# Task Planning for Highly Automated Driving

Chao Chen[1] and Andre Gaschler[1] and Markus Rickert[1] and Alois Knoll[2]

***Abstract*— A hybrid planning approach is presented in this paper with the focus of integrating task planning and motion planning for highly automated driving. In the context of task planning, the vehicle and environment states are transformed from the continuous configuration space to a discrete state space. A planning problem is solved by a search algorithm for an optimal task sequence to reach the goal conditions in the symbolic space, regarding constraints such as space topology, place occupation, and traffic rules. Each task can be mapped to a specific driving maneuver and solved with a dedicated motion planning method in the continuous configuration space. The task planning approach not only bridges the gap between high-level navigation and low-level motion planning, but also provides a modular domain description that can be developed and verified individually.**

**Our task planner for automated driving is evaluated in several scenarios with prior knowledge about the road-map and sensing range of the vehicle. Behavior that is otherwise complex to achieve is planned according to traffic rules and re-planned regarding the on-line perception.**

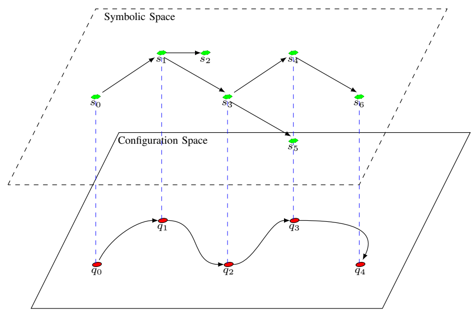


Fig. 1. Combined task planning and motion planning: The plane below is the continuous configuration space for motion planning with configurations as red dots. The plane above is the discrete state space for symbolic planning with states as green dots. The dashed blue lines show the mapping between the two spaces. First, the start $q_0$ and the goal $q_4$ are mapped to the symbolic states as $s_0$ and $s_6$, where a sequence of tasks is found through the states $\{s_0, s_1, s_3, s_4, s_6\}$. Then, the symbolic states are projected back to the configuration space as $\{q_0, q_1, q_2, q_3, q_4\}$, and concrete motions are planned from the start to the goal.

## I. Introduction

The research of autonomous driving has developed a rich spectrum of planning algorithms and systems from global navigation to local trajectory tracking. A navigation system deals with street maps, traffic conditions, fuel consumption, etc. from a global perspective to determine a route without motion details. A motion planning algorithm produces trajectories and control inputs for a concrete driving maneuver regarding vehicle kinematics and obstacles inside a local space and time horizon. It is usually impossible to plan a motion for an entire route due to insufficient perception and limited processing capacity of the system. Therefore, a motion planner is always responsible for sub-problems regarding the actual system state and milestones of the global route. As a result, an integration of these two kinds of planners is necessary for highly automated driving. In this paper, a domain specific task planning method is proposed to fulfill this purpose.

The idea is to design a hierarchical decision making strategy analogous to the behavior of human drivers. While having a rough route in mind, a human driver usually makes discrete decisions, such as overtaking, lane switching, or turning based on the actual traffic condition. Then, these driving tasks are performed with continuous steering, accelerating, or braking commands. If the situation is not clear, certain behaviors are performed to acquire additional information or negotiate with other traffic partners to resolve the uncertainties. If a task cannot be accomplished, the following actions and even the route are adapted. Such abilities and behaviors are especially addressed by artificial intelligence (AI) research as reasoning and problem solving, which are further developed in robotics domain as task planning. Fig. 1 shows the general concept of combining task planning in symbolic state space and motion planning in continuous configuration space.

The contribution of this work is a hybrid planning approach and introducing task planning into autonomous driving domain. In essence, our task planning approach is based on three necessary design principles that cannot be achieved by conventional state machine or signal-based architectures.

- **Symbolic-geometric hierarchy**: The whole problem is first processed in the symbolic space with limited discrete system states. Then, the tasks are passed to the motion planner as milestones and a sequence of motions are planned to perform these tasks. Thus, a large complicated problem is divided into sub-problems, which can be solved more efficiently with specific methods in adequate scopes.
- **Verifiability**: All symbolic domain description is organized as *modules that can be verified individually* and independently from the planning algorithm.
- **Generic AI planning**: Traffic rules and semantic information allow and require automatic planning and fully integrate into the task planning concept. 40 years of research in artificial intelligence have brought powerful task planning methods that are increasingly applied to robotics and can also be applied to automated driving.

[1]Chao Chen, Andre Gaschler and Markus Rickert are with fortiss GmbH, An-Institut Technische Universität München, Munich, Germany.

[2]Alois Knoll is with Robotics and Embedded Systems, Department of Informatics, Technische Universität München, Munich, Germany.

## II. Related Work

Classical planning is an important branch of artificial intelligence [1], which performs logical reasoning in a symbolic space and searches for a strategy or a sequence of actions to achieve the goal. A symbolical definition of problem domains can be found in STRIPS [2], PDDL [3], and their successors. In these formulations, a system state is represented with a number of predicates. Actions are defined with certain preconditions and effects. The preconditions decide whether an action is applicable to a state. The effects change the state when an action is performed. A planner searches for a sequence of actions that provides a valid transformation from the initial state to a state satisfying the goal conditions. The general approach to solve the classical planning problem is forward search [4] or backward chaining [1].

Symbolic planning applies well to problems that are discrete, deterministic, and fully-observed. However, the problems in the robotic domain are more intricate. The state of a robot and its environment is continuous and may be partially observed [5], [6]. The actions of a robot are also continuous and cost time. As a result, a symbolic planner cannot be directly applied, but as a task planning layer in a hierarchical planning architecture. It works with a set of abstractions of the world states and plans symbolic tasks for the robot based on the domain-specific semantic knowledge. These actions divide a large problem into sub-problems and limit the context for an efficient motion planning [7]. Only few works integrate symbolic and geometric constraints in mobile robotics. Plaku and Hager [8] develop a kinodynamic motion planner that can solve symbolic tasks. Their approach is to perform a forward search of motion that fulfills differential constraints, guided by a task planner. In the field of mobile manipulation, integrated task and motion planning is considered necessary to solve all but the simplest scenarios. A typical approach here is the belief-space hierarchical planner by Kaelbling and Lozano-Pérez [9]. This approach is quite different, as it searches backward from the goal state in a probabilistic state space and models uncertainty of both perception and manipulation actions.

In the autonomous driving domain, route planning problems are solved with graph search algorithms regarding a cost metric on a higher level. On the motion planning level, algorithms such as Hybrid-A* [10] or SEHS [11] produce collision-free trajectories for a vehicle in a known environment. In addition, the low-level motion planner can benefit from certain traffic knowledge [12]. However, sophisticated traffic rules are difficult to be modeled in the continuous configuration space. The common approach is to integrate all the conditions, rules, and actions into a state machine [13] [14]. In this case, the system complexity, in general, grows exponentially with respect to the number of states or conditions; test, verification, and modification of such systems are tedious and error prone. Kress-Gazit and Pappas [15] propose an approach to synthesize hybrid controllers with high level descriptions for urban driving behaviors, which guarantee correctness and facilitate the development of state machines. An alternative approach are behavior-based solutions, which select the current control inputs by likeliness or a fusion of predefined actions [16] [17]. However, the result behavior may be a local optimum that does not comply with the longer-term goals.

## III. Domain Definitions

The task planning domain for automated driving is defined with a tuple $\{\mathcal{W}, \mathcal{S}, \mathcal{P}, \mathcal{O}\}$. $\mathcal{W}$ is the continuous world model, which consists of the vehicle and environment states such as vehicle speed, object position, place geometry, etc. $\mathcal{S}$ is the symbolic state space, which holds the semantic information, e.g., parking-lot occupation, lane type. A set of predicates $\mathcal{P}$ provides a projection from $\mathcal{W}$ to $\mathcal{S}$ by verifying certain propositions of the world model. $\mathcal{O}$ is a set of operators, which have effects on the world state. A task is a function from a set of symbolic states to another. The preconditions are a set of predicates, which define the domain of the function. The value range of the function is determined by the effects of the task, i.e., a set of operators with parameters. Details are outlined in the following subsections.

### *A. World Model*

A world model consists of places and objects. A place $P$ is a particular space where a vehicle can perform its motion, e.g., a lane, a parking-lot. An object $O$ is a traffic related entity, e.g., a vehicle, a pedestrian or a traffic signal. Fig. 2 shows an example of a world model with three lanes, two junctions, and a vehicle.

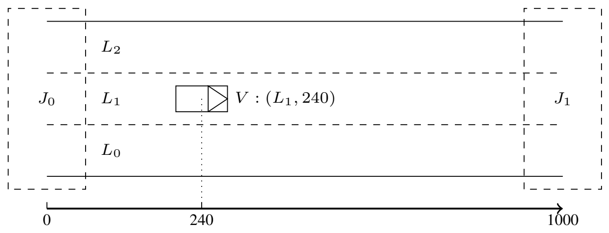


Fig. 2. Example of a world model: Places $L_0$, $L_1$ and $L_2$ are lanes and $J_0$ and $J_1$ are junctions. $L_2$ is left to $L_1$, while $L_0$ is on the right of $L_1$. $J_0$ is before the lanes and $J_1$ is after them. The axis below shows a lane coordinate starting from 0 to 1000 m. Object $V$ is a vehicle at the location $(L_1, 240)$, i.e., on the lane $L_1$ at position 240 m.

A place has attributes such as type $T$, and the topological relations to other places. It maintains a list of adjacent objects. Its geometry is used for spatial reasoning and generating motion planning requests. Two types of places are defined for the example scenarios in Section V.

- **Lane (L)**: A lane can have a type such as urban street, highway, motorway, etc. Further attributes are an identifier, a speed limit, and a driving direction. It is able to access the neighboring lanes and the adjacent junctions from it. Vehicles, traffic signals, and parking-lots are registered to a lane in different lists. They are located with a distance value $d$ in the lane coordinate as $(L, d)$ in Fig. 2. Thus, it is possible to reason about the relative positions of the objects and places. Finally, a lane has a certain geometric shape.

- **Junction (J)**: A junction holds the adjacency information of the inbound and outbound lanes. The connections have priorities and permissions according to the traffic rules and signals.

An object is described with attributes such as type, position and speed. It can be mapped to multiple places, e.g., a traffic light can be referred by several lanes of the same road. The state of a dynamic object can be changed and predicted. Two types of objects are introduced in the examples.

- **Vehicle (V)**: The state of a vehicle includes its location, speed, and driving behaviors such as light signals. The state of a vehicle is dynamic.
- **Traffic Signal (S)**: Traffic signals are lane markers, traffic signs, and traffic lights. A traffic light holds its current state and may also contain information about the phase period.

## B. Predicates

A predicate is a proposition about the world state. The arguments can be constants or variables from the perception. The value of a predicate is a trilean: *True*, *False* or *Unknown*. If the information is insufficient to verify a proposition, the value of the predicate is *Unknown*. A predicate can consist of sub-predicates or inherit from other predicates. The predicates used in the examples are:

1) $At(O, T)$ takes the value *True* if object $O$ is at a place of a specific type $T$.
2) $At(O, P)$ takes the value *True* if object $O$ is at any location of a specific place $P$.
3) $Before(O, L, d)$ takes the value *True* if object $O$ is on lane $L$ and before the location $(L, d)$. It is derived from the predicate $At(O, P)$.
4) $In(O, L, d_0, d_1)$ takes the value *True* if object $O$ is inside the range $[d_0, d_1]$ of lane $L$. It is the combination of $Before(O, L, d_1)$ and $\neg Before(O, L, d_0)$.
5) $Beside(L_1, L_2)$ takes the value *True* if lane $L_1$ is beside lane $L_2$.
6) $Free(L, d_0, d_1, t_0, t_1)$ takes the value *True* when an object can travel from $(L, d_0)$ to $(L, d_1)$ in the time duration $[t_0, t_1]$ without collision with other objects. Only the point locations of the objects are considered to verify this predicates.
7) $Connect(L_1, L_2, J)$ takes the value *True* if lane $L_1$ and lane $L_2$ are connected in junction $J$.
8) $Clear(L_1, L_2, J)$ takes the value *True* if the connection between lane $L_1$ and lane $L_2$ is clear to drive in junction $J$ for both space clearance and permission by traffic rules. It is derived from $Connect(L_1, L_2, J)$.

## C. Tasks

Tasks are primitive driving actions, which composed of *Preconditions* and *Effects*. The *Preconditions* are a set of predicates which decide whether a task is applicable. If one of the predicates is *Unknown*, the task is uncertain. The effects of a task are a set of operators, which mainly change the state of the ego vehicle. A task can take parameters, which serve as arguments for preconditions and effects. These parameters can be obtained directly from the world model or suggested by a third component reasoning about the semantic information. In addition, a task may also have a cost value, allowing the planner to choose an optimal solution regarding a cost metric. The following tasks are defined for the example scenarios:

1) $FollowLane(V, L, d, v)$: Vehicle $V$ follows lane $L$ to location $(L, d)$ with speed $v$.
**Preconditions:** Vehicle $V$ should be on lane $L$ and before the location $(L, d)$. Lane $L$ should be free at $\{(L, d_0, t_0), \ldots, (L, d, t_d)\}$ during the task. The start location and time $(L, d_0, t_0)$ can be obtained directly from the vehicle state. $t_d = t_0 + \frac{(d-d_0)}{v}$ is the estimated time at the final location $(L, d)$ with a constant speed $v$.
$$Before(V, L, d) \wedge Free(L, d_0, d, t_0, t_d)$$
**Effects:** Vehicle $V$ is at the desired position $d$ with the desired speed $v$ at time $t_d$.
**Cost:** The time cost of the task, $t_d - t_0$.
2) $SwitchLane(V, L_1, L_2)$: vehicle $V$ switches from lane $L_1$ to lane $L_2$.
**Preconditions:** Vehicle $V$ is on lane $L_1$, which is beside lane $L_2$. The both lanes should be free at $\{(L_{1,2}, d_0, t_0), \ldots, (L_{1,2}, d_s, t_s\}$ during the task. $(L_1, d_0, t_0)$ is the start location and time. $(L_2, d_s, t_s)$ with $t_s = t_0 + t_{\text{switch}}$ and $d_s = d_0 + v \times t_{\text{switch}}$ is the final location and time calculated with a constant time cost $t_{\text{switch}}$ and a constant speed $v$.
$$\begin{aligned} &At(V, L_1) \wedge Beside(L_1, L_2) \\ &\quad \wedge Free(L_1, d_0, d_s, t_0, t_s) \\ &\quad \wedge Free(L_2, d_0, d_s, t_0, t_s) \end{aligned}$$
**Effects:** Vehicle $V$ is at lane $L_2$ with distance $d_s$ at time $t_s$.
**Cost:** The time cost of the lane switching, $t_{\text{switch}}$.
3) $ChangeLane(V, L_1, L_2, J)$: Vehicle $V$ change from lane $L_1$ to lane $L_2$ at junction $J$.
**Preconditions:** Vehicle $V$ should be close to the end of lane $L_1$ within a range $d_{\text{change}}$. Lane $L_1$ and $L_2$ are connected in junction $J$ and is clear to drive. The end of lane $L_1$ and the begin of lane $L_2$ should be free during the duration $t_{\text{change}}$ of the lane changing. $l_1$ is the length of lane $L_1$. $t_0$ is the start time of the lane changing. The final time is $t_c = t_0 + t_{\text{change}}$.
$$\begin{aligned} &In(V, L_1, l_1 - d_{\text{change}}, l_1) \wedge Clear(L_1, L_2, J) \\ &\quad \wedge Free(L_1, l_1 - d_{\text{change}}, l_1, t_0, t_c) \\ &\quad \wedge Free(L_2, 0, d_{\text{change}}, t_0, t_c) \end{aligned}$$
**Effects:** Vehicle $V$ is on lane $L_2$ with distance $d_{\text{change}}$ at time $t_c$.
**Cost:** The time cost to change the lane, $t_{\text{change}}$.

These tasks are the very basic maneuvers concerning only the topology, length, and occupation of lanes. Further traffic rules can be applied to these tasks by extending the preconditions or providing the parameters regarding certain conditions. More details are presented in the example section.

## IV. Task Planning

According to the domain definition, a problem of task planning for automated driving can be represented with a triple $\{I, G, \mathcal{T}\}$, of which $I$ is an initial state, $G$ is a set of goal conditions, and $\mathcal{T}$ is a group of relevant tasks. The task planner follows a forward search approach.

### A. Forward Search with Heuristics

The general approach of the task planning is to propagate the world state forward until fulfilling the goal conditions as Alg. 1. A node of the search algorithm contains a world state $s$ and a task $t$ to reach the current state from the previous state. A heuristic is designed to improve the search efficiency with an actual cost $f$ for the cost from the start to the current state and a heuristic cost $h$ for the estimated rest cost to reach the goal. Nodes to be traversed are sorted in an open set $S_{\text{open}}$ according to the total costs $g = f + h$. The node with the smallest total cost is expanded with a set of tasks in each iteration. The evaluated nodes are saved to a closed set $S_{\text{closed}}$ to avoid revisiting the same node.

**Algorithm 1:** $\text{TaskPlanning}(I, G, \mathcal{T})$

```
1  $S_{\text{closed}} \leftarrow \emptyset$;
2  $S_{\text{open}} \leftarrow (I, t_{null})$;
3  while $S_{\text{open}} \neq \emptyset$ do
4      $(s_i, t_i) \leftarrow \text{PopTop}(S_{\text{open}})$;
5      if $\text{Verify}(s_i, G)$ then
6          return success;
7      else if $(s_i, t_i) \notin S_{\text{closed}}$ then
8          $\mathcal{T}_i \leftarrow \text{ChooseTasks}(\mathcal{T}, s_i)$;
9          for each $t_j \in \mathcal{T}_i$ do
10             if $\text{Verify}(s_i, \text{PreConditions}(t_j))$ then
11                 $s_{i,j} \leftarrow \text{ExecuteTask}(t_j, s_i)$;
12                 $S_{\text{open}} \leftarrow \{(s_{i,j}, t_j)\} \cup S_{\text{open}}$;
13         $S_{\text{closed}} \leftarrow \{(s_i, t_i)\} \cup S_{\text{closed}}$;
14 return failure;
```

The function $\text{Verify}(s, P)$ returns *True* if the world state $s$ satisfies a set of predicates $P$. The function $\text{PopTop}(S_{\text{open}})$ removes the node with the smallest $g$-value from the open set. The function $\text{ChooseTasks}(\mathcal{T}, s)$ selects a subset of tasks from $\mathcal{T}$ based on the world state $s$. Tasks are mapped to specific types of states in order to reduce the condition checks, e.g., a lane-switching task is only possible when a vehicle is in a lane. The task parameters are decided when a task is selected, e.g., the distance and speed of a lane-following task are decided based on the speed limit, traffic condition, and sensing range of the vehicle. If the preconditions of a selected task are satisfied, the effects of the task are applied to the current state with the function $\text{ExecuteTask}(t, s)$ to obtain a new state. In addition, the cost of the task is added to the $f$-value and the $h$-value is calculated for the new state. According to a route, the heuristic cost is estimated only based on the distance and speed limits regardless of the other constraints or driving efforts. The algorithm returns success when the goal conditions are fulfilled.

Due to the limited sensing capabilities of a vehicle, the available information may be insufficient to decide whether a precondition of a task is valid or not. In this case, the task is specified as uncertain, and still evaluated by the planner, with all the following nodes marked uncertain. If no solution with determinable tasks exists, the planner may return an uncertain result. The planner can further consider the success rate of the tasks and select the most promising solution. In contrast to re-planning with inexecutable tasks, a planner can generate an entire strategy assuming each uncertain task can fail to makes sure that the vehicle is always able to reach the goal or remain in a safe state.

### B. Planning Architecture

Task planning requires a route as heuristic and generates tasks as motion planning requests. It connects route planning and motion planning in a highly automated driving system as Fig. 3.

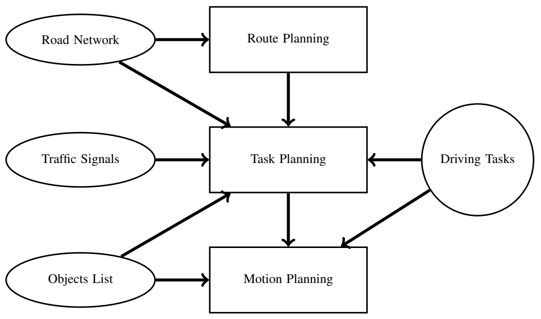


Fig. 3. Hybrid planning system architecture: The rectangles are the intelligent components which plan routes, tasks, and motions in different layers. The ellipses are the perception components that provide information such as road map, traffic signals, and objects list. A set of driving tasks are defined for the task planning and motion planning.

A high level navigation component provides the vehicle with a route, which serves as a global heuristic for tasks planning. A task planner sets way-points along the route as sub-goals to plan tasks on-line. If the task planner finds a selected target is unreachable, the route planner may receive a feedback and suggest an alternative route.

A task is then forwarded to create a request for motion planning. As a task can provide additional information about the driving behavior, a motion planner can solve the problem efficiently in a convenient scope. For example, a motion for a lane-following task should be performed only inside the lane, which can be generated with a compact behavior-based method. A motion planner can also give feedback to the task planner if a task is inexecutable for unsuccessful planning. The task planner can adapt the solution by suggesting an alternative task.

## V. Scenarios

To test the task planning concept for automated driving, two scenarios are implemented and verified: an overtaking scenario on the motorway and a rerouting scenario in a round-about. The former one puts emphasis on the traffic rules and the latter one deals with uncertainties and re-planning. A behavior-based method is employed to perform the lane-following task. The lane-switching and lane-changing tasks are planned with a traffic knowledge aided SEHS planner from [12].

### A. Overtaking on the Motorway

The overtaking scenario is demonstrated in Fig. 4. A motorway has three lanes in one direction with a speed limit of $50\,\mathrm{m\,s^{-1}}$. The lanes have the same length and their coordinates are aligned at the same start point. The ego vehicle is on the right lane $L_0$ at distance $0\,\mathrm{m}$ with a speed of $35\,\mathrm{m\,s^{-1}}$. There is another vehicle on the middle lane $L_1$ $80\,\mathrm{m}$ ahead with a speed of $30\,\mathrm{m\,s^{-1}}$. The goal is driving to $1000\,\mathrm{m}$ on any of the three lanes. According to German traffic rules [18], overtaking from the right is forbidden on the motorway. In addition, a vehicle should drive on the most possible right lane on the motorway. Therefore, the ego vehicle can choose its behavior as either staying on the right lane and reducing its speed or switching to the left lane $L_2$, overtaking the slow vehicle, and switching back to the right lane. Following the slow vehicle in the middle lane is not allowed because the right lane is not occupied.

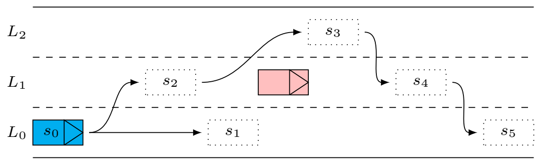
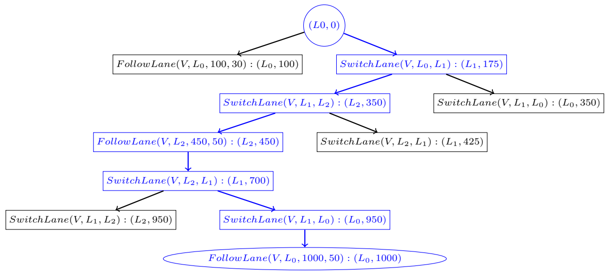


Fig. 4. Overtaking scenario: There are three lanes in one direction of a motorway. The blue rectangle is the ego vehicle and the red rectangle is a slower vehicle in front. The initial state is $s_0$. The arrows show the possible driving tasks planned with the internal states $\{s_1, s_2, s_3, s_4, s_5\}$.

Assuming the sensing range of the vehicle is $100\,\mathrm{m}$, so each lane-following task takes a distance parameter as $100\,\mathrm{m}$. The lane-following speed is the minimum value of the speed limit, the speed of the vehicles in front or on the left within a safety distance. Thus, a vehicle will not overtake other vehicles from the right or threaten the vehicle in front. Another precondition for lane-following task is that the lane on the right should not be free in a certain look-ahead distance. The lane-switching task has a constant time cost of $5\,\mathrm{s}$. The planning result is presented in Fig. 5, with a solution that first switches twice to the left lane, overtakes the slow vehicle, then switches back to the right lane and follows it to the target location.

If the overtaking problem is solved by a state machine approach, the states are defined with lane positions or types of driving maneuvers. Each possible transition between the states should be defined regarding the traffic rules and conditions. The state machine can only be applied to this specific scenario. Each tiny change of the traffic rules requires to modify the state machine, e.g., overtaking from the right is allowed in the urban street. In contrast, the task planning approach has a generic planner, which can handle infinite situations with a right selection of the task set.



$(L0, 0)$
$FollowLane(V, L_0, 100, 30) : (L_0, 100)$
$SwitchLane(V, L_0, L_1) : (L_1, 175)$
$SwitchLane(V, L_1, L_2) : (L_2, 350)$
$SwitchLane(V, L_1, L_0) : (L_0, 350)$
$FollowLane(V, L_2, 450, 50) : (L_2, 450)$
$SwitchLane(V, L_2, L_1) : (L_1, 425)$
$SwitchLane(V, L_2, L_1) : (L_1, 700)$
$SwitchLane(V, L_1, L_2) : (L_2, 950)$
$SwitchLane(V, L_1, L_0) : (L_0, 950)$
$FollowLane(V, L_0, 1000, 50) : (L_0, 1000)$

Fig. 5. Task planning for the overtaking scenario. The task and vehicle position of each node is listed. The solution is colored in blue.

### B. Round-About with Blockage

In the second scenario, the vehicle is driving through a round-about, where the initially planned exit is blocked. The scenario is illustrated in Fig. 6. For simplicity, the round-about has only one entrance and three exits with four lanes $L_0, L_1, L_2, L_3$ and four junctions $J_0, J_1, J_2, J_3$. The lanes $L_4, L_5, L_6, L_7$ in the round-about is $20\,\mathrm{m}$ long. The connection priorities are designed to let the vehicle take precedence while driving inside the round-about. The speed limit of all the lanes is $10\,\mathrm{m\,s^{-1}}$.

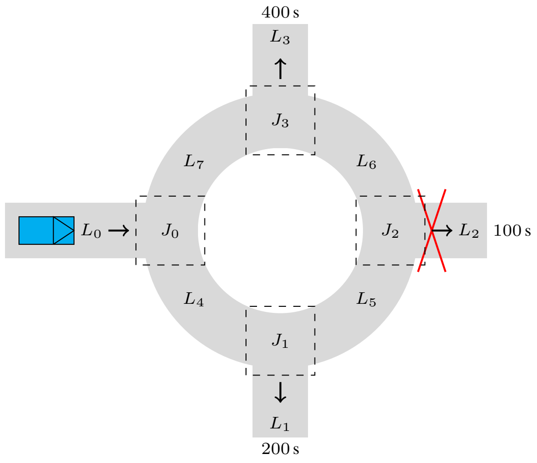


Fig. 6. Round-about scenario: $L_0$, $L_1$, $L_2$ and $L_3$ are lanes connecting to the round-about with the arrow directions. The four lanes in the round-about, $L_4$, $L_5$, $L_6$ and $L_7$, are counterclockwise directed. The four junctions, $J_0$, $J_1$, $J_2$ and $J_3$, connect the adjacent lanes. The blue rectangle is the vehicle coming from lane $L_0$ and can choose one of the three exits with different time to goal values. The red cross shows a temporary blockage at $L_2$, which can be detected only after the vehicle enters lane $L_5$.

The ego vehicle starts at the end of lane $L_0$ and the goal is at any of the three exits $L1, L2, L3$ with different time to goal values. The time cost of a lane-changing in a junction is $5\,\mathrm{s}$. Thus, the cost differences between the three exits are larger than driving around the whole round-about. The sensing range of the vehicle can only reach the next junction in this clustered environment. The optimal exit to $L_2$ is chosen at first as an uncertain solution in Fig. 7. After detecting the blockage, the vehicle continues driving in the round-about to the exit at lane $L_1$ with the second best cost, rather than the closer exit to $L_3$.

Alternative, the planner can assume that each uncertain

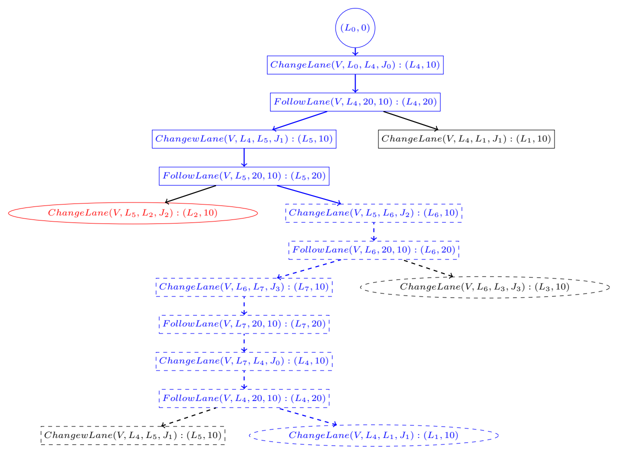

$(L_0, 0)$
$ChangeLane(V, L_0, L_4, J_0) : (L_4, 10)$
$FollowLane(V, L_4, 20, 10) : (L_4, 20)$
$ChangewLane(V, L_4, L_5, J_1) : (L_5, 10)$
$ChangeLane(V, L_4, L_1, J_1) : (L_1, 10)$
$FollowLane(V, L_5, 20, 10) : (L_5, 20)$
$ChangeLane(V, L_5, L_2, J_2) : (L_2, 10)$
$ChangeLane(V, L_5, L_6, J_2) : (L_6, 10)$
$FollowLane(V, L_6, 20, 10) : (L_6, 20)$
$ChangeLane(V, L_6, L_7, J_3) : (L_7, 10)$
$ChangeLane(V, L_6, L_3, J_3) : (L_3, 10)$
$FollowLane(V, L_7, 20, 10) : (L_7, 20)$
$ChangeLane(V, L_7, L_4, J_0) : (L_4, 10)$
$FollowLane(V, L_4, 20, 10) : (L_4, 20)$
$ChangewLane(V, L_4, L_5, J_1) : (L_5, 10)$
$ChangeLane(V, L_4, L_1, J_1) : (L_1, 10)$

Fig. 7. Task planning for Round-about scenario. The dashed tasks are planned after detecting the task $ChangeLane(V, L_5, L_2, J_2)$ in red is inexecutable. The node $ChangeLane(V, L_6, L_3, J_3)$ is also a solution, but $ChangeLane(V, L_4, L_1, J_1)$ is better with less time cost. The solution is colored in blue.

task may fail. When the vehicle plans its tasks at lane $L_0$, it has only enough information for the first task, $ChangeLane(V, L_0, L_4, J_0)$. If a lane-following task fails later, the goal in unreachable. In this case, a specific safety task should be added, which is acceptable as the final state when the goal is unreachable. If an exit lane-changing task fails, the vehicle can choose another connection to continue driving in the round-about. Thus, the entire tree in Fig. 7 is generated as a solution. At junction $J_1$, the vehicle is uncertain about the situations at $J_2$ and $J_3$, so it continues driving to junction $J_2$ as the best choice. At junction $J_3$, the vehicle has all the information about the junctions to decide which exit to take.

If the route is only planned by a navigation system, it will suggest to choose $L_2$ again by driving around the round-about. Additional feedback between motion planning and navigation is required to avoid such a behavior or deadlocks. A task planning can adapt the plan quickly and evaluate all the possibilities to provide a complete solution. The search algorithm checks the duplicated states to avoid a deadlock in the round-about when all the exits are blocked.

## VI. Conclusion and Future Work

This paper proposes a task planning approach to automated driving, based on generic AI planning in a symbolic-geometric hierarchy. It connects the high-level route planning with the low-level motion planning. The power of symbolic planning is that it can handle semantic information, especially the traffic rules. Specific motion planning methods can be applied to different tasks. As a result, the task planning method is more flexible to domain modifications, scales better with increasing complexity, and allows an incremental development. Furthermore, testing and verifying such system is easier than a conventional integrated approach, as the planner and the task domains can be *verified separately*.

As future work, we are going to develop a larger set of driving tasks with thorough traffic rules to evaluate this framework in diverse traffic scenarios. A comparison of different symbolic planning methods from artificial intelligence domain, e.g., Metric-FF, SHOP, is planned to choose the most efficient method for automated driving task planning. Finally, the task planner will be integrated in a real-time demonstrator with navigation and various motion planning methods for field tests.

## References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2009.

[2] R. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

[3] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, *et al.*, "PDDL - the planning domain definition language," Yale Center for Computational Vision and Control, Tech. Rep. CVC TR-98-003/DCS TR-1165, 1998.

[4] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[5] A. Gaschler, R. Petrick, M. Giuliani, M. Rickert, and A. Knoll, "KVP: A knowledge of volumes approach to robot task planning," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 202–208.

[6] A. Gaschler, R. Petrick, T. Kröger, A. Knoll, and O. Khatib, "Robot task planning with contingencies for run-time sensing," in *Proc. IEEE International Conference on Robotics and Automation, Workshop on Combining Task and Motion Planning*, 2013.

[7] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009.

[8] E. Plaku and G. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *Proc. IEEE International Conference on Robotics and Automation*, 2010, pp. 5002–5008.

[9] L. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[10] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[11] C. Chen, M. Rickert, and A. Knoll, "Combining space exploration and heuristic search in online motion planning for nonholonomic vehicles," in *Proc. IEEE Intelligent Vehicles Symposium*, 2013, pp. 1307–1312.

[12] C. Chen, M. Rickert, and A. Knoll, "A traffic knowledge aided vehicle motion planning engine based on space exploration guided heuristic search," in *Proc. IEEE Intelligent Vehicles Symposium*, 2014, pp. 535–540.

[13] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, *et al.*, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[14] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, *et al.*, "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[15] H. Kress-Gazit and G. Pappas, "Automatically synthesizing a planning and control subsystem for the DARPA Urban Challenge," in *Proc. IEEE Conference on Automation Science and Engineering*, 8 2008, pp. 766–771.

[16] J. Rosenblatt, "DAMN: A distributed architecture for mobile navigation," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2/3, pp. 339–360, 1997.

[17] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, *et al.*, "Odin: Team VictorTangos entry in the DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[18] "Verordnung zur Neufassung der Straßenverkehrs-Ordnung (StVO)," in *Bundesgesetzblatt*, Mar. 2013, vol. 1, no. 12, pp. 367–427.