Can we build it: formal synthesis of control strategies for cooperative driver assistance systems

WERNER DAMM, HANS-JÖRG PETER, JAN RAKOW and BERND WESTPHAL

Mathematical Structures in Computer Science / Volume 23 / Special Issue 04 / August 2013, pp 676 - 725
DOI: 10.1017/S0960129512000230, Published online: 08 July 2013

Math. Struct. in Comp. Science (2013), *vol.* 23, *pp.* 676–725.
doi:10.1017/S0960129512000230

Can we build it: formal synthesis of control strategies for cooperative driver assistance systems[†]

WERNER DAMM[‡], HANS-JÖRG PETER[§], JAN RAKOW[¶] and BERND WESTPHAL[‖]

[‡]*Carl von Ossietzky Universität Oldenburg, Interdisciplinary Research Centre on Safety Critical Systems and OFFIS Transportation, Germany*
Email: damm@offis.de
[§]*Universität des Saarlandes, Germany*
Email: peter@cs.uni-saarland.de
[¶]*Carl von Ossietzky Universität Oldenburg, Germany*
Email: jan.rakow@informatik.uni-oldenburg.de
[‖]*Albert-Ludwigs-Universität Freiburg, Germany*
Email: westphal@informatik.uni-freiburg.de

Received 3 April 2011; revised 11 November 2011

We propose a design and verification methodology supporting the early phases of system design for cooperative driver assistance systems, focusing on the realisability of new automotive functions. Specifically, we focus on applications where drivers are supported in complex driving tasks by safe strategies involving the coordinated movements of multiple vehicles to complete the driving task successfully. We propose a divide and conquer approach for formally verifying timed probabilistic requirements on successful completion of the driving task and collision freedom based on formal specifications of a set of given manoeuvring and communication capabilities of the car. In particular, this allows an assessment of whether they are sufficient to implement strategies for successful completion of the driving task.

1. Introduction

Future driver assistance systems will increasingly employ Car2X (Baldessari *et al.* 2007) communication capabilities and X-by-wire driving capabilities to support drivers in complex driving tasks or collision avoidance in either proposing or automatically carrying out coordinated manoeuvres involving multiple vehicles (Frese *et al.* 2008; Lee 2008; Baskar 2009). This paper proposes a design and verification methodology for the early phases of system design, focusing on the realisability of such new functions on a given platform of vehicle communication and manoeuvring capabilities. In particular, we analyse whether a given set of vehicle capabilities forms a sufficient basis for supporting the

[†] This research was partially supported by the German Science Foundation within the Transregional Collaborative Research Centre TR14 AVACS and the VW-Vorab Research Group IMOST.

implementation of a given new driving function, and if it is, we then synthesise a cooperation strategy allowing the given driving task to be completed successfully.

We will demonstrate our approach using an automated highway entry assistance system, where the task is to assure that an ego-car will be able to enter the highway within a specified time-bound and a given probability under specified assumptions characterising traffic density by engaging in a coalition with cars already on the highway. These cars, provided they are willing to adapt their driving behaviour according to a synthesised strategy, allow the ego-car to access the highway within the desired time-bound and probability, while ensuring collision avoidance. Typical communication service specifications take the form of timed probabilistic guarantees of message transmissions under given activation and invariant conditions characterising distance and overall communication load – see, for example, Eichler (2009). Typical platform manoeuvring services include maintaining a given distance to a lead car, lane changes and automated cruise control with head distance control, and following pre-computed parametrised trajectories, with parameter and trajectory dependent characterisations of minimal and maximal latencies from service initiation time to service completion. Manoeuvre service specifications include activation conditions characterising traffic situations under which this service may be invoked, assumptions on traffic environments that must hold throughout the specified latency (violation of such invariants will lead to controlled abortion of the service) and invariants guaranteed by the service. Violations of the assumptions may be caused by irregular behaviour of other cars (for example, from uncontained failures or driver behaviour), or unforeseen road conditions, such as icy conditions or obstacles on the road caused, for example, by a truck shedding a badly secured cargo. We assume that an autonomous layer provides for both sufficient detection of and reaction to such rare events, and that it will reduce the visible effect these have on the coordination strategy for abortion signals.

We choose two-player games as the fundamental computation model for capturing the reactive interplay between a hostile environment and a friendly controller (which is, in our case, the coalition formed by the ego-car). To express the different dimensions of vehicle dynamics, stochastic delays in Car2X communications and strategy synthesis for vehicles dynamically entering coalitions, we propose as the semantic domain, timed probabilistic hybrid topology-labelled (Bauer *et al.* 2007) game structures. While such a semantic domain can be formally defined, the main result of the current paper is the derivation of a proof-rule breaking down the formal proof of the existence of cooperation strategies meeting the given timeliness and success-rate requirements for driving tasks to a set of computationally tractable verification synthesis problems, each focusing on a few dimensions of expressivity. To this end, we propose a layered design structure for advanced cooperative driver assistance systems (ADAS), which completely decouples the determination of a cooperative control strategy from the realisation of the dynamic control strategy of individual driving tasks. In particular, the cooperation strategy determines coalition partners based on current traffic situation, initiates Car2Car communication to establish the coalition and determines, based on the observation of cars outside the coalition, for each car inside the coalition the initiation of services of the autonomous control layer. We provide an abstraction of the traffic environment around a representative highway segment, such as an entry ramp, which gives a discrete

safe over-approximation of the traffic environment around the ego-car within a predetermined distance, and dynamically creates or hides cars outside this perimeter. We prove that winning strategies synthesised in this abstract setting yield winning strategies in any concrete traffic environment, and use a simple off-line analysis to quantify the effect of message loss on the likelihood of establishing the desired coalition in a given time window. Taken together, these steps allow a reduction of the verification of time-bounded probabilistic requirements on successful completion of driving-tasks to a strategy synthesis in abstract traffic environments and a local verification of controller realisations for dynamic control, all of which can be fully automated.

The main contribution of this paper is to provide a design methodology for cooperative driver assistance systems allowing the use of recent advances in the automatic analysis of complex systems. In particular, it draws on results from:

— alternating logics (Alur *et al.* 2002; Henzinger and Prabhu 2006);
— spotlight abstraction (Westphal 2008; Wachter and Westphal 2007), which allows us to reduce reasoning about systems with an unbounded number of subsystems and dynamically changing communication topology to reasoning about a finite number of subsystems;
— probabilistic verification (see, for example, Bianco and de Alfaro (1995));
— verification of hybrid systems (see, for example, Alur *et al.* (1992) and Damm *et al.* (2006a));

and combines them to form a design approach for cooperative driver assistance systems that offers abstractions that are natural to designers of such systems. At the same time, it provides a modularisation of formal verification steps to create tractable subproblems, such as automata-based synthesis (see, for example, Pnueli and Rosner (1989), Schewe and Finkbeiner (2007) and Ehlers (2010)), to determine global cooperation strategies and the verification of hybrid controllers against their service specifications in order to address the purely local dynamic control capabilities invoked by the cooperation strategy.

Early research on intelligent cooperating vehicles includes the California PATH Project (Hsu *et al.* 1991; Tomlin *et al.* 1996), where control strategies for the autonomous control layer were proposed and have been verified against multiple prioritised requirements, such as safety, comfort and efficiency (Tomlin *et al.* 1996; Tomlin *et al.* 1998a). In particular, a layered approach was first proposed in the PATH project (Broucke and Varaiya 1997), though there it was restricted to the formation of platoons on intelligent highways using centralised roadside control through segment controllers. The technical report Lygeros *et al.* (1997) gives a detailed design of both the cooperation layer and the autonomous control layer for the leader, and establishes safety, comfort and efficiency requirements based on a third-order dynamic model of vehicle dynamics using zero-sum dynamic games. Lygeros *et al.* (1997) gives a fully analytic treatment, whereas we propose a design methodology aiming at automatic verification.

While our treatment is currently restricted to time-bounded reachability and safety, we have investigated in Damm and Finkbeiner (2011) the synthesis of optimal remorse-free strategies for a prioritised set of objectives given as LTL formulae, and we intend to integrate this into the proposed design methodology in later work.

On-line predictive control has been proposed (Althoff *et al.* 2007; Althoff *et al.* 2008) to anticipate the trajectories of surrounding vehicles to determine actions of the ego-car, such as in the large scale collaborative research centre SFB-TR 28 on cognitive vehicles[†], and the AMICI project in the Netherlands (Bellemans *et al.* 2006). In our approach, the strategy is synthesised at design-time and uses a full assessment of the dynamics of all vehicles in the perimeter of the ego-car in a sufficiently precise finite abstraction, employing invariants reflecting normal driving behaviours. We cater for any deviations between the real dynamic traffic situation and the abstraction used for the strategy synthesis (such as caused by traffic-rule violations, failures, or rare events) by incorporating a reflex layer that protects a safety envelope around each vehicle, which forces the aborting of manoeuvring services (which are otherwise guaranteed to complete) and autonomous correcting actions when there is a risk of a violation of a safety envelope. We also allow the strategy to re-determine both the desired coalition and the selection of manoeuvring services in the case of aborting. By contrast, Frese and Beyerer (2010) searches for strategies controlling *all* vehicles and employs heuristic methods from artificial intelligence, such as tree-search, to determine strategies for coordinated vehicle movements. An excellent survey for alternative methods for controlling all vehicles to perform collision-free driving tasks is given in Frese (2010). However, both methods restrict the analysis to a small number of vehicles, in contrast to our approach, which is based on safe abstractions guaranteeing collision freedom in achieving the driver's objectives, while taking into account the complete traffic situation.

In the aerospace domain, high-level objectives, such as maximising throughput and energy efficiency while maintaining safety, have led to new concepts for air-traffic control, such as free flight. There is a significant body of work on the verification of the safety of such control strategies as well as on design rules that ensure safety (Tomlin *et al.* 1998b; Lygeros and Lynch 1998; Lygeros *et al.* 1997; Richards and How 2002; Damm *et al.* 2006b; Damm *et al.* 2007). While the problem is *a priori* higher dimensional than for coordinated vehicle movements, the additional constraints for ground traffic coordination coming from expected and unexpected obstacles and the closer proximities, together with the drastically less regulated behaviour of drivers, make automatic vehicle traffic coordination much more challenging.

1.1. *Structure of the paper*

Section 2 provides background on advanced driver assistance systems, illustrated by a highway-entry assistance system example, which will be used throughout the paper. In particular, Section 2.2 describes the proposed design methodology, including a formal definition of service specifications for communication and manoeuvre platform capabilities, as well as what we call cooperation skeletons.

Section 3 gives a formal mathematical model allowing us to reason about the capabilities of a coalition of vehicles created to perform driving tasks based on vehicle capabilities characterised by service specifications in an arbitrary traffic environment.

[†] See `www.kognimobil.org` for details.

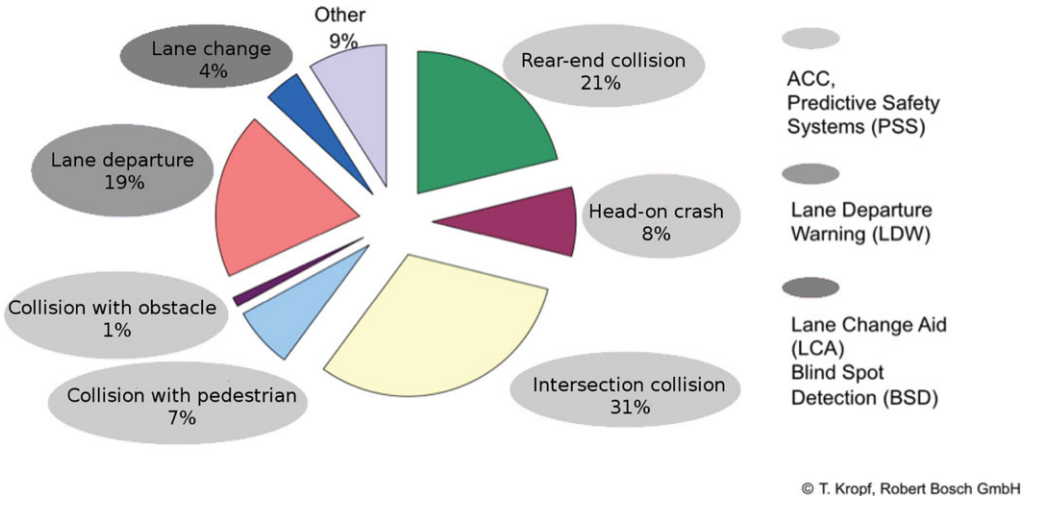


Fig. 1. (Colour online) Source of road accidents in Germany 2002.

Section 4 contains the main result of the paper in proving that the existence of winning strategies for time-bounded probabilistic completions of driving tasks can be established by lifting winning strategies in a certain finite abstract domain to the setting of unbounded traffic environments. This allows us to reduce the overall design problem into separate verification and synthesis problems, which have natural interpretations from an application perspective, and are at the same time computationally tractable using current verification technology.

In Section 5, we report on experimental results based on a prototype tool chain for synthesising winning strategies for cars in traffic environments modelled in the SMI formalism.

Finally, in Section 6, we assess the results achieved from an industrial perspective and point out directions for further research.

2. Driver assistance systems

Advanced driver assistance systems are seen as a key instrument for achieving the objectives of the European Commission in drastically reducing road fatalities. Figure 1 shows the source of road accidents in Germany based on statistics from BASt[†] for 2002, and how various classes of driver assistance systems are viewed as being able to cope with the particular class of road accidents.

Various projects[‡] are developing prototypes of such systems and evaluating their effect on contributing to the reduction of road accidents. We consider in this paper the design of a highway entry assistance system HWYASSIST as an example. This work has been

[†] See `http://www.bast.de` for details.

[‡] Examples include the Integrated Project Prevent – see `http://www.prevent-ip.org` for its publicly available final report.

carried out jointly in the IMOST project (Integrated Modelling for Safe Transportation)[†] with the DLR Institute of Transportation Systems[‡], which focuses on methods and tools for the design of advanced driver assistance systems. Such systems can range in functionality from pure warning systems, which alert the driver about risks stemming from recognised driver actions likely to cause an accident, through systems that take over control autonomously in driving situations where the driver can no longer prevent an accident, to the next generation of advanced cooperative driver assistance systems, which exploit Car2X networking so that multiple cars assist the ego-car in successfully completing a manoeuvre. In all these scenarios, the driver always maintains the right to take over control, as required by the Vienna convention. This requires significant additional complexity compared with fully autonomous driving scenarios, which can exclude interference by drivers.

2.1. *Towards formalising the requirements for driver assistance systems*

We will begin by highlighting some of the intricacies in requirement formalisation for advanced driver assistance systems. This section uses an incremental refinement of an informal high-level functional requirement for HWYASSIST to motivate the choice of a particular class of probabilistic timed ATL formulae. These will be used in the following sections to create formal requirement specifications for reasoning about an unbounded number of traffic agents for ADAS.

Consider first a typical high-level functional requirement for HWYASSIST:

Whenever the driver enters the ramp, the driver is safely guided onto the highway. (R0)

The term 'guided' suggests that an implementation that guarantees safety by just stopping the car on the highway access is not accepted. We will now develop this in a way that already leans towards the logic DPTATL we will introduce in Section 3.3. Note that because of both the driver's expectations and the finite length of the ramp, the ego-car should be able to enter the highway within some specified time t after entering the ramp. So we have:

$$\Box\,(ego.ramp \implies ((ego.safe \wedge ego.ramp)\,\mathbf{U}_{\leqslant t}\,(ego.safe \wedge ego.highway))) \qquad \text{(R1)}$$

In general, this requirement cannot be implemented – just consider the situation in Figure 2 (a), where there is a traffic accident on the highway causing a traffic jam. We will thus refine our requirement by introducing predicates characterising traffic situations. The formal model discussed in Section 3 will allow us to express properties such as 'there is a gap of sufficient size with relative distance d to the ego-car moving with relative speed s'. It will also allow us to refer to cars in front of and behind the gap and observe whether the adjacent lane is completely free, and otherwise characterise situations where other cars might potentially change lane and occupy a gap convenient for the ego-car. In general, a traffic situation is given by a predicate $\psi(ego, A_1, \ldots, A_n)$ ranging over variables

[†] See `http://imost.informatik.uni-oldenburg.de` for details.
[‡] See `http://www.dlr.de/fs/en`.

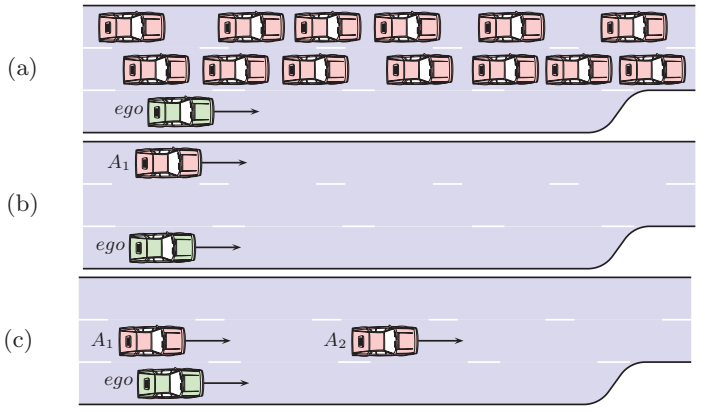


Fig. 2. (Colour online) Traffic situations an ADAS may encounter.

of type car, where *ego* refers to the car initiating a manoeuvre. For each car C, we can observe its position $C.pos$, its lane $C.l$, its speed $C.v$ and its acceleration $C.a$. Formally, a traffic situation is an expression in a first-order logic (see Section 3.4.2) defined over a signature induced by the data types underlying the formal model of traffic environments described in Section 3. For the case of HWYASSIST, the formal model subsumes an *a priori* unbounded extension of the highway to the left and right of the ramp, and the ramp itself, discretised into a grid consisting of grid fields of fixed size at infinitely many positions and finitely many lanes. We assume that the size of the positions is chosen such that cars completely fit into a grid field. Traffic situations can thus express conditions on the highway beyond what can be observed by the ego-car itself using on-board sensors alone. This expressivity can be seen as an abstraction of the situational awareness that can be created by Car2X communication, where cars can extend their purely local view of the traffic situation by using information provided either by the roadside infrastructure or other cars to construct a sufficiently precise internal model of the relevant traffic environment. Distributed algorithms achieving such sufficiently precise and up-to-date local views are beyond the scope of this paper – see, for example, Baskar (2009) for more details.

The predicates $ego.ramp$, $ego.safe$ and $ego.highway$ used in (R1) are particular instances of traffic situations. For example, $ego.safe$ would demand that a speed-dependent number of grid fields ahead of and behind the ego-car on the same lane as the ego-car are free. The two other predicates are specific to the application supported by HWYASSIST. In general, we assume some set M of manoeuvres, and assume that the completion of a manoeuvre m is expressible by a traffic predicate $ego.succ_m$. Drivers can initiate a particular manoeuvre at any time, causing the predicate $ego.start_m$ to be true in this time step. The manoeuvre will then remain active ($ego.active_m$) until completed or aborted.

We will now restate our top-level requirement for an arbitrary manoeuvre m, taking the above discussion into account.

$$\begin{aligned}&\Box\forall\, ego, A_1, \ldots, A_n : Car \bullet \psi(ego, A_1, \ldots, A_n) \wedge ego.safe \wedge ego.start_m \\ &\quad\Longrightarrow\ ((ego.active_m \wedge ego.safe)\ \mathbf{U}_{\leqslant t}\ ego.succ_m)\end{aligned} \tag{R2}$$

This requirement is still not implementable. For example, the traffic situation characterised by ψ might be as shown in Figure 2(b), where the right-hand lane of the highway is empty around the ramp area, and there is only a single car C, which is on the adjacent lane, albeit in a section parallel to the ramp. Drivers obedient to traffic rules would stay in the lane, while an aggressive driver in C might deliberately choose to prevent the ego-car from entering the highway. In our formal model, such situations would not be excluded *a priori*: at any point in time, the driver of a car can initiate a manoeuvre, such as in this particular case by initiating a lane change. While driver assistance systems must be designed to cater for such worst-case environment behaviour, the need for customer acceptance means they cannot possibly provide user guidance based on the worst-case dynamics of the ego-car's environment. Indeed, the driver assistance system should on the planning level mimic human drivers, which would base planning on expected traffic evolution around the ego-car, rather then worst-case behaviour. We will thus distinguish between *nominal* traffic evolution and *abnormal* traffic evolution, under which heading we include drivers violating traffic rules, failures of the car electronics, obstacles suddenly appearing on the highway, and so on. In particular, nominal driving requires that cars adjacent to the lane a car is moving to do not change lanes. Hence, the combination of nominal traffic evolution and the validity of $C.safe$ jointly entail the maintainability of a safety envelope around each car, which will only be intruded upon under abnormal conditions. In our formal development, we will assume a predicate $\psi_{abnormal}$, and will only require successful completion of a manoeuvre if no abnormal conditions occur during the complete time window up to the expected completion time:

$$\begin{aligned} &\Box \forall\, ego, A_1, \ldots, A_n : Car \bullet \\ &\quad \psi(ego, A_1, \ldots, A_n) \wedge ego.safe \wedge ego.start_m \wedge (\Box_{\leqslant t} \neg \psi_{abnormal}) \\ &\qquad \Longrightarrow ((ego.active_m \wedge ego.safe)\, \mathbf{U}_{\leqslant t}\, ego.succ_m) \end{aligned} \tag{R3}$$

While (R3) now captures precisely the conditions under which a successful completion of the manoeuvre can be required, the bounded look-ahead for t time units excluding abnormal behaviour cannot be implemented. Indeed, what sensors could be used to predict, as in the example scenario above, that a driver will become aggressive at some future point in time?

There are three approaches to addressing this problem.

The first, discussed in the paper Damm and Finkbeiner (2011), acknowledges up front that finding strategies for driver assistance systems able to guarantee (R2) is, for the reasons discussed above, impossible, and thus proposes replacing the key concept of *winning strategies* by the notion of *remorse-free strategies*. Basically, a strategy is said to be remorse-free if in situations where the strategy failed to complete the manoeuvre, no other strategy could have done better. The paper then provides a formal framework for ordering strategies according to remorse-freedom, and discusses a synthesis of optimal strategies with respect to this ordering.

A second alternative is to quantify the particular causes for abnormal traffic behaviour using empirical data, and we can then require time-bounded probabilistic reachability. In

this paper, we will demonstrate the viability of this approach, but restrict ourselves to one particular cause: message loss. We will come back to this after discussing the third variant.

The third approach revisits the 'aggressive driver' scenario and illustrates the strength of cooperative driver assistance systems. In cooperative driver assistance systems, the ego-car builds a coalition with a few traffic-situation-dependent cars in its environment. In situations (b) and (c) in Figure 2, such coalitioners are A_1 and $\{A_1, A_2\}$, respectively. By accepting membership of a coalition, a car enters a contract: whenever there are multiple possibilities for next moves, the car will choose one that benefits the objective of the ego-car in completing its manoeuvre. In this way, the contract eliminates the risk that, for example, the car A_1 in (b) chooses to block the right-hand lane, thereby helping the ego-car to enter the highway[†]. Other examples of helpful choices are accelerating or braking to form a gap of sufficient size, or changing lanes. In general, the choice of coalition partners will depend on the traffic situation: with increasing traffic density, the number of coalition partners required to enforce a timely successful completion of the manoeuvre will grow. In our setting, we assume that experts propose, for a given traffic situation ψ and manoeuvre m, a coalition $c_\psi \subseteq \{ego, A_1, \ldots, A_n\}$ of cars in the neighbourhood of the ego-car to achieve between them a successful completion of m.

The formalisation of the latter two proposals motivates the transition to timed probabilistic alternating logic to formulate our requirement. Recall that in alternating logic the eventuality operator '$\diamond\psi$' is prefixed by a list of logical variables denoting the systems forming a coalition aiming to establish the formula ψ.

Taking into account the loss of messages required to establish such a coalition, we will request that the manoeuvre can be completed within time-window t with some probability p. This motivates the following requirement formalisation:

$$\begin{aligned} &\Box \forall\, ego, A_1, \ldots, A_n : Car \bullet \\ &\quad \psi(ego, A_1, \ldots, A_n) \wedge ego.safe \wedge ego.start_m \wedge (\Box_{\leqslant t} \neg \psi_{abnormal}) \\ &\qquad \Longrightarrow \langle\langle c_\psi \rangle\rangle_{\geqslant p} (ego.safe \,\mathbf{U}_{\leqslant t}\, ego.safe \wedge ego.succ_m)\,. \end{aligned} \tag{R4}$$

We will refer to (*R*4) as the requirement specification for manoeuvre m in traffic situation ψ, denoted by $R(m, \psi)$. In general, the full specification $R(m)$ for manoeuvre m will be given as a conjunction of such formulae $R(m, \psi)$ for different traffic situations ψ.

Note that the services offered by cars for conducting manoeuvres are *not* fully controlled by the coalition because we assume that a reflex layer protects a safety envelope around each vehicle. Thus, there are no winning strategies that simply enforce the violation of the premise $\Box_{\leqslant t} \neg \psi_{abnormal}$.

[†] On a more detailed level, there is a need to differentiate between two types of driver interventions to be provided by the driver interface. In *emergency interventions*, the driver is required to take over control, thus overriding the contract in which 'his' driver assistance system is engaged. In all other interventions, the contractual requirements from entering the coalition are maintained.

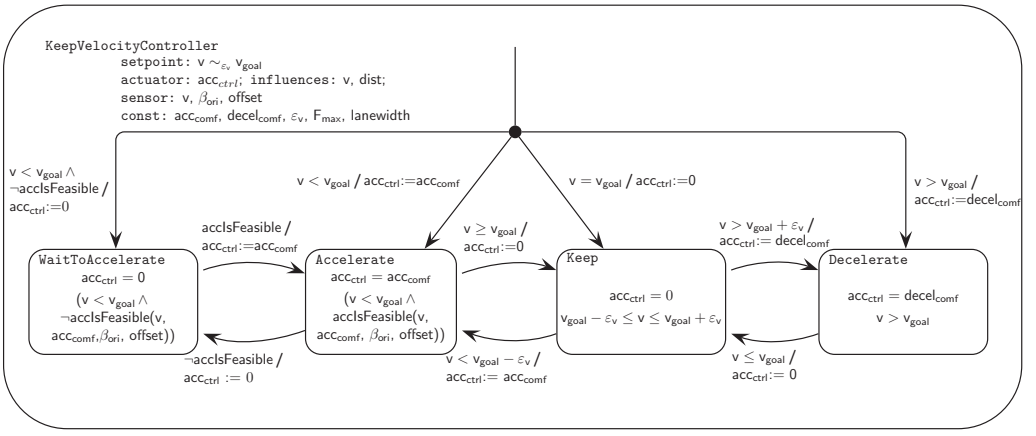


Fig. 3. Hybrid automaton of a keep velocity controller – see Wirtz *et al.* (2011).

2.2. *Layered design of driver assistance systems*

We are interested in cooperative driver assistance strategies where cars are dynamically engaging in coalitions to achieve a particular driving objective within a given time frame. Coalitions are formed according to the driving objective and traffic situations, and then resolved either on meeting the driving objective or on aborting the manoeuvre (due to unforeseen events, see below). Strategic roadmaps for the transportation sector (see Lee (2008)) view such intelligent cooperating vehicles as a major enabling technology for addressing today's transportation challenges, such as traffic congestion, emissions reduction and increased passenger safety. The previous section motivated the formalisation of requirements for cooperative driver assistance system, and introduced the highway-entry assistance system as our running example. This section proposes a layered design structure supporting platform-based design, component re-use and layered verification. In particular, we decompose the overall task of establishing $R(m)$ into computationally tractable synthesis and verification tasks.

2.2.1. *The autonomous layer.* Driving capabilities such as lane change, automatic cruise control, headway control, lane control or emergency breaking are realised as *services* within what we call the autonomous layer in a layered architecture of the car's electronic system. The term autonomous reflects the fact that such capabilities are executed without driver intervention.

The implementation of services will typically be based on hybrid automata (see, for example, Henzinger *et al.* (1996))[†]. Figure 3 shows a controller capable of adjusting the car's acceleration in order to maintain the set point of the car's velocity close to v_{goal}. The controller monitors changes of its real-valued sensor variables tracking the car's velocity and placement within the lane, and reacts by adapting the current control mode. For example, in 'Keep' mode, there is no acceleration or deceleration unless the velocity leaves

[†] See the references for papers on the efficient verification of hybrid controllers implementing such services.

its tolerance region (that is, the mode's invariant $v_{goal} - \epsilon_v \leqslant v \leqslant v_{goal} + \epsilon_v$ has been violated), in which case, depending on the velocity v, the controller will enter either the 'Accelerate' or 'Decelerate' mode.

In order to support platform-based design processes, which aim to maximise re-use of automotive functions across multiple platforms, we abstract from the concrete implementations of hybrid automata and describe services of the autonomous layer by what are called *service specifications*, which provide an assume–guarantee framework for temporal reasoning on traffic agent motion. Industrial design processes typically use Matlab-Simulink/Stateflow from Mathworks for developing controller descriptions that can be used for the tool-based generation of hybrid automata representations (see, for example, Silva *et al.* (2000)). For further details, consult the rich literature on hybrid system verification for methods and tools allowing the verification of hybrid automata against service specifications, which can be used as proof of the controller's compliance for use as a suitable implementation of the specification.

Formally, we assume that the autonomous layer offers a finite set

$$Srv = Srv_{tbnd} \cup Srv_{cont}$$

of time-bounded and continuous *services*. A time-bounded service $f \in Srv_{tbnd}$ changes the car's dynamics and position to achieve a certain post-condition within a fixed time frame, such as achieving a target acceleration or changing the lane to the adjacent right-hand lane. We assume that each service $f \in Srv_{tbnd}$ is formally described by a *service specification*

$$spec(f) = (pre(f), ass(f), inv(f), post(f), t_{\min}(f), t_{\max}(f))$$

where $t_{\min}(f) \leqslant t_{\max}(f)$ are minimal and maximal completion times, and the activation condition $pre(f)$ and the post-condition $post(f)$ are expressions describing traffic situations (see Section 3.4.2), which, in particular, refer to the position and dynamics of the ego-car. The activation condition $pre(f)$ characterises traffic conditions under which the service f can be started, the post-condition $post(f)$ characterises the effect of the service on the car's dynamics and position.

The (safety) assumption $ass(f)$ and the (guaranteed) invariant $inv(f)$ give, for each time offset $0 \leqslant d \leqslant t_{\max}(f)$, expressions $ass(f, d)$ and $inv(f, d)$, which may refer to the spatial position the car assumed when the service was started. Successful completion of service f is guaranteed if for each point in time d between the initiation and completion of the service, $ass(f, d)$ is satisfied by the traffic situation on the highway. For instance, the destination lane must be free in order to successfully complete the change lane service. If there is a violation of $ass(f, d)$, the service is aborted, but it is still guaranteed that the ego-car satisfies $inv(f)$. The invariant $inv(f, d)$ gives guarantees for the car's dynamics and physical position if service f is currently active for d time units.

Continuous services $g \in Srv_{cont}$ are activated at a set-point, such as the desired speed, requiring the controller implementing the service to stabilise the vehicle dynamics at the given set-point, and remain active until disabled. Each continuous service $g \in Srv_{cont}$ is characterised by a pre-condition $pre(g)$ and a post-condition $post(g)$ only. This inductively defines the progress achieved after t time units relative to the pre-condition.

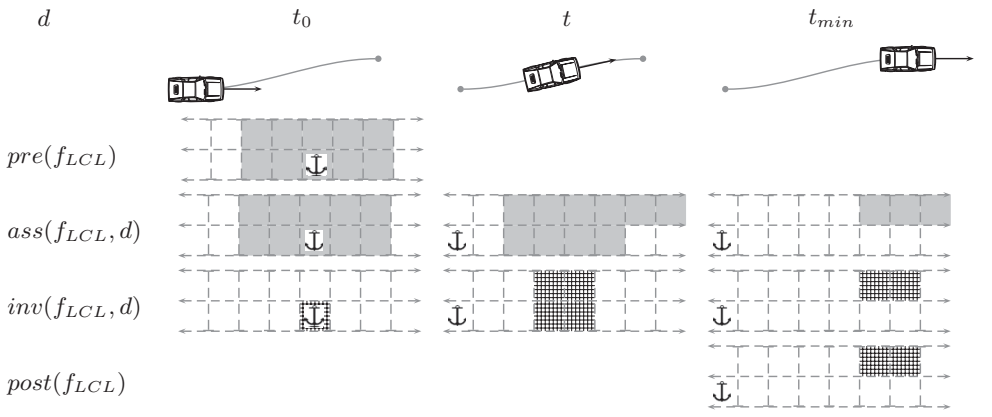


Fig. 4. Intuitive service specification for the 'lane change left' time-bounded service.

Additionally, each service is required to maintain the invariant *ego.safe*, where ego refers to the car in which the service is incorporated.

Figure 4 gives a service specification of a time-bounded service f_{LCL} for a lane change to the adjacent left-hand lane in the form of a visual representation, which we call *tiling* in the following. The precondition $pre(f_{LCL})$ requires that there is a safe gap in the current and destination lanes. We use ☐ tiles to depict the required absence of cars other than the ego-car; white fields may be occupied or free. The safety assumptions $ass(f, d)$ change over time as the car moves away from the *anchor*, that is, the point in space where the service was started (referred to by '⚓'). Close to completion of the service, the assumption requires that the destination lane is sufficiently free. The guaranteed invariant $inv(f_{LCL}, d)$ changes in a similar way to the assumption. We use ▦ tiles to indicate the positions possibly occupied by the car, that is, white fields ensure that these fields are not occupied by the car. In the example, the *tilings* indicate that at the beginning of the service the car only occupies the source lane, then both lanes, and, finally, towards completion of the service, only the destination lane. In the example, the post-condition $post(f_{LCL})$ coincides with the last invariant. Note that, in particular, $inv(f_{LCL}, d)$ need not change over time and, for example, could state that the car may constantly occupy the two lanes involved and that longitudinal acceleration does not change throughout the service.

In the following, we assume that any invariant $inv(f, d)$ allows us to deduce the car's position relative to the anchor field at position pos_{srv} and lane l_{srv}, where the car's position need not be exact in the sense that the car may occupy a subset of the fields indicated by $inv(f, d)$.

While time-bounded services are assumed to terminate within a certain time frame, continuous services are not required to terminate and are characterised by a simpler service specification $(pre(g), post(g))$, where the safety assumption (the pre-condition) and guaranteed invariant (the post-condition) are independent of time offsets expressed relative to the ego-car. An automatic cruise control system, which keeps a desired speed if possible, could be characterised as a continuous service – see Figure 5. The pre-condition states

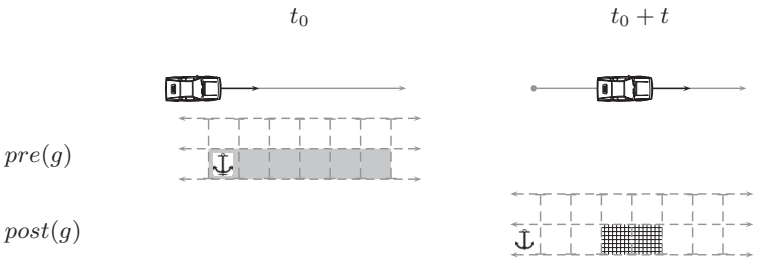


Fig. 5. Intuitive service specification for the 'keep velocity' continuous service.

that safety distances to the cars in front are maintained; the post-condition indicates that the ego-car is located in the region of ' ▦ ' tiles.

Note the difference between a change-lane service and the lane change manoeuvre supported by a driver assistance system. The latter will determine a coalition and a strategy building on a sequence of services executed by cars in the coalition to complete the manoeuvre. The former, in contrast, 'blindly' follows the service specification in achieving its invariant and post-condition, and reacts to violations of the assumptions.

Figure 10[†] illustrates a generic skeleton of an autonomous layer, which is formulated in terms of service specifications to abstract from the actual hybrid automata services.

While a finite set of time-bounded and continuous services, such as a lane change service $f_{LCL} \in Srv_{tbnd}$, are provided in the hierarchical ADAS state, the state 'Manual' represents driver behaviour. Intuitively, this state allows low-level access to the car's actuators resulting in a much less restricted behaviour compared with the services of the ADAS state.

In the ADAS state, a service is invoked by a dedicated $start_f$ message. Given that the activation condition $pre(f)$ of the service holds, the guaranteed invariants ($inv(f,d)$) are applied and the assumptions ($ass(f,d)$) are checked iteratively until either the service is completed in a timely fashion or an assumption has been violated, for example, due to an unforeseen appearance of a car in the region assumed to be free by $ass(f,d)$. In such cases, the control is returned to the cooperation layer ($coop$) described below, which may call another suitable service in reaction. The identifier $coop$ can be considered as a link-typed local variable of the automaton that stores the unique *process identity* of the car's cooperation layer process. It is here used to address the cooperation layer as the receiver in send edges.

The physical environment is considered as a process (hw – see Section 3.2) that keeps track of the traffic situation and, in particular, provides an interface for checking (chk) the pre-conditions and assumptions of a service, and applying (app) the effects given by the post-conditions and invariants.

[†] Note that skeletons of the three layers we consider are presented in Figures 6 to 10 from top to bottom. The highest layer, cooperation (see Section 2.2.3), interacts with the lower autonomous and communication layers (see Section 2.2.2).

2.2.2. *The communication layer.* In traditional driver assistance systems, capabilities at the autonomous layer are activated and de-activated by the driver. Cooperative driver assistance strategies rely on their perception of traffic situations built from on-car sensors and inter-vehicle and vehicle2infrastructure communication to pursue a shared strategy in activating and deactivating services of the autonomous layer to achieve successful completion of the supported manoeuvre in the prescribed time interval. Car2X communication serves multiple purposes in such applications Baldessari *et al.* 2007:

— They enhance a car's 'perception capabilities', in that locally available information about the current traffic condition (acquired through, for example, radar, video-cameras and lidar) is enhanced through the use of other cars' perceptions of their view on traffic conditions (allowing, for example, an extension to the horizon around curves, or to identify cars that are temporarily hidden by a truck). Such mental maps then serve as a basis for identifying possible coalition partners to meet driver-specified objectives (such as entering a high-way).
— They allow the negotiations required in the formation of a coalition.
— They allow synchronisation during the cooperative execution of agreed strategies.
— They allow information to be gained about abnormal events encountered during manoeuvre execution.

Using the principle of platform-based design, the communication services are shared across multiple applications, and hence encapsulated in a separate unit, which we call the communication layer. In particular, this supports timed probabilistic transmission guarantees for point-to-point communication using the communication standard provided by the Car2Car Consortium. For the purposes of this paper, the actual communication protocols employed are of no relevance – see Eichler (2009) for an excellent mathematical analysis of the Car2Car communication protocols leading to such service specifications.

Figure 9 shows a simple communication layer skeleton. Analogous to the abstraction of the hybrid behaviour to service specifications in the autonomous layer, this skeleton can be viewed as the result of an abstraction from a timed probabilistic automaton derived from the Car2Car standard. The skeleton is supposed to be read as the (syntactical) cross-product of the two given automata. Intuitively, the communication skeleton is, through the reception of the message *req*, asked to transmit the message req_{ext} to a particular process with the identity attached to the message by the link-typed parameter c_{rcv}. The receiver is informed about the cooperation layer (*coop*) that issued the request and a time budget (t_m) for manoeuvre execution. Note that the external message req_{ext} is only used between different communication layers. The transmission succeeds or fails with a certain probability defined by the weights w_s and w_f, respectively. In the latter case, the communication layer makes a transition to a state where no synchronisation with the other communication layer process takes place. However, when there is a successful transmission, the received message and its parameters are forwarded by the addressed communication layer to the cooperation layer assigned to it (that is, the one that is stored in its link-typed variable *coop*).

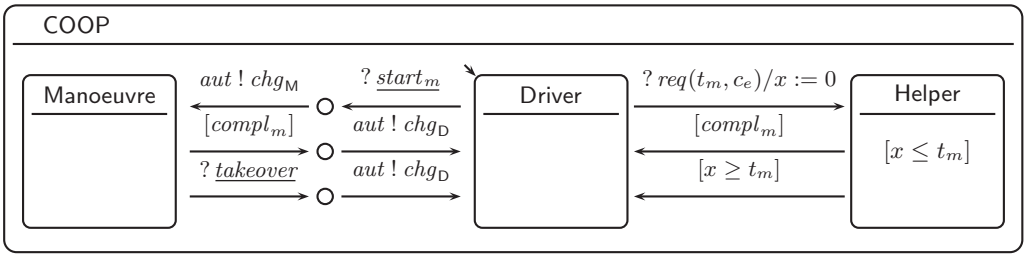


Fig. 6. Cooperation layer skeleton.

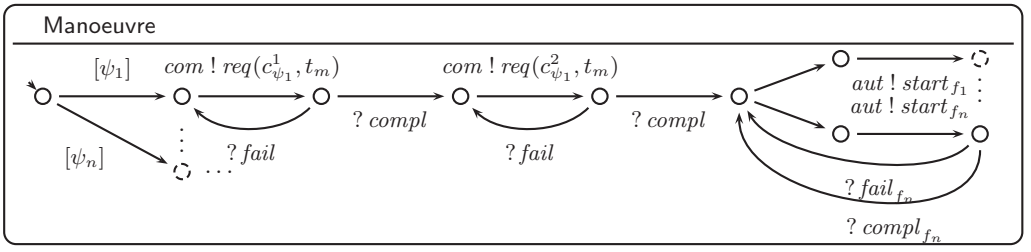


Fig. 7. Cooperation layer skeleton – Manoeuvre.

2.2.3. *The cooperation layer.* The services of the autonomous and communication layers provide a re-usable platform on which an ADAS supporting a particular manoeuvre can be built. In this section, we provide a generic skeleton of a cooperation layer, which employs the standardised interfaces of the autonomous and communication layer, and which is customised to a particular ADAS (such as HWYASSIST) by giving a *coordination strategy*. Intuitively, this strategy determines for each change in traffic situations and, for each car in the coalition, the activation time and duration of time-bounded services, the activation time and set-point for continuous services, and the calls for communication services, such as for building a coalition.

Figure 6 gives a high-level view of the top modes of the cooperation layer. Initially, the coordination layer is in 'Driver' mode, where the activation and deactivation of the autonomous layer services is under complete driver control. In particular, the driver may request the support of the ADAS to perform a manoeuvre m, such as entering the highway, by calling $start_m$. Upon completion of the manoeuvre ($[compl_m]$) or on an intervention by the driver taking over control (*takeover*), the driver mode is reactivated. Alternatively, the ADAS can be engaged in a coalition to support some ego-car in performing a manoeuvre, thereby taking the 'Helper' role. Note that the above architecture implies that a car cannot be simultaneously both in the role of an ego-car and a partner in a coalition formed by another ego-car.

For the 'Manoeuvre' mode, Figure 7 gives a generic skeleton of its internal structure, which consists, essentially, of a *non-deterministic choice* invoking the services of the autonomous and communication layers. In particular, whenever a service returns control to the cooperation layer by sending a dedicated *fail* or *compl* message, the automaton instantaneously calls a new service. Note that we have cut off parts of the diagram with dashed outlines in order to concentrate on the general structure. For the same reason,

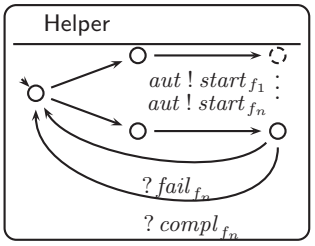


Fig. 8. Cooperation layer skeleton – Helper.

we have also omitted details such as an upper bound on the number of retries for inter-vehicle communication using the communication layer. The internal structure of the 'Helper' mode, shown in Figure 8, offers a non-deterministic choice of the services provided by the autonomous layer. Typically, in comparison with the choices available in the 'Manoeuvre' mode, the choices here are restricted to results giving more comfortable driving, for example, ruling out extreme acceleration.

It is the task of the coordination strategy to resolve the non-determinism inherent in the ADAS controlled modes 'Manoeuvre' and 'Helper'. Such strategies can be custom-designed, or synthesised automatically, using a sufficiently precise representation of traffic situations. This paper proposes the use of automatic synthesis tools for determining coordination strategies. From an industrial perspective, such automatically synthesised strategies provide evidence of the realisability of a new ADAS system for supporting a set of requirement specifications $R(m_1)$ to $R(m_n)$, and are thus used early in the concept validation phase. If it is impossible to synthesise such a coordination strategy, then it may be that the proposed set of coalitioners (or helpers) is too weak, or the platform is not sufficient to support this new class of manoeuvres. Later design phases could systematically derive a Stateflow representation from automatically synthesised strategies, leading to input-deterministic automata for both the 'Helper' and 'Manoeuvre' modes.

3. A formal model for cooperative driver assistance systems

As discussed in the introduction and previous section, the creation of a faithful and comprehensive model of cooperative cars, including their continuous dynamics and the traffic environment needs to take into account:

— The vehicle dynamics, as controlled by the autonomous layer, need to be modelled by hybrid automata. The autonomous layer switches between discrete modes and within modes, there is continuous movement on the highway space in dense time.
— The wireless car-to-car communication infrastructure may suffer probabilistic loss of packages.
— The cars, consisting of cooperation, autonomous and communication layers, can enter and leave the highway freely without a known fixed upper bound on the number of cars simultaneously in the highway system. There are also dynamic relations between cooperation layers. The cooperation layer in a car conducting a manoeuvre keeps track of the cars taking the helper role and *vice versa*.

— The design problem discussed in this paper reduces to a game-theoretic problem, namely, whether a coalition of cars is able to enforce success of a manoeuvre against a set of opponents. So a complete model needs to distinguish who controls the existing non-determinism, where, in the case of traffic agents, the coalition consists of a set of cars.

Supporting the development of cooperative driver assistance systems in such a rich and full model is beyond current technologies, so we propose to approach the problem by decomposing it into tractable subclasses.

In Section 2.2, we argued that we can assume local descriptions of the behaviour of the different layers executed in each car. The hybrid behaviour of the autonomous layer is reduced to a collection of service specifications with discrete variables and a discrete notion of time (see Section 2.2.1).

The probabilistic behaviour of the car-to-car communication infrastructure is assumed to be reduced to the essential information of the probability that the communication required to acquire a helper succeeds in the absence of race conditions. This information can be obtained by probabilistic analysis of the protocol stack in use, and does not depend on the autonomous or cooperation layers (see Section 2.2.2)

In order to model the remaining aspects, Section 3.1 provides a variant of Dynamic Concurrent Probabilistic Game Structures (DCPGS) as introduced in Ehlers *et al.* (2011). A DCPGS, as descibed in Section 3.1, is an untimed model of unbounded creation of agents where each agent has:

— a unique identity;
— a behaviour determined by a finite state description with non-determinism and probabilistic branching;
— a finite set of local variables (possibly of infinite range) plus link-valued variables that carry the identities of other agents;
— a finite set of moves.

Also, at each point in time, only finitely many agents are alive. Note that the definition of a DCPGS in Ehlers *et al.* (2011) considers (dense) time and turn-based games, but our Definition 3.1 is concurrent.

In Section 3.2, we describe how to obtain a DCPGS from the local descriptions of the behaviour of the three layers presented in Section 2. Note that we do not formalise each of the local behaviour descriptions as a DCPGS, but sketch how an overall model of traffic environments with an *a priori* unbounded number of cars in the form of a game structure can be obtained from the local behaviour descriptions. To this end, we assume zero-time rendezvous communication between the agents. One transition of the game structure will correspond to a finite number of zero-time agent communications.

In Section 2.1, we discussed the formalisation of requirements on driver assistance systems using temporal logic with an intuitive understanding. In Section 3.3, we will introduce a probabilistic variant of (discrete) Timed Alternating-Time Temporal Logic (TATL*) (Henzinger and Prabhu 2006) for systems with a dynamically growing number

of agents. The formulae of this logic are interpreted over the DCPGS introduced in Section 3.1.

In Section 3.4, we observe that requirement (R4) from Section 2.1 is a formula in our logic over the formal model of a driver assistance system from Section 3.2.

3.1. *Dynamic concurrent probabilistic game structures.*

Definition 3.1 (dynamic concurrent probabilistic game structure). Let $\mathbb{I}$ be a countable set of *agent identities*. A *dynamic concurrent probabilistic timed game structure* (DCPGS) over $\mathbb{I}$ is a quintuple

$$\mathscr{S} = (S, s_{init}, A, \Gamma, \delta)$$

where:

— S is a set of *game states* on which $\mathrm{supp} : S \to 2^{\mathbb{I}}_{fin}$ is a function that indicates the (finite) set of agents *alive* in a given game state;
— $s_{init} \in S$ is the initial state;
— A is a set of *game actions*;
— $M = \mathbb{I} \times A$ is the set of *agent moves*;
— $\Gamma : S \to 2^M$ is a function that assigns to a given game state the set of available moves; and
— $\delta : S \times 2^M \to (2^{\mu(S)} \setminus \emptyset)$ is the *transition function* that assigns to a game state and a set of moves a set of probability distributions on the game states, where $\mu(Q)$ denotes the set of all finite discrete *probability distributions* over a set Q, that is, for each $P \in \mu(Q)$, $\sum_{q \in Q} P(q) = 1$ and $\{q \in Q \mid P(q) > 0\}$ is finite;

such that the following well-formedness constraints hold:

(1) In each game state $s \in S$, at least one action is available to each alive agent, that is, for each agent $u \in \mathrm{supp}(s)$, there is an action $a \in A$ such that $(u, a) \in \Gamma(s)$.
(2) The transition function ensures *monotonic growth* of the set of agents alive in states, that is, for states $s, s' \in S$ and a set of moves $\kappa \subseteq M$, for each $P \in \delta(s, \kappa)$, $P(s') > 0$ implies $\mathrm{supp}(s') \supseteq \mathrm{supp}(s)$ ('monotone frame semantics').

The course of the game played on a given dynamic concurrent probabilistic game structure proceeds as follows. The game begins in the initial state s_{init}. When the game is in a state $s \in S$, all live agents simultaneously and independently propose their moves. To do this, each agent $u \in \mathrm{supp}(s)$ selects an available move (u, a) from $\Gamma(s)$ choosing his action $a \in A$. Let κ denote the set of moves selected by the live agents. The set κ determines a set of probability distributions by $\delta(s, \kappa)$, such that each $P \in \delta(s, \kappa)$ determines the probability of reaching a certain successor state.

A finite (or infinite) *path* of the DCPGS $\mathscr{S} = (S, s_{init}, A, \Gamma, \delta)$ starting in game state $s_0 \in S$ is a finite (or infinite) sequence

$$\omega = s_0 \xrightarrow{\kappa_1} s_1 \xrightarrow{\kappa_2} s_2 \xrightarrow{\kappa_3} \dots$$

with $s_i \in S$ and $\kappa_i \subseteq \Gamma(s_{i-1})$ for $i \in \mathbb{N}^+$. We write:

— $\omega(k)$ to denote s_k, the kth game state of ω;

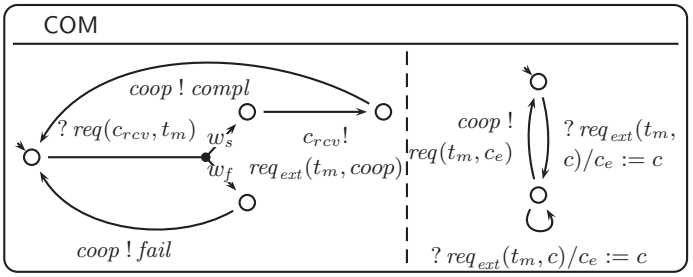


Fig. 9. Communication layer skeleton.

— $\omega_{0..k}$ to denote the prefix of ω up to and including the kth game state of ω;
— ω/k to denote the suffix from and including the kth state of ω;
— last(ω) to denote the last state of a finite path ω;
— $\mathsf{FinPth}(s)$ ($\mathsf{InfPth}(s)$) to denote the set of finite (infinite) paths starting in $s \in S$.

We say a set of moves $\kappa \subseteq M$ is *complete* with respect to a set $I \subseteq \mathbb{I}$ of identities if and only if κ has exactly one move per agent, that is, for each $u \in I$,

$$|\kappa \cap (\{u\} \times A)| = 1.$$

A *strategy* starting at game state s is a function

$$\pi : \mathsf{FinPth}(s) \to 2^M$$

such that $\pi(\omega) \subseteq \Gamma(\text{last}(\omega))$. That is, a strategy assigns to any finite path ω starting in s a set of moves that are available in the last state of ω.

We say strategy π starting at $s \in S$ is a strategy for team $T \subseteq \mathbb{I}$ if and only if $\pi(\omega)$ is complete with respect to T for each $\omega \in \mathsf{FinPth}(s)$. A strategy $\bar{\pi}$ is called a counter-strategy against team T if and only if $\bar{\pi}(\omega)$ is complete with respect to the complement of team T, that is, complete with respect to $\text{supp}(\text{last}(\omega)) \setminus T$ for each $\omega \in \mathsf{FinPth}(s)$. Note that the complement team in our dynamic games may grow along finite path ω in accordance with the monotone frame semantics. For the more general case of a dynamically growing team, see Ehlers *et al.* (2011). A *scheduler* starting at game state $s \in S$ is a function

$$\pi_{sc} : \mathsf{FinPth}(s) \times (2^{\mu(S)} \setminus \emptyset) \to \mu(S)$$

that resolves any remaining non-determinism, that is, for each finite path ω and each non-empty set of probability distributions $\mathscr{P} \subseteq \mu(S)$, we have $\pi_{sc}(\omega, \mathscr{P})$ is a probability distribution $P \in \mathscr{P}$.

Let π be a strategy for team T, $\bar{\pi}$ be a counter-strategy against T and π_{sc} be a scheduler, with all three starting in game state $s \in S$. The set $\mathsf{Outcomes}(s, \pi, \bar{\pi}, \pi_{sc})$ consists of all paths starting in s that may arise when the team chooses its moves according to strategy π, the complement of the team chooses its moves according to $\bar{\pi}$ and any remaining non-determinism is resolved by the scheduler π_{sc}.

Formally, $\mathsf{Outcomes}(s, \pi, \bar{\pi}, \pi_{sc})$ is the set of infinite paths

$$\omega = s_0 \xrightarrow{\kappa_1} s_1 \xrightarrow{\kappa_2} s_2 \xrightarrow{\kappa_3} \dots$$

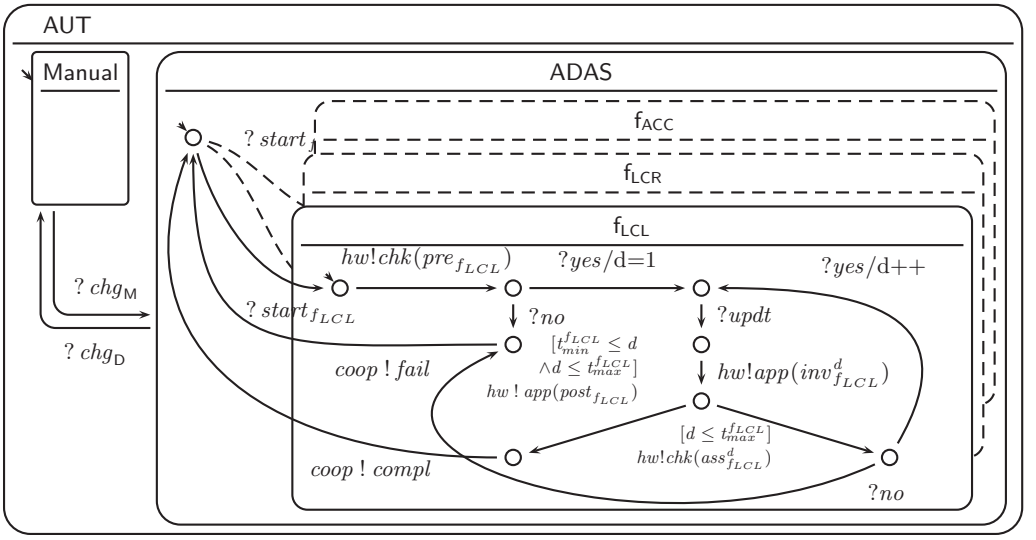


Fig. 10. Autonomous layer skeleton.

such that $s_0 = s$ and for all $i \in \mathbb{N}_0$,

$$\kappa_{i+1} = \pi_{sc}(\omega_{0..i}, \pi(\omega_{0..i}) \cup \bar{\pi}(\omega_{0..i}))$$

and $P(s_{i+1}) > 0$ for

$$P = \pi_{sc}(\omega_{0..i}, \delta(s_i, \kappa_{i+1})).$$

Following Ehlers *et al.* (2011), we define the probability measure P on sets of paths as follows. Let

$$\mathsf{Outcomes}_{fin}(s, \pi, \bar{\pi}, \pi_{sc})$$

be the set of all finite prefixes of paths in

$$\mathsf{Outcomes}(s, \pi, \bar{\pi}, \pi_{sc}).$$

Let $\mathscr{F}_s^{\pi,\bar{\pi},\pi_{sc}}$ be the smallest σ-algebra on $\mathsf{Outcomes}(s, \pi, \bar{\pi}, \pi_{sc})$ that contains the sets

$$\{\omega \in \mathsf{Outcomes}(s, \pi, \bar{\pi}, \pi_{sc}) \mid \omega_{0..|\omega'|} = \omega'\}$$

for all $\omega' \in \mathsf{Outcomes}_{fin}(s, \pi, \bar{\pi}, \pi_{sc})$. Then

$$Prob : \mathsf{Outcomes}_{fin}(s, \pi, \bar{\pi}, \pi_{sc}) \to [0, 1]$$

is defined inductively by

$$Prob(\omega) = \begin{cases} 1 & \text{iff } |\omega| = 1 \\ Prob(\omega') \cdot \pi_{sc}(\omega', \delta(\text{last}(\omega'), \kappa))(s') & \text{iff } \omega = \omega' \xrightarrow{\kappa} s'. \end{cases}$$

The measure P on $\mathscr{F}_s^{\pi,\bar{\pi},\pi_{sc}}$ is now the unique measure such that

$$P(\{\omega \in \mathsf{Outcomes}(s, \pi, \bar{\pi}, \pi_{sc}) \mid \omega_{0..|\omega'|} = \omega'\} = P(\omega').$$

Note that we may choose the set of identities $\mathbb{I}$ to be finite and that we may have a transition function δ that yields only trivial probability distributions, that is, each $P \in \delta(s, \kappa)$ is of the form $\{s' \mapsto 1.0\}$ for some $s' \in S$. In this case, a DCPGS is a classical concurrent game structure with some uncontrollable non-determinism.

3.2. *Formalising the behaviour of the ADAS incorporating vehicle dynamics and the traffic environment.*

Formally, a cooperative driver assistance system is a DCPGS $\mathscr{S}$ over a countably infinite set of identities

$$\mathbb{I} = \mathbb{I}_{COOP} \dot{\cup} \mathbb{I}_{AUT} \dot{\cup} \mathbb{I}_{COM} \dot{\cup} \mathbb{I}_{HW} \dot{\cup} \mathbb{I}_{SP}$$

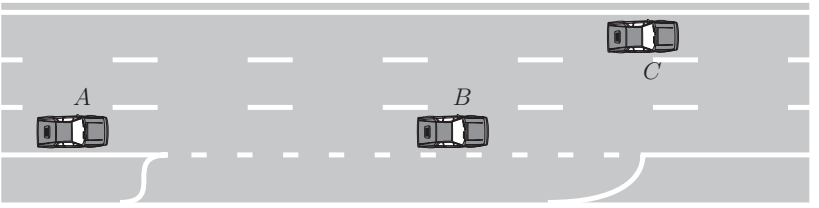
that is equally partitioned into identities for five agent types, $COOP$, AUT, COM, HW and SP. An agent identity from $\mathbb{I}_{COOP}$, for instance, models an instance of the $COOP$ automaton from Section 2. We assume exactly one instance of HW, a highway agent, which together with AUT realises a minimal plant model. The particular role of HW is to provide a unified view of the physical highway situation to all AUT instances.

For technical reasons, we assume exactly one instance of SP, a spawn agent, which creates new cars consisting of interlinked instances of triples of $COOP$, AUT and COM. Furthermore, SP provides uncontrollable non-deterministic effects such as interactions by the driver. Even if a car is part of a coalition, the driver is not, so this cannot be modelled simply by non-determinism in the cooperation layer. In the model, it must still be possible for the driver to take over at any time and thereby inhibit the success of a manoeuvre.

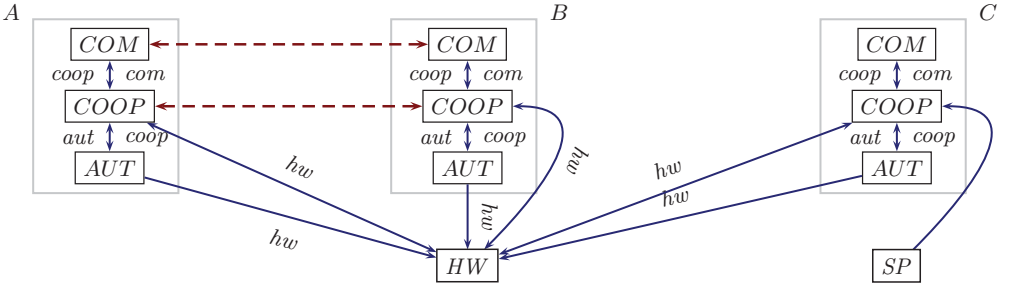
3.2.1. *Game states of the ADAS model.* We assume that the set of game states S of $\mathscr{S}$ consists of structures that provide valuations for the following local variables of the live agents:

(1) $COOP$:
   (a) the current location in the local behaviour description (see Figure 6);
   (b) link-valued variables of $COOP$-type that keep track of helpers in its own coalition or of another car that is currently being helped (see Figure 11);
   (c) link-valued variables *aut* and *com* of type AUT and COM that provide the connection to the underlying layers in the car – they are established when the car is constructed and remain unchanged over the car's lifetime.

(2) AUT:
   (a) the current location in the local behaviour description;
   (b) the currently active service $srv \in Srv$ (see Section 2.2.1);
   (c) the anchor position of the currently active service, consisting of integer position pos_{srv} and lane l_{srv}, ranging over $0, \ldots, L$;
   (d) the time d_{srv}, that is, the number of steps for which the current service is active, ranging over $0, \ldots, t_{\max}(srv)$;
   (e) the autonomous layer may also keep track of (discretised) physical parameters such as the lateral and longitudinal velocity , $v^{\leftrightarrow}$ and $v^{\updownarrow}$, and acceleration, $a^{\leftrightarrow}$ and $a^{\updownarrow}$, ranging over finite lists of intervals



(a) Situation of a highway with three cars which is modelled by the game state shown in Figure 11(b) below.



(b) Visualisation of a game state corresponding to the highway situation shown in Figure 11(a) above. Each solid rectangle represents an agent, it is labelled with the local behaviour description it instantiates. Solid edges indicate fixed links, dashed edges indicate dynamic links. Each car A, B, C is modelled by three interlinked agents, one for each layer, in the game structure. In addition, there is exactly one instance of HW and SP.

Fig. 11. (Colour online) Architecture of the formal model of multi-agent traffic systems.

(in order to simplify the update procedure, these variables are assumed to be part of the AUT instances, rather than in HW, for instance);

(f) a link-valued variable *coop* of type $COOP$ that provides the connection back to the overlying cooperation layer – this is established when the car is constructed and remains unchanged over the car's lifetime;

(g) one link-valued variable *hw* of type HW that provides the connection to the shared resource.

(3) COM :

(a) the current location in the local behaviour description;

(b) possibly, link-valued variables of type COM that are employed to realise communication between COM layers of different cars – these will change over the lifetime of a car;

(c) one link-valued variable *coop* of type $COOP$ that provides the connection back to the overlying cooperation layer – this is established when the car is constructed and remains unchanged over the car's lifetime.

(4) HW :

(a) possibly, a current location in a local behaviour description;

(b) conceptually a mapping from position/lane pairs (pos, l) to sets of $COOP$ identities so that the HW instance provides a unified view on the physical situation of all (finitely many) cars currently on the highway.
We do not make any particular assumptions about how this information is stored. Note that the physical position of each car can be uniquely reconstructed from the triple consisting of the current service, the anchor position and the time spent in the service so far.

(5) SP:

(a) possibly, a current location in a local behaviour description;
(b) a variable *coop* of type set-of-$COOP$ that carries the identities of all currently live $COOP$ agents.

3.2.2. *Game actions and moves of the ADAS model.* The set of game actions A of $\mathscr{S}$ is determined by the capabilities of the five kinds of agents. In the given setting, we will basically only model the actions of the $COOP$ instances as actions. At each game state, a $COOP$ can issue at most one service request each to the car's AUT and COM layers. A request to AUT is a request for the start of a service (which automatically stops the currently running service). A request to COM is a request for a communication, with a COM identity passed over as a parameter. In addition, there is a dedicated no-op(eration) request τ, which asks AUT or COM, or both, to continue with the current service.

For simplicity, we assume that at each point in time, AUT is executing exactly one service. Autonomous layers that offer the execution of multiple services in parallel can be modelled by adding one service for each combination of services executable in parallel.

The agents modelling AUT, COM, HW and SP instances only have τ-actions available, which correspond to the non-deterministic choice in the local behaviour descriptions.

3.2.3. *Available moves function of the ADAS model.* In this section we show how the available moves function Γ and the destination function δ can be constructed from the local behaviour descriptions.

As outlined above, we assume that the local behaviour descriptions employ a rendezvous communication that is executed in zero-time, and that a finite sequence of interleaved zero-time synchronisations make up one transition of the game structure $\mathscr{S}$ (see Section 3.2.4). Regarding the behaviour, we assume that:

(1) HW

(a) can be queried for whether an assumption of an active service given in form of $ass(f)$ is consistent with the invariants of the services active in all cars on the highway, as given by $inv(f)$ per car (see Section 2.2.1)
(in Figure 10, this query is issued by the autonomous layer in the form of synchronisation *chk*, which expects responses *yes* or *no*);
(b) can receive updates of anchor positions;
(c) can be queried for the identities of cars at certain highway positions in order to model the sensor-based detection of other cars.

(2) COM

(a) provides, in addition to requests for a service, a synchronisation for reporting back to its $COOP$ instance whether communication was successful or failed;

(b) can engage in synchronisation with other COM layers, which may ask to join a coalition.

There is a possible source for conflicts, or *race conditions*, here. It may be the case that two cars want to ask the same third car for help in a manoeuvre, and it may even be the case that two cars want to ask each other to join into a coalition. We assume that in this case one communication will be successful, that is, exactly one non-deterministically chosen request will be positively acknowledged, and all others will receive negative acknowledgements.

(3) AUT

(a) provides, in addition to requests for service, a synchronisation for reporting back to its $COOP$ instances that there was a failure due to unsatisfied service assumptions or that a service was successfully completed;

(b) uses the functions of HW.

Note that the hybrid automaton modelling the autonomous layer is assumed to be *(input) deterministic*, that is, given the same service requests and environmental conditions, it ensures the same behaviour. However, viewed from its $COOP$ layer, the autonomous layer does not behave deterministically since it reacts to the behaviour of other cars on the highway, and the behaviour of these other cars may, in particular, be determined by their (highly non-deterministic) drivers.

In the model proposed here, the plant non-determinism (wind, elevation of the road, and so on) is integrated into AUT, so AUT is non-deterministic.

(4) $COOP$

(a) is sensitive to driver interactions issued by SP;

(b) uses the functions of AUT and COM.

(5) SP

(a) non-deterministically creates cars in the form of three interlinked layers at highway entry ramps;

(b) through the *coop* link, non-deterministically interacts with any $COOP$ instance in order to model driver interaction. That is, it is the SP agent that synchronises on *takeover* with instances of $COOP$ – see Figure 6.

Driver interaction is not modelled as part of the cooperation layer because, from the perspective of an assistance system, the driver is one of the opponents: if the driver takes over, the assistance system can no longer control the success of the manoeuvre. Instead of having an additional driver agent in the model, we propose to locate the non-determinism stemming from the driver in SP.

We assume that local behaviour descriptions are *input enabled*, that is, at each point in time, they provide transitions for all synchronisations they can possibly engage in. In particular, these transitions may not have any effect on the local state of the agent, that is, the agent may effectively simply ignore certain synchronisations in certain local states.

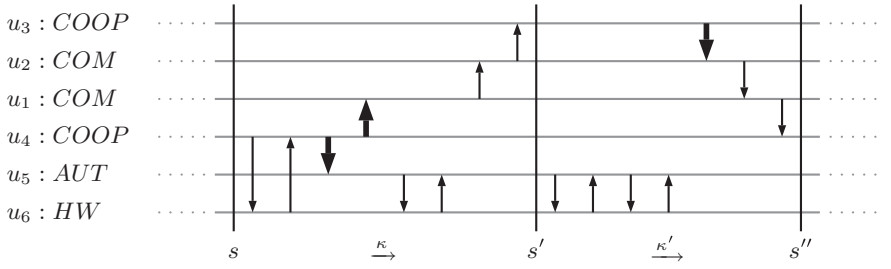


Fig. 12. Game states represent stable situations between finite sequences of zero-time synchronisations and local transitions of the live agents. The diagram can be viewed as a sequence chart rotated by 90 degrees. Each horizontal line represents one agent. The horizontal lines begin and end with dashes as they are supposed to show three consecutive game states in a longer path. Arrows represent zero-time synchronisation between agents. Thick arrows become moves of the agent. The example shows a game state s with live agents $\{u_1, \ldots, u_6\} \subseteq \mathrm{supp}(s)$. Starting from s, the local behaviour descriptions of $COOP$, AUT, COM interact with each other and HW in any possible interleaving. For instance, the $COOP$ instance u_4 may first query HW for the current highway situation, for example, the identity of a nearby car. Then u_4 switches its AUT instance u_5 to a certain service. Afterwards u_4 ask its COM instance u_1 to contact the car that has just been detected. This is carried out through the COM instance u_2 of the destination instance. Before the communication, AUT instance u_5 updates the car position (the AUT instance of u_3 is not shown). Here, the move of u_4 in κ would consist of the two thick arrows corresponding to services, the move of u_3 consists of two no-ops and the other agents have τ actions available. In the example, u_5 continues to execute the service without intervention from u_4, while the interaction between u_3 and u_2 will be part of the move of u_3 in κ'.

In this way, the synchronisation between agents never blocks, and there can only be races (see (2 b) above), which are represented in $\mathscr{S}$ by the *set-valued* transition function δ and resolved by a scheduler π_{sc} – see Section 3.1.

3.2.4. *Destination function of the ADAS model.* Let s be a game state and κ be a set of moves providing one move for each live agent.

The possible resolutions of conflicts between COM layer instances (see (2 b) above) can be enumerated. For each resolution, there will be one probability distribution P in $\delta(s, \kappa)$. Let $s' \in S$. We set $P(s')$ to be the probability of reaching s' by executing a finite number of zero-time synchronisations and local transitions in the live agents until a dedicated stable situation is reached again (see Figure 12). We assume that stable situations are uniquely determined by the language used to describe the behaviour, such as the one used in Figures 6 to 10.

Time invariants and guards as shown in Figure 6 in 'Helper' mode and on the transition back to 'Driver' are understood as counting steps in $\mathscr{S}$, that is, we assume that a counter is introduced that counts steps in $\mathscr{S}$ and, for example, takes the transition back to 'Driver' if the corresponding guard over the counter is satisfied.

Note that a probability less than one can only stem from the behaviour of the *COM* layer and that we assume that in a single transition of $\mathscr{S}$, at most one non-trivial probabilistic transition of *COM* is involved.

The set $\Gamma(s)$ of moves available at game state s is determined by the local states of the agents alive in s and the local behaviour descriptions discussed in Section 2. The relevant communications from *COOP* to *AUT* and *COM* can be extracted and merged into a pair, using the no-op action if no communication is taking place in the transition under consideration.

In the following, we assume that there is a fixed and given correspondence between the unit of time modelled by single transitions in $\mathscr{S}$ and the time the communication employed for building a coalition takes, that is, sending out a request to the prospective helper and receiving a (positive or negative) acknowledgement.

3.3. *Dynamic probabilistic alternating-time temporal logic*

Having introduced dynamic concurrent probabilistic game structures as the semantic domain of traffic agent systems in Section 3.1, in this section we recall a variant of DPTATL (Ehlers *et al.* 2011), the dynamic probabilistic variant of TATL* (but assuming discrete rather than continuous time). The logic we present is, in particular, expressive enough to provide a formal underpinning of the requirement (R4) from Section 2.1. DPTATL is inspired by the well-known logics TATL* (Henzinger and Prabhu 2006), PCTL (Hansson and Jonsson 1994) and METT (Bauer *et al.* 2006), and is enriched with an abstract expression language that allows us to refer to an agent's state. The presentation in this section follows Ehlers *et al.* (2011).

3.3.1. *Expression language.* Let $(v \in)\ \mathscr{V}$ be a set of typed *logical variables* and let $(U \in)\ \mathscr{U} \subseteq \mathscr{V}$ be a finite set of *agent types*. We assume an *expression language* that may, in particular, refer to the local variables of agents (see Section 3.2). We assume at least the following:

— A is an expression of agent-type U if A is a logical variable of agent type U.
— $expr_1 = expr_2$ is a Boolean expression if $expr_1, expr_2$ are expressions of agent type U.
— If x is a local variable with type T in agents of type U_i and $expr_1$ is an expression of type U_i or of type set-of-U_j, then $expr.x$ is an expression of type T.
— If x is a local variable with agent-type U_j in agents of type U_i and $expr$ is an expression of type U_i or of type set-of-U_j, then $expr.x$ is an expression and its type is set-of-U_j-agents.

In addition, there is a set of typed function symbols of arity $n \in \mathbb{N}_0$,

$$f(expr_1, \ldots, expr_n),$$

including constants *true* and *false*, constants corresponding to the domain of the finite local variables types, basic arithmetics and predicates such as '=', '$\leqslant$' and '$<$' on the arithmetic types.

We assume a function $\mathscr{D}$ that gives for each type T a semantical domain $\mathscr{D}(T)$. An expression over agent types $U_1, \ldots, U_n$, with $n \in \mathbb{N}$, can be evaluated for DCPGSs over $\mathbb{I}$ where $\mathscr{D}(U_i) \subseteq \mathbb{I}$, with $1 \leqslant i \leqslant n$. In the following, we assume such a DCPGS $\mathscr{S}$.

In the following, we also assume an interpretation $\mathscr{I}$ for the function symbols in the expression language, and we assume that we can access the value of local variable x of agent u, such as location or velocity (see Section 3.2), in game state s as $s[u].x$. Let β be a type-consistent valuation of the logical variables. We define the interpretation of expression *expr* in a game state s of $\mathscr{S}$ under valuation β inductively over the structure of expressions:

— $\mathscr{I}[\![v]\!](s, \beta) = \beta(v)$.
— $\mathscr{I}[\![expr_1 = expr_2]\!](s, \beta) = true$ if and only if

$$\mathscr{I}[\![expr_1]\!](s, \beta) = \mathscr{I}[\![expr_2]\!](s, \beta).$$

— $\mathscr{I}[\![expr.x]\!](s, \beta) = s[u].x$ if and only if *expr* is an expression of type U_i and $u = \mathscr{I}[\![expr]\!](s, \beta)$.
— $\mathscr{I}[\![expr.x]\!](s, \beta) = s[u].x$ if and only if *expr* is an expression of type set-of-U_j and $\mathscr{I}[\![expr]\!](s, \beta)$ is the singleton set $\{u\}$.
$\mathscr{I}[\![expr.x]\!](s, \beta)$ is undefined otherwise.

The interpretation of expressions using function symbols is defined inductively as usual, except that the interpretation of an expression is undefined when the semantical value of one of its sub-expressions is undefined.

There are sufficient syntactic criteria on the description of the local behaviour to ensure that a set-typed link variable is a singleton. In the following, we assume that the interpretation of expressions is always defined.

3.3.2. *State and path formulae.* The set of state formulae is defined with respect to an expression language:

— *expr* is a state formula if *expr* is a Boolean expression from the expression language.
— $\neg\varphi_1$ and $\varphi_1 \vee \varphi_2$ are state formulae if and only if φ_1 and φ_2 are state formulae.
— $\forall v : T \bullet \varphi_1$ is a state formula if and only if $v \in \mathscr{V}$ is a logical variable of type T and φ_1 a state formula.
— $\langle\langle A_1, \ldots, A_n\rangle\rangle_{\geqslant p}\, \psi$ is a state formula if and only if $A_1, \ldots, A_n \in \mathscr{U}$, with $n \in \mathbb{N}_0$, are logical variables of agent type, $p \in [0, 1]$ is a probability bound and ψ is a path formula.

The set of path formulae is defined analogously with respect to an expression language:

— Any state formula φ_1 is a path formula.
— $\neg\psi_1$ and $\psi_1 \vee \psi_2$ are path formulae if and only if ψ_1 and ψ_2 are path formulae.
— $\mathbf{X}\,\psi_1$ is a path formula if and only if ψ_1 is a path formula.
— $\psi_1\ \mathbf{U}\ \psi_2$ is a path formula if and only if ψ_1 and ψ_2 are path formulae.

We will use the normal abbreviations for logical connectives, and $\Diamond\psi$ and $\Box\psi$ for $true\ \mathbf{U}\ \psi$ and $\neg\Diamond\neg\psi$, respectively. In addition, we obtain the bounded reachability $\Diamond_{\leqslant t}\psi$,

bounded invariant $\Box_{\leqslant t}\psi$ and bounded until $\psi_1 \mathbf{U}_{\leqslant t}\psi_2$ operator by introducing the following abbreviations:

$$\Diamond_{\leqslant t}\psi \equiv \bigvee_{0\leqslant i\leqslant t} \mathbf{X}^i\,\psi$$
$$\Box_{\leqslant t}\psi \equiv \bigwedge_{0\leqslant i\leqslant t} \mathbf{X}^i\,\psi$$
$$\psi_1\,\mathbf{U}_{\leqslant t}\,\psi_2 \equiv \bigvee_{0\leqslant i\leqslant t}\ \bigwedge_{0\leqslant j<i} \mathbf{X}^j\,\psi_1 \wedge \mathbf{X}^i\,\psi_2.$$

Furthermore, we use $[\psi]_{\geqslant p}$ as an abbreviation for $\langle\langle\emptyset\rangle\rangle_{\geqslant p}\,\psi$.

Let s be a game state of $\mathscr{S}$, and β be a valuation of the logical variables. The satisfaction relation between game states, valuations and state formulae is defined inductively over the structure of path formulae:

— $s,\beta \models expr$ if and only if $\mathscr{I}[\![expr]\!](s,\beta) = true$.
— $s,\beta \models \neg\varphi_1$ if and only if $s,\beta \not\models \varphi_1$.
— $s,\beta \models \varphi_1 \vee \varphi_2$ if and only if $s,\beta \models \varphi_1$ or $s,\beta \models \varphi_2$.
— $s,\beta \models \forall v : T \bullet \varphi_1$ if and only if for all $u \in \mathbb{I}$, if $u \in \mathrm{supp}(s)$ and $u \in \mathscr{D}(T)$, then $s,\beta[v := u] \models \varphi_1$.
— $s,\beta \models \langle\langle A_1,\ldots,A_n\rangle\rangle_{\geqslant p}\,\psi$ if and only if there is a strategy π starting in s for team $T = \{\beta(A_i) \mid 1 \leqslant i \leqslant n\}$ such that for any counterstrategy $\bar{\pi}$ and any scheduler π_{sc} starting in s, we have

$$Prob(\mathsf{Outcomes}(s,\pi,\bar{\pi},\pi_{sc}) \cap \{\omega \in \mathsf{InfPth}(s) \mid \omega,\beta \models \psi\}) \geqslant p.$$

Let ω be an infinite path of $\mathscr{S}$ and β be a valuation of the logical variables. The satisfaction relation between game states, valuations and path formulae is defined inductively over the structure of path formulae:

— $\omega,\beta \models \varphi_1$ if and only if $\omega(0),\beta \models \varphi_1$.
— $\omega,\beta \models \neg\psi_1$ if and only if $\omega,\beta \not\models \psi_1$.
— $\omega,\beta \models \psi_1 \vee \psi_2$ if and only if $\omega,\beta \models \psi_1$ or $\omega,\beta \models \psi_2$.
— $\omega,\beta \models \mathbf{X}\,\psi_1$ if and only if $\omega/1,\beta \models \psi_1$.
— $\omega,\beta \models \psi_1\,\mathbf{U}\,\psi_2$ if and only if there exists $k \in \mathbb{N}_0$ such that $\omega/k,\beta \models \psi_2$ and for each $0 \leqslant j < k$, we have $\omega/j,\beta \models \psi_1$.

We write $\mathscr{S} \models \psi$ if and only if for each strategy π for the empty team $T = \emptyset$, each counterstrategy $\bar{\pi}$ and each scheduler π_{sc}, all starting in $s_{init} \in S$, we have

$$\forall\,\omega \in \mathsf{Outcomes}(s_{init},\pi,\bar{\pi},\pi_{sc}) : \omega,\emptyset \models \psi.$$

Note that the semantical value of state and path formulae will always be well defined because of the assumption of monotone frame semantics and the definition of quantification ranging over the live agents, and due to our assumptions on SP in Section 3.2, namely that it creates cars with regular singleton interlinking.

In contrast to Ehlers *et al.* (2011), we do not explicitly assign newly created agents to the coalition or to the set of opponents. In our case study, the coalition never spawns

agents, only the opponents do (in the form of SP), and these new agents are implicitly assigned to the opponents in our semantics.

3.4. *Formalising the requirements on the driver assistance system*

In Section 3.4.3, we will rephrase the top-level requirement (R4) from Section 2.1 using the logic we have just introduced. In order to do this, we will first construct the required 'vocabulary' in the form of car observables in Section 3.4.1. We will then formally define the traffic situations referred to in (R4) as expressions over car observables in Section 3.4.2.

3.4.1. *Car observables.* In the following, we assume a number of expressions for referring to particular aspects of the local state of a car.

First, we assume a unary predicate *nominal* on car identities such that $nominal(C)$ holds for a given logical variable $C \in \mathcal{V}$, which denotes a car identity, if and only if the dynamics of that car are currently classified as nominal (see Section 2.1). We assume a predicate $driverint(C)$ that holds if and only if the current state of car C was reached by driver interaction. That is, this predicate holds, in particular, when the driver forces an abort of a manoeuvre. Technically, these properties can be observed within the local state of a $COOP$ instance.

We assume that predicate $incoalition(C)$ holds if and only if car C is currently participating in some coalition. The predicate $established(C_1, \ldots, C_n)$ is assumed to hold if and only if cars $C_1, \ldots, C_n$ are currently participants of the same coalition – note, however, that the whole coalition may be larger. Technically, the $incoalition(C)$ and $established(C_1, \ldots, C_n)$ properties can be expressed in terms of the corresponding link-valued variables of $COOP$ instances. At least the car initiating the coalition has a link that keeps track of the cars that have already joined the coalition.

We assume that for each manoeuvre $m \in M$ offered by cars, there are predicates $start_m(C)$, $succ_m(C)$ and $active_m(C)$ holding, respectively, when car C has just changed from 'cruise' mode to the mode corresponding to manoeuvre m, when that manoeuvre has been successfully completed, that is, just before returning to 'cruise' mode, and when it is in between these two stages. These conditions can be observed by means of local states of the manoeuvre mode.

For readability, we may write $C.pred$ instead of $pred(C)$ for unary predicates.

We will assume that we can refer to the physical situation of cars using expressions $expr.pos_{min}$, $expr.pos_{max}$, and $expr.l_{min}$ and $expr.l_{max}$, where $expr$ is of agent type AUT. There is in general no unique position and lane since the description by service specifications introduce a certain vagueness.

Since we can navigate from a $COOP$ instance to the associated AUT instance using the *aut* link, we may write, for example, $A.pos_{min}$ instead of $A.aut.pos_{min}$ if A is of type $COOP$. Furthermore, we use $A.l = c$ as an abbreviation for $A.l_{min} = A.l_{max} = c$.

From these observables, we can construct more complex expressions to refer to the physical situation of cars on the highway (see Table 1), starting from the lateral and longitudinal distance between two cars, through the characterisation of a safe situation by $SAFE^{\rightarrow}(A, B)$ and finishing with expressions referring to the relative positions of cars.

Table 1. *Abbreviations used to refer to car distances and positions.*

$$
\begin{aligned}
\delta^{\rightarrow}(A,B) &:= B.pos_{min} - A.pos_{max} \\
\delta^{\leftarrow}(A,B) &:= A.pos_{min} - B.pos_{max} \\
\delta^{\leftrightarrow}(A,B) &:= \min(\max(0,\delta^{\rightarrow}(A,B)),\max(0,\delta^{\leftarrow}(A,B))) \\
\delta^{\uparrow}(A,B) &:= B.l_{min} - A.l_{max} \\
\delta^{\downarrow}(A,B) &:= A.l_{min} - B.l_{max} \\
\delta^{\updownarrow}(A,B) &:= \min(\max(0,\delta^{\uparrow}(A,B)),\max(0,\delta^{\downarrow}(A,B))) \\
SAFE^{\rightarrow}(A,B) &:= \bigwedge_{1\leqslant j\leqslant n} A.v = c_j \wedge \delta^{\rightarrow}(A,B) > 0 \implies \delta^{\rightarrow}(A,B) > dist_j \\
BEFORE(A,B,c) &:= A \neq B \wedge (B.l_{min} = c \vee B.l_{max} = c) \wedge \\
&\qquad \forall\, C : COOP \bullet C \neq A \wedge C \neq B \wedge (C.l_{min} = c \vee C.l_{max} = c) \wedge \\
&\qquad \delta^{\rightarrow}(A,C) > 0 \implies \delta^{\rightarrow}(A,C) \geqslant \delta^{\rightarrow}(A,B) \\
BEFORE(A,B) &:= \exists\, c \bullet BEFORE(A,B,c) \\
SAFE^{\rightarrow}(A) &:= \forall\, B : COOP \bullet BEFORE(A,B) \implies SAFE^{\rightarrow}(A,B) \\
SAFE\_GAP(A,k) &:= \forall\, B : COOP \bullet B \neq A \implies (BEFORE(A,B,k) \implies \\
&\qquad SAFE^{\rightarrow}(A,B)) \wedge (BEHIND(A,B,k) \implies SAFE^{\leftarrow}(A,B))
\end{aligned}
$$

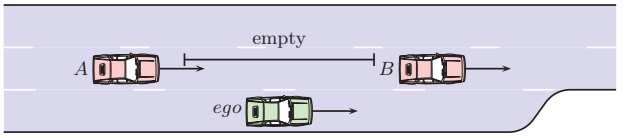


Fig. 13. (Colour online) Example traffic situation. The ego-car is on the entry ramp and there are two cars A and B on the neighbouring lane, which are the spatially nearest cars behind and in front of *ego* on that lane. There may be other cars behind A and in front of B. There are no restrictions on the presence or absence of cars on any other lanes besides these two.

3.4.2. *Traffic situations.* Formally, a traffic situation is an expression

$$\psi(ego, A_1, \ldots, A_n)$$

with free logical variables $ego, A_1, \ldots, A_n$ of agent type $COOP$.

Traffic situation expressions may be restricted to refer to local variables representing physical properties such as position and speed (local variables of the AUT instance linked to the denoted $COOP$ instance), but we may also need to refer to information such as whether a car is currently in a coalition as visible in the form of link-typed local variables of $COOP$ instances.

For example, Figure 13 depicts a game state that satisfies the traffic situation

$$\exists\, c : 0, \ldots, L \bullet ego.l = c \wedge BEFORE(ego, A_1, c+1) \wedge BEHIND(ego, A_2, c+1). \qquad (1)$$

We assume that for each traffic situation ψ, there is a proposal c_ψ for a coalition in the form of a subset of the free logical variables in ψ, that is, $c_\psi \subseteq \{ego, A_1, \ldots, A_n\}$.

We further assume that a traffic situation expression implies that the logical variables $ego, A_1, \ldots, A_n$ are bound to *different* cars. In (1), this is ensured implicitly by the definition of *BEFORE* and *BEHIND*.

3.4.3. *Formalising the requirements for driver assistance systems.* We can now revisit requirement (R4) in Section 2.1 as a formula of the logic we have just defined over the car observables introduced in Section 3.4.1.

The task is to check whether

$$\begin{aligned} \mathscr{S} \models \Box \forall\, ego, A_1, \ldots, A_n : COOP \bullet \\ \psi(ego, A_1, \ldots, A_n) \wedge ego.safe \wedge ego.start_m \wedge (\Box_{\leqslant t_{2\psi}} \neg \psi_{abnormal}) \\ \Longrightarrow \langle\langle c_\psi \rangle\rangle_{\geqslant p} \left(ego.safe \,\mathbf{U}_{\leqslant t_{2\psi}}\, ego.safe \wedge ego.succ_m \right) \end{aligned} \tag{2}$$

where $\mathscr{S}$ is the cooperative driver assistance system constructed from *COM*, *AUT*, *COOP*, *HW* and *SP* as described in Section 3.2. Note that the logical variables in (2) now range over *COOP* agents instead of the *ad hoc* notation *Car* we used in Section 2.1. Each car consists of three interlinked agents, one *COOP*, one *AUT* and one *COM* instance – see Section 3.2. Intuitively, the *COOP* agents forming the coalition and playing against all agents in other cars, and also against their own *AUT* and *COM* layers because the non-determinism in *AUT* and *COM* stems from the integration of an environment model into *AUT* and the characteristics of unreliable communication in *COM*, neither of which can be controlled by the cooperation layer.

The maximum allowed time for manoeuvre m in traffic situation ψ is $t_{2\psi}$. The expression $\psi_{abnormal}$ concretises to

$$(\exists\, C : COOP \bullet near(ego, C) \Longrightarrow C.incoalition) \tag{3}$$

$$\vee \left(\exists\, C : COOP \bullet \bigvee_{m \in M} (\neg(C.start_m \Longrightarrow C = ego) \,\mathbf{U}\, established(c_\psi)) \right) \tag{4}$$

$$\begin{aligned} \vee \Diamond_{\leqslant t_{2\psi}} \big(\exists\, C : COOP \bullet near(ego, C) \wedge \neg C.nominal \\ \vee driverint.ego \vee driverint.A_1 \vee \cdots \vee driverint.A_n \big) \end{aligned} \tag{5}$$

where the predicate $near(ego, C)$ characterises a certain spatial neighbourhood around the ego-car – this will play a role when constructing the final abstraction of the system.

Disjunct (3) assumes that certain cars are not yet in a coalition (as observed by *incoalition*).

Disjunct (4) assumes that until the desired coalition as given by c_ψ is established, at most *ego* is starting any manoeuvre. The idea is that there is an area (possibly marked by traffic signs) where seeking coalitions is forbidden. For other cars on the entry lane, we can assume a 'first come first serve' principle by which the next following car could be notified by the first car when it can start to build its coalition, or where the next following car simply perceives a different traffic situation that may indicate when it is time to start building its own coalition.

Thus, it is at most *ego* that may be looking for coalition partners in its neighbourhood, so there will be no race conditions for coalition requests. Under this assumption, all requests to join a coalition will be granted. This corresponds to probability theory's 'most

helpful' scheduler, so we can effectively refer to the probability of manoeuvre success where only unreliable communication is responsible for not meeting a probability bound.

Finally, disjunct (5) requires that all relevant cars have nominal behaviour during manoeuvre time $t_{2\psi}$, and that no driver interaction takes place. Driver interaction releases the assistance system from the obligation to complete the manoeuvre successfully.

4. Formal realisability analysis

4.1. *Approach*

Recall the overall property (R4) from Section 2 in the precise form given in Section 3.4.2:

$$\begin{aligned}\mathscr{S} \models \Box \forall\, ego, A_1, \ldots, A_n : COOP \bullet \\ \psi(ego, A_1, \ldots, A_n) \wedge ego.safe \wedge ego.start_m \wedge (\Box_{\leqslant t_{2\psi}} \neg \psi_{abnormal}) \\ \Longrightarrow \langle\langle c_\psi \rangle\rangle_{\geqslant p} \left(ego.safe \; \mathbf{U}_{\leqslant t_{2\psi}} \; ego.safe \wedge ego.succ_m\right)\end{aligned} \tag{6}$$

Because we assume that some reflex functionality of the autonomous layer ensures the safety of cars (see Section 2), we will continue without considering the safety of the ego-car explicitly, that is,

$$\begin{aligned}\mathscr{S} \models \Box \forall\, ego, A_1, \ldots, A_n : COOP \bullet \\ \psi(ego, A_1, \ldots, A_n) \wedge ego.start_m \wedge (\Box_{\leqslant t_{2\psi}} \neg \psi_{abnormal}) \\ \Longrightarrow \langle\langle c_\psi \rangle\rangle_{\geqslant p} \Diamond_{\leqslant t_{2\psi}} ego.succ_m.\end{aligned} \tag{7}$$

We split the problem into probabilistic and game-theoretic problems by distinguishing two phases:

(1) Establishing the coalition through (unreliable) communication – prospective coalition members do not work 'in favour' of *ego* yet.

(2) Conducting the actual manoeuvre in the established coalition without communication, and starting from the traffic situations possibly reachable after the first phase.

We assume that both phases are assigned amounts of time $t_1, t_2 > 0$ such that $t_1 + t_2 = t_{2\psi}$ (see Figure 14).

We first establish that the likelihood of a successful coalition formation is at least p, that is, we establish

$$\begin{aligned}\mathscr{S} \models \Box \forall\, ego, A_1, \ldots, A_n : COOP \bullet \\ \psi(ego, A_1, \ldots, A_n) \wedge ego.start_m \wedge (\Box_{\leqslant t_1} \neg \psi_{abnormal}) \\ \Longrightarrow \left[\Diamond_{\leqslant t_1} established(c_\psi)\right]_{\geqslant p}\end{aligned} \tag{8}$$

using an off-line analysis (see Section 4.2).

Then, from the traffic situations possibly evolving out of ψ within t_1 time units, characterised by predicate

$$\psi + t_1,$$

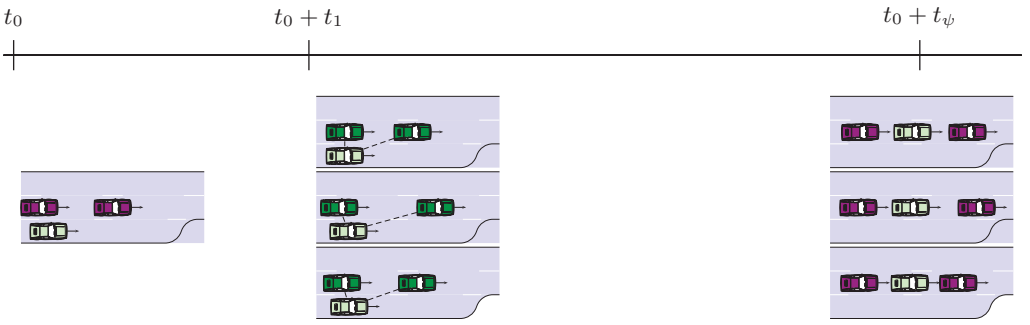


Fig. 14. (Colour online) Time budgeting.
The coalition is established between t_0 and $t_0 + t_1$, and during this time, the traffic situation evolves into possibly multiple situations, which can be precomputed. The manoeuvre is carried out between $t_0 + t_1$ and $t_0 + t_1 + t_2 = t_0 + t_{2\psi}$.

we check whether

$$\begin{aligned}\mathscr{S} \models \Box \forall\, ego, A_1, \ldots, A_n : COOP \bullet \\ (\psi + t_1)(ego, A_1, \ldots, A_n) \wedge ego.active_m \wedge \\ established(c_\psi) \wedge (\Box_{\leqslant t_2} \neg \psi_{abnormal}) \\ \Longrightarrow \langle\langle c_\psi \rangle\rangle \Diamond_{\leqslant t_2} ego.succ_m.\end{aligned} \tag{9}$$

Equation (9) refers to an infinite state transition system because $\mathscr{S}$ imposes no bound on the number of agents, and the grid in the HW instance is infinite. Therefore, we establish

$$\begin{aligned}\mathscr{S}^{\#} \models \Box \forall\, ego, A_1, \ldots, A_n : COOP \bullet \\ (\psi + t_1)(ego, A_1, \ldots, A_n) \wedge ego.active_m \wedge \\ established(c_\psi) \wedge (\Box_{\leqslant t_2} \neg \psi_{abnormal}) \\ \Longrightarrow \langle\langle c_\psi \rangle\rangle \Diamond_{\leqslant t_2} ego.succ_m\end{aligned} \tag{10}$$

where $\mathscr{S}^{\#}$ is a finite abstraction of $\mathscr{S}$ – see Section 4.3 for details of the construction of $\mathscr{S}^{\#}$ from $\mathscr{S}$.

Equation (10) is a classical finite synthesis problem, that is, discrete, non-hybrid, finite and non-probabilistic. In particular, the logical variables now range over a finite set of identities $\mathbb{I}_{fin}$. By symmetry†, it is even the case that we only need to consider a single variable valuation β of $ego, A_1, \ldots, A_n$ that binds these $n + 1$ logical variables to different $COOP$ agent identities from $\mathbb{I}_{fin}$ because traffic situations are required to ensure that the different logical variables denote different agents (see Section 3.4.2). That is, the single

† The behaviour of an agent does not depend on its identity as the local behaviour descriptions refer to identities only symbolically. Identities can be stored and compared, but an agent cannot determine whether its identity is the value u_1 or u_2 and change its behaviour accordingly. Therefore, undesired behaviour can be observed for pairwise different cars $u_1, \ldots, u_n$ if and only if undesired behaviour can be observed for any other set of pairwise different identities $u'_1, \ldots, u'_n$, provided creation is assumed to assign identities to newly created agents fully non-deterministically – see Westphal (2008) for details.

synthesis task

$$\begin{aligned}\mathscr{S}^{\#}, \beta \models \Box \big(& ((\psi + t_1)(ego, A_1, \ldots, A_n) \wedge ego.active_m \wedge \\ & established(c_\psi) \wedge (\Box_{\leqslant t_{2\psi}} \neg \psi_{abnormal}) \\ & \Longrightarrow \langle\langle c_\psi \rangle\rangle \Diamond_{\leqslant t_2} ego.succ_m \big)\end{aligned} \tag{11}$$

is equivalent to (10).

From Section 4.4, we can conclude (9) from (10). From (8) and (9), we can conclude (7) because we can obtain the strategy required in (7) by prefixing the deterministic coalition establishing the procedure of COOP (see Section 4.2) to the strategy obtained from (9). From (7), we can then obtain (R4).

Note that in the two-phase setting, the approach proposed here is exact if all possible time budgets are enumerated, and there are finitely many of them in our discrete-time model.

4.2. *Determining the likelihood of coalition formation*

For the following off-line analysis, we assume that there is a deterministic procedure in $COOP$ that tries to establish coalition c_ψ when the manoeuvre is started.

By the assumptions in (8), the probability of a coalition being successfully established is exclusively determined by the communication reliability because the condition $\neg\psi_{abnormal}$ excludes races: there is no competition for helpers during t_1.

Thus, we can compute the probability of successful coalition establishment for this procedure as follows:

— Assume that the start of the manoeuvre as indicated by $ego.start_m$ corresponds to a location loc in the $COOP$ description.
— Assume that the successful establishment of the coalition is indicated by a unique second location loc'.
— As COM gives the probability of successful inter-car communication in one step of the game structure, we can substitute the probabilistic edges from COM into the procedure in $COOP$ and compute the probability for reaching loc' as the sum of the products of probabilities along the paths from loc to loc'.

The resulting probability p is a lower bound for the probability of successfully completing manoeuvre m in traffic situation ψ. The overall lower bound can be increased by considering other time budgets, that is, other times t_1', t_2' such that $t_1' + t_2' = t_{2\psi}$.

4.3. *A safe finite abstraction of the infinite grid*

In Section 4.3.1, we define a finite abstraction of $\mathscr{S}$ in terms of $\mathscr{S}$, though this is not, in general, an effective procedure. In Section 4.3.2, we outline a procedure for how we can effectively obtain the finite abstraction using a modification of the agents HW and SP.

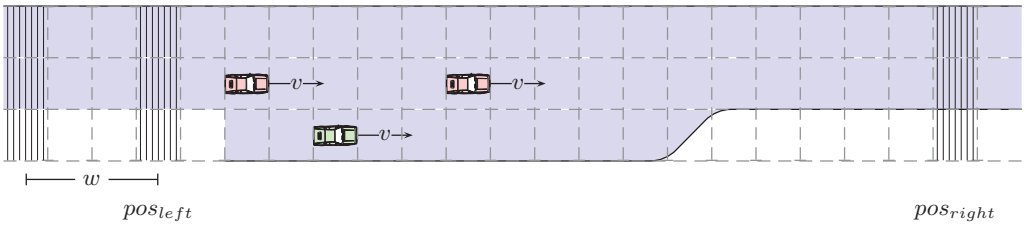


Fig. 15. (Colour online) The highway segment is the part of the highway ranging from position $pos_{left} - w$ to pos_{right} and is assumed to include an entry ramp.

4.3.1. *Definition in terms of the infinite system.* Let $pos_{left} < pos_{right}$ be positions on the grid such that a representative entry ramp is covered and the neighbourhood of ego-cars on the ramp is included (see Section 3.4.3).

Let w be the maximal spatial length of a service, that is, the maximal distance between the anchor point and the rightmost field referred to by an expression in a service specification. In the service specification example in Section 2.2.1, this would be the horizontal distance between the anchor and any light or dark grey tile. In the following, we call the highway positions from pos_{left} to pos_{right} the highway segment (see Figure 15). The length w will be added to the left of the highway segment to provide sufficient space for anchors.

Let $\mathscr{S}$ be a game structure over $\mathbb{I}$ partitioned into identities for HW, SP, $COOP$, AUT and COM as described in Section 3.2. Let $\mathbb{I}_{fin}$ be a finite set of identities providing:

— at least as many identities for $COOP$, AUT and COM such that we have a grid of size

$$(pos_{right} - pos_{left} + w) \cdot L,$$

where L is the number of lanes, in order to have enough identities to cover each field of the area between $pos_{left} - w$ and pos_{right} with a car; and
— at least one identity for the $SP$and HW instances.

For simplicity, we will assume that cars inside the chosen area never have relations with cars outside the area – this is not a problem of principal since we can treat it by introducing a dedicated 'other' identity per agent type.

Let $s, s' \in S$ be two states of $\mathscr{S}$. We use $in(s, s')$ to denote the set of cars that appear first in the highway segment in game state s', that is, the set of $u \in \mathbb{I}_{COOP}$ such that the set of currently possible positions of car u as given by an instantiation of the invariant of the current service$s[u].srv$ with service duration time $s[u].d_{srv}$ and anchor position $(s[u].pos_{srv}, s[u].l_{srv})$, that is,

$$inv(s[u].srv, s[u].d_{srv})(s[u].pos_{srv}, s[u].l_{srv}),$$

has an empty intersection with the highway segment, and the set of positions possible in game state s' as given by

$$inv(s'[u].srv, s'[u].d_{srv})(s'[u].pos_{srv}, s'[u].l_{srv})$$

has a non-empty intersection with the highway segment.

We use $In(s, s')$ to denote the set of quadruples

$$(srv, pos_{srv}, l_{srv}, d_{srv})$$

such that there exists an identity $u \in in(s, s')$ with

$$\begin{aligned} s[u].srv &= srv \\ s[u].pos_{srv} &= pos_{srv} \\ s[u].l_{srv} &= l_{srv} \\ s[u].d_{srv} &= d_{srv}. \end{aligned}$$

We assume that no two different cars have the same service configuration and anchor at the same point in time as this would imply a collision, a case which is excluded by the assumption of property (10).

Correspondingly, we use $out(s, s')$ to denote the set of cars that appear last in the highway segment in game state s, that is, the set of $u \in \mathbb{I}_{COOP}$ such that the set of currently possible positions of u in s,

$$inv(s[u].srv, s[u].d_{srv})(s[u].pos_{srv}, s[u].l_{srv}),$$

has a non-empty intersection with the highway segment, and the set of currently possible positions in s',

$$inv(s'[u].srv, s'[u].d_{srv})(s'[u].pos_{srv}, s'[u].l_{srv}),$$

has an empty intersection with the highway segment. We use $Out(s, s')$ to denote the set of quadruples

$$(srv, pos_{srv}, l_{srv}, d_{srv})$$

such that there exists $u \in out(s, s')$ with

$$\begin{aligned} s'[u].srv &= srv \\ s'[u].pos_{srv} &= pos_{srv} \\ s'[u].l_{srv} &= l_{srv} \\ s'[u].d_{srv} &= d_{srv}. \end{aligned}$$

The abstract game structure $\mathscr{S}^{\#}$ is constructed as follows. The game states $S^{\#}$ are the states from S projected onto $\mathbb{I}_{fin}$ for a reasonable definition of projection, for example, if we view $s \in S$ as a function mapping agent identities to valuations of the local variables of agents, then $s^{\#}$ is the function restriction of this function to $\mathbb{I}_{fin}$.

We write $s \supseteq s^{\#}$ if and only if $s^{\#}$ is the result of projecting s onto $\mathbb{I}_{fin}$. Note that the local variables storing the anchor position now only need to range over

$$(pos_{right} - pos_{left} + w) \cdot L,$$

so we obtain a finite state system. The initial state $s^{\#}_{init}$ is the projection of s_{init} onto $\mathbb{I}_{fin}$.

The set of actions $A^{\#}$ coincides with A, and the set of moves in $\mathscr{S}^{\#}$ is made up of the moves of $\mathscr{S}$ projected onto $\mathbb{I}_{fin}$, that is,

$$M^{\#} = M \cap (\mathbb{I}_{fin} \times A).$$

The available moves are obtained similarly since

$$\Gamma^{\#}(s^{\#}) = \Gamma(s) \cap M^{\#}.$$

This is well defined by construction since the available moves per agent depend only on the local state of the traffic agent, and not on any others.

The abstract destination function

$$\delta^{\#} : S^{\#} \times 2^{M^{\#}} \to (2^{\mu(S^{\#})} \setminus \emptyset)$$

is constructed using a multi-step procedure following the classical approach to provide at least one abstract transition for each transition in the concrete system $\mathscr{S}$. We can also eliminate probabilistic edges because, for the realisability analysis, it is only important that coalitions can be established, and not the probability that they are established.

Let $s^{\#} \in S^{\#}$ be an abstract state and $\kappa^{\#}$ be a moveset in $\mathscr{S}^{\#}$. Let $s \sqsupseteq s^{\#}$ and $\kappa \sqsupseteq \kappa^{\#}$ be concretisations of $s^{\#}$ and $\kappa^{\#}$. For each $P \in \delta(s, \kappa)$, we add

$$\{s^{\#\prime} \mapsto 1.0\}$$

to $\delta^{\#}$ (that is, we substitute all probabilistic branching by non-determinism) where $s^{\#}$ is obtained as follows (see Figure 16):

— We project s' onto $\mathrm{supp}(s^{\#})$, that is, onto the agents that have already been on the highway segment in $s^{\#}$.
— We remove the $COOP$, AUT, and COM instances given by $out(s, s')$.
— For each service state

$$(srv, pos_{srv}, l_{srv}, d_{srv}) \in In(s, s'),$$

we choose an unused identity (possibly from the just removed ones) from $\mathbb{I}_{fin}$ for $COOP$, AUT and COM, and construct a car that corresponds to a car that started the service srv at anchor position (pos_{srv}, l_{srv}) at d_{srv} time units ago.[†]

This choice is possible because of our assumption above regarding the size of $\mathbb{I}_{fin}$.

That is, the moves of SP and HW are disregarded in practice (though their effects are still visible: cars are created on the entry ramps that lie within the highway segment).

The transition system $S^{\#}$ has a finite representation which, despite the fact that it is strictly speaking a probabilistic and dynamic game structure, is actually an ordinary game structure, so existing procedures apply.

4.3.2. *Constructive definition of the finite abstraction.* The declarative construction we have given in Section 4.3.1 can also be obtained by an effective procedure in which we define SP using corresponding create actions.

[†] By symmetry, we can choose the identities in ascending order to optimise the size of the state space.

Fig. 16. Definition of the abstract transition relation.
In $\mathscr{S}^{\#}$, there is a transition from $s^{\#}$ to $s^{\#\prime}$ because:
(i) $s^{\#}$ can, in particular, be extended to s. Note that the car identities in $s^{\#}$ and s coincide within the highway segment – outside the highway segment, cars may carry any other identity from $\mathbb{I}$ or $\mathbb{I}_{fin}$:
(ii) There is a transition from s to s' in $\mathscr{S}$. Note that car identities do not change during this transition.
(iii) $s^{\#\prime}$ is a projection of s' on $\mathbb{I}_{fin}$ in the following sense. We remove u_3 from s' because it now lies completely outside the highway segment. The cars from $\mathbb{I} \setminus \mathbb{I}_{fin}$ are removed in all cases. Note that u_4 (which we assume to be from $\mathbb{I} \setminus \mathbb{I}_{fin}$ for the example) is in $In(s, s')$. It enters the highway segment in the transition from s to s'. To obtain $s^{\#\prime}$, we create u_3 (that is, we re-use the just removed identity) with the same configuration as found with car u_4.

We need to mimic the over-approximation at the right-hand side of the highway segment using 'modified sensors'. A modified grid HW is supposed to give any possible answer to queries regarding conflicts between assumptions and invariants (see Section 3.2).

Similarly, at the left-hand border pos_{left}, the modified HW non-deterministically gives any possible answer to conflict queries for tilings that reach beyond the left-hand border. This may lead to cars running into accidents, but if they do, the $\neg\psi_{abnormal}$ premise no longer holds, so the case is effectively not considered in the analysis of property (10).

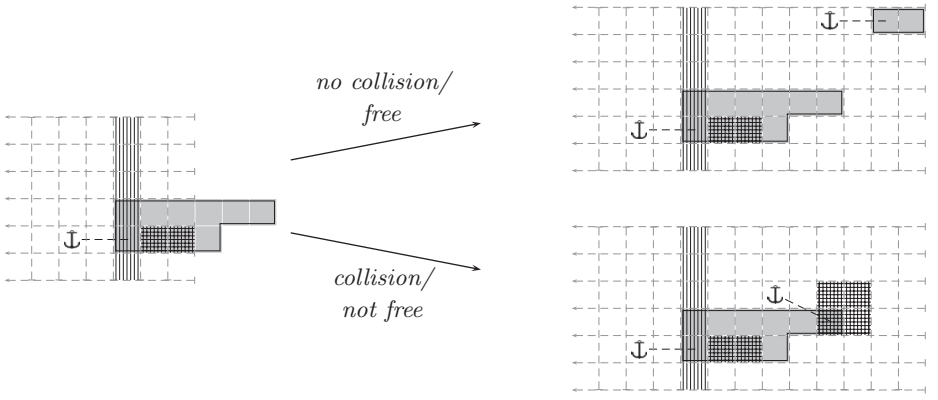


Fig. 17. Blurred sensors by non-deterministic replies from HW.
If there is a concretisation of $s^{\#}$ such that a tiling extending beyond the right-hand border has a non-empty overlap with another tiling, HW is supposed to give a positive answer to queries for overlap (lower case). As the region to the right of the right-hand border may always be empty, a negative answer is always possible if there are no existing overlaps in the highway segment.

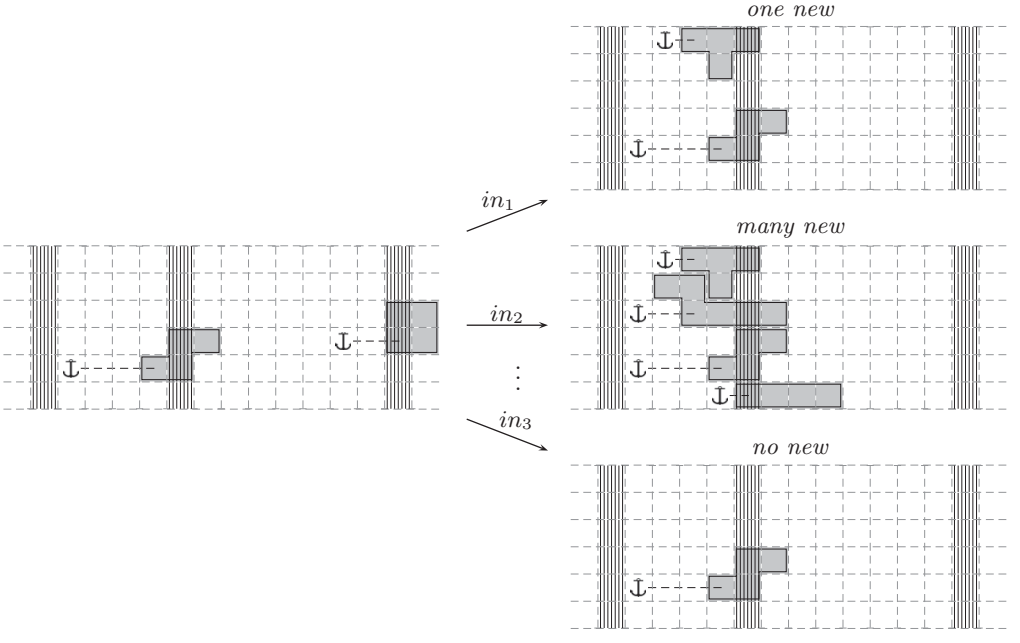


Fig. 18. Enter/leave segment.

There is a range of possiblities for over-approximating the behaviour at the left-hand border. At one extreme, there is 'full chaos', where new cars are created, possibly in the middle of a service, and possibly in a collision situation, though collision situations will not be included in the analysis because of the constraints in property (10). Alternatively,

cars may be created sensibly, in particular, avoiding crashes. But in this approach, it is more difficult to ensure that $\mathcal{S}^{\#}$ is an over-approximation of $\mathcal{S}$.

4.4. *Lifting bounded-horizon strategies to unbounded traffic environments*

4.4.1. *Simulation relation.* Let

$$\mathcal{S} = (S, s_{init}, A, \Gamma, \delta)$$
$$\mathcal{S}^{\#} = (S^{\#}, s^{\#}_{init}, A^{\#}, \Gamma^{\#}, \delta^{\#})$$

be dynamic concurrent probabilistic timed game structures over $\mathbb{I}$ and $\mathbb{I}_{fin} \subseteq \mathbb{I}$. In particular, we assume that $\mathcal{S}^{\#}$ is trivial with respect to probabilism. This is easy to achieve by replacing probabilistic branching by true non-determinism.

A relation $h \subseteq S \times S^{\#}$ is said to be a *simulation relation* if and only if for each $(s, s^{\#}) \in h$:

(1) For each well-typed Boolean expression *expr*,

$$s^{\#}, \beta \models expr \implies s, \beta \models expr$$

if *expr* is link-local to $\mathbb{I}_{fin}$ under β, that is, for each sub-expression $expr_1$ of type U_i or set-of-U_i, we have

$$\mathcal{I}[\![expr_1]\!](s^{\#}, \beta) \in \mathbb{I}_{fin}$$

or

$$\mathcal{I}[\![expr_1]\!](s^{\#}, \beta) \subseteq \mathbb{I}_{fin}.$$

So, in particular, β binds logical variables of agent type only to identities from $\mathbb{I}_{fin}$, that is, $\beta(A) \in \mathbb{I}_{fin}$ for each $A \in \mathcal{U}$.

(2) Let $T = \{u_1, \ldots, u_n\} \subseteq \mathbb{I}_{fin}$ be a team that is alive in s, so the team has to be alive in $s^{\#}$. Let $\kappa \subseteq \Gamma(s)$ be a set of moves, and $\kappa_T := \{\gamma_1, \ldots, \gamma_n\}$ be the moves of the team. Then

$$s \xrightarrow{\kappa} s'$$

implies that there exists $\kappa^{\#} \subseteq \Gamma(s^{\#})$ such that $\kappa^{\#}_T = \kappa_T$ and that there exists $s^{\#\prime}$ such that

$$s^{\#} \xrightarrow{\kappa^{\#}} s^{\#\prime}$$

and $(s', s^{\#\prime}) \in h$.

We say $\mathcal{S}^{\#}$ simulates $\mathcal{S}$ if and only if $\mathcal{S}$ and $\mathcal{S}^{\#}$ are dynamic concurrent probabilistic timed game structures over $\mathbb{I}$ and $\mathbb{I}_{fin} \subseteq \mathbb{I}$, and there exists a simulation relation h as defined above.

4.4.2. *The finite abstraction simulates the infinite system* For $\mathcal{S}, \mathcal{S}^{\#}$ from Section 4.3.1,

$$h = \{(s, s^{\#}) \in S \times S^{\#} \mid s \supseteq s^{\#}\}$$

is a simulation relation.

Let *expr* be an expression that is link-local to $\mathbb{I}_{fin}$ under a valuation β of the logical variables in *expr*. As the interpretation of expressions is defined inductively on the structure, it is sufficient to consider the agent and set-of-agent typed expressions:

— $\mathcal{I}[\![v]\!](s^{\#}, \beta) = \beta(v) = \mathcal{I}[\![v]\!](s, \beta)$.
— $\mathcal{I}[\![expr.x]\!](s^{\#}, \beta) = s^{\#}[\mathcal{I}[\![expr]\!](s^{\#}, \beta)].x = s[\mathcal{I}[\![expr]\!](s^{\#}, \beta)].x$ because *expr* is link-local to $\mathbb{I}_{fin}$ and $s^{\#}$ and s coincide on $\mathbb{I}_{fin}$.

Let $(s, s^{\#}) \in h$ and $T = \{u_1, \ldots, u_n\} \subseteq \mathbb{I}_{fin}$ be a team that is alive in s. The team is alive in $s^{\#}$ because $s \supseteq s^{\#}$.

Let $\kappa \subseteq \Gamma(s)$ be a set of moves consisting of the team moves $\kappa_T := \{\gamma_1, \ldots, \gamma_n\}$, and let

$$s \xrightarrow{\kappa} s'.$$

Set

$$\kappa^{\#} = \{(u, a) \in \kappa \mid u \in \mathbb{I}_{fin}\}.$$

Then $\kappa^{\#} \supseteq \kappa_T$ and $\kappa^{\#} \in \Gamma^{\#}(s^{\#})$ by construction of $\Gamma^{\#}$. Using $s^{\#\prime}$ as constructed above based on $In(s, s')$ and $Out(s, s')$, we have

$$s^{\#} \xrightarrow{\kappa} s^{\#\prime}$$

by construction of $\delta^{\#}$.

4.4.3. *Lifting theorem.*

Theorem 4.1. Let $\mathcal{S}$ and $\mathcal{S}^{\#}$ be dynamic concurrent trivially probabilistic timed game structures such that $\mathcal{S}^{\#}$ simulates $\mathcal{S}$, and let

$$\varphi \equiv \Box expr_1 \wedge (\Box_{\leqslant t} expr_2) \implies \langle\langle ego, A_1, \ldots, A_n \rangle\rangle \Diamond_{\leqslant t} expr_3 \tag{12}$$

be a state formula with free logical variables $ego, A_1, \ldots, A_n$ and path formulae ψ_1, ψ_2, ψ_3. If φ is link-local to $\mathbb{I}_{fin}$ under valuation β, then

$$\mathcal{S}^{\#}, \beta \models \varphi \implies \mathcal{S}, \beta \models \varphi. \tag{13}$$

Proof. Let $\mathcal{S}^{\#}$ simulate $\mathcal{S}$ and let φ be a state formula with a structure as given above. Let β be a valuation of the logical variables $ego, A_1, \ldots, A_n$ such that φ is link-local to $\mathbb{I}_{fin}$ under β.

Assume $\mathcal{S}^{\#}, \beta \models \varphi$. By the semantics of state formulae (see Section 3.3.2), there exists, for each game state $s^{\#} \in S^{\#}$ such that

$$s^{\#}, \beta \models expr_1 \wedge (\Box_{\leqslant t} expr_2), \tag{14}$$

a strategy $\pi^{\#}$ starting in $s^{\#}$ for team

$$T = \{\beta(ego)\} \cup \{\beta(A_i) \mid 1 \leqslant i \leqslant n\}$$

such that for any counter-strategy $\bar{\pi}^{\#}$ and any scheduler $\pi^{\#}_{sc}$ starting in $s^{\#}$, we have

$$Prob(\mathsf{Outcomes}(s^{\#}, \pi^{\#}, \bar{\pi}^{\#}, \pi^{\#}_{sc}) \cap \{\omega \in \mathsf{InfPth}(s^{\#}) \mid \omega^{\#}, \beta \models \psi\}) \geqslant 1.0.$$

Let $s \in S$ be a game state such that

$$s, \beta \models expr_1 \wedge (\Box_{\leqslant t} expr_2).$$

As φ is link-local, there exists $s^{\#} \in S^{\#}$ such that $s \supseteq s^{\#}$ and $(s, s^{\#}) \in h$.

We now construct a strategy $\pi : \mathsf{FinPth}(s)$ in $\mathscr{S}$ from $\pi^{\#}$ from $s^{\#}$ as follows. Let

$$\omega = s_0 \xrightarrow{\kappa_1} s_1 \ldots \xrightarrow{\kappa_n} s_n \in \mathsf{FinPth}(s).$$

Because $\mathscr{S}^{\#}$ simulates $\mathscr{S}$, there is

$$\omega^{\#} = s_0^{\#} \xrightarrow{\kappa_1^{\#}} s_1^{\#} \ldots \xrightarrow{\kappa_n^{\#}} s_n^{\#} \in \mathsf{FinPth}(s^{\#}).$$

Set $\pi(\omega) = \pi^{\#}(\omega^{\#})$. We can now obtain a strategy in $\mathscr{S}$ because $\pi^{\#}$ gives moves for the team that is the same in $\mathscr{S}$ by definition of β.

We now assume π is not a winning strategy from s in $\mathscr{S}$ in order to show a contradiction. That is, there exists a counter-strategy $\bar{\pi}$, a scheduler π_{sc} and a path

$$\omega = s_0 \xrightarrow{\kappa_1} s_1 \ldots \xrightarrow{\kappa_t} s_t \cdots \in \mathsf{Outcomes}(s, \pi, \bar{\pi}, \pi_{sc})$$

such that

$$\omega, \beta \not\models \Diamond_{\leqslant t} expr_3.$$

So

$$s_i, \beta \not\models expr_3, 0 \leqslant i \leqslant t.$$

By the simulation relation between $\mathscr{S}$ and $\mathscr{S}^{\#}$, there is a path

$$\omega^{\#} = s_0^{\#} \xrightarrow{\kappa_1^{\#}} s_1^{\#} \ldots \xrightarrow{\kappa_t^{\#}} s_t^{\#} \ldots$$

where the coalition T follows strategy $\pi^{\#}$, that is, $\pi^{\#}(\omega_{0..i}) \subseteq \kappa_i^{\#}$.

From $\omega^{\#}$, we can construct a counter-strategy $\bar{\pi}^{\#}$ and a scheduler $\pi_{sc}^{\#}$ by considering the complement of the team moves. Then we have

$$\omega^{\#} \in \mathsf{Outcomes}(s^{\#}, \pi^{\#}, \bar{\pi}^{\#}, \pi_{sc}^{\#})$$

and, by the simulation property,

$$s_i^{\#}, \beta \not\models expr_3, 0 \leqslant i \leqslant t,$$

thus

$$\omega^{\#}, \beta \not\models \Diamond_{\leqslant t} expr_3.$$

So $\pi^{\#}$ is not winning, which contradicts the assumption. □

The desired relation is now obtained as follows.

Corollary 4.1. (10) implies (9).

Proof. (10) is link-local if β only maps logical variables of agent type to identities from $\mathbb{I}_{fin}$. By symmetry, we can choose a representative β with that property. □

5. Experimental results

In this section, we report on some practical experience of using our approach. We have developed a prototype synthesis tool chain that computes a winning strategy for the coalition in a finite-state system represented in the SMI formalism (Brockmeyer 1999; Wittich 1999).

We have obtained a finite-state system from a given infinite-state system by manually applying the service-specification-based abstraction technique as presented in Section 4.3.

5.1. *Tool Chain*

The first step in the chain is carried out by a conversion tool that transforms the SMI file into a BLIFMV (Kukimoto 1996) file. BLIFMV is a language for describing discrete sequential systems with non-determinism. A BLIFMV model consists of a declaration of:

(1) input signals, which represent the actions of the coalition and the environment;
(2) output signals, which represent the configurations of the system; and
(3) commands, which describe how output signals are set according to the current input signals.

Hence, each BLIFMV model induces a labelled transition system over Boolean variables, where each transition is labelled with an input signal. By associating the controllability of each input signal with either the coalition or the environment, we obtain a finite game structure. Together with a goal state, identified by a dedicated output signal *goal*, we finally obtain a two-player reachability game in which the coalition (that is, the reachability player) tries to reach the goal state independent of the behaviour of the environment (that is, the safety player).

For a reachability game specified by a BLIFMV model and a partitioning of its input signals, the second step in the tool chain is the synthesis of a winning strategy for the coalition (or to report that no winning strategy exists). This task is carried out by a prototype synthesis tool for reachability objectives, which we implemented using the framework of the LTL synthesis tool UNBEAST (Ehlers 2011). The implemented algorithm is based on the standard BDD-based fixpoint algorithm for the symbolic solution of reachability games (see, for example, Alur *et al.* (2005) for a description). The implementation uses the CUDD BDD library (Somenzi 2009) for the symbolic representation and manipulation of state sets, and relies on the automatic on-the-fly reordering heuristics implemented in the CUDD library for finding efficient orderings of the Boolean variables.

The symbolic game solving algorithm computes those states from which the reachability player has a winning strategy – known as the *attractor*. Starting with the immediate winning states (where *goal* is set), in each iteration of the fixpoint algorithm, the intermediate attractor is enlarged by those states from which the reachability player can enforce a state in the attractor in one step. As the set of states is finite, the algorithm is guaranteed to terminate eventually with a fixpoint. There exists a winning strategy for the reachability player if and only if the initial state is contained in the attractor.

When there is a winning strategy, the synthesis tool, as a postprocessing step, extracts a (for the reachability player) deterministic strategy from the attractor. This extraction is guided by a minimisation heuristic that, in the case where several winning successor states can be chosen, computes the set of the successor states that guarantee progress with respect to the reachability winning condition, and then chooses the state with the smallest index with respect to a lexicographical order on the Boolean variables. As we only consider reachability winning objectives, we represent a synthesised strategy as a finite tree, where each node is labelled with a system configuration. The edges of the tree correspond to possible actions (moves) of the environment leading to a new node representing a new configuration. Each internal node is also labelled with a winning move for the coalition (which is supposed to be executed upon entering the node), while each leaf node represents a system configuration in which the reachability goal has been reached.

5.2. *System under consideration*

The system under consideration is the highway entry assistance system as introduced in Section 2. The subject of the strategy synthesis task is a BLIFMV model of a cooperation skeleton as introduced in Section 2.2.3. In the BLIFMV model, the cooperation skeleton is a non-deterministic program, where the execution of a service, as introduced in Section 2.2.1, is controlled by a set of input signals *per car*. Therefore, the synthesised winning strategy is a decision tree that proposes the execution of a certain service by the cars in the coalition according to the behaviour of the environment, including the cars that are not part of the coalition.

The model consists of abstract versions of the continuous acceleration and deceleration services, and the time-bounded change lane service. Technically, the assumptions of the services (see Figure 4 for an example) are incorporated into the winning condition in such a way that whenever a service assumption is violated, the goal state becomes unreachable.

In the model, following Sections 3 and 4.3, both time and space are discrete and finite, that is, we obtain a grid as shown in Figure 15.

We consider traffic situations where three cars are relevant, the ego-car on the entry lane together with one coalition car and one opponent car, which remain on the rightmost regular lane. Therefore, it is sufficient to distinguish two lateral positions. The model is parametrised in the finite number of longitudinal positions.

We always assume that the system starts in a uniform initial configuration, independent of the actual traffic situation that should be analysed. When the model is executed, the first (deterministic) step is to set up a specific traffic situation by assigning concrete initial values to the state bits. Also in the initial configuration, we assume the coalition involving the ego-car is already established. That is, in terms of Section 2.2.3, the ego-car is in mode *Manoeuvre*, the other car in the coalition is in mode *Helper* and the opponent car is in mode *Driver* (see Figure 6). The reachability player controls the input signals that determine the execution of the services in the ego and coalition cars. The safety player controls the input signals that determine the execution of the services in the opponent car.

Table 2. *Running times for synthesising a reachability winning strategy, and strategy sizes for different degrees of precisions of the abstraction*

Grid length	Fixpoint iterations	Strategy size [nodes]	Running time [sec]
10	20	49	17
11	21	89	19
12	21	81	20
13	21	89	34
14	22	137	28
17	23	201	41
22	24	273	106
27	26	361	59
32	28	689	98

The winning condition is that the ego-car reaches the end of the regular lane without any prior violation of the service assumptions.

5.3. *Results*

We executed the tool chain on different instances of the finite-state model corresponding to an application of our abstraction technique to the infinite system with different degrees of precision. The more precise the abstraction, the bigger the grid length, and thus, the larger the number of states. All experiments were conducted on a 2.4 GHz Intel Core Duo computer running Ubuntu 10.04.

In our first experiment, we assumed an initial traffic situation in which a winning strategy for the coalition exists. The results are shown in Table 2. From left to right, the table shows the size of the model in terms of the grid length, the number of fixpoint iterations, the size of the synthesised strategy in terms of nodes in the strategy tree and the running time of our tool. As one might expect, the number of fixpoint iterations required to compute the attractor increases with the size of the model. Similarly, the running times and the sizes of the winning strategies also increase in roughly the same way. Note that the heavy oscillation effect that can be seen in the running times is due to the intricate caching and reordering heuristics, whose activation depends on the structure of the BDDs that arise during the fixpoint construction. Also note that the small irregularity in the strategy size for a grid length of 12 is due to our strategy extraction heuristic, which relies on the syntactic structure of the Boolean game encoding.

In a second experiment, we chose the initial traffic situation in such a way that no winning strategy for the coalition exists (that is, the specification given implicitly by the non-deterministic BLIFMV model is unrealisable). The results are shown in Table 3. The first column shows the size of the model in terms of the grid length, the second column shows the number of fixpoint iterations and the third column shows the running time of our tool. As in the previous experiment, the number of fixpoint iterations increases with the size of the model. However, the fixpoint of the attractor (which does not contain

Table 3. *Running times for establishing the non-existence of a reachability winning strategy for different degrees of precision of the abstraction*

Grid length	Fixpoint iterations	Running time [sec]
10	6	16
11	6	16
12	7	15
13	7	12
14	7	10
17	8	13
22	10	12
27	11	9
32	13	3

the initial state) is reached much earlier. Recall here that the initial traffic situation is not defined through the initial state: it is selected in an initialisation step, and thus has an impact on the transition structure of the model. This is why the maximal number of iterations required to compute the fixpoint can vary according to the initial traffic situation. Interestingly, the running times decrease with the size of the model, even though more fixpoint iterations are required. This is because the BDDs representing the attractors have a much simpler structure than the one produced in the first experiment. Furthermore, this effect becomes stronger as the model becomes bigger.

6. Conclusion

The key application domain targeted in this paper, cooperative driver assistance systems, is generally seen as a key enabler for achieving safe and fuel-efficient driving. We have shown how to formalise both requirements and models for such classes of applications. However, the expressivity demands in the underlying mathematical models go beyond what can be reasonably analysed. So how can we build such applications if we cannot analyse them?

The answer elaborated in this paper exploits the well-established engineering principles of layered, platform-based design, which has allowed us to decompose the overall verification complexity into tractable subproblems, each of which has an intuitive meaning and relevance in the application domain itself:

Service verification. The development and characterisation of components of the platform in terms of the services they offer for re-use in multiple car models is part of product-line development related activities. From the mathematical analysis point of view, these lead to well-defined problems for the verification of hybrid systems. Numerous approaches addressing the scalability of hybrid verification tools have been developed recently (Frehse 2005; Frehse 2006; Frehse 2008; Tabuada *et al.* 2004; Henzinger *et al.* 2001; Damm *et al.* 2011b; Damm *et al.* 2011a). In particular, one of the co-authors established, in joint work with the late Amir Pnueli, a component-based

approach to hybrid system design supporting compositional, incremental verification of both the safety and stability of hybrid systems (Damm *et al.* 2010).

Synthesis of cooperation strategies. The key design question then becomes, assuming these platform services, can we build cooperation strategies on top of them that guarantee timely completion of driver initiated manoeuvres with guaranteed probabilities despite lossy Car2Car communication? We have reduced this problem to a setting where winning strategies, if they exist, can be synthesised automatically. These determine for each car in the coalition and each traffic situation, which service of the platform should be activated.

Factoring out abnormal situations. Our approach allows such synthesis to work on expected behaviours, mimicking a human driver's approach in planning driving tasks, in that dealing with abnormal behaviours (such as aggressive driver behaviours, failing components) is completely encapsulated by the autonomous layer. We see this as a key asset for gaining customer acceptance of cooperative driver assistance systems.

Acknowledgements

The authors would like to thank Rüdiger Ehlers and Boris Wirtz for helpful comments and fruitful discussions.

References

Althoff, M., Stursberg, O. and Buss, M. (2007) Online verification of cognitive car decisions. In: *Intelligent Vehicles Symposium*, IEEE 728–733.

Althoff, M., Stursberg, O. and Buss, M. (2008) Verification of uncertain embedded systems by computing reachable sets based on zonotopes. In: *Proceedings of the 17th IFAC World Congress* 5125–5130.

Alur, R., Courcoubetis, C., Henzinger, T. A. and Ho, P.-H. (1992) Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Grossman, R. L., Nerode, A., Ravn, A. P. and Rischel, H., (eds.) Hybrid Systems. *Springer-Verlag Lecture Notes in Computer Science* **736** 209–229.

Alur, R., Henzinger, T. A. and Kupferman, O. (2002) Alternating-time temporal logic. *Journal of the ACM* **49** (5) 672–713.

Alur, R., Madhusudan, P. and Nam, W. (2005) Symbolic computational techniques for solving games. *Software Tools for Technology Transfer: STTT* **7** (2) 118–128.

Baldessari, R. *et al.* (2007) Car-2-car communication consortium – manifesto.

Baskar, D. (2009) *Traffic Management and Control in Intelligent Vehicle Highway Systems*, Ph.D. thesis, University of Delft.

Bauer, J., Schaefer, I., Toben, T. and Westphal, B. (2006) Specification and verification of dynamic communication systems. In: *Proceedings of Sixth International Conference on Application of Concurrency to System Design: ACSD 2006*, IEEE 189–200.

Bauer, J., Toben, T. and Westphal, B. (2007) Mind the shapes: abstraction refinement via topology invariants. In: Namjoshi, K. S., Yoneda, T., Higashino, T. and Okamura, Y. (eds.) Proceedings of the 5th international conference on automated technology for verification and analysis: ATVA'07. *Springer-Verlag Lecture Notes in Computer Science* **4762** 35–50.

Bellemans, T., De Schutter, B. and De Moor, B. (2006) Model predictive control for ramp metering of motorway traffic: A case study. *Control Engineering Practice* **14** (7) 757–767.

Bianco, A. and de Alfaro, L. (1995) Model checking of probabilistic and nondeterministic systems. In: Thiagarajan, P. (ed.) Foundations of Software Technology and Theoretical Computer Science. *Springer-Verlag Lecture Notes in Computer Science* **1026** 499–513.

Brockmeyer, U. (1999) *Verifikation von Statemate Designs*, Ph.D. thesis, Carl von Ossietzky Universität Oldenburg.

Broucke, M. and Varaiya, P. (1997) The automated highway system: A transportation technology for the 21st century. *Control Engineering Practice* **5** (11) 1583–1590.

Damm, W. and Finkbeiner, B. (2011) Does it pay to extend the perimeter of a world model? In: Proceedings of the 17th International Symposium on Formal Methods.

Damm, W., Dierks, H., Oehlerking, J. and Pnueli, A. (2010) Towards component based design of hybrid systems: Safety and stability. In: Manna, Z. and Peled, D. (eds.) Essays in Memory of Amir Pnueli. *Springer-Verlag Lecture Notes in Computer Science* **6200** 96–143.

Damm, W., Ihlemann, C. and Sofronie-Stokkermans, V. (2011b) Decidability and complexity for the verification of safety properties of reasonable linear hybrid automata. In: Frazzoli, E. and Grosu, R. (eds.) *Proceedings of the 14th international conference on Hybrid systems: computation and control: HSCC '11*, ACM 73–82.

Damm, W. *et al.* (2006a) Automatic verification of hybrid systems with large discrete state space. In: Graf, S. and Zhang, W. (eds.) Automated Technology for Verification and Analysis – Proceedings 4th International Symposium, ATVA 2006. *Springer-Verlag Lecture Notes in Computer Science* **4218** 276–291.

Damm, W. *et al.* (2006b) Automatic verification of hybrid systems with large discrete state space. In: Proceedings of the 4th Symposium on Automated Technology for Verification and Analysis. *Springer-Verlag Lecture Notes in Computer Science* **4218** 276–291.

Damm, W. *et al.* (2007) Exact state set representations in the verification of linear hybrid systems with large discrete state space. In: Namjoshi, K. S., Yoneda, T., Higashino, T. and Okamura, Y. (eds.) Proceedings of the 5th international conference on Automated technology for verification and analysis: ATVA'07. *Springer-Verlag Lecture Notes in Computer Science* **4762** 425–440.

Damm, W. *et al.* (2011a) Exact and fully symbolic verification of linear hybrid automata with large discrete state spaces,. *Science of Computer Programming, Special Issue on Automated Verification of Critical Systems.* **77** (10-11) 1122–1150.

Ehlers, R. (2010) Symbolic bounded synthesis. In: Touili, T., Cook, B. and Jackson, P. (eds.) CAV. *Springer-Verlag Lecture Notes in Computer Science* **6174** 365–379.

Ehlers, R. (2011) Unbeast: Symbolic bounded synthesis. In: Abdulla, P. A. and Leino, K R. M. (eds.) TACAS. *Springer-Verlag Lecture Notes in Computer Science* **6605** 272–275.

Ehlers, R. *et al.* (2011) Dynamic communicating probabilistic timed automata playing games. Reports of SFB/TR 14 AVACS 75, SFB/TR 14 AVACS. ISSN: 1860-9821 (Available at `http://www.avacs.org`.)

Eichler, S. (2009) *Solutions for Scalable Communication and System Security in Vehicular Network Architectures*, Ph.D. thesis, Technische Universität München.

Frehse, G. (2005) Phaver: Algorithmic verification of hybrid systems past hytech. In: Morari, M. and Thiele, L. (eds.) HSCC. *Springer-Verlag Lecture Notes in Computer Science* **3414** 258–273.

Frehse, G. (2006) On timed simulation relations for hybrid systems and compositionality. In: Asarin, E. and Bouyer, P. (eds.) FORMATS. *Springer-Verlag Lecture Notes in Computer Science* **4202** 200–214.

Frehse, G. (2008) Phaver: algorithmic verification of hybrid systems past hytech. *Software Tools for Technology Transfer: STTT* **10** (3) 263–279.

Frese, C. (2010) A comparison of algortihms for planning cooperative motions. Technical Report IES-2010-14, Lehrstuhl für Interaktive Echtzeitsysteme.

Frese, C., Batz, T., Wieser, M. and Beyerer, J. (2008) Life cycle management for cooperative groups of cognitive automobiles in a distributed environment. In: *Intelligent Vehicles Symposium, 2008*, IEEE 1125–1130.

Frese, C. and Beyerer, J. (2010) Planning cooperative motions of cognitive automobiles using tree search algorithms. In: Dillmann, R., Beyerer, J., Hanebeck, U. D. and Schultz, T. (eds.) KI 2010: Advances in Artificial Intelligence. *Springer-Verlag Lecture Notes in Computer Science* **6359** 91–98.

Hansson, H. and Jonsson, B. (1994) A logic for reasoning about time and reliability. *Formal Aspects of Computing* **6** 102–111.

Henzinger, T. A., Ho, P.-H. and Wong-toi, H. (1996) Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control* **43** 225–238.

Henzinger, T. A., Minea, M. and Prabhu, V. S. (2001) Assume-guarantee reasoning for hierarchical hybrid systems. In: Benedetto, M D. D. and Sangiovanni-Vincentelli, A. L. (eds.) HSCC. *Springer-Verlag Lecture Notes in Computer Science* **2034** 275–290.

Henzinger, T. A. and Prabhu, V. S. (2006) Timed alternating-time temporal logic. In: Asarin, E. and Bouyer, P. (eds.) FORMATS. *Springer-Verlag Lecture Notes in Computer Science* **4202** 1–17.

Hsu, A., Eskafi, F., Sachs, S. and Varaiya, P. (1991) The design of platoon maneuver protocols for IVHS. PATH Report UCB-ITS-PRR-91-6, University of California.

Kukimoto, Y. (1996) Blif-mv. Technical report, The VIS group, University of California, Berkeley.

Lee, K. (2008) Advanced research for integrated active transportation system. In: *The National Workshop on New Research Directions for High Confidence Transportation Cyber Physical Systems*, Washington, D.C., U.S.A.

Lygeros, J. and Lynch, N. A. (1998) Strings of vehicles: Modeling and safety conditions. In: Henzinger, T. A. and Sastry, S. (eds.) Proceedings Hybrid Systems: Computation and Control, First International Workshop, HSCC'98. *Springer-Verlag Lecture Notes in Computer Science* **1386** 273–288.

Lygeros, J., Tomlin, C. and Sastry, S. (1997) Multiobjective hybrid controller synthesis. In: Maler, O. (ed.) HART. *Springer-Verlag Lecture Notes in Computer Science* **1201** 109–123.

Pnueli, A. and Rosner, R. (1989) On the synthesis of a reactive module. In: *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages: POPL'89* 179–190.

Richards, A. and How, J. (2002) Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In: *Proceedings of the American Control Conference, 2002*, volume 3 1936–1941.

Schewe, S. and Finkbeiner, B. (2007) Bounded synthesis. In: Namjoshi, K. S., Yoneda, T., Higashino, T. and Okamura, Y. (eds.) Proceedings of the 5th international conference on Automated technology for verification and analysis: ATVA'07. *Springer-Verlag Lecture Notes in Computer Science* **4762** 474–488.

Silva, B., Richeson, K., Krogh, B. and Chutinan, A. (2000) Modeling and verifying hybrid dynamic systems using CheckMate. In: *Proceedings Conference on Automation of Mixed Processes: Hybrid Dynamic Systems* 323–328.

Somenzi, F. (2009) CUDD: CU Decision Diagram package release 2.4.2.

Tabuada, P., Pappas, G. J. and Lima, P. U. (2004) Compositional abstractions of hybrid control systems. *Discrete Event Dynamic Systems* **14** (2) 203–238.

Tomlin, C., Lygeros, J. and Sastry, S. (1998a) Synthesizing controllers for nonlinear hybrid systems. In: Henzinger, T. A. and Sastry, S. (eds.) Proceedings Hybrid Systems: Computation and Control, First International Workshop, HSCC'98. *Springer-Verlag Lecture Notes in Computer Science* **1386** 360–373.

Tomlin, C., Pappas, G., Kosecka, J., Lygeros, J. and Sastry, S. (1998b) Advanced air traffic automation: A case study in distributed decentralized control. In: Siciliano, B. and Valavanis, K. (eds.) Control Problems in Robotics and Automation. *Springer-Verlag Lecture Notes in Control and Information Sciences* **230** 261–295.

Tomlin, C., Pappas, G. J., Lygeros, J., Godbole, D. N. and Sastry, S. (1996) Hybrid control models of next generarion air traffic management. In: Antsaklis, P. J., Kohn, W., Nerode, A. and Sastry, S. (eds.) Hybrid Systems. *Springer-Verlag Lecture Notes in Computer Science* **1273** 378–404.

Wachter, B. and Westphal, B. (2007) The spotlight principle. On combining process-summarising state abstractions. In: Cook, B. and Podelski, A. (eds.) VMCAI. *Springer-Verlag Lecture Notes in Computer Science* **4349** 182–198.

Westphal, B. (2008) *Specification and Verification of Dynamic Topology Systems*, Ph.D. thesis, Carl von Ossietzky Universität Oldenburg.

Wirtz, B., Strazny, T., Rakow, A. and Rakow, J. (2011) A lane change assistance system: Cooperation and hybrid control. Reports of SFB/TR 14 AVACS SFB/TR 14 AVACS.

Wittich, G. (1999) *Ein problemorientierter Ansatz zum Nachweis von Realzeiteigenschaften eingebetteter Systeme*, Ph.D. thesis, Carl von Ossietzky Universität Oldenburg.