Decision-making for automated vehicles using a hierarchical behavior-based arbitration scheme

Piotr F. Orzechowski¹, Christoph Burger² and Martin Lauer²

Abstract—Behavior planning and decision-making are some of the biggest challenges for highly automated systems. A fully automated vehicle is confronted with numerous tactical and strategical choices. Most state-of-the-art automated vehicle platforms implement tactical and strategical behavior generation using finite state machines. However, these usually result in poor explainability, maintainability and scalability. Research in robotics has raised many architectures to mitigate these problems, most interestingly behavior-based systems and hybrid derivatives.

Inspired by these approaches, we propose a hierarchical behavior-based architecture for tactical and strategical behavior generation in automated driving. It is a generalizing and scalable decision-making framework, utilizing modular behavior components to compose more complex behaviors in a bottom-up approach. The system is capable of combining a variety of scenario- and methodology-specific solutions, like POMDPs, RRT* or learning-based behavior, into one understandable and traceable architecture. We extend the hierarchical behavior-based arbitration concept to address scenarios where multiple behavior options are applicable but have no clear priority against each other. Then, we formulate the behavior generation stack for automated driving in urban and highway environments, incorporating parking and emergency behaviors as well. Finally, we illustrate our design in an explanatory evaluation.

I. INTRODUCTION

Recent years have shown significant research and engineering progress in the field of automated driving and advanced driver assistance systems. In particular, the performance of state-of-the-art environment perception methods is improving quickly due to the advances in deep learning and other AI technologies.

Besides perception, one of the biggest challenges for highly automated systems is behavior planning and decision-making. In urban driving, traffic participants are confronted with numerous tactical and strategical choices. Humans decide in most of these situations, like stopping at a zebra crossing, choosing an appropriate gap when merging or yielding at intersections, reactively. Long-term decisions, like goal and route selection or the choice of driving style and behavior preferences, consider longer time horizons, though.

So far, considerable results in behavior and trajectory planning have been already achieved for some complex scenarios [1]–[3]. However, no generalizing and scalable decision-making framework has been found that is capable of combining a variety of such scenario- and methodology-specific approaches into one understandable and traceable architecture.

How and when should an automated vehicle switch from a regular ACC controller to a lane change, cooperative zip merge or parking planner? How can we support POMDPs, hybrid A* and any other planning method in our behavior generation?

Most state-of-the-art fully automated vehicles that have at least proven successful in the DARPA Urban Challenge [4]–[6] or during test rides on public roads [7], [8] have used finite state machines (FSMs) for tactical and/or strategical behavior generation. FSMs are a useful tool for simple systems with a small number of behavior options and maneuvers where each state represents one maneuver or driving mode. In practice FSMs, even hierarchical FMSs, turn out to be unsuitable for more complex tasks due to their poor explainability (about the reason why a behavior is executed), maintainability (the effort to refine existing behavior) and scalability (the effort to achieve a high number of behaviors). These shortcomings motivate the search for other architectures that can be used for tactical and strategical behavior generation.

Decision-making is a well known research field in robotics, also referred to as “robot control” or “action selection” [9]. Generally, the various approaches can be classified into knowledge- or behavior-based systems.

Knowledge-based systems, like FSMs, typically perform the action selection in a centralized, top-down manner using a knowledge database that contains a fused and abstracted representation of all available sensor data. As a result, the engineer designing the action selection module (in FSMs the state transitions) has to be aware of the conditions, effects and possible interactions of all behaviors at hand.

Behavior-based systems, on the other hand, decouple actions into atomic simple behavior components that should be aware of their conditions and effects themself. These modular behavior components are then combined to more complex behaviors in a bottom-up approach. Many architectures for behavior coordination have been proposed. The most prominent are the subsumption architecture [10], activation networks [11] and voting systems [12].

In this publication, we propose a hybrid approach combining the best from both worlds: A hierarchical behavior-based architecture for tactical and strategical behavior generation.

¹Piotr F. Orzechowski is with Intelligent Systems and Production Engineering, FZI Research Center for Information Technology, Karlsruhe, Germany orzechowski@fzi.de
²Christoph Burger and Martin Lauer are with the Institute of Measurement and Control Systems, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany {christoph.burger,martin.lauer}@kit.edu

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.
in automated driving. We combine atomic behavior components to more complex behaviors using generic arbitrators. Arbitrators can again be combined with other arbitrators or behavior components to generate an even more complex system behavior. A similar approach has proven very successful in robot soccer [13].

In automated driving, each behavior represents a tactical driving maneuver like changing the lane, crossing an unsignaled intersection or parking into a nearby parking lot. The integration of these self-contained maneuvers into the arbitration structure also realizes strategical behaviors like switching from urban to highway driving.

The behavior components make use of an environment model to assess if they are applicable in the current state and make this information available to their parent arbitrator. Arbitrators do not know the nature of their underlying behavior options by design (functional decomposition). Instead, given their arbitration strategy, they select an appropriate behavior based on abstract information only (e.g., expected utility or priority).

Such a hierarchical behavior-based arbitration scheme has great advantages:

- Scenario-specific solutions can be combined easily.
- It supports different planning approaches.
- The resulting behavior can be well explained.
- It can be iteratively extended by more behaviors.
- The modularity improves robustness and efficiency.
- Complex behavior emerges from simple components.

The remainder of this paper is structured as follows: Section II summarizes the main concepts of hierarchical behavior-based arbitration schemes as found in [13]. The main contribution of this publication is presented in Section III, where we formulate the behavior generation for automated driving using a hierarchical behavior-based arbitration scheme. We extend the existing arbitration approach, develop a suitable maneuver representation, define a set of fundamental driving behaviors and combine these to an overall system behavior using arbitrators. Section IV validates this approach with explanatory experiments on an urban route. Finally, Section V concludes with the key advantages and a brief outlook.

II. FUNDAMENTALS

A first concept of hierarchical behavior-based arbitration schemes for behavior generation has been presented in detail in [13]. This chapter highlights the main ideas that also have been briefly outlined in section I.

The concept is based on simple modular behavior components and generic arbitrators.

A. Behaviors — How to do things

Behavior components are the fundamental building blocks of a behavior-based architecture. They describe how and when things can be done.

A behavior component provides three main functionalities: The invocation condition indicates if this behavior is applicable in the current situation. The commitment condition of a currently active behavior signalizes if it can be continued or not. If either the invocation or commitment condition is true, the behavior can be selected to generate and execute its actual behavior command.

B. Arbitrators — Which thing to do

Arbitrators hierarchically combine behaviors to produce more complex behavior strategies. They decide which thing to do.

An arbitrator contains a list of behavior options to choose from. A specific selection logic determines which option is chosen based on abstract information.

Any knowledge and decision logic necessary for the selection and execution of a behavior component is completely encapsulated inside the behavior component itself. As a result, arbitrators do not need any knowledge about the nature of their underlying behavior options.

This bottom-up design approach leads to strong functional and semantic decomposition.

To generate even more complex behavior, an arbitrator can also be a behavior option of a hierarchically higher arbitrator.

The following selection schemes have been proposed: The highest priority first arbitrator organizes its behavior options in a list ordered by priority. An applicable option with the highest priority is chosen. The sequence arbitrator executes its options based on a fixed predefined order. A random arbitrator assigns probabilities to its behavior options and selects one among all applicable options randomly.

Additional arbitration schemes that are necessary for, but not limited to automated driving are introduced in section III-D.

III. APPLICATION TO AUTOMATED DRIVING

This chapter describes the main contribution of this publication: how a hierarchical behavior-based arbitration scheme can be utilized for decision-making in automated driving.

In contrast to classical behavior-based systems each behavior component is not directly connected to the sensors and actors. Instead, the input is an abstract environment model that contains a fused, tracked and filtered representation of the world. The behaviors’ output is also in a more generic form that can be passed to a trajectory planner or controller. In this sense, we follow the sense-plan-act paradigm in the overall software structure [9] but employ a behavior-based approach in the decision-making module.

A. Environment Model

The environment model in our implementation contains a lanelet map [14], ego motion state and detected objects with prediction. The map describes drivable areas, distinct lanes, parking lots, traffic rules, etc. The ego motion state mainly depicts the current pose and velocity of the ego vehicle. Currently, we assume that the objects are provided with a decoupled prediction, but integrated planning and prediction within the behavior components is also possible. We think our design should support both open-loop and closed-loop prediction.
B. Maneuver Representation

As we aim for a generalizing approach that is applicable to various driving environments our behavior representation should be as task-agnostic as possible. It should fit all relevant use cases and environments of automated driving, namely highway, urban and parking. However, the proposed representation and interfaces would also work for other environments like off-road driving.

Our smallest behavior building blocks, the behavior components, represent basic driving maneuvers such as “follow the ego lane”, “merge into traffic” or “park near goal”. In general, we can distinguish between maneuvers in a structured or unstructured environment. Urban and highway scenarios provide road boundaries or even distinct lanes, while parking lots and off-road areas feature open space like scenarios.

Therefore, we use a twofold maneuver representation:

Driving commands in structured maneuver environments use a corridor-based maneuver representation. It consists of a maneuver corridor, reference line, predicted objects and the chosen maneuver variant. The corridor is usually generated from map data [14], but could also be provided online, e.g. from semantic segmentation [15]. The reference line is an approximation of the centerline and can serve as a rough positional reference. Additionally, velocity objectives are given along this line, e.g. derived from the speed limit and curvature. The object list contains all objects that are relevant for this maneuver, their predictions as well as virtual objects indicating stop positions. Finally, the maneuver variant defines the chosen homotopy class, as discussed in [16]. An example of a corridor-based driving command is shown in Fig. 1.

Driving commands in unstructured environments directly use a trajectory to represent the requested maneuver. We did not choose a more abstract representation in this case, in order to support a wide variety of use cases in such environments.

Depending on the command representation type, the system following the decision-making module runs different pipelines to execute these maneuvers. Corridor-based maneuvers are passed to a trajectory planner, e.g. [17] or [18], followed by an appropriate controller. While trajectory-based driving commands are directly handed over to a trajectory controller that is tuned for slow and capable of backward driving, as needed for parking and similar maneuvers.

C. Driving Maneuvers — How to drive

Following the behavior-based approach we begin with designing atomic behavior components for simple tasks, before stacking them together in section III-D. Here, we do not attempt to present a feature-complete list with all necessary behaviors. Instead, we focus on explaining the main design concept using some hand-picked example behaviors, that should compile a decent start to develop an automated vehicle. This stack can then be extended iteratively by more specialized behavior components addressing specific driving situations. Furthermore, a behavior component can compute its maneuver command with any preferred state-of-the-art method.

An urban environment is probably the most challenging one for automated driving. We can think of at least three basic driving maneuvers needed in an urban setting:

**FollowEgoLane** As long as the ego pose matches any urban lane of our route and this lane does not intersect other lanes, our vehicle could probably follow it. Thus, the invocation condition of *FollowEgoLane* is true as long as such a matching lane exists. Executing this behavior for one time step will keep the vehicle in a well-defined state, so we can leave the commitment condition false to allow other behaviors to be selected after a *FollowEgoLane* command.

**CrossIntersection** One characteristic of urban environments are numerous signalized or unsignalized intersections that need specific behavior. An automated vehicle has to make sure to yield to superordinate traffic participants (such as vehicles and vulnerable road users (VRUs)). Additionally, in some intersections occlusions ask for even higher caution [19].

The invocation condition of *CrossIntersection* is true as long as the current ego lane intersects other lanes within its planning horizon. Crossing intersections can be critical such that its commitment condition will be true until the intersection has been passed. This should help to e.g. clear the intersection as soon as reasonably possible and not to unintendedly change lanes in the intersection.

**ChangeLane** Lane changes, on the other hand, are only possible when the current ego lane has a directly adjacent reachable lane on the left or right side with a safe distance to the following and leading vehicles. The *ChangeLane* component is defined w.r.t. the supposed changing direction and instantiated once for each direction to improve reusability.

As a result, the invocation condition of *ChangeLane* is true as long as the current ego lane has a directly adjacent reachable lane in the respective direction with
a big enough gap to safely change into. In order to produce consistent driving behavior, the commitment condition is true until the lane change maneuver has been completed or properly aborted, e.g. in case the selected gap becomes too small.

However, in dense traffic it might be necessary to perform a lane change in three consecutive phases — ApproachGap, IndicateIntention and MergeIntoGap [20]. These can be designed as behavior components as well and put into sequence in section III-D.

For better clarity and conciseness, the remaining behaviors will only be described briefly.

**ApproachGap** A gap approach should be possible, if the current ego lane has a directly adjacent reachable lane, regardless of the nearest gap size.

**IndicateIntention** Once the gap has been reached but is too tight to merge into directly, the vehicle will stay in its lane but indicate its intention using the turn signals and by laterally approaching the other lane.

**MergeIntoGap** As soon as the gap size is big enough, the vehicle can safely merge into it.

Another typical application for automated vehicles is driving on highways. Many occurring behaviors, like lane changing and ACC like lane following, are similar to those provided for urban environments. High velocities, special traffic rules and the absence of VRUs justify distinct behavior components though, in order to properly adjust these for highway environments.

**MergeOntoHighway** One of the most critical maneuvers on highways is merging onto the highway from an onramp. High relative velocities and sometimes short onramps pose a challenge to select an adequate gap, approach it with possibly high acceleration and finally merge into it. Similarly to lane changes in dense urban traffic, MergeOntoHighway could be modeled with sequential sub-behaviors, to further decompose the problem.

**FollowHighwayLane** Following a highway lane is comparable to FollowEgoLane, except that the environment is much more structured. Additionally, it can be assumed that no unpredictable VRU will enter or even cross the highway, which prevents overcautious behavior enabling driving at a higher speed. Furthermore, in countries like Germany, special highway laws prohibit overtaking other traffic participants on the right side.

**ChangeHighwayLane** Again, comparable to lane changes in dense urban traffic, changing lanes on highways can be modeled as a multi-phase behavior or as one integrated interaction aware behavior, using e.g. POMDPs [21].

**ExitFromHighway** Exiting from highways can be as simple as changing to a new diverging lane or as challenging as crossing traffic that is meanwhile entering the highway. Such scenarios can be found in cloverleaf interchanges, for example, where offramps overlap onramps.

In the beginning, end or even during an automated drive, the vehicle has to park in a suitable place. Either to drop and collect passengers or for standby and charging. Usually, path or trajectory planners based on graph search methods are used in these environments [22].

**LeaveGarage** Typically a ride starts with the car parked in a parking lot or garage. In this case, LeaveGarage is the only applicable behavior.

**ParkNearGoal** As soon as the automated vehicle is close to its goal and a suitable parking lot is found, the vehicle can reduce its speed and park into this parking lot. Notice, that the search for a parking lot is not included here. It might be modeled as another behavior component or supplied by the routing module.

Finally, we add fail-safe emergency behaviors, in case a dangerous unforeseen traffic situation evolves or as a fall back if no other behavior component provides feasible commands.

**EmergencyStop** In case — for whatever reason — an unavoidable collision will be anticipated, the EmergencyStop behavior will provide a full-stop trajectory to reduce damage and fatalities.

**EvadeObject** If a longitudinally unavoidable collision will be anticipated which could be avoided or at least mitigated by an evasive maneuver like [23], EvadeObject will provide such a behavior option.

**SafeStop** As a fail-safe fallback for any system failure or if none of the other behavior components provide feasible commands, SafeStop will bring the vehicle to a safe stop, preferably in a safe position.

**D. Arbitration Scheme — Which maneuver to drive**

Now that we have developed a couple of basic behavior components, we can use them to compose the overall behavior for automated driving, starting bottom-up.

We follow a similar notation to [13], denoting the behavior options of an arbitrator with \( O_{RBAN} \), using round brackets “(“) for an ordered list and curly brackets “{“} for a set of options. Basic behavior components are highlighted with *ItalicNames* and arbitrators with *CapitalNames*.

In an urban environment FollowEgoLane, CrossIntersection, ChangeLane and MergeIntoLane are possible behaviors. Typically, these have no clear and consistent priority over each other, yet the most reasonable one should be chosen. As none of the existing arbitration schemes (by priority, sequence or random) is sufficient for this task, we define a new cost-based arbitrator that selects the behavior option with the lowest expected cost. By introducing such a cost arbitrator, the decision-making concept can be extended to dynamically changing preferences.

Now, such a cost-based **URBAN** arbitrator can be used to select among the urban behavior components:

\[
O_{URBAN} = \{ \text{FollowEgoLane}, \text{CrossIntersection}, \\
\text{ChangeLaneLeft}, \text{ChangeLaneRight} \\
\text{MergeIntoLANELEFT}, \
\text{MergeIntoLANERIGHT} \}
\]

The cost-based arbitrator uses a behavior-agnostic cost estimation module to estimate the expected average travel velocity, incorporate routing costs and penalize lane changes.
As discussed in section III-C lane changes in dense traffic can be decomposed into three stages. As a result, a sequence-based arbitrator is used to compose \( O_{\text{MERGEINTOLANE}} \):

\[
O_{\text{MERGEINTOLANE}} = (\text{ApproachGap, IndicateIntention, MergeIntoGap})
\]

Similar to the urban case, highway behavior options are combined using a cost-based arbitrator:

\[
O_{\text{HIGHWAYDRIVING}} = \{\text{MergeOntoHighway, FollowHighwayLane, ChangeHighwayLaneLeft, ChangeHighwayLaneRight, ExitFromHighway}\}
\]

In case of \( O_{\text{PARKING}} \) at most one option — parking in or out — is feasible after all, such that a trivial priority-based arbitrator can be used:

\[
O_{\text{PARKING}} = (\text{LeaveGarage, ParkNearGoal})
\]

The emergency maneuvers for unavoidable collisions are grouped together using a cost-based arbitrator estimating the expected damage. In such a way, it chooses the option with the lowest expected damage:

\[
O_{\text{AVOIDCOLLISIONINLASTRESORT}} = \{\text{EmergencyStop, EvadeObject}\}
\]

Finally, these arbitrators and the \( O_{\text{SAFESTOP}} \) fallback are composed together to the top-most priority-based \( O_{\text{AUTOMATEDDRIVING}} \):

\[
O_{\text{AUTOMATEDDRIVING}} = (O_{\text{AVOIDCOLLISIONINLASTRESORT}}, \text{PARKING, URBANDRIVING, HIGHWAYDRIVING, SafeStop})
\]

The resulting arbitration graph is shown in Fig. 2.

IV. Experiments

In this section, we show the applicability of the proposed concept to utilize a hierarchical behavior-based architecture for behavior generation in automated driving.
The explanatory example performs basic urban driving behaviors on a simulative 5.7 km test track based on our real-world test route in Karlsruhe, Germany. The route, shown in Fig. 4, contains segments with speed limits of 30 km/h, 50 km/h and 60 km/h, is crossing or turning at 12 intersections, traversing one roundabout and ends at a parking lot.

We use the ROS-based open-source simulation framework CoInCar-Sim [24]. One great advantage of this framework is that it provides the same interface as our test vehicle Bertha [7]. Hence, we can develop, test and deploy the same behavior and planning pipeline in CoInCar-Sim and Bertha.

Our basic example maneuvers for this track are: ParkNearGoal, FollowEgoLane, ChangeLane (one instantiation for left, another for right lane changes) and SafeStop. Lane following and both lane change behaviors are combined within a cost-based URBAN-DRIVING arbitrator. Whereas parking, urban driving and the safe stop fallback constitute the overall behavior using a priority-based AUTOMATED-DRIVING arbitrator. Fig. 5 illustrates this arbitration graph.

This design has the following motivation. ParkNearGoal is only applicable in the vicinity of the goal and a nearby parking lot. Thus, as long as the ego vehicle is still on the route FollowEgoLane is and ChangeLaneLeft or ChangeLaneRight might be applicable. URBAN-DRIVING will select the most promising one, w.r.t. the expected average velocity, routing costs and lane change penalties. As soon as the vehicle approaches its goal, FollowEgoLane will bring it to a stop within the last lanelet. Then, ParkNearGoal will become applicable, chosen by priority and lead the car into its parking lot. When the parking maneuver is finished, ParkNearGoal will render inapplicable again. At that point also none of the URBAN-DRIVING behaviors is applicable any more because the car has left the route. As a result AUTOMATED-DRIVING selects the lowest priority behavior SafeStop. This is a good illustration of how the fallback behavior prevents undefined states and keeps the vehicle in a safe position.

B. Results

Fig. 3 shows the resulting behavior selection over time. The whole route takes 9:40min and features the expected behavior characteristics. The vehicle starts leaving the campus area by following the lane. At intersection A, it changes to the right lane in order to take a turn into a north-east direction. At point B, it takes another right turn following the ego lane and has to merge into traffic on the left lane. When approaching the next intersection C, the ego vehicle changes onto the exit lane in order to turn into south-east direction. At $t = 339$ s it approaches and passes the roundabout D.

Fig. 6 shows the two applicable behavior options at point E, where the route leads onto the “Adenauerring” again. The route continues with a right turn from the rightmost lane, while the ego is on the leftmost lane still. This is a suitable scenario to explain the cost-based arbitration in detail. The urban driving cost estimate incorporates the average expected
travel velocity, routing costs and penalizes lane changes:

\[ J = -\hat{v} + n_{LC-needed} \cdot J_{LC-needed}, \quad \text{without lane change} \]
\[ J = -\hat{v} + n_{LC-needed} \cdot J_{LC-needed} + J_{LC-Maneuver}, \quad \text{otherwise} \]

\( \hat{v} \) depicts the expected average velocity of this maneuver. Each of the \( n_{LC-needed} \) needed lane changes to follow the route after this command is charged with \( J_{LC-needed} = 10 \text{ km/h} \). Lane change behaviors themselves are penalized with a lower \( J_{LC-Maneuver} = 5 \text{ km/h} \). Hence, the arbitrator generally prefers the follow lane behavior as long as it matches the route. As soon as one or multiple lane changes will be necessary, this maneuver will become more favorable.

At point \( E \), the behaviors have these costs:

\[ J_{FollowEgoLane} = -6.944 + 1 \cdot 2.778 + 0 = -4.167 \]
\[ J_{ChangeLaneRight} = -9.283 + 0 \cdot 2.778 + 1.389 = -7.894 \]

Consequently the cost-based arbitrator chooses \( \text{ChangeLaneRight} \), which has lower cost than \( \text{FollowEgoLane} \).

An interesting part is directly after taking the right turn at point \( E \) from \( t = 422 \text{ s} \) to \( t = 436 \text{ s} \). Here, the vehicle performs two consecutive lane changes in order to pass this two-lane road from the rightmost lane to the exit lane. This is especially noteworthy, as no double lane change or other hand-crafted behavior has been defined for such a scenario. The behavior emerges purely because the routing has been incorporated into the cost estimate.

The road leads back to the campus again, where the vehicle comes slows down and stops at the end of the route. Finally, the parking behavior becomes active and brings the car into its parking lot. After finishing the parking maneuver, the safe stop behavior is the last suitable option and keeps the car at a standstill.

V. CONCLUSIONS AND FUTURE WORK

In this publication, we presented the following contributions:

An extension to the hierarchical behavior-based arbitration concept introduced in [13]. We introduced a cost-based arbitration scheme that is helpful when multiple behavior options are applicable but have no clear and consistent priority against each other.

We have formulated a behavior generation stack for automated driving based on the hierarchical behavior-based arbitration scheme. It consists of maneuvers for urban and highway environments, contains parking and emergency behaviors, and prevents undefined states with a fallback safe stop behavior.

We have shown the usefulness and applicability of our design in an explanatory evaluation on a simulation route. The key advantages of the approach are:

- **Scenario-specific solutions can be combined easily.**
  In the experiments, five different behaviors have been employed to handle various scenarios, from four-way intersections, T-junctions, a roundabout to multilane bypass roads and parking.

- **It supports different planning approaches.**
  We utilized two different trajectory planners in our experiments. Urban corridor-based maneuvers activated an optimization-based planner similar to [17], while the parking maneuver used an RRT* motion planner to generate Hybrid Curvature trajectories [22].

- **The resulting behavior can be well explained.**
  The strongly modular design significantly improves understandability compared to FSMs or classical behavior-based systems. Each invocation condition can be well understood; the selection logic of arbitrators is comprehensive. As a result, the hierarchical decision-making process can be well explained and traced over time.

- **It can be iteratively extended by more behaviors.**
  In order to add the parking behavior to our behavior generation, the definition of invocation and commitment conditions for parking was sufficient to add it to the AUTOMATEDDRIVING arbitrator. Thanks to the strong decoupling, no changes to any other behavior component were necessary.

- **The modularity supports robustness and efficiency.**
  As each of the behavior components is self-contained, such that occurring failures are contained as well and do not affect the overall system stability. In case of a failure, the system will degrade seamlessly by ignoring this behavior option. Furthermore, the atomic structure allows to evaluate behavior options in parallel to increase efficiency. Strong modularity has many more advantages, among others, reusability and maintainability.

- **Complex behavior emerges from simple components.**
  Complex system behavior, as multiple consecutive lane changes to approach an exit lane, emerges from the arbitration scheme without the need for hand-crafted decision or planning logic.

These benefits have led to a smooth development process with promising results, as outlined in section IV. Thus, we look forward to further enhance the numerous existing behavior components, extend the behavior stack by e.g. our
MIQP approach for cooperative zip merges [25] and most excitingly to integrate this stack on our test vehicle Bertha.

REFERENCES

[1] S. Hoermann, F. Kunz, D. Nuss, S. Renter, and K. Dietmayer, “Entering crossroads with blind corners. A safe strategy for autonomous vehicles”, in 2017 IEEE Intelligent Vehicles Symposium (IV), Jun. 2017.

[2] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller, “Automated Driving in Uncertain Environments: Planning With Interaction and Uncertain Maneuver Prediction”, IEEE Transactions on Intelligent Vehicles, vol. 3, no. 1, Mar. 2018.

[3] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, “Scalable Decision Making with Sensor Occlusions for Autonomous Driving”, in 2018 IEEE International Conference on Robotics and Automation (ICRA), May 2018.

[4] M. Buehler, K. Iagnemma, and S. Singh, Eds., The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, red. by B. Siciliano, O. Khatib, and F. Groen, vol. 56, ser. Springer Tracts in Advanced Robotics, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[5] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholz, et al., “Odin: Team victortango’s entry in the darpa urban challenge”, Journal of Field Robotics, vol. 25, no. 8, 2008.

[6] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, et al., “Junior: The Stanford entry in the Urban Challenge”, Journal of Field Robotics, vol. 25, no. 9, 2008.

[7] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, et al., “Making Bertha Drive – An Autonomous Journey on a Historic Route”, IEEE Intelligent Transportation Systems Magazine, vol. 6, no. 2, Sum. 2014.

[8] M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, et al., “Experience, Results and Lessons Learned from Automated Driving on Germany’s Highways”, IEEE Intelligent Transportation Systems Magazine, vol. 7, no. 1, Spr. 2015.

[9] B. Siciliano and O. Khatib, Eds., Springer Handbook of Robotics, ser. Springer Handbooks, Springer International Publishing, 2016.

[10] R. Brooks, “A robust layered control system for a mobile robot”, IEEE Journal on Robotics and Automation, vol. 2, no. 1, Mar. 1986.

[11] Pattie Maes, “How to do the right thing”, Connection Science, vol. 1, no. 3, 1989.

[12] Julio K. Rosenblatt, “DAMN: A distributed architecture for mobile navigation”, Journal of Experimental & Theoretical Artificial Intelligence, vol. 9, no. 2-3, 1997.

[13] M. Lauer, R. Hafner, S. Lange, and M. Riedmiller, “Cognitive concepts in autonomous soccer playing robots”, Cognitive Systems Research, vol. 11, no. 3, 2010.

[14] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, et al., “Lanelet2: A high-definition map framework for the future of automated driving”, in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Nov. 2018.

[15] A. Meyer, N. O. Salscheider, P. F. Orzechowski, and C. Stiller, “Deep Semantic Lane Segmentation for Mapless Driving”, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2018.

[16] P. Bender, O. Ş. Taş, J. Ziegler, and C. Stiller, “The combinatorial aspect of motion planning: Maneuver variants in structured environments”, in 2015 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2015.

[17] J. Ziegler, P. Bender, T. Dang, and C. Stiller, “Trajectory planning for Bertha – A local, continuous method”, in 2014 IEEE Intelligent Vehicles Symposium Proceedings, Jun. 2014.

[18] B. Gutjahr, L. Gröll, and M. Werling, “Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC”, IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 6, Jun. 2017.

[19] P. F. Orzechowski, A. Meyer, and M. Lauer, “Tackling Occlusions & Limited Sensor Range with Set-based Safety Verification”, in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Nov. 2018.

[20] J. Nilsson, M. Brännström, E. Coelingh, and J. Fredriksson, “Lane Change Maneuvers for Automated Vehicles”, IEEE Transactions on Intelligent Transportation Systems, vol. PP, no. 99, 2016.

[21] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, “A Belief State Planner for Interactive Merge Maneuvers in Congested Traffic”, in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Nov. 2018.

[22] H. Banzhaš, M. Dolgov, J. Stellet, and J. M. Zöllner, “From Footprints to Beliefprints: Motion Planning under Uncertainty for Maneuvering Automated Vehicles in Dense Scenarios”, in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Nov. 2018.

[23] M. Werling and D. Liccardo, “Automatic collision avoidance using model-predictive online optimization”, in 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), Dec. 2012.

[24] M. Naumann, F. Poggenhans, M. Lauer, and C. Stiller, “CoInCar-Sim: An Open-Source Simulation Framework for Cooperatively Interacting Automobiles”, in 2018 IEEE Intelligent Vehicles Symposium (IV), Jun. 2018.

[25] C. Burger and M. Lauer, “Cooperative Multiple Vehicle Trajectory Planning using MIQP”, in 2018 21st
