BDI Model of Connected and Autonomous Vehicles

Inga Rüb[1] and Barbara Dunin-Kęplicz[1]

University of Warsaw

Abstract. It is expected that connected and autonomous vehicles (CAVs) will become a regular mean of transportation by the year 2022. To fully leverage the potential of this new technology it is necessary to equip such cars with efficient algorithms permitting them to drive in a safe and possibly optimal manner. Thereby we aim to design and implement tools for evaluation of strategies for driving and interactions in various settings. In this paper we present results of the first stage of our bigger research program on a simulation framework of CAVs.

A search for balance between huge complexity of representing of real-world CAVs and comprehensibility of the solution led us to the paradigm of multi-agent systems. Beliefs-Desires-Intentions (BDI) systems offer useful abstractions for activities of a single self-driving car and a whole systems of such vehicles. Indeed, the BDI framework helps to combine two distinct natures of a self-driving car: its reactiveness and proactiveness. Moreover, modularity of the resulting architectures for an individual CAV and urban traffic induced by these cars makes the design easily extensible and resilient. We also consider technical aspects of implementation for a regular desktop computer on a large scale of hundreds and thousands vehicles. Our prototype verifies feasibility of this concept.

1 Modelling CAVs

Representatives of biggest automotive manufacturers predict that connected and autonomous vehicles (CAVs) will be fully deployed in five years [28,19,27,9]. Such self-driving cars, trucks and buses have been heralded as a solution to many problems in transportation [7,12,20]: from car accidents, through traffic jams, to insufficient number of parking slots. However, in order to observe the expected improvements in the nearest future, this promising technology needs to be carefully evaluated, for example with the use of a simulation. Hence, our ultimate goal is creating an efficient and flexible framework suitable for simulating CAVs, in particular their interactions and driving strategies. For this purpose we need to model traffic induced by CAVs on a detailed, microscopic level.

Due to huge complexity of real-world technologies applied to a CAV, we do not aim to consider the engineering details. Rather, we represent CAVs as entities capable of getting from a location A to another location B along a given path (the route is determined a priori). They possibly differ in size or certain properties but implement the same algorithms and protocols. The vehicles should be able to exchange data with the infrastructure and all neighbours within a distance not exceeding a specified maximum while keeping their driving **safe** and possibly **optimal**.

As far as **safety** is concerned, in the ideal case, CAVs should prevent all collisions that are physically avoidable. The drive is considered to be **optimal** if there are no unnecessary delays. Thus, the agent should head towards its destination with the maximal allowed speed and do not stop or slow down unless it is compelled to do so. When informed of unexpected situations that make the current route impassable, it should ask the navigation system for an alternative path. Other ways of improving driving are optional and involve coordinated interactions with the environment.

To explore this kind of optimizations, we focus on traffic conditions of **modern urban and suburban areas**: district-size territories which are densely cut by streets, jammed with cars in rush hours. The environment includes just the basic elements of road infrastructure: roads, traffic lights and a traffic management system. We do not pay special attention to non-CAV traffic members, though their presence manifests itself in unexpected disturbances. Vehicle sensors are assumed to function perfectly as long as they are responsive, whereas possible malfunctions contribute to the already significant **non-determinism**.

To obey conditions and requirements agents need to behave cautiously: keep a proper distance to a car in front and not overestimate the limited precision of the collected data. There is hardly a possibility for a single CAV to **improve its performance** in terms of driving time. This is why interactions become essential to optimizing the system as a whole and CAVs need to communicate not only to learn about the environment but also to influence it. In fact, their intelligence is sufficient for self-organization while complex interactions are essential for their behaviour.

Hence, to model CAVs, an appropriate paradigm of MAS should be used: the one that combines goal-directed reasoning with event-driven behaviour (for a discussion see [30] and references there). Our research leads us to the classic and well-described BDI concept. While referring to beliefs, desires and intentions, the paradigm seems to be the one that satisfies the requirements (for a discussion see [5] and references there). The BDI framework introduces useful abstractions for most of CAV activities, like collecting data or planning actions. By mimicing human-like behaviour, BDI allows us to represent complex reasoning in a comprehensible way: this would allow end-users to analyze and modify decision algorithms. However, to create efficient simulation tools we need to modify slightly the classic understanding of the BDI notions [26] and create a tailor-made version of this framework.

Here we present our attempt to design BDI architectures for modelling both: a single self-driving car and a whole system of such vehicles. The paper is organized as follows. In Section 2 more aspects of BDI are discussed in the context of CAVs. Then we describe the architectures: for a single self-driving car in Section 3 and for the multi-agent system in Section 4. Section 5 reviews the proposed architecture and outlines our future work.

2 The BDI approach

Among advantages of the BDI framework, especially compelling is the ability of agents to differentiate between goals that **should** be pursued in general and obligations that **must** be fulfilled at a given moment [29]. Such categorization helps to express motivations for actions planned by an agent and allows CAVs to coordinate their behaviour in accordance with global priorities, e.g.: a vehicle cannot optimize its route at the cost of safety of other self-driving cars.

2.1 Interactions

The need for coexistence between individual CAVs implies two basic rules that govern the simulated interactions:

- a CAV **should** try to optimize its travel time;
- a CAV **must** help others in their attempt to drive faster only if there is no conflict of interests.

A good example of interactions that rely on these rules is *vehicle platooning* [4]: while almost all of CAVs travelling as a group derive significant advantages, there is no benefit for the leader. Even though, if leading a platoon causes no losses, a CAV is required to agree whenever asked for the favour.

In other situations, like crossing an intersection with no signal controls, self-driving cars are forced to reach an agreement even though it may slow down at least some of the involved agents. Yet, it is possible to solve such conflicts of interests, e.g. by ordering inconsistent requests in accordance with timestamps or introducing more sophisticated mechanisms (like efficient scheduling [14] or negotiations [3]). Reaching a consensus applies also to planning optimal routes: to minimize congestion or travel time, agents can try to avoid traffic jams by changing their paths voluntarily [31].

The above examples prove that modelled CAVs should be able to constitute a co-operating system with no central management. To choose the right solution together, CAVs have to take a comprehensive view on the situation, specifically, they need to be aware of individual commitments and long-term goals. Hence, our model of a self-driving car embraces abstractions of **beliefs**, **desires** and **intentions**, whose meaning underwent slight changes to better suit the context of CAVs.

2.2 Beliefs

Since we assume correctness of collected data, we deal with knowledge rather than beliefs. At the same time, the access to information is limited: vehicles do not receive regular messages about distant cars or areas, nor explore them remotely. The only exception is an emergency situation, when a CAV gets alerted to a road becoming impassable. For this reason the procedures of driving are designed to run algorithms on restricted input, whereas data of a CAV can be categorized in the following way:

- knowledge the car is endowed with at the very beginning of its travel, which include information about the map, kinematics and other mechanisms of the environment; for example, an agent understands the way signal control functions and is able to compute the appropriate acceleration to reach a given speed;
- regularly updated values of the environment, such as: a phase of traffic lights, other agents within the range and their parameters;
- alerts about non-deterministic events, e.g.: a car accident or an obstacle;
- self-awareness about the physical and mental state of itself, e.g.: the own length and width, its current position, selected actions and – desires.

2.3 Desires

Desires refer to an agent's pro-attitudes: everything that a CAV wants to achieve or maintain. We divide them into ***basic*** and ***meta desires***. *Basic desires* are always valid and bring about short-term modifications of the physical state. Their opposite, *meta desires* are triggered by events and make a long-term impact on the mental state.

	basic desires	***meta desires***
are	always valid	triggered by events
influence	physical state	mental state
have	short-term effects	long-term effects

Table 1: different kinds of desires.

Basic desires propel a vehicle to go along the given path in a safe and optimal manner:

– **drive fast** – do not slow down for no reason;
– **follow the route** – do not change the path for no reason;
– **obey the traffic rules** – do not violate the Highway Code for no reason;
– **do not crash** – avoid any obstacles.

As crucial for the agent's survival and mission, they need to be taken into account for each action of a CAV. For example, if a vehicle intends to turn left it needs to verify whether: it would be better to accelerate first, this decision is in accordance with the route, the phase of traffic lights allows to go in this particular direction and, finally, there is no risk of a collision. However, the reoccurring phrase 'for no reason' signalizes that the agent may decide to pursue all its desires but the last one only to some extent. Specifically, the order of itemization reflects priorities: the wish to drive fast is the least important whereas avoiding collisions is absolutely crucial and can invalidate preceding desires. While the priorities are fixed, a particular basic desire may undergo changes dictated by a meta desire.

Meta desires are supposed to make the CAV reason and react about external stimuli in a different way:

– **replan the route** – if anything unexpected happens;
– **cooperate with platoons** – if there is such a possibility;
– **coordinate traffic** – if car approaches an intersection.

Again, desires are ordered deliberately in a linear manner. E.g., 'replan the route' should be considered before joining a group – otherwise it might be necessary to cancel the participation almost immediately. Also, as long as a CAV follows the platoon leader it does not need to coordinate traffic at any intersections. However, in contrast to basic desires, meta desires can be considered in a different sequence with no other side-effects than decrease in efficiency. In contrast to basic desires, meta desires do not influence the agent's behaviour until certain events take place. When triggered, they result in long-term modification of an agent: coordination of traffic influences beliefs about the environment; joining a platoon replaces basic desire ‘follow the route’ with a desire ‘stay in platoon’; finally, replanning the route changes the vehicle’s *goals*.

In our model ‘**goals**’ are defined as the vehicle’s achievement (not maintenance) tasks that are assumed to be consistent, valid and specified in advance. In practice, goals constitute the reference points for desire ‘follow the route’: they determine sequence of physical parameters, like position, speed and orientation, that need to be obtained by the agent.

A CAV’s response to a mixture of various constraints and obligations is modelled as its *intention*: commitment to an action planned along with the desires and their priorities. Here our approach diverges again from the standard model. We propose the design, where a commitment results from the practical reasoning that involves not one but all basic desires. The process is performed by the core of the agent architecture.

3 Agent architecture

The architecture of a CAV defines modules that manage mental and physical aspects of a self-driving car as well as their interdependencies. For our application we propose the scheme presented in Figure 1.

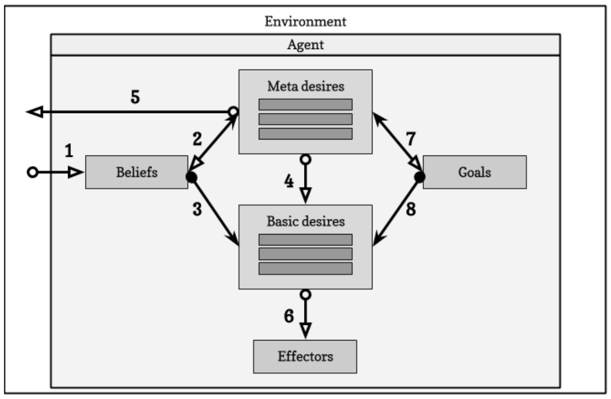


Fig. 1: the agent architecture; an arrow from A to B informs that B is directly influenced by some changes in A: if B uses A an arrowhead is filled; if B is modified by A an arrowhead is empty.

3.1 Knowledge

The base of the reasoning process is the agent’s knowledge: findings about the environment and the internal state. The former are represented as a nested hierarchy of interacting instances, which are compliant with some specific *schemas*. The other kind of knowledge refers to the way in which various mechanisms work. This information is encoded within *algorithmic procedures* that allow to predict consequences of potential actions and future events, e.g. if an agent *knows* how speed changes due to acceleration, it is able to compute its braking distance. Concerning a CAV's internal state, its parameters have a finite and well-defined accuracy.

For the sake of example, Algorithm 1 presents an inbuilt method which is used by a car to check, if it should slow down and thereby increase the distance to the neighbour in front. The function takes the CAV instance as its only parameter and returns a boolean value. It is allowed to access all of the agent's data, just like in line 2, where a call to another procedure provides information about the vehicle's speed. In order to find the shortest breaking distance for an arbitrary deceleration and the given velocity value, the algorithm refers to known facts – kinematics laws (line 3). Other sources of beliefs, sensors and communication, are used for retrieving the model of the car in front (line 4). Next, in line 5, an agent is able to verify relevant conditions and match appropriate instructions with the current state of the environment: unless there is a car in front, it is not sensible to consider decelerating any further. In the opposite case, the procedure accesses data contained in the model of the other CAV, computes the braking distance for the speed value and gets information concerning the distance between itself and its neighbour.

Algorithm 1 An examplary algorithm used by a modelled CAV

```
Input: $self$ – the CAV that runs the computations
Output: true if it should slow down or false otherwise
 1: procedure $should\_the\_car\_slow\_down$
 2:     $my\_speed \leftarrow self.$GETCURRENTSPEED()
 3:     $my\_braking\_distance \leftarrow$ COMPUTEBRAKINGDISTANCE($my\_speed$)
 4:     $car\_in\_front \leftarrow self.$GETCARINFRONT()
 5:     if $car\_in\_front = $ NULL then
 6:         return false
 7:     end if
 8:     $their\_speed \leftarrow car\_in\_front.$GETCURRENTSPEED()
 9:     $their\_braking\_distance \leftarrow$ COMPUTEBRAKINGDISTANCE($their\_speed$)
10:     $distance\_between \leftarrow self.$GETDISTANCETO($car\_in\_front$)
11:     return $my\_braking\_distance > their\_braking\_distance + distance\_between$
12: end procedure
```

3.2 Goals

Goals are defined at the very beginning of the vehicle's 'lifetime', but may be changed adequately to the situation later on. A single goal **determines a physical state** the CAV should achieve: it requests the agent to get to a certain place and be oriented towards a given direction. Additionally, the vehicle might be asked to acquire particular speed or acceleration.

Since goals describe the CAV's route, an agent needs to purse them in a fixed order. Also, the tasks are one-off: as soon as any of them is fulfilled, it becomes no longer valid. Thus, the most appropriate structure for organizing them is a **stack**: a CAV always strives to complete a target at the top. It is assumed that subsequent tasks are consistent, but future goals can be altered by meta desires (arrow 7 in Figure 1).

3.3 Reasoning pipelines

Basic and meta desires should be separated. First, they generate behaviours of different nature and independently interact with other modules. Secondly, they cannot share computational resources: costly deliberation must not stop the agent from reacting to rapid changes in its immediate environment.

Basic reasoning pipeline The basic-reasoning module is divided into separate *units* of code that correspond to individual basic desires and create a plan of actions for the nearest simulation step. The plan is modified and confirmed in its final version by each unit. Since some basic desires are more important than other, the whole process is organized as a pipeline. It is performed **sequentially** and in a **fixed order** (the central controller calls units one by one): the initial plan of actions is first modified by low-priority layers and then it is redesigned and acknowledged by layers of higher priorities. That way, consecutive modifications to a planned commitment can be automatically applied (they are more important than a currently designed plan): contrary to similar subsumption architecture [2], there is no need for interfaces nor communication between the layers.

The basic-reasoning process is presented in Figure 2. It starts with creating an *initial plan* of actions, which orders the car to move forward with its current configuration. Afterwards, this neutral proposal is altered in accordance with basic desires, where units of high priority can modify or reject low-priority suggestions. For example, a self-driving vehicle may modify its route to avoid a car crash. The resulting plan is executed by the vehicle's effectors (arrows 6 in Figure 1).

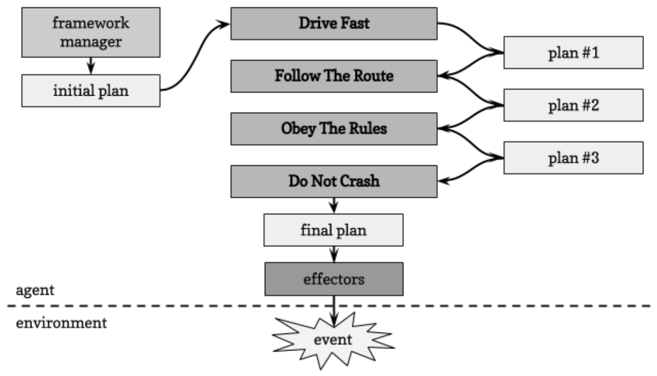


Fig. 2: the pipeline constituted by basic desires.

We assume that a CAV is ready to perform basic reasoning always when it is necessary and the process takes a fixed amount of time. Things are different in case of meta desires.

Meta reasoning pipeline Whereas basic desires make the agent react to the environment, meta desires propel vehicles to **shape the environment** (arrow 5 in Figure 1).

Meta desires involve time-consuming computations (e.g. when replanning the route) or communication with other cars (e.g. when creating a platoon). Initially, we designed each of them to work in a separate thread (MT) and spawn an additional worker thread (WT) if needed, not to block or slow down other modules of the system. Since meta desires can impact decisions of each other (e.g: replanning the route may open opportunities to join a platoon), their corresponding units exchange messages via the framework controller. Figure 3 presents the resulting pipeline.

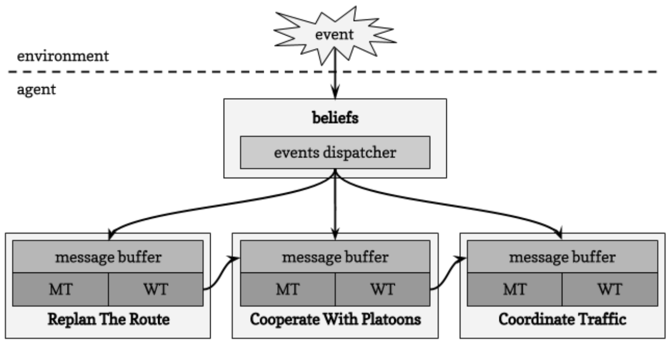


Fig. 3: the pipeline constituted by meta desires.

4 System architecture

Apart from the agent architecture another mechanism needs to be designed: the one that manages the simulation of hundreds of CAVs as a whole. Here we need to take into account limited computing resources of a desktop computer.

4.1 Single-thread agents

We begin with determining the potential number of threads needed for the simulation. Mainly due to separation of meta-desire units, hundreds of CAVs result in at least thousands of threads and, thereby, in significant overhead for switching the context as well as troublesome synchronization. To limit the number of threads, we decide not to handle meta desires in parallel, but rather to perform both kinds of reasoning in the same single thread. Now the meta-reasoning pipeline is similar to its basic counterpart: a CAV regularly considers meta desires in accordance with their linear order.

To apply this solution, vehicle needs to initialize the pipeline with a sufficient frequency to be able to react in time. For the same reason, algorithms performed formerly by main threads of meta desires cannot consume too much of resources. Advanced computations (e.g. finding a common path with a platoon's one) should be offloaded to a **thread pool**: a group of threads spawned by the program at the start. An optimal pool's size can be determined, based on the probability distribution of task requests [21].

To simplify the agent architecture further, communication between vehicles and infrastructure is not handled in a parallel thread. Thanks to determinism of protocols, after CAVs publish all indispensable data, each car is able to find the outcome of potential communication without exchanging messages. For example, an agent intending to join the platoon can directly check whether the number of grouped CAVs allows him to join the platoon: in contrast to the real-world situations, it does not need to ask for acknowledgements. The resulting practical reasoning is presented as Algorithm 2. Algorithm 3 shows an examplary procedure `check conditions` of a unit handling meta desire *replan the route*. The procedure, given information about the environment (`I`) and messages from preceding meta desires (`M`), decides on modifications to a vehicle route and switches between basic desires: *wander around* and *follow the route*.

Algorithm 2 Practical reasoning of a CAV

```
procedure CREATE A PLAN
    for each item $D$ in $meta\ desires$ do
        $M \leftarrow D.read\ messages\ from\ the\ buffer$
        $I \leftarrow D.learn\ about\ the\ environment$
        $triggered \leftarrow D.check\ conditions(M, I)$
        if $triggered$ then
            $D.compute\ and\ prepare\ optimizations$
        end if
        $D.adjust\ basic\ desires$                              ▷ optionally
    end for
    $P \leftarrow empty\ plan$                                  ▷ just keep going forward
    for each item $D$ in $basic\ desires$ do
        $I \leftarrow D.learn\ about\ the\ environment$
        $P \leftarrow D.adjust\ the\ plan(P, I)$
    end for
    return $P$
end procedure
```

4.2 Single-thread system

The altered agent architecture is supposed to be a compromise between technology restrictions and ability to mimic the world accurately enough. Single-thread agents introduce an additional difficulty: simulated CAVs need to provide the information that is necessary for others in a synchronized way. In practice each CAV has to know at least the plan prepared by the car in front (only then it can adjust its speed and avoid a rear-end collision). It is not straightforward how to reconcile this condition with other properties of the simulation:

1. CAVs run their computations in parallel,
2. CAVs can ‘predict’ future t_p seconds of their local environment,

Thereby, we introduce new variables:

– t_c – time to communicate and get data,
– t_r – time to reason about actions for next t_p seconds.

Algorithm 3 Unit replan-the-route

```
procedure CHECK CONDITIONS(M, I)
    if $any\ route\ became\ passable(I)$ then
        $triggered \leftarrow$ true; $update\ beliefs$
    end if
    if $WT\ has\ finished\ its\ work$ then
        if $there\ is\ no\ route$ then
            $triggered \leftarrow$ true; $wander\ around \leftarrow$ true
            return
        end if
        if $should\ apply\ the\ new\ route(I, M)$ then
            $save\ the\ old\ route\ as\ fallback\ and\ apply\ the\ new\ one$
        end if
    end if
    if $current\ goal\ cannot\ be\ obtained(I, M)$ then
        if $no\ fallback\ route\ is\ applicable\ now$ then
            $triggered \leftarrow$ true; $wander\ around \leftarrow$ true; return
        end if
        $apply\ the\ fallback\ route$
    end if
    if $future\ goals\ cannot\ be\ obtained(I, M)$ then
        $triggered \leftarrow$ true; return
    end if
    $follow\ the\ route \leftarrow$ true                ▷ do not wander around
end procedure
```

Values of t_c and t_r depend on t_p: the more predictions to be made and the more actions to design, the greater are t_c and t_r. For purposes of the following examples, let the variables be assigned some fixed values: $t_p = 6\,\mathrm{s}$, $t_c = 1\,\mathrm{s}$ and $t_r = 1\,\mathrm{s}$.

Single-agent scenario Given these constraints, we consider a situation, where a single vehicle drives on an empty road. It is able to deduce what the environment will look like during next $6\,\mathrm{s}$ and needs $1\,\mathrm{s}$ to plan its actions. As soon as it finishes the reasoning, the resulting plan is valid for $5\,\mathrm{s}$. The optimal strategy for a CAV is presented in Figure 4. Light grey color designates intervals for which the vehicle has already designed actions. Time spent on executing plans is marked with the dark shadow, whereas periods for which there were/are no commitments are left blank. A red hourglass represents the reasoning.

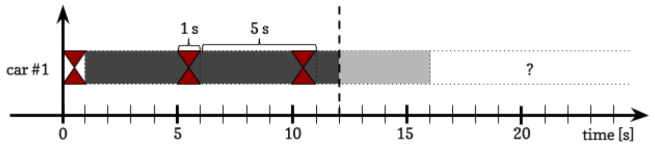


Fig. 4: after $12\,\mathrm{s}$ the agent realized 1/5 of the current plan and has commitments for next $4\,\mathrm{s}$.

During the first second the car remains still (it has no plan of what to do) and reasons about the future. For the following interval of $4\,\mathrm{s}$ it executes planned actions. In the 6th second of its drive the vehicle continues to perform designed actions but also, simultaneously, reasons about the forthcoming $5\,\mathrm{s}$. This way the CAV can drive smoothly while the reasoning is performed as rarely as possible.

Multiagent scenario In the multiagent scenario there are n vehicles on a one-lane road that passed the traffic lights and one that has to wait for the green light (the car m), as shown in Figure 5. Each of them plans its actions using the same basic-reasoning. However, in this case, the single-car strategy cannot be applied: CAVs function concurrently but they are not fully independent. For each $1 < i \leq n$, commitments of car #$(i-1)$ are essential for decisions of car #i. Thus, a vehicle needs to perform the reasoning more often than it would for its own purpose. Figure 6 illustrates what happens if a CAV #1 applies the single-agent scenario.

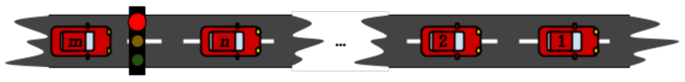


Fig. 5: a multiagent scenario on a one-lane road.

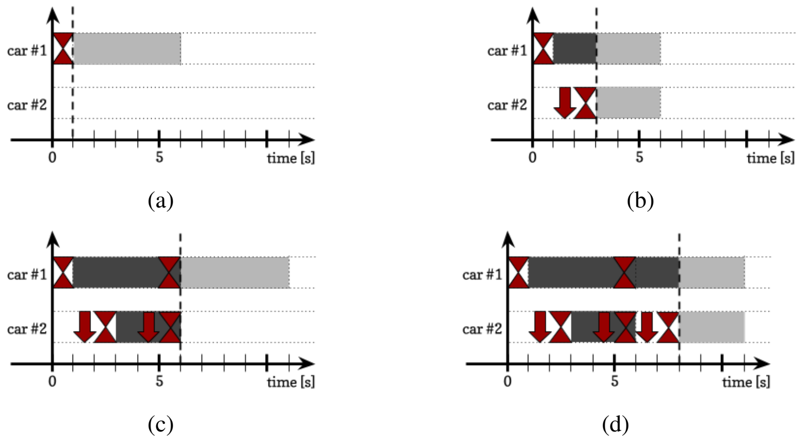
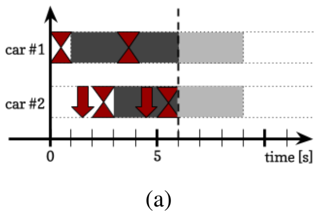


Fig. 6: the single-agent strategy in the multiagent scenario.

Like previously, car #1 starts with no plan and during the first second it prepares actions for the future (6a). As soon as car #1 finishes the planning, it sends required data to car #2. This process takes time t_c and is symbolised by a down arrow. After $3\,\mathrm{s}$, both CAVs know what actions to take during the following interval of $3\,\mathrm{s}$ (6b). Car #2 is not able to prepare a full-length plan, because the neighbour in front has not yet made any decisions regarding the 7th and 8th second. When car #1 in front starts to consider the next period (6c), it is too late for car #2 to prepare for the following interval of $2\,\mathrm{s}$. After $8\,\mathrm{s}$, the situation repeats itself and both vehicles again follow their plans for the three-second future (6d).

Frequency of reasoning Unlike in the discussed situation, agents have to be able to design their actions in advance. Hence, CAVs need to perform the reasoning whenever neighbours ask for information. We call this *Reasoning On Demand* and present it in Figure 7 (7a and 7b correspond to 6c and 6d respectively).



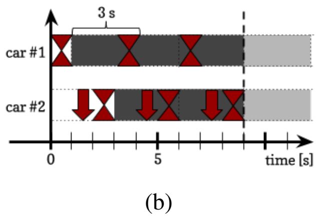


Fig. 7: car #1 creates plans incrementally.

After first $3\,\mathrm{s}$ car #2 requests data necessary for practical reasoning. Car #1 has already defined the commitments for the next $2\,\mathrm{s}$ but, to provide as much data as possible, it plans additional actions for further $3\,\mathrm{s}$. When the car at the back obtains the data both agents have their plans ready until the 9th second (7a). Then, another cycle begins: car #1 has to provide the neighbours with necessary information. Due to this arrangement, the CAV in front plans actions for shorter intervals but more frequently – every $3\,\mathrm{s}$ (Subfigure 7b). For n vehicles in the line, the period T, with which agents need to perform the reasoning equals $t_p - nt_r - (n-1)t_c$. Also, T must satisfy the condition $t_r < T$, otherwise CAVs are not able to prepare their plans on time.

Our solution For simplifcation purpose, *Reasoning On Demand* can be reduced to the mechanism presented in Figure 8. Here, the simulation time 'freezes' while agents

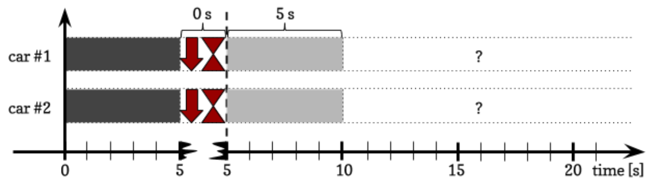


Fig. 8: Simplified *Reasoning On Demand*, $T = 5\,\mathrm{s}$

collect necessary information and plan what to do for the next T seconds. After the normal time flow is restored, CAVs execute their actions. That way, the scenario where they exchange messages and drive simultaneously is imitated with just one difference. *Reasoning On Demand* requires a car #i to prepare a plan based on current observations and commitments of a car #$(i-1)$ adopted from t_c to $(t_c + t_r)$ seconds ago, whereas in the proposed arrangement CAVs reason about the up-to-date environment and **equally new plans** of the car in front. The discrepancy matters only in non-deterministic world, where a plan prepared later takes into account reactions to recent random changes and the sooner one does not. For example, the car at the back is informed in advance that the car in front will brake due to an unexpected event. Hence, during the reasoning phase, the CAV at the back pretends not know about non-deterministic situations before they actually take place.

Consequently, the modified *Reasoning On Demand* is consistent with the original version and permits to run computations for all agents in a **sequential manner**, resulting in:

– simplicity of the mechanism: an easier way to implement, analyse and debug the program;
– repeatable behaviour of agents, which is not altered by overheads and management of multiple threads;
– a guarantee that vehicles get necessary data with no delays: if CAVs perform their tasks in parallel their threads interleave in accordance to an uncontrolled schedule
– ability to control the value of T: one of the goals for our future research is determining the optimal period of reasoning and its dependency on other parameters, e.g. maximal allowed speed.

On the downside, there is a concern, that the single-thread solution does not utilize multicore processors. To address this issue we propose **splitting the computations** into parts that involve mutually independent areas of the simulation environment. It is possible thanks to signal controls that stop the traffic along a given direction, as illustrated by Figure 5: car #m, does not need any data from car #n if the red phase of traffic lights lasts for at least another T seconds.

The discussed improvements result in a powerful and efficient architecture for the simulation framework, presented in Figure 9. Interactions between particular modules are marked with numbered arrows (similarly as in case of Figure 1):

1. A controller manages meta reasoning within a single independent area. It keeps agents organized in a line and initializes the process sequentially. If a CAV awaits information from another vehicle it is pushed at the back of the queue and resumes its computations later.
2. The second controller operates in an analogical way but is responsible for basic reasoning.

3–4. Time-consuming algorithms used for meta reasoning are offloaded to a pool of worker threads, which is shared between all the agents.

5–6. Vehicles are able to influence the environment directly, e.g. by coordinating the traffic at intersections with no signal control.

7. CAVs know their own state. These data can be accessed directly by other agents via imitated communication.

8–9. Beliefs representing the public knowledge are models and procedures common for all cars. Information that a CAV is allowed to retrieve is limited in distance.
10. Every change in the environment is immediately recorded and expressed as available data.

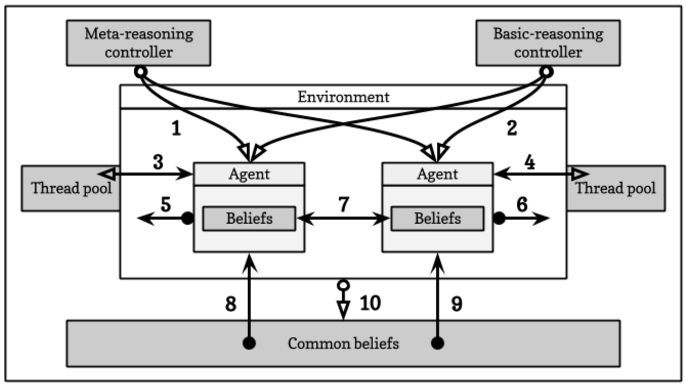


Fig. 9: the architecture of the whole system.

5 Discussion

Even though BDI has been recently considered in the context of CAVs its potential has not been utilized: existing architectures are either solely reactive [11] or focus on just a single kind of interactions [17]. More general solutions that combine reactiveness and proactivity, like [16,23], or classical MAS, like [24,6], are not oriented towards unique properties of self-driving cars and their environment. To our best knowledge validated models of CAVs like [13,18,25] do not incorporate all of the features characteristic for our design. However, the proposed architecture derives inspiration from existing solutions [2,10,8,22,1,15] and incorporates some of their suitable ideas.

A unique feature of our system is comprehensibility of procedures that implement mechanisms of driving and dependencies between them. This enables end-users to modify algorithms of CAV steering and the configuration easily. We created a prototype version of the simulation framework which proves feasibility of our design. The program can be used to test various techniques that coordinate traffic at intersections and to find out how making a road impassable influences total time travelled by the CAVs. We aim to enhance the tool with an additional module for finding routes that satisfy specific requirements, investigate how agents of different attitudes are able to coexist and verify if gamification systems could be an efficient solution to traffic jams in cities.

Concerning the expected future of our simulation, most important properties of the proposed architecture include:

- **extensibility** – assured by the system modularity, especially in case of reasoning pipelines, where manoeuvres are implemented by separate units;
- **resilience** – guaranteed in case of malfunction of a desire-related unit (units are not aware of each other); also, if no meta procedures are executed, the basic reasoning is capable of driving the car on its own;
- **scalability** – is problematic, mainly due to the sequential reasoning process and micro-scale precision.

Currently our prototype is meant for **small-scale traffic**: the maximal considered area should correspond to an average district in a big city with a thousand of travelling vehicles at most. Such scale of computations already allows us to observe major traffic phenomena (e.g. creation of traffic jams, efficiency of coordinated intersections) and can be supported by a single PC.

To run the simulation for a larger scale and keep all its advantages intact, we need to leverage high reasoning costs, for example by parallelizing the computations as suggested in the previous section. If, in the following stages of our research program, there is a need to increase efficiency of our framework further, the main property of the model that should be preserved at all costs is its BDI basis: these abstractions and mechanisms allowed us to simulate variety of interactions between CAVs, including paradigmatic activities like cooperation, coordination and communication, while enforcing modularity and simplicity of the system.

References

1. Behere, S., Torngren, M.: A functional architecture for autonomous driving. In: 2015 First International Workshop on Automotive Software Architecture (WASA). pp. 3–10 (May 2015)
2. Brooks, R.: A robust layered control system for a mobile robot **2**, 14 – 23 (04 1986)
3. de Campos, G.R., Falcone, P., Sjöberg, J.: Autonomous cooperative driving: A velocity-based negotiation approach for intersection crossing. In: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013). pp. 1456–1461 (Oct 2013)
4. Davila, A., Nombela, M.: Platooning - safe and eco-friendly mobility (04 2012)
5. Dunin-Kęplicz, B., Verbrugge, R.: Teamwork in multi-agent systems: A formal approach (may 2010)
6. E. Pollack, M., Israel, D., Bratman, M.: Towards an architecture for resource-bounded agents (1987)
7. Fagnant, D.J., Kockelman, K.: Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. Transportation Research Part A: Policy and Practice **77**(Supplement C), 167 – 181 (2015)
8. Ferguson, I.A.: Touring machines: autonomous agents with attitudes. Computer **25**(5), 51–55 (May 1992)
9. Frank, S.: Die zukunft nach dem abgas-skandal (Apr 2016), http://www.focus.de/finanzen/news/wirtschaft-und-geld-die-zukunft-nach-dem-abgas-skandal_id_5457885.html
10. Georgeff, M.P., Ingrand, F.F.: Decision-making in an embedded reasoning system. In: Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 2. pp. 972–978. IJCAI'89 (1989)
11. Gora, P., Rüb, I.: Traffic models for self-driving connected cars. Transportation Research Procedia **14**(Supplement C), 2207 – 2216 (2016), transport Research Arena TRA2016

12. Gruel, W., Stanford, J.M.: Assessing the long-term effects of autonomous vehicles: A speculative approach. Transportation Research Procedia **13**(Supplement C), 18 – 29 (2016), towards future innovative transport: visions, trends and methods 43rd European Transport Conference Selected Proceedings
13. Guériau, M., Billot, R., El Faouzi, N.E., Hassas, S., Armetta, F.: Multi-agent dynamic coupling for cooperative vehicles modeling. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. pp. 4276–4277. AAAI'15, AAAI Press (2015)
14. Jin, Q., Wu, G., Boriboonsomsin, K., Barth, M.: Multi-agent intersection management for connected vehicles using an optimal scheduling approach. In: 2012 International Conference on Connected Vehicles and Expo (ICCVE). pp. 185–190 (Dec 2012)
15. Johansson, R., Nilsson, J., Bergenhem, C., Behere, S., Tryggvesson, J., Ursing, S., Söderberg, A., Törngren, M., Warg, F.: Functional Safety and Evolvable Architectures for Autonomy, pp. 547–560 (2017)
16. K Lincoln, N., Veres, S., Dennis, L., Fisher, M., Lisitsa, A.: An agent based framework for adaptive control and decision making of autonomous vehicles **1** (01 2010)
17. Kamali, M., Dennis, L.A., McAree, O., Fisher, M., Veres, S.M.: Formal verification of autonomous vehicle platooning. Science of Computer Programming **148**(Supplement C), 88 – 106 (2017)
18. Krajzewicz, D.: Traffic Simulation with SUMO – Simulation of Urban Mobility, pp. 269–293. Springer New York, New York, NY (2010)
19. Lambert, F.: Bmw will launch the electric and autonomous inext in 2021, new i8 in 2018 and not much in-between (May 2016), https://electrek.co/2016/05/12/bmw-electric-autonomous-inext-2021/
20. Leary, K.: Japan is testing driverless buses to help the elderly get around (sep 2017), https://futurism.com/japan-is-testing-driverless-buses-to-help-the-elderly-get-around/
21. Ling, Y., Mullen, T., Lin, X.: Analysis of optimal thread pool size. SIGOPS Oper. Syst. Rev. **34**(2), 42–55 (Apr 2000)
22. Lygeros, J., Godbole, D.N., Sastry, S.: A design framework for hierarchical, hybrid control. Tech. rep., Machines and Robotic Laboratory, University of California, Berkeley (1997)
23. Maleš, L., Ribarić, S.: A model of extended bdi agent with autonomous entities (integrating autonomous entities within bdi agent). In: 2016 IEEE 8th International Conference on Intelligent Systems (IS). pp. 205–214 (Sept 2016)
24. Müller, J., Pischel, M.: The agent architecture interrap: Concept and application (07 1993)
25. Passos, L., Rossetti, R., Kokkinogenis, Z.: Towards the next-generation traffic simulation tools: A first appraisal pp. 1–6 (01 2011)
26. Rao, A.S., Georgeff, M.P.: Bdi agents: From theory to practice. In: IN PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS (ICMAS-95. pp. 312–319 (1995)
27. Sage, A., Lienert, P.: Gm executive credits silicon valley for accelerating development of self-driving cars (Aug 2016), http://www.reuters.com/article/us-ford-autonomous/ford-plans-self-driving-car-for-ride-share-fleets-in-2021-idUSKCN10R1G1
28. Stoll, J.D.: Gm executive credits silicon valley for accelerating development of self-driving cars (May 2016), https://www.wsj.com/articles/gm-executive-credits-silicon-valley-for-accelerating-development-of-self-driving-cars-1462910491
29. Thangarajah, J., Harland, J., Morley, D.N., Yorke-Smith, N.: On the life-cycle of BDI agent goals. In: ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings. pp. 1031–1032 (2010)
30. Wooldridge, M.: An Introduction to MultiAgent Systems. Wiley Publishing, 2nd edn. (2009)
31. Zhang, R., Rossi, F., Pavone, M.: Routing autonomous vehicles in congested transportation networks: Structural properties and coordination algorithms (06 2016)