + \mathsf{v} \cdot t + \frac{1}{2}\,\mathsf{acc} \cdot t^2 \\ p'_b &= \mathsf{v}_{\mathsf{goal}} \cdot t + p_b \\ \mathsf{dist}_{\mathsf{goal}} &= p'_b - p'_a \\ \mathsf{v}_{\mathsf{goal}} &= \mathsf{v} + \mathsf{acc} \cdot t \end{aligned}$$

We use $t = \frac{\mathsf{v}_{\mathsf{goal}} - \mathsf{v}}{\mathsf{acc}}$ to derive:

$$\begin{aligned} p'_a &= p_a + \mathsf{v} \cdot \frac{\mathsf{v}_{\mathsf{goal}} - \mathsf{v}}{\mathsf{acc}} + \frac{1}{2}\,\mathsf{acc} \cdot (\frac{\mathsf{v}_{\mathsf{goal}} - \mathsf{v}}{\mathsf{acc}})^2 \\ p'_b &= \mathsf{v}_{\mathsf{goal}} \cdot \frac{\mathsf{v}_{\mathsf{goal}} - \mathsf{v}}{\mathsf{acc}} + p_b \\ \mathsf{dist}_{\mathsf{goal}} &= p'_b - p'_a. \end{aligned}$$

Which is equivalent to

$$\begin{aligned} p'_a &= p_a + \frac{\mathsf{v} \cdot \mathsf{v}_{\mathsf{goal}} - \mathsf{v}^2}{\mathsf{acc}} + \frac{1}{2}\,(\frac{\mathsf{v}^2_{\mathsf{goal}} - 2\,\mathsf{v}_{\mathsf{goal}} \cdot \mathsf{v} + \mathsf{v}^2}{\mathsf{acc}}) \\ p'_b &= \frac{\mathsf{v}^2_{\mathsf{goal}} - \mathsf{v} \cdot \mathsf{v}_{\mathsf{goal}}}{\mathsf{acc}} + p_b \\ \mathsf{dist}_{\mathsf{goal}} &= p'_b - p'_a. \end{aligned}$$

Using $\mathsf{dist}_{\mathsf{goal}} = p'_b - p'_a$ we get

$$\mathsf{dist}_{\mathsf{goal}} = \frac{\mathsf{v}^2_{\mathsf{goal}} - \mathsf{v} \cdot \mathsf{v}_{\mathsf{goal}}}{\mathsf{acc}} + p_b - (p_a + \frac{\mathsf{v} \cdot \mathsf{v}_{\mathsf{goal}} - \mathsf{v}^2}{\mathsf{acc}} + \frac{1}{2}\,(\frac{\mathsf{v}^2_{\mathsf{goal}} - 2\,\mathsf{v}_{\mathsf{goal}} \cdot \mathsf{v} + \mathsf{v}^2}{\mathsf{acc}}))\,.$$

Which is transformed to

$$\mathsf{dist}_{\mathsf{goal}} = \frac{\mathsf{v}^2_{\mathsf{goal}}}{\mathsf{acc}} - \frac{\mathsf{v} \cdot \mathsf{v}_{\mathsf{goal}}}{\mathsf{acc}} + p_b - p_a - \frac{\mathsf{v} \cdot \mathsf{v}_{\mathsf{goal}}}{\mathsf{acc}} + \frac{\mathsf{v}^2}{\mathsf{acc}} - \frac{1}{2}\,\frac{\mathsf{v}^2_{\mathsf{goal}}}{\mathsf{acc}} + \frac{\mathsf{v}_{\mathsf{goal}} \cdot \mathsf{v}}{\mathsf{acc}} - \frac{1}{2}\,\frac{\mathsf{v}^2}{\mathsf{acc}}\,.$$

By addition we get

$$\mathsf{dist}_{\mathsf{goal}} = \frac{1}{2}\,\frac{\mathsf{v}^2_{\mathsf{goal}}}{\mathsf{acc}} + \frac{1}{2}\,\frac{\mathsf{v}^2}{\mathsf{acc}} - \frac{\mathsf{v} \cdot \mathsf{v}_{\mathsf{goal}}}{\mathsf{acc}} + p_b - p_a = \frac{1}{2}\,\frac{(\mathsf{v}_{\mathsf{goal}} - \mathsf{v})^2}{\mathsf{acc}} + \mathsf{dist}\,.$$

Hence the agent reaches the distance $\mathsf{dist}_{\mathsf{goal}}$ with velocity $\mathsf{v}_{\mathsf{goal}}$ when using the acceleration as below and if the agent is slower than its reference object and too close to it, or if the agent is faster than the reference object and too far away from it.

$$\mathsf{acc} = \frac{1}{2}\,\frac{(\mathsf{v}_{\mathsf{goal}} - \mathsf{v})^2}{(\mathsf{dist}_{\mathsf{goal}} - (p_b - p_a))} = \frac{1}{2}\,\frac{(\mathsf{v}_{\mathsf{goal}} - \mathsf{v})^2}{(\mathsf{dist}_{\mathsf{goal}} - \mathsf{dist})}$$

This result might be puzzling at the first glance. There is no dependence on the time and the formula is independent of absolute velocities. But note that to accelerate from a given velocity v_1 to a target velocity v_2, we certainly have a variable time—$v_2 = v_1 + t \cdot \mathsf{acc}$. This means, the sooner we want the car to reach the velocity v_2 the stronger it has to accelerate. But since we also fix a certain distance that the car has to cover, we can eliminate the time.

Also, the absolute velocities of the cars do not influence the target acceleration, as we refer only to the relative distance.

A.3 Mode Invariants

In Table Table 1 on page 13 in Sect. 2 we gave invariants for the Approach Velocity Controller of Fig. 9. Here we show how to derive the mode invariants on the example of (CoD3) and (CoD4).

The Invariant of mode `ComfDecel` includes the disjunction of (CoD1)-(CoD4).

state	invariant	
`ComfDecel`	$\mathsf{v} > 0 \wedge ($	
	$(\mathsf{dist} \geq \mathsf{dist}_{\mathsf{goal}} \wedge \mathsf{v} > \mathsf{v}_{\mathsf{goal}} \wedge \lvert\mathsf{decel}_{\mathsf{target}}\rvert \geq \lvert\mathsf{decel}_{\mathsf{comf}}\rvert)$	(CoD1)
	$\vee\,(\mathsf{dist} < \mathsf{dist}_{\mathsf{goal}} \wedge \mathsf{v} \geq \mathsf{v}_{\mathsf{goal}})$	(CoD2)
	$\vee\,(\mathsf{dist} < \mathsf{dist}_{\mathsf{goal}} \wedge \mathsf{v} < \mathsf{v}_{\mathsf{goal}} \wedge \mathsf{acc}_{\mathsf{target}} < \mathsf{acc}_{\mathsf{comf}} \wedge \mathsf{t}_{\mathsf{reason}} \leq \mathsf{t}^{\mathsf{acc}}_{\mathsf{comf}})$	(CoD3)
	$\vee\,(\mathsf{dist} < \mathsf{dist}_{\mathsf{goal}} \wedge \mathsf{v} < \mathsf{v}_{\mathsf{goal}} \wedge \mathsf{acc}_{\mathsf{target}} < \mathsf{acc}_{\mathsf{comf}} \wedge \mathsf{t}_{\mathsf{reason}} < \mathsf{t}^{\mathsf{acc}}_{\mathsf{target}}))$	(CoD4)

Here we derive the clauses (CoD3) and (CoD4). The general scenario can be described as follows: The car is too slow but also too close ($\mathsf{dist} < \mathsf{dist}_{\mathsf{goal}} \wedge \mathsf{v} < \mathsf{v}_{\mathsf{goal}}$). It can use a constant (very) low acceleration that would take it (very) long to reach the goal velocity—at least longer than reasonable; the distance increases in the mean time as the velocity is less than the goal velocity.

We distinguish two cases: *(i)* using maximal comfortable acceleration would take longer than $\mathsf{t}_{\mathsf{reason}}$ and *(ii)* using the target acceleration takes longer than $\mathsf{t}_{\mathsf{reason}}$ but $\mathsf{acc}_{\mathsf{comf}}$ takes at most $\mathsf{t}_{\mathsf{reason}}$.

We get $\mathsf{dist} < \mathsf{dist}_{\mathsf{goal}} \wedge \mathsf{v} < \mathsf{v}_{\mathsf{goal}}$ to express that the distance is less than the goal distance and the velocity is less than the goal velocity. In this case a target acceleration exists.

$\mathsf{acc}_{\mathsf{target}} < \mathsf{acc}_{\mathsf{comf}}$ expresses that the target acceleration is still comfortable.

The following implication means that given we can reach the goal velocity and distance using maximal comfortable acceleration within reasonable time, then we only further decelerate if the target acceleration takes longer than $\mathsf{t}_{\mathsf{reason}}$.

$$\mathsf{t}^{\mathsf{acc}}_{\mathsf{comf}} < \mathsf{t}_{\mathsf{reason}} \Rightarrow \mathsf{t}^{\mathsf{acc}}_{\mathsf{target}} > \mathsf{t}_{\mathsf{reason}}$$

Put together we get

$$\mathsf{dist} < \mathsf{dist}_{\mathsf{goal}} \wedge \mathsf{v} < \mathsf{v}_{\mathsf{goal}} \wedge \mathsf{acc}_{\mathsf{target}} < \mathsf{acc}_{\mathsf{comf}} \wedge \mathsf{acc}_{\mathsf{target}} < \mathsf{acc}_{\mathsf{comf}} \wedge (\mathsf{t}^{\mathsf{acc}}_{\mathsf{comf}} < \mathsf{t}_{\mathsf{reason}} \Rightarrow \mathsf{t}^{\mathsf{acc}}_{\mathsf{target}} > \mathsf{t}_{\mathsf{reason}})$$

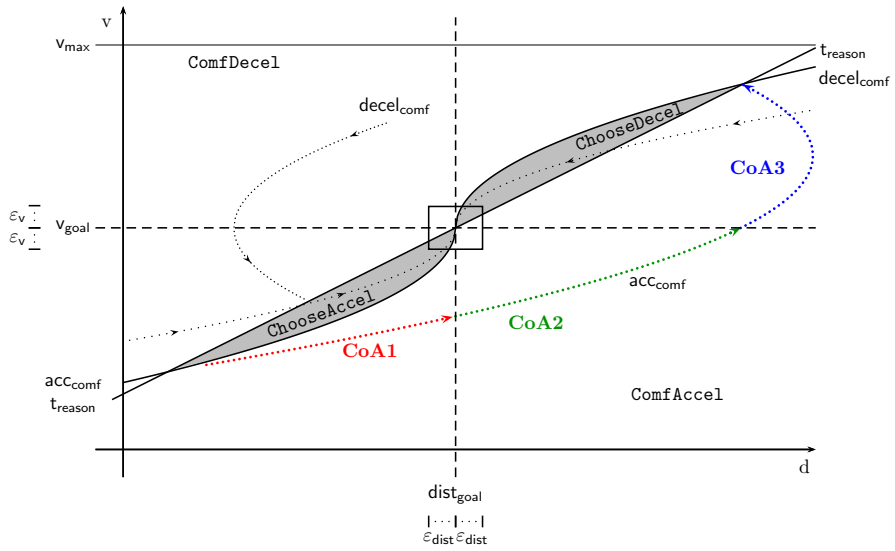


Figure 38: Example Trajectory.

which can be transformed into:
$(\mathsf{dist} < \mathsf{dist}_{goal} \wedge \mathsf{v} < \mathsf{v}_{goal} \wedge \mathsf{acc}_{target} < \mathsf{acc}_{comf} \wedge \mathsf{t}_{reason} \leq \mathsf{t}^{acc}_{comf})$
$\vee (\mathsf{dist} < \mathsf{dist}_{goal} \wedge \mathsf{v} < \mathsf{v}_{goal} \wedge \mathsf{acc}_{target} < \mathsf{acc}_{comf} \wedge \mathsf{t}_{reason} < \mathsf{t}^{acc}_{target})$

Analogously the conditions (CoA3) and (CoA4) can be derived. Figure 38 shows an exemplary trajectory with (CoA3).

List of Figures

List of Tables

References

[1] Jörg Bauer, Ina Schaefer, Tobe Toben, and Bernd Westphal. Specification and verification of dynamic communication systems. In *Sixth International Conference on Application of Concurrency to System Design, 2006. ACSD 2006.*, pages 189–200. IEEE Computer Society Press, 2006.

[2] Werner Damm, Alfred Mikschl, Jens Oehlerking, Ernst-Rüdiger Olderog, Jun Pang, André Platzer, Marc Segelken, and Boris Wirtz. Automating verification of cooperation, control, and design in traffic applications. In Cliff Jones, Zhiming Liu, and Jim Woodcock, editors, *Formal Methods and Hybrid Real-Time Systems*, volume 4700 of *Lecture Notes in Computer Science*, pages 115–169. Springer, 2007.

[3] Werner Damm, Hans-Jörg Peter, Jan Rakow, and Bernd Westphal. Can we build it: Formal synthesis of control strategies for cooperative driver assistance systems. *Mathematical Structures in Computer Science, Special Issue on Practical and Lightweight Formal Methods for the Design, Modeling and Analysis of Software Systems*, 2011. Accepted for publication.

[4] Ann Hsu, Farokh Eskafi, Sonia Sachs, and Pravin Varaiya. The design of platoon maneuver protocols for IVHS. PATH Research Report UCB-ITS-PRR-91-6, Inst. of Transportation Studies, University of California, April 1991. ISSN 1055-1425.

[5] Christian Laugier and Thierry Fraichard. Decisional architectures for motion autonomy. In L. Vlacic, M. Parent, and F. Harashima, editors, *Intelligent Vehicle Technologies*, pages 333–392. Elsevier, 2001.

[6] Michel Parent. The car of the future. In L. Vlacic, M. Parent, and F. Harashima, editors, *Intelligent Vehicle Technologies*, pages 3–18. Elsevier, 2001.

[7] Steven Shladover. Automated highway systems research: The influence of Pravin Varaiya. In E.H. Abed, editor, *Advances in Control, Communication Networks, and Transportation Systems*, Systems and Control: Foundations Applications, pages 267–282. Birkhauser, 2005.

[8] Tobe Toben, Bernd Westphal, and Jan-Hendrik Rakow. Spotlight abstraction of agents and areas. In Bengt Jonsson, Jörg Kreiker, and Marta Kwiatkowska, editors, *Quantitative and Qualitative Analysis of Network Protocols*, number 10051 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2010. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[9] Pravin Varaiya. Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control*, 38:195–207, 1993.

[10] Bernd Westphal. *Specification and Verification of Dynamic Topology Systems.* PhD thesis, Carl von Ossietzky Universität Oldenburg, May 2008.