Linking Discrete and Continuous Models, Applied to Traffic Manoeuvrers*

Ernst-Rüdiger Olderog, Anders P. Ravn and Rafael Wisniewski

Abstract The interplay between discrete and continuous dynamical models is discussed, and a systematic approach to developing and combining these models together is outlined. The combination is done with linking predicates that define refinement relations between the models. As a case study, we build an abstract, discrete spatial model and a concrete, continuous dynamic model for traffic manoeuvrers of multiple vehicles on highways. In the discrete model we show the safety (collision freedom) of distance keeping and lane-change manoeuvrers using events and actions to specify state transitions. By linking the discrete and continuous model via suitable predicates that express the discrete events and actions as distances and set-points in the continuous model, the safety carries over to the concrete model.

1 Introduction

Hybrid systems were introduced in order to model dynamical systems with a complex interaction between discrete actions and continuous evolutions in their trajectories [15]. Semantic models in the form of Hybrid Automata with the underlying transition systems [2, 29] were soon developed, and simulation tools like Stateflow [30] and Ptolemy II [24] appeared as well. Due to the success of model checking

Ernst-Rüdiger Olderog
Department of Computing Science, University of Oldenburg, Germany,
e-mail: olderog@informatik.uni-oldenburg.de

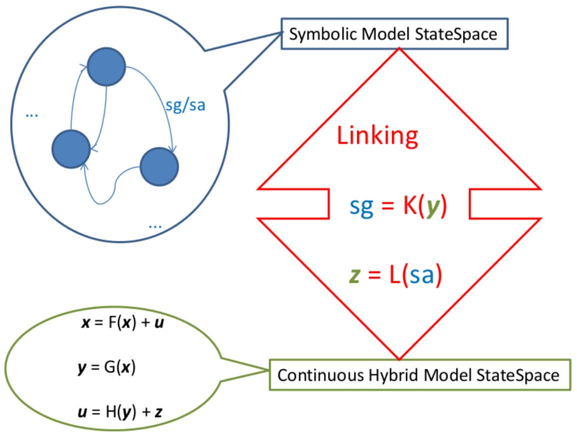
Anders P. Ravn
Department of Computer Science, Aalborg University, Denmark, e-mail: apr@cs.aau.dk

Rafael Wisniewski
Department of Automation, Aalborg University, Denmark, e-mail: raf@es.aau.dk

* This research was partially supported by the German Research Foundation (DFG) in the Transregional Collaborative Research Center (SFB/TR 14) AVACS (www.avacs.org). This chapter is a revised and extended version of the conference paper [35].

for timed automata [3], much effort has been directed towards analysis tools which use over- and under-approximations of hybrid automata [12, 14, 51], because it was clear from the outset that decidability was impossible even for very simple models.

There has been much progress both in analysis tools and in the amount of case studies, but it is still hard to find general composition principles. Often a system is decomposed into simpler subsystems that are loosely coupled [20, 42, 4] and thus can be analyzed individually. This loose coupling among concurrently operating subsystems was illustrated in [7], and it was analysed at a semantic level for heterogeneous subsystems in [38]. One observation though is that verification is usually done on subsystem models abstracted from detailed continuous models. It is this decomposition that is in the focus of this work. In a search for a more general and perhaps even teachable approach to performing this abstraction, we have tried to extract the principles from our continued efforts in modelling and verifying vehicle manoeuvrers in traffic, because it is a setting with a complex state space, a demand for decentralized control, and hybrid behaviours.



The discrete model is a collection of discrete automata with transitions governed by symbolic guards, *sg*, and with symbolic actions, *sa*. The underlying symbolic state space is hybrid with time evolutions. However, it is asserted that time steps do not change the value of guards and actions. The continuous model is a conventional control model which accepts set points, *z*. Linking is given via suitable functions *K* and *L*.

Fig. 1 Modelling approach

Here, we have reached the conclusion that a key point in the abstraction is to keep the discrete part, often a supervisory layer, on symbolic and finite level without any direct reference to time, because it allows for exhaustive verification using conventional techniques. However, this will in itself leave the continuous dynamics as an unexplored postulate. Thus, there is a need for linking the symbolic quantities of the discrete model to the concrete continuous model by a proper refinement relation. Via linking, behavioural properties of the abstract model are preserved for the concrete model. An inspiration is here the data refinement relations explored in program verification [8]. However, in the reactive setting, linking predicates as in the approach of UTP (Unifying Theories of Programming) [23] are more suitable. In summary, the approach presented has the following steps, illustrated in Fig. 1. For the symbolic, discrete model:

1. build a qualitative model of the context with symbolic representation of states of objects.,
2. formulate rules for interaction as finite state machines operating on the symbolic state (If the state machines use communication protocols, timeout transitions may be used to compensate for lost messages),
3. specify safety and liveness properties of the symbolic state,
4. verify the properties.

These steps are illustrated on the case of vehicle manoeuvrers in Sections 2 and 3.

When this part has gone through a number of iterations and the result is satisfactory, consider the concrete model:

5. identify a concrete dynamical model for the objects including available or at least plausible sensors and actuators,
6. link the models by relating the symbolic state variables to concrete observables that are computed by a controller for the dynamical system using available sensors and concrete models of the individual objects, also link symbolic actions to set points for the control,
7. design and validate the controllers and observers.

These steps are illustrated on the case in Sections 4 to 6.

Note that often the two models may develop concurrently. When this happens, it is important to keep the linkage stable when doing separate iterations.

A pragmatic consideration when designing the linking in the concrete case has been to design a system where a smart car can navigate among ordinary dumb cars. There is no need to require all cars to be smart and able to communicate with other cars. This has implications for the sensors and actuators, see Fig. 3 in Section 5, as well as impact on the symbolic guards and actions.

In Section 7, we comment on generally related work, while the conclusion in Section 8 considers limitations of the approach and potential for tool support.

2 Symbolic Model

In this section, we summarize and adapt the model of [22]. In this model, a multi-lane highway has an infinite extension with positions represented by real numbers in $\mathbb{R}$ and with lanes represented by a finite set of natural numbers, $\mathbb{L} = \{0, \ldots, N\}$. We assume that all traffic proceeds in one direction, with increasing position values, in pictures shown from left to right. The highway is populated by cars with unique identities denoted by capital letters $\mathbb{I} = \{A, B, \ldots\}$.

At each moment in time, we represent the traffic on the highway by a *traffic snapshot*. It records for each car the current position *pos* (at the rear end of the car) and speed *spd*, and on which lanes the car *reserves* or *claims* space. The idea is that a reserved space is owned by a unique car. Thus for safety, we have to show that reserved spaces of different cars are mutually exclusive. In contrast, a claimed space is used in preparation of a lane change and may still overlap with claimed or reserved spaces of other cars. However, then the lane change must not take place. The length of reserved and claimed spaces is given by the *safety distance*, which is the length of the car plus a safe estimate of the (speed-dependent) braking distance that the car will need to come to a complete standstill.

Definition 1. A *traffic snapshot* $\mathscr{T}$ comprises the functions pos, spd, res, clm

- $pos : \mathbb{I} \to \mathbb{R}$ such that $pos(C)$ is the position of car C along the lanes,
- $spd : \mathbb{I} \to \mathbb{R}$ such that $spd(C)$ is the current speed of the car C,
- $res : \mathbb{I} \to \mathscr{P}(\mathbb{L})$ such that $res(C)$ is the set of lanes C reserves,
- $clm : \mathbb{I} \to \mathscr{P}(\mathbb{L})$ such that $clm(C)$ is the set of lanes C claims.

We denote the set of all traffic snapshots by $\mathbb{T}$.

Note that in $\mathscr{T}$, it is not specified which space is occupied on the reserved and claimed lanes. This information is given by an uninterpreted function *se* for *safety envelope*. For a given traffic snapshot $\mathscr{T}$, we introduce for each car C its *safety envelope* $se_{\mathscr{T}}(C)$ as the interval $se_{\mathscr{T}}(C) = [pos(C), pos(C) + d(C)]$ starting at the current position $pos(C)$ of the car and of some uninterpreted length $d(C) > 0$, which is intended to be the safety distance of car C dependent on its current speed $spd(C)$. The exact value of $d(C)$ is not known in the symbolic model, but will be determined in the concrete dynamic model.

2.1 View

For our safety proof, we restrict ourselves to finite parts of a traffic snapshot $\mathscr{T}$ called *views*; the intuition being that safety depends on local information only.

Definition 2. A *view* $V = (L, X, E)$ consists of an interval of lanes visible in the view, $L = [l, n] \subseteq \mathbb{L}$, and the extension visible in the view, $X = [r, t] \subseteq \mathbb{R}$, and $E \in \mathbb{I}$, the identifier of the car under consideration.

A *subview* of V is obtained by restricting the lanes and extension we observe. For this we use sub- and superscript notation: $V^{L'} = (L', X, E)$ and $V_{X'} = (L, X', E)$, where L' and X' are subintervals of L and X, respectively.

For a car E and a traffic snapshot $\mathscr{T} = (pos, spd, res, clm)$ its *standard view* is

$$V_s(E, \mathscr{T}) = (\mathbb{L}, [pos(E) - ho, pos(E) + ho], E) \ ,$$

where the *horizon ho* is chosen such that a car driving at maximum speed can, with lowest deceleration, come to a standstill within the horizon.

2.2 Spatial Logic

To specify properties of traffic snapshots within a given view in an intuitive and yet precise way, we use a two-dimensional spatial interval logic, MLSL (Multi-Lane Spatial Logic) [22]. In this logic, the horizontal dimension is continuous, representing positions on a highway, and the vertical dimension is discrete, representing the number of a lane on a highway. In the syntax, variables ranging over car identifiers are denoted by small letters c, d, u and v. To refer to the car owning the current view, we use a special variable *ego*. By Var we denote the set of all these variables. Additionally, the letter γ ranges over car identifiers or elements in Var.

Definition 3 (Syntax). The syntax of the *multi-lane spatial logic MLSL* is given by the following formulae:

$$\phi ::= true \mid u = v \mid free \mid re(\gamma) \mid cl(\gamma) \mid$$

$$\phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid \exists v\colon \phi_1 \mid \phi_1 \frown \phi_2 \mid \begin{array}{c}\phi_2\\ \phi_1\end{array}$$

We denote the set of all MLSL formulae by Φ.

Formulae of MLSL express the spatial status of neighbouring lanes on a multi-lane highway. For a lane, the spatial status describes whether parts of it are reserved or claimed by a car or completely free. To this end, MLSL has atoms $re(\gamma)$, $cl(\gamma)$, and *free*, and two chop operators: the horizontal chop $\phi_1 \frown \phi_2$ expresses that an interval can be divided into two horizontally adjacent parts such that ϕ_1 holds in the left part and ϕ_2 in the right part, and the vertical chop $\begin{array}{c}\phi_2\\ \phi_1\end{array}$ expresses that an interval can be divided into two vertically adjacent parts where ϕ_1 holds on the lower part and ϕ_2 on the upper part. We use juxtaposition for the vertical chop to have a correspondence to the visual layout in traffic snapshots.

The logic is given a semantics that defines the when traffic snapshots satisfy a given formula.

Definition 4 (Semantics). The *satisfaction* $\models$ of formulae is defined inductively with respect to a *model* $\mathcal{M} = (\mathcal{T}, V, \nu)$ comprising a traffic snapshot $\mathcal{T}$, a view $V = (L, X, E)$ with $L = [l, n]$ and $X = [r, t]$, and a valuation $\nu : \mathbb{I} \cup \mathrm{Var} \to \mathbb{I}$ *consistent* with V, i.e., with $\nu(ego) = E$ and $\nu(C) = C$ for $C \in \mathbb{I}$:

$$\begin{array}{lcl}
\mathcal{M} \models true & & \text{for all } \mathcal{M} \\
\mathcal{M} \models u = v & \Leftrightarrow & \nu(u) = \nu(v) \\
\mathcal{M} \models free & \Leftrightarrow & |L| = 1 \text{ and } |X| > 0 \text{ and} \\
 & & \forall C \in \mathbb{I}\colon L \subseteq res(C) \cup clm(C) \Rightarrow se_{\mathcal{T}}(C) \cap (r, t) = \emptyset \\
\mathcal{M} \models re(\gamma) & \Leftrightarrow & |L| = 1 \text{ and } |X| > 0 \text{ and} \\
 & & L \subseteq res(\nu(\gamma)) \text{ and } X \subseteq se_{\mathcal{T}}(\nu(\gamma)) \\
\mathcal{M} \models cl(\gamma) & \Leftrightarrow & |L| = 1 \text{ and } |X| > 0 \text{ and } L \subseteq clm(\nu(\gamma)) \text{ and } X \subseteq se_{\mathcal{T}}(\nu(\gamma))
\end{array}$$

$$\begin{array}{lll}
\mathscr{M} \models \phi_1 \land \phi_2 & \Leftrightarrow & \mathscr{M} \models \phi_1 \text{ and } \mathscr{M} \models \phi_2 \\
\mathscr{M} \models \neg\phi & \Leftrightarrow & \text{not } \mathscr{M} \models \phi \\
\mathscr{M} \models \exists v\colon \phi & \Leftrightarrow & \exists \alpha \in \mathbb{I}\colon (\mathscr{T}, V, \nu \oplus \{v \mapsto \alpha\}) \models \phi \\
\mathscr{M} \models \phi_1 \frown \phi_2 & \Leftrightarrow & \exists s\colon r \le s \le t \text{ and} \\
 & & (\mathscr{T}, V_{[r,s]}, \nu) \models \phi_1 \text{ and } (\mathscr{T}, V_{[s,t]}, \nu) \models \phi_2 \\
\mathscr{M} \models \begin{array}{c}\phi_2\\ \phi_1\end{array} & \Leftrightarrow & \exists m\colon l-1 \le m \le n+1 \text{ and} \\
 & & (\mathscr{T}, V^{[l,m]}, \nu) \models \phi_1 \text{ and } (\mathscr{T}, V^{[m+1,n]}, \nu) \models \phi_2
\end{array}$$

We write $\mathscr{T} \models \phi$ if $(\mathscr{T}, V, \nu) \models \phi$ for all views V and consistent valuations ν.

For the semantics of the vertical chop, we set the interval $[l,m] = \emptyset$ if $l > m$. A view V with an empty set of lanes satisfies only *true* or an equivalent formula. Both chop modalities are associative. Other logical operators like $\lor, \to, \leftrightarrow$ and $\forall$ are treated as abbreviations. Also, we use the notation $\langle\phi\rangle$ for the two-dimensional modality *somewhere* ϕ, defined in terms of both chop operators:

$$\langle\phi\rangle \equiv \mathit{true} \frown \begin{pmatrix} \mathit{true} \\ \phi \\ \mathit{true} \end{pmatrix} \frown \mathit{true}.$$

For example, $\mathit{Safe} \equiv \forall c,d : c \neq d \to \neg\langle re(c) \land re(d)\rangle$ expresses the safety property that any two different cars have disjoint reserved spaces.

2.3 Transition System

A traffic snapshot is an instant picture of the highway traffic. The following *transitions* describe how it may change. Time may pass or a car may perform several actions when attempting and performing a lane change. We use the overriding notation $\oplus$ for function updates [46].

$$\begin{array}{lll}
\mathscr{T} \xrightarrow{t} \mathscr{T}' & \Leftrightarrow & \mathscr{T}' = (pos', spd', res, clm) \\
 & & \land \forall C \in \mathbb{I}\colon pos'(C) > pos(C) \qquad (1) \\
\mathscr{T} \xrightarrow{\mathsf{c}(C,n)} \mathscr{T}' & \Leftrightarrow & \mathscr{T}' = (pos, spd, res, clm') \\
 & & \land |clm(C)| = 0 \land |res(C)| = 1 \\
 & & \land \{n+1, n-1\} \cap res(C) \neq \emptyset \\
 & & \land clm' = clm \oplus \{C \mapsto \{n\}\} \qquad (2) \\
\mathscr{T} \xrightarrow{\mathsf{wd\ c}(C)} \mathscr{T}' & \Leftrightarrow & \mathscr{T}' = (pos, spd, res, clm') \\
 & & \land clm' = clm \oplus \{C \mapsto \emptyset\} \qquad (3)
\end{array}$$

$$\mathscr{T} \xrightarrow{\mathsf{r}(C)} \mathscr{T}' \quad \Leftrightarrow \quad \begin{aligned} &\mathscr{T}' = (,pos,spd,res',clm') \\ &\wedge clm' = clm \oplus \{C \mapsto \emptyset\} \\ &\wedge res' = res \oplus \\ &\quad \{C \mapsto res(C) \cup clm(C)\} \end{aligned} \tag{4}$$

$$\mathscr{T} \xrightarrow{\mathsf{wd\ r}(C,n)} \mathscr{T}' \quad \Leftrightarrow \quad \begin{aligned} &\mathscr{T}' = (pos,spd,res',clm) \\ &\wedge res' = res \oplus \{C \mapsto \{n\}\} \\ &\wedge n \in res(C) \wedge |res(C)| = 2. \end{aligned} \tag{5}$$

In (1), time passes, which results in the cars moving along the highway to the right. However, note that reservations, *res*, and claims, *clm*, cannot change during time passing transitions. The new position and speed of each car is determined by the dynamics of them, which is described at the concrete level. A car may *claim* a neighbouring lane n (2) if and only if it does not already claim a lane or is in the progress of changing the lane and therefore reserves two lanes . Furthermore, a car may *withdraw* a claim (3) or *reserve* a previously claimed lane (4) or withdraw the reservation of all but one of the lanes it is moving on (5).

3 Abstract Controllers

In this section we present abstract car controllers for keeping distance and changing lanes. By abstract, we mean that properties, invariants and guards of transitions are given by MLSL formulas. The controllers should guarantee that at any moment the spaces reserved by different cars are disjoint. This is expressed concisely by

$$Safe \equiv \forall c,d : c \neq d \Rightarrow \neg \langle re(c) \wedge re(d) \rangle ,$$

stating that in any lane any two different cars have disjoint reserved spaces. The quantification over lanes arises implicitly by the negation of the somewhere modality in *Safe*. A traffic snapshot $\mathscr{T}$ is *safe* if $\mathscr{T} \models Safe$ holds.

3.1 Keeping Distance

A distance controller DC of a car E should guarantee the safety as long as E is driving along the highway without making any new claim or reservation. This is expressed by time transitions among traffic snapshots: $\mathscr{T} \xrightarrow{t} \mathscr{T}'$. From the perspective of the car E, safety means that the following *collision check* remains false:

$$cc \equiv \exists c \colon c \neq ego \wedge \langle re(ego) \wedge re(c) \rangle .$$

Thus we require:

(DC) The distance controller DC of a car E keeps the property $\neg cc$ invariant under time transitions, i.e., for every transition $\mathscr{T} \xrightarrow{t} \mathscr{T}'$ whenever $\mathscr{T} \models \neg cc$, also $\mathscr{T}' \models \neg cc$.

3.2 Changing Lanes

We specify an abstract controller by a *timed automaton* [3] with clocks ranging over $\mathbb{R}_{\geq 0}$ and data variables ranging over $\mathbb{L}$ and $\mathbb{I}$. Strictly speaking, the single clock x, which is used in the automaton, is unnecessary for proving safety; it is added to ensure liveness. MLSL formulae appear in transition guards and state invariants. This can be seen in the lane-change controller in Fig. 2, where the MLSL formulae ϕ_1 and ϕ_2 are kept symbolic. The abstract lane-change controller LCP of [22] is an instantiation of this controller, except that it has the invariant $\neg cc$ in the initial state q_0. Here this property is ensured invariantly by the distance controller DC.

LCP assumes that every car, E, knows the full extension of claims and reservations of all cars within its view. It thus has *perfect knowledge* of its neighbouring cars (hence the letter P in the name of the controller); E perceives another car C as soon as C's safety envelope enters the view of E. In the following and in Section 5, we identify the car variables *ego* and c with their values, the cars E and C, respectively.

At the initial state q_0 of LCP, the car has reserved exactly one lane, which is saved in the variable n. An auxiliary variable l stores the lane the *ego* car wants to move to. Suppose *ego* intends to change to a neighboring lane, then it adheres to the following protocol. First, it claims a space on the target lane adjacent to and of the same extension as the reservation on its current lane, moving to state q_1. Subsequently, it checks for a *potential collision* (pc), i.e., whether its claim intersects with the reservation or claim of any other car. This is expressed by the MLSL formula

$$pc \equiv \exists c : c \neq ego \wedge \langle cl(ego) \wedge (re(c) \vee cl(c)) \rangle \, .$$

If pc occurs, *ego* withdraws its claim and returns to state q_0, giving up the wish to change lanes for the moment. Otherwise, *ego* turns its claim into reservations and thus reserves two lanes. This is in state q_3, During this double reservation *ego* changes lane within t_{lc} time units, an upper time bound for the lane change. Then *ego* withdraws its reservation on the original lane and continues to drive on the target lane, being again in state q_0. In this protocol, only turning the claim into a reservation (in the transition from state q_2 to state q_3) may violate the safety property. Thus in LCP of Fig. 2, we instantiate $\phi_1 \equiv pc$ and $\phi_2 \equiv \neg pc$.

In order to ensure liveness in the states q_0 and q_1, they are to be left within t_0 time units. Liveness in state q_0 could be ensured by adding an invariant asserting that the state should be left when a claim is made. The lane change timeout t_{lc} should strictly speaking be replaced by a symbolic guard that would be asserted by the concrete model when a lane change was completed. This symbolic guard would then be linked to either a sensor value or most likely to a timer in the concrete model.

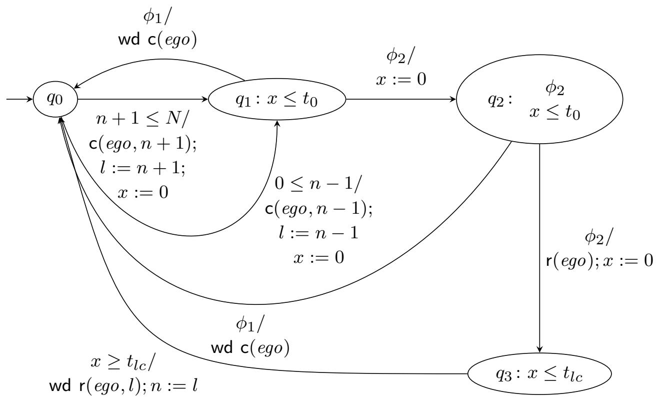


Fig. 2 The lane-change controller LCP with $\phi_1 \equiv pc$ and $\phi_2 \equiv \neg pc$.

3.3 Safety

We stipulate now that every car is equipped with the controllers DC and LCP (or that its driver manually follows its protocol). Under these assumptions, we can show:

Theorem 1 (Safety of DC and LCP). *Let $\mathscr{T}_0$ be an initial safe traffic snapshot. Then every traffic snapshot $\mathscr{T}$ that is reachable from $\mathscr{T}_0$ by transitions allowed by the controllers DC and LCP is safe.*

Proof. As in [22], we fix an arbitrary car E and shows that $\neg cc$ holds for every traffic snapshot $\mathscr{T}$ reachable from $\mathscr{T}_0$. The argument is by induction on the number of transitions needed to reach $\mathscr{T}$ from $\mathscr{T}_0$, and the crucial case in the induction step is that of the reservation transition. In contrast to [22], the initial state q_0 of LCP in Fig. 2 does not have $\neg cc$ as a built-in invariant. However, since the distance controller DC is running in parallel to LCP, the safety property $\neg cc$ is an invariant for this state. Moreover, it is also invariant under any transition that is not creating any new reservation. Regarding LCP, we thus have that $\neg cc$ holds in the start state q_2 of the reservation transition from state q_2 to state q_3 in LCP. As in [22], it can be shown that performing the reservation transition in state q_2 satisfying both $\neg cc$ and $\neg pc$ leads to q_3 satisfying $\neg cc$. $\square$

4 Concrete Model

The aim of this section is to present a physical model of a vehicle, which describes the position $pos(C)$ and the speed $spd(C)$ of a vehicle C. It will lay the basis for the controller design in Section 6.

4.1 Longitudinal Motion

A vehicle C is characterised by its velocity given in $[m/s]$ at the current time t given in $[s]$, $v_C : \mathbb{R}_+ \rightarrow \mathbb{R}_+$. Both the time and the velocity are considered non-negative reals. The acceleration and braking of the vehicle C is realised by a torque $T \equiv T_C : \mathbb{R}_+ \rightarrow \mathbb{R}$ given in $[Nm]$. The torque is applied to the wheels from the transmission and braking system, and it belongs at any given time to an interval $[\underline{T}, \overline{T}] \equiv [\underline{T_C}, \overline{T_C}]$, where $\underline{T_C} < 0$ is the maximal torque of the brakes, and $\overline{T_C} > 0$ is the torque at full throttle.

To model aerodynamic drag force, we introduce a drag coefficient C_{W}. The drag force is proportional to the square of the velocity

$$C_{\mathrm{W}}(t)v_C^2(t).$$

As indicated in the above equation, C_{W} varies in time. Specifically, C_{W} is characterised as follows. Suppose a vehicle D drives in front of the vehicle under consideration C. The drag coefficient is an empirical quantity approximated by

$$C_{\mathrm{W}}(\delta, v_D) = C_C \left(1 - \exp\left(-\frac{a\delta}{C_D v_D}\right)\right)^2,$$

where C_C, C_D are the aerodynamic coefficients of the vehicles C and D, and a is a constant [47]. In short, the aerodynamic coefficient of a vehicle is determined by its geometry: shape and size. The drag coefficient is positive, $\mathrm{Image}(C_{\mathrm{W}}) \subseteq [0, C_C]$. It converges to C_C for small distances δ and large velocities v_D.

As a consequence, the dynamics of the vehicle C is given by

$$(Mr^2 + J)\dot{v}_C(t) = -C_{\mathrm{W}}(\delta(t), v_D(t))r^2 v_C(t)^2 + rT(t),$$

where M is the mass of the vehicle C $[kg]$, J is the combined moments of inertia of the wheels $[kgm^2]$, and r is the radius of the wheels $[m]$.

Let X be the state space of the vehicle C (with the vehicle D driving in in front). It is the linear space of vectors comprising of the velocity v_C of the vehicle C, and the distance δ from C to D, i.e., $X = \mathbb{R}^2$. We assume that both the velocity and the distance are available as indicated in Fig. 3, where sensor $\hat{v}$ measures v_C and $\hat{d}_1$ measures δ. If the vehicle D is out of range the distance sensor delivers the value ∞.

A feedback controller is a function $T : X \rightarrow [\underline{T}, \overline{T}]$ that takes the current state to the torque. Negative values are realised by the braking system; whereas, the positive values are realised by the transmission (the throttle). As a consequence, $T(t) = T\left(v_C(t), \delta(t)\right)$.

To simplify the notation, we introduce

$$\begin{aligned} x(t) = (x_1(t), x_2(t)) &\equiv (\delta(t), v_C(t)) \in \mathbb{R}^2 \\ z(t) &\equiv v_D(t) \in \mathbb{R} \\ b &\equiv \frac{r}{Mr^2 + J} \in \mathbb{R} \\ a(x_1, z) &\equiv rbC_{\mathrm{W}}(x_1, z) \in C^\infty(\mathbb{R}^2, \mathbb{R}_+) \\ u(t) &\equiv bT \in \mathbb{R} \\ (-\underline{u}, \overline{u}) &\equiv (-b\underline{T}, b\overline{T}) \in \mathbb{R}^2_+ \\ x_0 &\equiv (d^0, v_C^0) \in \mathbb{R}^2. \end{aligned} \tag{6}$$

As a result, the equations of motion are given by the following Cauchy problem with $x(0) = x_0$:

$$\begin{aligned} \dot{x}_1(t) &= z(t) - x_2(t) \\ \dot{x}_2(t) &= -a(x_1(t), z(t))\, x_2(t)^2 + u(t), \end{aligned} \tag{7}$$

where $u(t) \in [\underline{u}, \overline{u}]$. The subscripts of x refer to the components of the vector x.

Remark 1. The equation (7) can be used to compute the safety or braking distance $d_s(v_c^0)$ as a function of the initial velocity v_c^0 of the vehicle C. To this end, let $z(t) = 0$, i.e., the vehicle in front instantaneously stops

$$\dot{x}_1(t) = -x_2(t) \quad \text{and} \quad \dot{x}_2(t) \le \underline{u}$$

for $x_0 = (0, v_C^0)$. To compute the braking distance, we apply the Gronwall lemma [48], which we state now for completeness. Suppose that k is a non-negative and bounded function on an interval $[t_0, t_1]$ and l a non-decreasing function on the same interval. If

$$v(t) \le l(t) + \int_{t_0}^{t} k(s) v(s) ds$$

for $t \in [t_0, t_1]$, then

$$v(t) \le \exp\left(\int_{t_0}^{t} k(s) ds\right) l(t).$$

Consequently, by the Gronwall lemma, the time to stop is $t \le \hat{t} \equiv -\frac{v_C^0}{\underline{u}}$ (notice that $\underline{u} < 0$). Hence, the braking distance is at most $d_s(v_C^0) = -\frac{(v_C^0)^2}{2\underline{u}}$.

4.2 Lateral Motion

So far, we have not discussed lateral motion. For the details of modelling, we refer to [37]. In short, the kinematic model of the vehicle C is given by the global position

$$\dot{X} = v_C \cos(\psi + \beta) \tag{8a}$$
$$\dot{Y} = v_C \sin(\psi + \beta), \tag{8b}$$

where v_C is the velocity of the vehicle C, β is the slip angle of the vehicle defined below, and ψ is the yaw angle, that defines the orientation angle of the vehicle w.r.t. the x-axis

$$\dot{\psi} = \frac{v_C}{l} \cos(\beta) \tan(\theta). \tag{9}$$

In (9), l is the vehicle base, the distance between the rear and the front wheels, and θ is the angle between the front wheel and the longitudinal axis of the vehicle, with $\theta \in [\underline{\theta}, \overline{\theta}]$ for $\underline{\theta} < 0$ and $\overline{\theta} > 0$; θ as the control input.

The slip angle of the vehicle is given by the relation

$$\beta \equiv \beta(\theta) = \tan^{-1}\left(\frac{l_\mathrm{r} \tan(\theta)}{l}\right),$$

where l_r is the distance between the centre of gravity and the rear wheel.

5 Linking

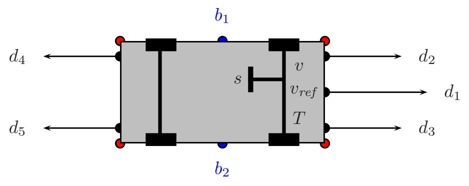
To link the abstract and the concrete model, we must map the symbolic observables and events to observer functions in the controllers. In this work, we assume that each car is equipped with the observers, realised by suitable sensors, and actuators listed in Fig. 3.

The abstract controller LPC takes a view of the traffic snapshot, represented by MLSL formulae built with the atoms $\mathit{free}, \mathit{re}(c), \mathit{cl}(c)$. By Theorem 1, this suffices for the safety check at the abstract level. However, the check *assumes* that the reserved or claimed spaces are large enough. Whether this assumption is true, depends on the concrete controller based on the car dynamics.

5.1 Distance Controller

We first turn to the distance controller DC in each car as formalized by assumption **DC**. Every car E keeps the property $\neg cc$ invariant under time transitions, expressing that “no collision” occurs:

$$\neg cc \equiv \neg \exists c\colon\ c \neq \mathit{ego} \wedge \langle \mathit{re}(\mathit{ego}) \wedge \mathit{re}(c) \rangle\,.$$



We assume that each car is equipped with the following observers:

- $\hat{v}$ gives its own velocity,
- $\hat{d}_1$ gives the distance to the car ahead in the same lane,
- $\hat{d}_2$ ($\hat{d}_3$) give the distance to the car ahead in the left (right) neighboring lane,
- $\hat{d}_4$ ($\hat{d}_5$) give the distance to the car behind in the left (right) neighboring lane, and
- $\hat{b}_1$ ($\hat{b}_2$) tell whether a car on the lane next to the left (right) one is "blinking", indicating a desired lane change to the left (right) neighboring lane.

Also, each car has its blinkers (here shown as small circles at the four corners of the car) and a torque T as actuators. Steering s and desired reference velocity v_{ref} are inputs from the driver.

Fig. 3 Car with observers and actuators

Since the overlap $re(ego) \wedge re(c)$ is symmetric, the distance controller in *ego* must check forward or backward for any other car c. However, considering all cars together, it suffices that each car *ego* checks only that there is "no collision *forward*". Let *c ahead ego* abbreviate the following MLSL formula expressing that car c is immediately ahead of *ego*:

$$c\ ahead\ ego \equiv \begin{pmatrix} \neg re(ego) \\ \wedge \\ \neg re(c) \end{pmatrix} \frown \begin{pmatrix} re(ego) \frown \neg re(ego) \\ \wedge \\ \neg re(c) \frown re(c) \frown \neg re(c) \end{pmatrix}.$$

Then we replace the invariant $\neg cc$ by the following formula:

$$\neg ccf \equiv \neg \exists c \colon c \neq ego \wedge \langle re(ego) \wedge re(c) \rangle \wedge \langle c\ ahead\ ego \rangle .$$

We recall the resulting "forward looking" distance controller DC_f. Note that logically $\neg ccf$ in DC_f is *weaker* than $\neg cc$ in DC, admitting more traffic snapshots. However, when all cars check $\neg ccf$ instead of $\neg cc$, safety remains guaranteed. This is formalized as follows. Consider the abstract setting A, where *all* cars are equipped with DC, and the abstract forward setting A_f, where *all* cars are equipped with DC_f.

Proposition 1 (Safety of DC_f). *Every time transition among traffic snapshots permitted in A_f is also permitted in A.*

In the concrete controller, we have the observable d that implements the abstract safety distance function $d(ego)$ for car *ego* at its current speed. Also, there is the concrete observable $\hat{d}_1$ measuring the distance to the next car c ahead. The formula $\neg ccf$ is satisfied if the inequality $d < \hat{d}_1$ holds. Thus the linking predicate relating the abstract and concrete levels is here

$$\neg ccf \Leftarrow d < \hat{d}_1 .$$

Note that the implication indicates that $d < \hat{d}_1$ admits no more traffic snapshots than $\neg ccf$ does.

5.2 *Lane-Change Controller*

To link the abstract lane change controller LCP to the observers at the concrete level, the MLSL formulae appearing as guards in LCP are replaced by suitable comparisons of observer values read at the concrete level.

Since the distance controller DC is running in parallel to LCP, the safety property $\neg cc$ holds as long as the reservation transition from state q_2 to state q_3 in LCP is *not* performed (cf. Fig. 2 and the proof of Theorem 1). Note that we can *weaken* the guard of any transition in LCP, except for this reservation transition, and the altered lane change controller will stay safe. For example, we may even weaken the guard ϕ_1 to $true$. Then a claim can always be withdrawn, but this does not violate safety.

Regarding the reservation transition from state q_2 to state q_3, the controller will stay safe as long as we *strenghten* its guard ϕ_2, which in LCP is given by the formula

$$\neg pc \equiv \neg\exists c : c \neq ego \wedge \langle cl(ego) \wedge (re(c) \vee cl(c))\rangle$$

expressing that no potential collison occurs. To link $\neg pc$ with the concrete controller, we distinguish the cases of reservation and claim of c.

Case 1 : $\phi_{re} \equiv \neg\exists c : c \neq ego \wedge \langle cl(ego) \wedge re(c)\rangle$. This formula states that no (other) car c on *ego*'s target lane has a reservation that overlaps with *ego*'s claim. The car c may be (i) ahead of *ego* (or aligned with *ego*) or (ii) behind *ego*. In subcase (i), the concrete controller looks forward using the observables d giving the safety distance needed for car *ego* at its current speed and $\hat{d}_t$ (with t either 2 or 3) measuring the distance to the next car c in front of *ego* on the *target* lane of its lane change maneuver. The concrete controller checks the inequality $d_s < \hat{d}_t$. In subcase (ii), the concrete controller looks *backward* using the observables $\hat{d}_b$ (with b either 4 or 5) measuring the distance to the next car behind *ego* on the target lane and $d_{s,max}$, the maximal braking distance of any car, i.e., an *overapproximation* of the actual braking distance of that car. The concrete controller checks the inequality $d_{s,max} < \hat{d}_b$. Thus, the linking predicate relating the abstract and concrete levels is in this case

$$\phi_{re} \Leftarrow d < \hat{d}_t \wedge d_{s,max} < \hat{d}_b .$$

Due to the over-approximation in $d_{s,max}$ the check at the concrete level may be *stronger* than necessary, permitting fewer lane changes than $\neg pc$, but it preserves safety.

Case 2 : $\phi_{cl} \equiv \neg\exists c : c \neq ego \wedge \langle cl(ego) \wedge cl(c)\rangle$.

The formula states that no other car c on *ego*'s target lane has a claim that overlaps with *ego*'s claim. Such a car c may only be in a lane *next* to *ego*'s target lane. In this case, the concrete controller checks with its sensor b_t (with t either 1 or 2) on the side of the target lane for a turn signal of some car c on the lane next to the target lane. The formula ϕ_{cl} is satisfied if $\neg b_t$ holds. Thus, the linking predicate relating the abstract and concrete levels is in this case

$$\phi_{cl} \Leftarrow \neg b_t .$$

Summarising, at the concrete level, we instantiate

$$\phi_2 \equiv (d < \hat{d}_t \wedge d_{s,max} < \hat{d}_b) \wedge \neg b_t ,$$

which by the linking predicates for ϕ_{re} and ϕ_{cl} implies $\neg pc$ at the abstract level.

For the guards of the two withdrawal transitions from state q_1 to state q_3 and from state q_2 to state q_0 in Fig. 2, we put $\phi_1 \equiv \neg\phi_2$ for the above instantiation of ϕ_2. Thus compared with the abstract controller LCP, the guard ϕ_1 is weakened, permitting more withdrawals, but as argued before, this preserves safety.

Altogether, instantiating in the controller in Fig. 2 the formula ϕ_2 by the distance inequalities and blinker sensor values as stated above and ϕ_1 by its negation, we obtain a concrete lane-change controller that we call LCP$_c$. Consider the abstract setting ALC, where all cars are equipped with LCP, and the concrete setting CLC, where all cars are equipped with LCP$_c$.

Proposition 2 (Safety of LCP$_c$). *Every reservation transition among traffic snapshots permitted in CLC is also permitted in ALC.*

Combining Propositions 1 and 2, we obtain:

Theorem 2 (Safety of DC$_c$ and LCP$_c$). *Let $\mathscr{T}_0$ be an initial safe traffic snapshot. Then every traffic snapshot $\mathscr{T}$ that is reachable from $\mathscr{T}_0$ by transitions allowed by the controllers DC$_c$ and LCP$_c$ is safe.*

6 Concrete Controllers

The main focus in this section will be on the longitudinal motion control. Nonetheless, for completeness we will provide a control for changing the lane.

6.1 Longitudinal Control

We will address the assumptions for the distance controller used in Section 5.1 linking the safety to the safety envelope through the variable d. To this end, we propose a sliding mode controller for a vehicle C that maintains the velocity of the vehicle at the reference v_{ref} until the distance d between C and the vehicle D in front is reached. Subsequently, the distance d is kept. If D is out of range of the distance sensor, the controller keeps the velocity at v_{ref}. In the following, we assume that at full throttle, the control $\overline{u}$ is strong enough to overcome the drag. To this end, we notice that $a(x_1,z) \in [0, rbC_C]$ for any $(x_1,z) \in R^2_+$, where the constant b is defined in (6). Let the speed limit be denoted by $\bar{v}$. Consequently, we assume that the maximal control $\overline{u} > rbC_C\bar{v}^2$. By a safe control, we understand a control that keeps the motion of a vehicle safe.

Definition 5 (Safe Control). A *safe controller* for the control system (7) and a function $z : \mathbb{R}_+ \to [0,\bar{v}]$ is a function $u : \mathbb{R}^3 \mapsto \mathbb{R}$ such that the solutions of the dynamical system (7) with $u(t) = u(x(t), z(t))$ satisfy the following condition: If $x_1(0) \geq d$, then $x_1(t) > 0$ for all $t \in \mathbb{R}_+$.

In plain words, Definition 5 says that an on-board controller is safe if: whenever the distance from the controlled vehicle to a vehicle in front is initially greater than d then a collision between these two vehicles will never happen.

Proposition 3 (Existence of a safe controller). *Consider the control system* (7) *and a function* $z : \mathbb{R}_+ \to [0,\bar{v}]$. *Let* $0 \leq v_{\text{ref}} < \bar{v}$, $d \equiv d(\bar{v})$, *and* $\alpha \equiv rbC_C\bar{v}^2$. *Suppose that* $\underline{u} < 0$. *Let* $k > 0$, *and define two affine maps*

$$L_1(x) \equiv x_2 - v_{\text{ref}},\ L_2(x,z) \equiv z - x_2 + k(x_1 - d), \tag{10}$$

and a polyhedral set

$$P(z) \equiv \{x \in \mathbb{R}^2 |\ L_1(x) \leq 0 \text{ and} - L_2(x,z) \leq 0\}. \tag{11}$$

Then the control

$$u(x,z) = \begin{cases} \underline{u} & \text{for } \ x \in \mathbb{R}^2 \setminus P(z) \\ \overline{u} & \text{for } \quad x \in P(z) \end{cases} \tag{12}$$

is safe. Furthermore, the following two properties for the vehicle controlled by the u in (12) *hold:*

1. *If* $x_2(0) > v_{\text{ref}}$ *then* $x_2(t) < x_2(0)$ *for all* $t \in \mathbb{R}_+$ *and there is* $\tau \in \mathbb{R}^+$ *such that* $x_2(t) \leq v_{\text{ref}}$ *for* $t > \tau$.
2. *Let* $\beta \equiv \inf\{\dot{z}(t) |\ t \in \mathbb{R}_+\}$ *and* $\gamma \equiv \sup\{\dot{z}(t) |\ t \in \mathbb{R}_+\}$. *Suppose that* $\underline{u} < \beta$ *and* $\overline{u} > \alpha + \gamma$, *and assume* $0 < k < \min\{\beta - \underline{u}, \overline{u} - \alpha - \gamma\}/\bar{v}$. *Then*
   a. *Let* $0 \leq x_1(0) < d$, *and suppose that the controller* (12) *is such that* $u(t) = \overline{u}$ *holds on an interval* $[0,\tau]$. *Then* $x_1(t) > x_1(0)$ *for all* $t \in [0,\tau]$.
   b. $\lim_{t\to\infty} x_1(t) = d$.

Proof. If $x_1(0) \in \mathbb{R}^2 \setminus P(z)$, then the following holds. There is a family of open intervals $\{(\underline{\tau}_\alpha, \overline{\tau}_\alpha) |\ \alpha \in \Lambda\}$ such that $x(\underline{\tau}_\alpha) \in P(z)$ and if $t \in (\underline{\tau}_\alpha, \overline{\tau}_\alpha)$ then $x(t) \in \mathbb{R}^2 \setminus P(z)$, hence $u(t) = \underline{u}$, and from (7), $x_1(t) > 0$. If $t \in \mathbb{R} \setminus \bigcup_{\alpha\in\Lambda}(\underline{\tau}_\alpha, \overline{\tau}_\alpha)$ then $x(t) \in P(z(t))$, and thus $x_1(t) \geq d$. The last statement follows from the following. If $x(t) \in P(z(t))$, then

$$k(x_1(t) - d) \geq x_2(t) - z(t). \tag{13}$$

And, we consider two cases: $x_2(t) \geq z(t)$ and $z_2(t) < z(t)$. If $x_2(2) \geq z(t)$, then from (13), $x_1(t) \geq d$. If $z_2(t) < z(t)$, then from (7), $x_1(t) \geq x_1(0) \geq d$. Hence, the control (12) is safe.

We prove Property 1 and Property 2 of the proposition. To this end, we observe that for $x \in \mathbb{R}^2 \setminus P(z)$,

$$\dot{L}_1(x,z) = -a(x_1,z)x_2^2 + \underline{u} \leq \underline{u} < 0 \tag{14}$$

$$\begin{aligned}\dot{L}_2(x,z,\dot{z}) &= \dot{z} + a(x_1,z)x_2^2 + k(z-x_2) - \underline{u} \\ &\geq \beta - k\bar{v} - \underline{u} > 0.\end{aligned} \tag{15}$$

Whereas, for $x \in P(z)$,

$$\dot{L}_1(x,z) = -a(x_1,z)x_2^2 + \overline{u} \geq -\alpha + \overline{u} > 0 \tag{16}$$

$$\begin{aligned}\dot{L}_2(x,z,\dot{z}) &= \dot{z} + a(x_1,z)x_2^2 + k(z-x_2) - \overline{u} \\ &\leq \gamma + \alpha + k\bar{v} - \overline{u} < 0.\end{aligned} \tag{17}$$

By (14), Property 1 holds.

We will show Property 2.a. To this end, we notice that $u(t) = \overline{u}$ whenever $x(t) \in P_{z(t)}$. We consider two cases $z(t) > x_2(t)$ and $z(t) \leq x_2(t)$. If $z(t) > x_2(t)$ then $\dot{x}_1(t) = z(t) - x_2(t) > 0$ and Property 2.a follows. Suppose that $z(t) \leq x_2(t)$. Then $0 < k(x_1(t) - d) \geq z(t) - x_2 + k(x_1(t) - d) = L_2(x(t), z(t)) \geq 0$, which is a contradiction.

To show Property 2.b, we observe that by Inequalities (14)–(17), any flow line of (7) intersects the boundary of P at a point say $\tilde{x}$ (transversally), i.e., there is $t_1 \geq 0$ such that $x(t_1) = \tilde{x}$. If $L_1(\tilde{x}) = 0$, then the solution (in a Filippov sense) $x(\cdot)$ is such that $L_1(x(t)) = 0$ for all $t \in [t_1, t_2]$, where t_2 is the time at which $L_2(x(t_2), z(t_2)) = 0$. Subsequently, the Fillipov solution $x(\cdot)$ is such that $L_2(x(t), z(t)) = 0$ for all $t \geq t_2$. As a consequence, $z(t) - x_2(t) + k(x_1(t) - d) = 0$, which is equivalent to

$$\frac{\mathrm{d}}{\mathrm{d}t}(x_1(t) - d) = -k(x_1(t) - d).$$

Hence, $\lim_{t\to\infty} x_1(t) = d$. $\square$

The above proposition shows that there is a control that keeps the distance from the vehicle C to the vehicle in front safe while the velocity of C does not exceed the reference. Also whenever the vehicle C accelerates, $u(t) = \overline{u}$, and initially the distance $x_1(0)$ is less than d then the distance increases, i.e., the traffic situation is no less safe than it was at the beginning. If the distance between C and D was greater than d then there is no future time that they will hit each other.

To avoid discontinuous control and hence abrupt switches between acceleration $\overline{u}$ and deceleration $\underline{u}$, the control (12) can be replaced by a continuous approximation. To this end, we will need an ε-neighbourhood $\partial P^{\varepsilon}(z)$ of the boundary $\partial P(z)$ of the polyhedral set $P(z)$. Subsequently, in $P \setminus \partial P^{\varepsilon}(z)$, we will use u equal to $\overline{u}$, in $\mathbb{R}^2 \setminus (P(z) \cup \partial P^{\varepsilon}(z))$, we will use u equal to $\underline{u}$ and in $\partial P^{\varepsilon}(z))$, we will use the control that is a linear combination of $\overline{u}$ and $\underline{u}$ weighted by the distance to $\partial P(z)$.

These constructions will be detailed below. For this purpose, recall the definitions of L_1, L_2 in (10), and P in (11), and consider

$$\mathbb{L}_1 \equiv L_1^{-1}(0) = \{x \in \mathbb{R}^2 |\, L_1(x) = 0\} \text{ and } \mathbb{L}_{2,z} \equiv \{x \in \mathbb{R}^2 |\, L_2(x,z) = 0\},$$
$$\mathbb{H}_1 \equiv \{x \in \mathbb{R}^2 |\, L_1(x) \leq 0\} \text{ and } \mathbb{H}_{2,z} \equiv \{x \in \mathbb{R}^2 |\, -L_2(x,z) \leq 0\}.$$

For an $\varepsilon > 0$, we define a map $h : [-\varepsilon, \varepsilon] \to [0,1]$ by $y \mapsto \frac{1}{2}\left(\frac{1}{\varepsilon}y + 1\right)$. Let $\mathbb{L}_1^{\varepsilon}$ be the (closed) ε-neighborhood of $\mathbb{L}_1$ (with respect to the Hausdorff metric), $\mathbb{L}_{2,z}^{\varepsilon}$ be the ε-neighborhood of $\mathbb{L}_{2,z}$, $\mathbb{H}_1^{\varepsilon}$ be the ε-neighborhood of $\mathbb{H}_1$, and $\mathbb{H}_{2,z}^{\varepsilon}$ be the ε-neighborhood of $\mathbb{H}_{2,z}$. Furthermore, we define the ε-neighbourhood $P^{\varepsilon}(z)$ of P by

$$P^{\varepsilon}(z) \equiv \mathbb{H}_1^{\varepsilon} \cap \mathbb{H}_{2,z}^{\varepsilon}.$$

Let $x^i(x) \equiv x - \pi_{\mathbb{L}_i}(x)$ for $i \in \{1,2\}$, where $\pi_{\mathbb{L}_1}$ and $\pi_{\mathbb{L}_2}$ are the projections on $\mathbb{L}_1$ and $\mathbb{L}_{2,z}$, respectively. For $l \equiv l(x) = \operatorname{argmax}\{|x^i(x)|| \; i \in \{1,2\}\}$ let

$$y(x) = |x^l| \operatorname{sign}(\langle n^l, x^l \rangle),$$

where $\langle \cdot,\cdot \rangle$ is the scalar product on $\mathbb{R}^2$, n^1 and n^2 are the normal vectors to $\mathbb{L}_1(\cdot)$ and $\mathbb{L}_{2,z}(\cdot)$ pointing into P,

$$n^1 = (0,-1), n^2 = (k,-1).$$

Finally, we are able to define the ε-neighbourhood $\partial P^{\varepsilon}(z)$ of the boundary of $P(z)$

$$\partial P^{\varepsilon}(z) \equiv P^{\varepsilon}(z) \setminus (\mathbb{R}^2 \setminus (\mathbb{H}_1^{\varepsilon} \cup \mathbb{H}_{2,z}^{\varepsilon})).$$

We define $\bar{h} : P^{-\varepsilon}(z) \to [0,1]$ by

$$\bar{h}(x) = h(y(x)).$$

The function $\bar{h}$ takes a point x in the ε-neighbourhood of $\partial P(z)$ and delivers a number between 0 and 1 dependent on the distance to $\partial P(z)$: 0 when the distance is ε and x is outside P and 1 when the distance is ε and x is inside P. The control is then

$$u(x,z) = \begin{cases} \underline{u} & \text{for} \quad x \in \mathbb{R}^2 \setminus P^{\varepsilon}(z) \\ (1-\bar{h}(x))\underline{u} + \bar{h}(x)\overline{u} & \text{for} \quad x \in P^{-\varepsilon}(z) \\ \overline{u} & \text{for} \quad x \in P(z) \setminus P^{-\varepsilon}(z). \end{cases}$$

The parameter ε is to be chosen as a tradeoff between the accuracy of tracking the distance d and "evenness" of the control. The bigger ε is, the more even and less accurate is the control.

6.2 Lane Change

The control for lateral motion is discussed in [37]. For completeness of our study, we propose a facile feedforward control for changing the lane. To avoid a collision during the maneuver of changing the lanes, it is assumed that the minimum distance d to the front vehicles in the current lane and the neighboring target lane is big enough, i.e., greater than the sum of the maximal braking distance of the vehicle C and the distance $\int_0^{t_{lc}} v_C(t)dt$ traveled by C during the lane change.

Recall the lateral motion given by the lateral position Y in (8b) and the yaw angle ψ in (9). We will use the notation

$$b(\theta) \equiv b(\theta, v_C) \equiv \frac{v_C}{l} \cos(\beta(\theta)) \tan(\theta).$$

The next proposition characterises the the lateral motion

Proposition 4. *Suppose* $b(\theta) \neq 0$. *Then the solution of* (8b) *and* (9) *belongs to the graph* $\Gamma \equiv \{(\psi, Y) \in]-\pi, \pi[\times \mathbb{R} |\ Y = F(\psi)\}$ *of the function*

$$F \equiv F_{\theta, y_0, \psi_0, v_C} : \psi \mapsto \tilde{y}_0(y_0, \psi_0) - \frac{v_C}{b(\theta)} \cos(\psi + \beta(\theta)),$$

where $\tilde{y}_0(y_0, \psi_0) = y_0 + \frac{v_C}{b(\theta)} \cos(\psi_0 + \beta(\theta))$, *and* y_0 *is the initial lateral position, and hence* ψ_0 *is the initial yaw angle.*

Proof. The tangent space $T_{(\psi,Y)}\Gamma$ to the graph γ at any point $(\psi, Y) \in \Gamma$ is given by

$$T_{(\psi,Y)}\Gamma = \left\{ \alpha \left(1, \frac{\partial F}{\partial \psi}(\psi, Y)\right) \in \mathbb{R}^2 \middle|\ \alpha \in \mathbb{R} \right\},$$

but $\frac{\partial F}{\partial \psi}(\psi, Y) = \frac{v_C}{b(\theta)} \sin(\psi_0 + \beta(\theta))$, and hence by (8b) and (9) we have $(\dot{\psi}, \dot{Y}) \in T_{(\psi,Y)}\Gamma$. $\square$

To change the lane, we change the state (Y, ψ) from $(y_0, 0)$ to $(y_1, 0)$. Without loss of generality, it is assumed that $y_0 > y_1$.

6.2.1 Manoeuvre with Constant Velocity

If we suppose that the velocity v_C during the entire manoeuvrer is kept constant, then suppose that $(\theta_0, \theta_1) \in [\underline{\theta}, 0) \times (0, \overline{\theta}]$ are such that the equation $F_{\theta_0, y_0, 0, v_C}(\psi) = F_{\theta_1, y_1, 0, v_C}(\psi)$, or equivalently

$$\tilde{y}_0(y_0, 0) - \tilde{y}_0(y_1, 0) + v_c \left(\frac{\cos(\psi + \beta(\theta_1))}{b(\theta_1)} - \frac{\cos(\psi + \beta(\theta_0))}{b(\theta_0)} \right) = 0,$$

has the solution $\hat{\psi}$. The proposed manoeuvre consists of

1. turning the front wheels from 0 to the angle $\theta_0 > 0$,
2. waiting until the orientation angle ψ is $\hat{\psi}$,
3. turning the wheels to the angle $\theta_1 < 0$,
4. waiting until the orientation angle ψ reaches 0,
5. finally turning the front wheels back to 0.

6.2.2 Manoeuvre with Varying Velocity of the Vehicle

Suppose the vehicle velocity v_C is piecewise constant on possibly very short time intervals. Let $\theta^*(t)$ be the solution of the following equation

$$F^*(\theta^*(t)) \equiv F_{\theta^*(t),Y(t),\psi(t),v_C(t)}(0) = y_1.$$

Notice that $\theta^*(t)$ depends on the current velocity $v_C(t)$.

Then the lane-change manoeuvre consists of

1. turning the front wheel from 0 to the angle θ_0,
2. waiting until the yaw angle $\psi(t)$ reaches ψ^* for some $\psi^* \in]0, \pi/2[$,
3. keeping the wheels at the angle $\theta(t) = \theta^*(t)$ until the orientation of the vehicle reaches 0 yaw angle.
4. turning the front wheels back to 0.

Both proposed controllers are feed-forward, thus a linear control [37] is to be implemented to remove deviations from the lateral reference y_1. The time t_{lc} of the manoeuvre depends on the vehicle velocity, v_C, and it is used in the guard of the abstract controller LCP depicted in Fig. 2.

7 Related Work

In the following, we consider related work within the categories of verification, hierarchical design approaches, spatial logics, and traffic maneuvers.

Automatic Verification. Most approaches to the automatic verification of hybrid systems represent discrete control and continuous dynamics together in one formal model, e.g., a hybrid automaton [2] or a hybrid program [36]. Whereas the reachability of locations is decidable for timed automata [3], this is in general not true for hybrid automata [18]. These limitations are overcome by using suitable abstractions and symbolic representations.

Model checking of linear hybrid automata by examining the reachable state space started with the tool HyTech [19]. More advanced techniques are incorporated in the tools PHAVer [12] and SpaceEx [13]. An alternative to these reachability-based methods are bounded versions of model checking using SAT-based techniques modulo the theory of ordinary differential equations [11, 10]. The concept of local theory extensions has been applied to proving safety properties of hybrid systems in [6]. Interactive theorem proving for hybrid systems in the context of an extended dynamic logic is pursued in [36]. For Hybrid CSP an experimental tool was developed [49].

Hierarchical Design. To simplify the analysis of hybrid systems, several approaches to controller design for hybrid systems have pursued a *separation* of the dynamics from the control layer.

An early work with an example of keeping distance between vehicles, is the paper by Nadjm-Tehrani and Strömberg [34], where they study the mapping from the continuous state space to the discrete state space. In the approach, the two models are combined to a hybrid model, and the linkage from the modes of the continuous model to the discrete modes is done by a *Characterizer* that generates events and a *Selector* for set-points. These could be characterized by linking predicates as done in this paper, that would allow a clearer separation of the models.

Raisch et al. [32, 31] introduce abstraction and refinement to support a hierarchical design of hybrid control systems. However, this line of work stays within the same underlying model. Instead, the work here operates with separate models, because they can be tailored to the reasoning tools available for respectively automata and logics and those available for conventional control theory. Here, we are more in accordance with the work in [38], that deals with semantic alignment of heterogeneous models. The linking predicates introduced in the current work may make the alignment easier, because it relates only specific quantities and not full models.

Another inspiration for our work has been the approach pursued by Van Schuppen et al. [17] that works upwards from what we call the concrete model and introduce synthesis of control laws for piecewise-affine hybrid systems based on simplices, resulting in a discrete controller with transitions between the simplices. This may be an approach to finding a symbolic state space, when there is no obvious way to partition it.

Spatial Logic. Work on spatial logic often focusses on qualitative spatial reasoning [43] as exemplified in the region connection calculus [39]. We have used the spatial logic MLSL [21] to reason abstractly on highway traffic. The logic gives a compact formulation of properties and configurations, and an ability to compose and decompose them as well as a potential for deductions [26]. MLSL is inspired by interval temporal logic [33], the Duration Calculus [50], and the Shape Calculus [40]. It is a two-dimensional extension of interval temporal logic, where one dimension has a continuous space (the position in each lane) and the other has a discrete space (the number of the lane).

In [41], hybrid automata are considered where invariants and guards are expressed in a spatio-temporal logic $S4_u$. However, there is no separation of space and dynamics as in our approach.

Traffic Maneuvers. A very influential effort was the California PATH (Partners for Advanced Transit and Highways) project on automated highway systems for cars driving in groups called platoons [44]. The manoeuvres include joining and leaving the platoon, and lane change. Lygeros et al. [28] sketch a safety proof for car platoons taking car dynamics into account, but admitting *safe collisions*, i.e., collisions at a low speed. Not all scenarios of multi-lane traffic are covered in their proof.

Platzer et al. [5, 27] represent traffic applications in a differential dynamic logic $d\mathscr{L}$ that is supported by the theorem prover KeYmaera [36]. This logic does not separate space (symbolic model) from dynamics (concrete model), that is at the heart of our approach. The paper [1] proposes a bottom-up strategy, where a concrete model is gradually abstracted to Markov chains, for which the set of reachable states is analysed.

On highways, the analysis of safety is simplified because all cars drive in one direction. More difficult to analyse are country roads with opposing traffic. The safety of overtaking manoeuvres on such roads has been proven in [21]. Even more degrees of freedom in traffic manoeuvres can be found in urban traffic. The manoeuvres at crossings has been studied in [45].

Since driving assistants are liable to hit the road very soon, the effort at providing clear modelling and verification for this application area is very important.

Linking. For linking abstract and concrete data-manipulating systems the concepts of data and operation refinement with corresponding simulation-based proof techniques are well-known [8, 9]. Note that these techniques start by relating abstract and concrete data *variables*, that is not quite suitable in our setting, where we have to relate abstract *predicates* on reservations and claims to concrete sensor values. The transfer of temporal properties from abstract to concrete transitions systems via simulations and bisimulations is well-understood in the area of model checking [16].

8 Conclusion

The paper has presented an approach to hybrid systems modelling where an abstract model is built in theories that are decidable modulo symbolic guards and actions while a concrete model uses conventional continuous time for which controllers are developed. The key point is that these two worlds are connected by *linking predicates*, so the concrete model is a refinement of the abstract one.

In the following, we discuss pros and cons of the approach for the individual steps and for the overall work.

Symbolic Model. A symbolic model is well known from a controller side, which can be built using timed automata. Also the use of symbolic guards and actions is intuitively easy. Note that time should enter only as timeouts on communications. These timeouts occur at the interface to the lower level concrete model or in communication protocols for interaction between the state machines.

When this is done, it is feasible to use model checking with a simplified environment model that assigns suitable values from finite domains to the predicates, and accept actions of similar finite types. Thus, an exhaustive automated verification is possible, although it has not been done in this paper, because we consider the decomposition and linking the main points. Also, encoding the spatial model is a major effort. Steps in this direction have been taken by S. Linker in formalising a safety proof for a controller specification of [25] using the theorem prover ISABELLE.

Defining a suitable state space is intrinsically difficult. We have used a spatial logic to structure it. The logic gives a compact formulation of properties and configurations, and an ability to compose and decompose them as well as a potential for deductions. However, if a developer is not familiar with logic, it may be easier to stay with set theory, i.e., use the semantics underlying the logic. This would also be the case if a model checking tool is used, because the logic would have to be semantically encoded in most cases. The simple CTL or LTL logics used in model checkers are not nearly as expressive as spatial logics. Thus, the logic is not essential for the approach or even the application case, but it is a neat shorthand.

Concrete Model. Identification of the concrete model and controller development is well known and is highly application dependent. In the current presentation, the modelling and controller design is very general. For real applications there is much engineering to do, but this is not relevant for this exposition.

During the development, one must have an eye on the predicates of the symbolic model, so it is feasible to construct observers that match the guards, and handle set points presented by the actions.

Linkage. The linking predicates are the formal outcome of elaborate discussions concerning the interface of the two models. They represent the point where many real application projects fail, because engineering traditions from software development and control system development meet. The advantage of the approach is that the two sides have to meet and agree. An issue that is common to top-down approaches is that the defined interface turns out to be either unimplementable in the concrete or inadequate for the abstract verification. Here, we see no magic bullet.

Overall comments. The approach seems well suited for application areas, where a collection of semi-autonomous entities have to coordinate to achieve common objectives. In a tightly coupled application, where there is a tight centralized supervisor, it is most likely easier to stay with a one level concrete model, typically a conventional hybrid automaton.

Acknowledgements. We thank three anonymous reviewers for their helpful comments.

References

1. M. Althoff, O. Stursberg, and M. Buss. Safety assessment of autonomous cars using verification techniques. In *American Control Conference (ACC) 2007*, pages 4154–4159. IEEE, 2007.
2. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138(1):3–34, 1995.
3. R. Alur and D. L. Dill. A theory of timed automata. *TCS*, 126(2):183 – 235, 1994.

4. A. D. Ames, E. A. Cousineau, and M. J. Powell. Dynamically stable bipedal robotic walking with nao via human-inspired hybrid zero dynamics. In *HSCC 2012*, pages 135–144. ACM, 2012.
5. N. Arechiga, S. M. Loos, A. Platzer, and B. H. Krogh. Using theorem provers to guarantee closed-loop system properties. In *American Control Conference (ACC) 2012*, pages 3573–3580. IEEE, 2012.
6. W. Damm, C. Ihlemann, and V. Sofroni-Stokkermans. PTIME parametric verification of safety properties for reasonable linear hybrid systems. *Mathematics in Computer Science*, 5(4):469–497, 2011.
7. W. Damm, E. Möhlmann, and A. Rakow. Component based design of hybrid systems: A case study on concurrency and coupling. In *HSCC 2014*, pages 145–150. ACM, 2014.
8. W.-P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*. Cambridge University Press, 1998.
9. J. Derrick and E. A. Boiten. *Refinement in Z and Object-Z: Foundations and Advanced Applications*. Springer, 2014.
10. A. Eggers, M. Fränzle, and C. Herde. SAT modulo ODE: A direct SAT approach to hybrid systems. In S. D. Cha, J. Choi, M. Kim, I. Lee, and M. Viswanathan, editors, *ATVA 2008*, volume 5311 of *LNCS*, pages 171–185. Springer, 2008.
11. M. Fränzle and C. Herde. HySAT: An efficient proof engine for bounded model checking of hybrid systems. *Formal Methods in Syst. Design*, 30(3):179–198, 2007.
12. G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. *STTT*, 10(3):263–279, 2008.
13. G. Frehse, C. Guernic, A. Donzé, S. Cotton, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *CAV 2011*, volume 6806 of *LNCS*, pages 379–395. Springer, 2011.
14. G. Frehse, R. Kateja, and C. L. Guernic. Flowpipe approximation and clustering in space-time. In *HSCC 2014*, pages 203–212, 2013.
15. R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors. *Hybrid Systems*, volume 736 of *LNCS*. Springer, 1993.
16. O. Grumberg. Abstraction and reduction in model checking. In H. Schwichtenberg and R. Steinbrüggen, editors, *Proof and System-Reliabilty*, volume 62 of *Nato Science Series II. Math., Physics and Chemistry*, pages 213–260. Kluwer Academic Publishers, 2002.
17. L. Habets, P. Collins, and J. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Trans. on Automatic Control*, 51(6):938–948, 2006.
18. T. A. Henzinger. The theory of hybrid automata. In *LICS 1996*, pages 278–292. IEEE, 1996.
19. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. *STTT*, 1(1-2):110–122, 1997.
20. A. Hereid, S. Kolathaya, M. S. Jones, J. Van Why, J. W. Hurst, and A. D. Ames. Dynamic multi-domain bipedal walking with atrias through slip based human-inspired control. In *HSCC 2014*, pages 263–272. ACM, 2014.
21. M. Hilscher, S. Linker, and E.-R. Olderog. Proving safety of traffic manoeuvres on country roads. In Z. Liu, J. Woodcock, and H. Zhu, editors, *Theories of Programming and Formal Methods*, volume 8051 of *LNCS*, pages 196–212. Springer, 2013.
22. M. Hilscher, S. Linker, E.-R. Olderog, and A. P. Ravn. An abstract model for proving safety of multi-lane traffic manoeuvres. In S. Qin and Z. Qiu, editors, *ICFEM 2011*, volume 6991 of *LNCS*, pages 404–419. Springer, 2011.
23. C. A. R. Hoare and J. He. *Unifying Theories of Programming*. Prentice Hall, 1998.
24. E. A. Lee and H. Zheng. Operational semantics of hybrid systems. In *HSCC 2005*, pages 25–53, 2005.
25. S. Linker. *Proofs for Traffic Safety: Combining Diagrams and Logic*. PhD thesis, Dept. of. Comp. Sci, Univ. of Oldenburg, 2015.
26. S. Linker and M. Hilscher. Proof theory of a multi-lane spatial logic. *Logical Methods in Computer Science*, 11(3), 2015. See: www.lmcs-online.org/ojs/regularissues.php?id=46.

27. S. M. Loos, A. Platzer, and L. Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In M. J. Butler and W. Schulte, editors, *FM 2011*, volume 6664 of *LNCS*, pages 42–56. Springer, 2011.
28. J. Lygeros, D. N. Godbole, and S. S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Trans. on Automatic Control*, 43(4):522–539, 1998.
29. N. A. Lynch, R. Segala, and F. W. Vaandrager. Hybrid I/O automata revisited. In *HSCC 2001*, pages 403–417, 2001.
30. MathWorks. Stateflow, 1995.
31. T. Moor, J. Raisch, and J. Davoren. Admissiblity criteria for a hierarchical design of hybrid systems. In *Proc. IFAD Conf. on Analysis and Design of Hybrid Systems*, pages 389–394, St. Malo, France, 2003.
32. T. Moor, J. Raisch, and S. O'Young. Discrete supervisory control of hybrid systems based on l-complete approximations. *Discrete Event Dynamic Systems*, 12:83–107, 2002.
33. B. Moszkowski. A temporal logic for multilevel reasoning about hardware. *Computer*, 18(2):10–19, 1985.
34. S. Nadjm-Tehrani and J. Strömberg. From physical modelling to compositional models of hybrid systems. In H. Langmaack, W. P. de Roever, and J. Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems, Third International Symp. Organized Jointly with the Working Group Provably Correct Systems – ProCoS*, volume 863 of *LNCS*, pages 583–604. Springer, 1994.
35. E.-R. Olderog, A. Ravn, and R. Wisniewski. Linking spatial and dynamic models for traffic maneuvers. In *54th IEEE Conference on Decision and Control (CDC)*. IEEE, Dec 2015. 8 pp.
36. A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Spinger, 2010.
37. R. Rajamani. *Vehicle Dynamics and Control*. Mechanical engineering series. Springer Science, 2006.
38. A. Rajhans and B. H. Krogh. Compositional heterogeneous abstraction. In *HSCC 2013*, pages 253–262. ACM, 2013.
39. D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proc. 3rd Int'l Conf. Knowledge Representation and Reasoning*, 1992.
40. A. Schäfer. A calculus for shapes in time and space. In Z. Liu and K. Araki, editors, *ICTAC 2004*, volume 3407 of *LNCS*, pages 463–478. Springer, 2005.
41. Z. Shao and J. Liu. Spatio-temporal hybrid automata for cyber-physical systems. In Z. Liu, J. Woodcock, and H. Zhu, editors, *ICTAC 2013*, volume 8049 of *LNCS*, pages 337–354. Springer, 2005.
42. K. Sreenath, C. R. Hill, Jr., and V. Kumar. A partially observable hybrid system model for bipedal locomotion for adapting to terrain variations. In *HSCC 2013*, pages 137–142. ACM, 2013.
43. J. van Benthem and G. Bezhanishvili. Modal logics of space. In M. Aiello, I. Pratt-Hartmann, and J. Benthem, editors, *Handbook of Spatial Logics*, pages 217–298. Springer Netherlands, 2007.
44. P. Varaija. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, AC-38(2):195–207, 1993.
45. M. Werling, T. Gindele, D. Jagszent, and L. Gröll. A robust algorithm for handling traffic in urban scenarios. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 168–173, Eindhoven, NL, 2008.
46. J. Woodcock and J. Davies. *Using Z – Specification, Refinement, and Proof.* Prentice Hall, 1996.
47. M. Zabat, N. Stabile, S. Farascaroli, and F. Browand. The aerodynamic performance of platoons: A final report. http://escholarship.org/uc/item/8ph187fw, UC Berkeley, 1995.
48. J. Zabczyk. *Mathematical Control Theory – An Introduction*. Birkhäuser, 2008.
49. N. Zhan, S. Wang, and H. Zhao. Formal modelling, analysis and verification of hybrid systems. In Z. Liu, J. Woodcock, and H. Zhu, editors, *Unifying Theories of Programming and Formal Engineering Methods*, volume 8050 of *LNCS*, pages 207–281. Springer, 2013.

50. C. Zhou, C. Hoare, and A. Ravn. A calculus of durations. *IPL*, 40(5):269–276, 1991.
51. J. Ziegler, P. Bender, T. Dang, and C. Stiller. Trajectory planning for bertha – A local, continuous method. In *2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, June 8-11, 2014*, pages 450–457, 2014.