# HTN Planning Deserves Better Domains: Engineering Realistic Models for Autonomous Vehicles in Space and on Earth

Ebaa Alnazer, Ilche Georgievski, Marco Aiello
Service Computing Department, IAAS
University of Stuttgart
Stuttgart, Germany
e-mail: {ebaa.alnazer | ilche.georgievski | marco.aiello}@iaas.uni-stuttgart.de

***Abstract*—Developing operational AI planning systems requires domain models that faithfully capture the characteristics of their corresponding real-world applications. However, existing planning domain models exhibit two key limitations: they often underrepresent realistic aspects, such as alternative ways to accomplish tasks, uncertainty, and risk, and they provide limited coverage of societally relevant applications. We address these limitations in the context of Hierarchical Task Network (HTN) planning, an expressive AI planning technique well-suited for representing rich domain knowledge. We extend well-established domains of Rover and Satellite by incorporating overlooked realistic aspects, and engineer a new Self-Driving Car domain that has not previously been considered in the HTN planning literature. We design and apply a systematic domain-engineering approach grounded in existing methodologies and perform a quantitative structural analysis of the resulting domains. The results show that these domains exhibit increased structural diversity and richer representations of application knowledge. Beyond these results, the analysis highlights structural patterns in how HTN domains evolve when enriched with realistic aspects. We argue that these types of contributions not only bring planning domains closer to real-world applications but also provide a stronger foundation for advancing HTN planning frameworks and for developing and evaluating HTN planners.**

***Keywords-HTN planning; knowledge engineering; domain models; risk; uncertainty; alternative methods.***

## I. Introduction

Intelligent autonomous systems are increasingly deployed in safety-critical and high-stakes environments, such as self-driving cars, satellites, and planetary rover missions to Mars. To operate reliably in such settings, these systems must reason about their actions and generate plans that achieve complex objectives under operational and safety constraints. These capabilities are studied in AI planning, a subfield of AI concerned with computing courses of action that, when executed from a given initial state, achieve specified objectives [2].

Among planning paradigms, HTN planning has attracted sustained interest due to its expressive power and its ability to represent structured application knowledge [3]. By decomposing tasks into subtasks using methods, HTN planning supports the explicit modelling of procedural knowledge and multiple alternative ways to achieve tasks. This makes it particularly suitable for complex, real-world applications.

While planning algorithms have advanced significantly, comparatively less attention has been given to the systematic engineering of domain models, including HTN ones. AI planning is a model-based technique, meaning to construct valid and executable plans, accurate and adequate knowledge about the application domain must be elicited, conceptualised, and formalised as a domain model. In HTN planning, this dependency is especially strong, as rich application knowledge is explicitly encoded in task hierarchies and decomposition methods. Constructing such domain models requires eliciting domain expertise, identifying relevant operational aspects, and translating them into formal structures, a knowledge-engineering process that remains labour-intensive and often ad hoc [4][5].

Many benchmark HTN domains, particularly those derived from classical benchmarks and introduced through International Planning Competitions (IPCs), are intentionally simplified to align with planners' capabilities and support their evaluation [6]. They often (1) restrict alternative task decompositions to a minimal set, (2) assume deterministic action outcomes or fixed costs, and (3) omit explicit modelling of risk and uncertainty. As a result, domains that shape much of HTN planning research abstract away characteristics intrinsic to real-world environments. This creates a discrepancy between the expressive potential of HTN planning and the simplified benchmarks typically used to evaluate the planners. This discrepancy motivates our central focus: how to engineer HTN planning domain models that more faithfully capture realistic, application-inherent aspects while remaining structurally analysable and comparable.

Building on our previous work [1][7][8], here we integrate three complementary elements into a single perspective on HTN domain engineering: the structured identification of realistic domain aspects, the incorporation of richer alternative task structures and explicit modelling of uncertainty and risk in HTN domain models, and the quantitative structural analysis of the resulting domain models. Together, these elements allow us not only to enrich domain models but also to assess how such enrichment changes their properties.

We propose a systematic domain-engineering approach grounded in existing knowledge-engineering processes [5][9][10]. We apply a conceptual framework for identifying and capturing realistic aspects [11], which guides the elicitation and structured incorporation of alternative courses of action, uncertainty in action costs, and risk. These aspects are modelled within the state-based framework of HTN planning [3] and its risk-aware extension [8], where risk and uncertainty are represented through probability distributions over action costs. Finally, we propose a set

of metrics to enable a quantitative assessment of domain model properties, such as structural characteristics and domain richness, thereby enabling a more systematic comparison between simplified benchmarks and their enriched or newly engineered counterparts.

To demonstrate and analyse our approach, we pursue two complementary strategies. First, we revisit two benchmark domains, Satellite and Rover, which are widely used in both classical and HTN planning research (e.g., [6][7][12]). Their prominence makes them representative of modelling assumptions prevalent in the field. At the same time, they represent autonomous vehicles in space operations, an inherently uncertain and risk-sensitive application area. They also capture important real-world planning challenges, such as coordinating multiple satellite or rover missions under strict resource constraints, including limited power, restricted target access, and high operational costs [13][14]. This makes them suitable case studies for extending simplified benchmarks to incorporate richer alternative task structures and probabilistic cost representations within risk-aware HTN planning.

Second, we engineer a new HTN domain for self-driving cars. Unlike Satellite and Rover, this domain is not derived from an existing benchmark but developed from scratch, informed by expert knowledge collected from the literature, allowing realistic aspects to be embedded during initial domain construction. The terrestrial autonomous driving setting provides a contemporary example of a realistic domain whose characteristics have received limited attention in AI planning. Many of these characteristics pose significant challenges for both modelling and solving planning problems. They arise from the domain's inherent dynamics and uncertainty, including road incidents, varying road conditions, risk factors, diverse driving tasks, and resource constraints, all of which require continuous tracking of the vehicle's state and its surrounding environment.

Taken together, we position HTN domain engineering and analysis as the central focus for realistic domain models for the first time. By integrating structured aspect identification, risk-aware modelling, systematic domain enrichment, and quantitative structural analysis, we provide a consolidated and extended account of how HTN planning domains can be developed to better reflect the complexity of real operational environments.

The remainder of the paper is organised as follows. Section II provides the necessary background. Section III contains the approach we follow to engineer HTN planning domains. Sections IV and V present our extensions and engineering of existing Satellite and Rover HTN planning domains with more alternatives and risk-awareness, respectively. Section VI contains our approach to engineering the Self-Driving Car domain. Section VII provides our quantitative analysis of the developed HTN planning domains, and Section IX presents discussions beyond the presented domains and concluding remarks.

## II. Background

HTN planning is a widely used AI planning technique across many real-world domains, such as robotics, e.g., [15], healthcare, e.g., [16], and building automation, e.g., [17]. It enriches planning models with procedural knowledge about how tasks can be accomplished [3]. This domain knowledge, together with the intrinsic hierarchical structure of tasks, enables HTN planning to naturally capture structured decision-making processes.

An HTN planning problem is defined by an initial state, an initial *task network* representing the objective to be achieved, and domain knowledge consisting of networks of *primitive* and *compound* tasks. A task network represents a hierarchy of tasks, each either primitive or compound, and may include constraints that restrict their ordering. Primitive tasks correspond to actions that can be executed directly, whereas compound tasks must be decomposed into subtasks via *methods*, which describe how to accomplish them. In *state-based HTN planning*, the search starts at an initial state with an empty plan. The objective is to construct a plan that accomplishes the initial task network. During the search, when a compound task is encountered, it is decomposed using an *applicable* method, i.e., a method that can decompose the task and whose *preconditions* are satisfied in the current state. Decomposition refines the task network at the next hierarchical level while remaining in the same state. When a primitive task is encountered, it is executed, removed from the task network, and appended to the plan, after which the search proceeds to the resulting successor state.

Risk-aware HTN planning is a framework that extends classical HTN planning with constructs that consider the *uncertainty* of real-world environments [8]. It enables modelling of risk- and uncertainty-inducing actions through probability distributions over their costs and effects, where costs are defined as unbounded negative functions. The framework can be tailored for planning problems where actions have deterministic effects but variable costs. In our modelling approach, we adopt this variant as an initial step toward incorporating risk and uncertainty into HTN planning domain models.

In risk-aware HTN planning, the cost functions of actions can be of different types depending on the factors/sources of the costs. The cost function can be (1) *external* ($c^{ie}(a)$), i.e., it depends on external factors not explicitly modelled in the domain, such as market electricity prices or taxes, (2) *state-dependent* ($c^{is}(a)$), i.e., it is based on the system's current state, like the vehicle's charging level or position, (3) *constant* ($c^{ic}(a)$), i.e., it remains the same for every execution of an action, such as a fixed charging price, or (4) external and state-dependent, i.e., a *hybrid function* ($c^{ies}(a)$), which depends on both current state and external factors, where $a$ denotes an action.

## III. Engineering HTN Domain Models Approach

We design a systematic approach to ensure that HTN planning domains are systematically enhanced or engineered with realism in mind, supporting the development of planning

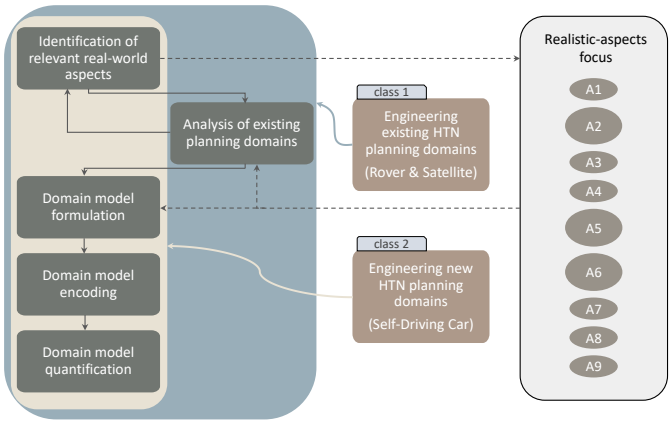



Figure 1. Approach for engineering HTN planning domains and classes of HTN planning domains.

models that are both expressive and applicable to real-world settings.

The approach is illustrated in Figure 1. It consists of several sequential phases, as suggested in the existing literature on knowledge engineering for planning [4][5][9][10], and distinguishes two classes of domains based on whether domain models are already available. *Class 1* concerns domain models that already exist and can be extended to increase realism, while *Class 2* focuses on domain models for which no domain model exists and must be engineered from scratch.

The approach begins by identifying the relevant aspects of the application to be reflected in the resulting domain model, regardless of class. When engineering knowledge for a Class 1 domain model, this phase must be combined with an analysis of the existing domain model. Be it a Class 1 or Class 2 domain, the next phase is to map the identified aspects into appropriate HTN constructs to formulate the extended or newly engineered domain model. The formulation of domain knowledge is followed by a phase in which the final outcome is produced by encoding the domain model using a planning language. Finally, the last phase is a quantitative assessment to evaluate how well the domain models reflect targeted aspects.

### A. Identification of Relevant Real-World Aspects

HTN planning is a model-based technique in which planning systems rely on relevant and adequate knowledge of the domain in which they are supposed to operate [4][10]. Engineering such knowledge requires domain requirements analysis, which entails identifying relevant aspects that characterise the application domain [5]. We therefore apply an existing conceptual framework for identifying and categorising aspects of real-world planning domains [11]. We select seven aspects of the framework (**A1**–**A7**) and complement them with two additional necessary aspects (**A8** and **A9**) related to the real-world grounding of planning domains [7]. We refer to those two as *foundational aspects*. We place particular emphasis on **A2**, **A5**, and **A6**, as they capture alternative ways of accomplishing tasks, uncertainty, and risk associated with action costs.

**A1:** An important aspect in engineering domain models is defining their *objectives*. We consider hard goals or requests, which define a mandatory behaviour of plans generated by the planning system, according to which planning problems in the domain should be solved.

**A2:** Another key aspect concerns the granularity of *tasks and their hierarchical relationships*. Higher-level tasks represent abstractions that can be decomposed into subtasks, forming a hierarchy that naturally introduces structured causality and supports reasoning across different abstraction levels. A high-level task may admit multiple refinement options, i.e., alternative ways for accomplishing the same task. These refinements may only be valid under specific *constraints*.

**A3:** In addition to constraints restricting which refinements are possible in a given world state, we consider constraints governing the order in which tasks should be performed. Depending on the scenario, tasks may follow a total order, a partial order, or no order at all.

**A4:** It may be necessary to define who or what executes actions in the domain. These actors may include human agents, non-human agents, or mixed teams. We refer to them collectively as *agents*.

**A5:** Understanding the sources of uncertainty present in real-world environments is particularly important. When considering the executing agents, uncertainty can originate internally, from the agent itself (e.g., system reliability, human limitations, irrational behaviour), or externally, from environmental factors beyond the agent's control. Both internal and external sources can further be classified as either random (stochastic, rare, and unpredictable) or regular (pattern-driven and consistent).

**A6:** Executing actions alters the environment and incurs costs, i.e., consumes resources, such as money, time, fuel, or effort, which are predictable in a certain world. These costs represent one type of *quantity* that should be considered. In a fully predictable world, such costs are fixed. Under uncertainty, however, action costs become variable and may differ each time an action is executed. This variability stems from the underlying source of uncertainty. When the source is random, cost variability is unpredictable and cannot be addressed in offline planning. When the source is regular, the variability can be more precisely defined. When cost distributions of actions are known or statistically inferable, actions are considered *risk-inducing*. When the distributions instead reflect the decision maker's beliefs, the actions are *uncertainty-inducing*.

**A7:** Another salient consideration when identifying relevant aspects is the *structural diversity* of a domain. Planning domains should exhibit structural diversity by capturing the specific characteristics of the real-world applications they represent, rather than merely reproducing tasks that appear in other, similar domains. Many benchmark domains, for example, *Logistics*, *Rover*, and *Transport*, share common task structures, such as moving between locations. While such tasks are essential, modelling domains mainly around these generic task structures risks producing domain models that are structurally similar and insufficiently representative of their intended applications. To achieve meaningful structural

diversity, domain models should incorporate knowledge specific to the application context.

**A8:** One foundational aspect concerns the extent to which a domain model represents an *actual or plausible application* of planning technology, rather than an oversimplified or toy problem.

**A9:** The second foundational aspect addresses *evolving technological awareness*. It highlights the expectation that domain models should, when appropriate, reflect recent technologies and emerging trends in real-world applications. By doing so, planning models remain relevant and aligned with technological progress rather than being anchored solely in static or outdated assumptions.

### *B. Analysis of Existing Planning Domains*

The Analysis of Existing Planning Domains phase involves examining the current domain representation, including its tasks, methods, actions, and structural relations, to determine how the identified aspects are currently reflected in the domain model. In doing so, the analysis highlights gaps or simplifications in the existing representation that prevent the domain from capturing important characteristics of the underlying application. The insights gained during this analysis may also prompt refinement of the aspects identified in the previous phase, leading to an iterative process between aspect identification and domain analysis until a consistent understanding of the domain model is achieved.

### *C. Domain Model Formulation*

The next phase concerns formulating domain knowledge to construct the actual domain model [4]. A *domain model* is an abstract conceptual description of an application domain used to represent knowledge within a planning application. This conceptual description is derived from knowledge obtained in the preceding phase(s) and covers domain dynamics in relation to the types of problems the planning engine must solve, and the types of plans (solutions) that need to be produced as output [9]. Our focus is on state-based HTN planning and its risk-aware variant.

The domain model formally describes persistent knowledge and represents entities that are invariant across all planning problems [9][18]. These include objects with their relations and properties, actions that can change the environment's state, and other constructs, such as tasks and methods in HTN planning. A corresponding *problem instance* is needed to formally describe specific planning scenarios, which include the initial world state and the goal to be achieved.

HTN methods are particularly important as they model alternative ways to achieve tasks. We propose a categorisation of HTN methods based on their potential to serve as alternatives to other methods, distinguishing between strict alternatives, weak alternatives, and no alternatives [7]. Methods are considered strict alternatives if they share the same components and their preconditions do not contradict each other, allowing them to be simultaneously applicable. In contrast, two or more methods are classified as weak alternatives when their preconditions differ but are not mutually exclusive. In such cases, the domain model alone cannot determine whether the methods will be applicable in the same world state; this can only be resolved during planning, when variables are instantiated with concrete objects from the problem instance. A method is classified as having no alternative if it is the sole method capable of decomposing a given task, or if its preconditions (or the preconditions of the actions resulting from the decompositions of its corresponding task) contradict those of all other methods associated with that task. Formulating all types of alternatives entails that the added methods for task decomposition may introduce new task networks that include compound and primitive tasks not present in the original model of Class 1 domains.

Similarly, incorporating risk awareness is important as real-world applications often involve uncertainty that affects the outcomes and resource consumption of actions. In many situations, the exact cost of performing an action (e.g., resource consumption) cannot be defined in advance but can instead be characterised by a probability distribution. When such distributions are known or can be inferred, i.e., when risk-inducing actions are taken, the situation corresponds to decision-making under risk. Risk can be incorporated into HTN planning domains by modelling action costs as probability distributions. This representation allows planning decisions, such as choices between alternative methods, to be evaluated with risk attitude. Thus, incorporating risk into HTN planning domains may also entail enriching them with multiple alternatives for achieving tasks.

### *D. Domain Model Encoding*

The next phase is to encode domain models (and problem instances) in well-defined syntax, such as the Hierarchical Planning Definition Language (HPDL) [19], Hierarchical Domain Definition Language (HDDL) [20], or another suitable planning language. In our approach, we use HDDL and our own extension of it to enable risk-aware encoding [1][8]. We refer to this HDDL variant as *risk-aware HDDL* (rHDDL), which allows encoding actions with probability distributions over their costs, representing each possible cost alongside its associated probability.

### *E. Domain Model Quantitative Analysis*

As a final phase of our approach, we analyse the resulting domain models quantitatively to better understand their properties. The purpose of this analysis is not to evaluate planner performance but to examine how the undertaken engineering efforts affect the structural characteristics of the resulting HTN domain models.

Several of our introduced aspects, such as risk-aware action costs, influence planning behaviour during execution rather than the structural composition of a domain model itself. Since no metrics currently exist to quantify these properties at the modelling level, our analysis focuses on aspects that can be assessed directly from the domain structure. In particular, we focus on structural characteristics of engineered domains and analyse them using the following *structural metrics* [7]:

- total number of methods **#m**,
- number of methods that are strict alternatives **#ms**,
- number of methods that are weak alternatives **#mw**,
- number of methods that do not form an alternative **#mn**,
- number of tasks **#t**,
- number of direct recursive tasks **#rt**, where the direct recursive task is the task that is contained in the task network of one of the methods that can decompose it, and
- number of methods that have $x$ subtasks **#m-tn**.

Beyond structural metrics, further insight into domain realism can be obtained by examining *knowledge richness*, a characteristic enabled by the expressive power of HTN planning. Since no widely accepted definition of knowledge richness exists, one might argue that knowledge richness can be quantified by the number of tasks, methods, alternative methods, or expressive constructs. However, objectively assessing whether the addition of such constructs truly enriches domain knowledge remains challenging. Instead, we adopt a more pragmatic approach and measure knowledge richness using metrics that capture the diversity of domain objects and their relationships. The *domain-richness metrics* are [7]:

- the number of predicates **#p**,
- the number of predicates that define relations between two or more objects **#pm**,
- the number of child object types, corresponding to the leaves of the type hierarchy **#ot**,
- the number of object types that serve as parents in the type hierarchy **#otp**, and
- the number of actions **#a**.

### F. Application Domains

We apply our domain-engineering approach to three application domains. To illustrate how different realistic aspects can be incorporated into HTN planning domains, we consider domains that emphasise different characteristics of their underlying applications. For Class 1, we focus on space missions involving rovers and satellites. In particular, we select the Rover domain model [21],[1] and the Satellite domain [22],[2] The Rover domain has been used as a benchmark in IPCs, while the Satellite domain we consider is a hierarchical adaptation of a classical IPC benchmark, used to evaluate specific planning approaches, e.g., [22]–[24]. We extend the Rover domain by focusing on alternative decompositions, reflecting the multiple operational strategies available in planetary exploration missions. In contrast, we extend the Satellite domain by focusing on risk awareness, capturing the uncertainty and variability inherent in space operations. For Class 2, we focus on terrestrial driving involving autonomous cars. We develop an HTN domain model from scratch while focusing on both alternative decompositions and risk awareness, reflecting the dynamic and uncertain nature of autonomous driving environments.[3]

## IV. Class 1: Rover Domain with Alternative Decompositions

The Rover application domain involves multiple autonomous rovers operating in planetary exploration scenarios. These rovers must navigate across a set of predefined waypoints to perform scientific tasks such as collecting rock or soil samples, capturing photographs of designated targets, and transmitting the gathered data back to a lander for further analysis.

We enrich the Rover domain by introducing additional alternative task decompositions. Accordingly, our analysis focuses on aspects related to task structure and decomposition, namely **A1**, **A2**, **A3**, **A4**, and **A7**. As this extension does not introduce risk awareness, aspects related to uncertainty and quantities (i.e., **A5** and **A6**) are outside the scope of the engineering process. We additionally discuss the domain with respect to the foundational aspects **A8** and **A9**. In the following, we describe all phases of the approach, but the last one, which has its own dedicated section. Moreover, we merge the first and second phases for a cleaner presentation and discussion.

### A. Identification of Relevant Real-World Aspects and Analysis of the Existing Domain

Rovers have the objective of exploring planetary environments by investigating targets and performing multiple tasks, such as collecting data and samples and taking images of various objects [25] (**A1**). The existing planning benchmark [26][27] reflects several of these activities, where tasks are organised across multiple abstraction levels and exhibit structured causality, with complex tasks achieved through the decomposition into multiple subtasks (**A2**). In this domain model, multiple rovers explore a planetary environment by navigating between locations, collecting rock and soil samples, and taking images of target objects. Getting a rock or soil sample involves navigating to the data location, emptying the rover's storage, collecting the sample, navigating to a location from which the lander is visible, and, lastly, transmitting the collected data to the lander. The execution of these tasks should follow this total order (**A3**). Getting images follows a similar structure. It involves calibrating the camera, navigating to a location from which the target is visible, capturing the image, and relaying it to the lander by navigating to a location from which the lander is visible and transmitting it to the lander. The rovers and the lander represent the agents operating in this domain (**A4**).

Rovers are designed to operate on specific types of terrain and may therefore be unable to traverse certain areas. With technological advancements, however, the capabilities of rovers for planetary exploration have expanded. In particular, *space drones* have been introduced to support rovers, and perhaps astronauts in the future, in exploring terrains that are hard to reach by rovers. Space drones are "small uncrewed vehicles

[1]https://github.com/panda-planner-dev/ipc2020-domains/tree/master/partial-order/Rover

[2]https://github.com/panda-planner-dev/domains/tree/master/total-order/Satellite-PANDA

[3]The full domain models are available on Github: https://github.com/PlanX-Universe/htn-planning-domain-models.

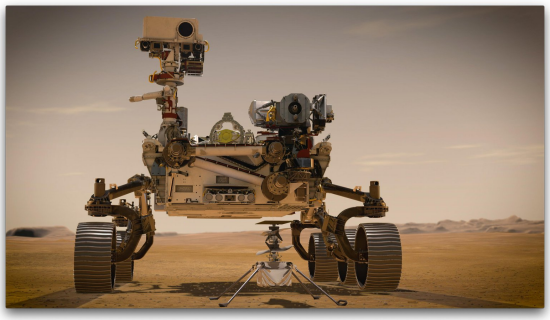


Figure 2. NASA's Mars 2020 Perseverance rover and NASA's Ingenuity Mars Helicopter. Courtesy NASA/JPL-Caltech.

guided by remote or autonomous means" [28]. Figure 2 shows a notable example, which is *Ingenuity* (also known as the Mars Helicopter), the first space drone sent to Mars by NASA in July 2020. This drone is attached to the *Perseverance rover*, which landed safely on Mars in February 2021 [29].

Extending the Rover domain by allowing rovers to request assistance from space drones during exploration missions, therefore captures additional characteristics of contemporary planetary exploration missions. In particular, adding space drones as helpers for rovers broadens the range of achievable tasks and objectives and introduces alternative ways of accomplishing space mission tasks (**A2**), an aspect lacking in the original domain.

The extension can include several alternatives: a space drone may send collected data to a lander, a space drone may collect data instead of a rover, and data gathering may be a collective effort between rovers and drones. Extending with the first alternative, for example, requires the extended domain to contain drones located at different positions such that each drone belongs to a rover. In this setting, each task of communicating image, soil, or rock data to the lander can be accomplished in two ways. The first option is for the rover to communicate the collected data directly to the lander by navigating towards it. The second option allows the rover to delegate this task to a drone. To send data to the lander using a space drone, additional tasks with their ordering constraints should be introduced (**A2** and **A3**). In particular, the drone navigates to the rover's location, the rover transfers the data to the drone, the drone navigates to a location from which the lander is visible, and lastly, it transmits the data to the lander. In the extended domain, space drones also represent an additional category of agents (**A4**).

Regarding **A7**, the original Rover domain already includes tasks that capture the specificities of the application, such as communicating rock data to the lander. It therefore goes beyond merely restating generic tasks that appear across many domains, such as navigation. Extending the domain with space drones to help rovers further exploit these application-specific characteristics by introducing operational capabilities.

The original Rover domain also covers **A8**, as it was modelled on the basis of an actual application of planning technology, namely the Mars exploratory rover mission of 2003, rather than an oversimplified or toy scenario. However, planetary exploration missions have evolved significantly since then, with substantial advances in capabilities and technology. One such development is the integration of space drones as assistants to rovers, which can be viewed as a transformative technology in rover missions. By incorporating drones, the extended Rover domain captures these developments and therefore addresses **A9**.

### *B. Domain Model Formulation*

Having identified the relevant realistic aspects of the Rover domain, we now formulate them within the HTN domain model. The knowledge about rover exploration activities is represented using compound and primitive tasks. Compound tasks capture abstraction levels, causal relationships, recursion, conditions, and the representation of alternative ways of achieving tasks. Each compound task can be decomposed into subtasks through multiple methods.

An abstract graphical overview of the resulting HTN domain model is presented in Figure 3, where blue nodes denote compound tasks, orange nodes denote primitive tasks, and grey nodes $v_{m_0}, v_{m_1}, \ldots, v_{m_{13}}$ denote methods. Nodes highlighted in green correspond to our extensions. For clarity, the figure shows only the portion of the domain related to sampling and transmitting soil data. The domain components for rock sampling follow analogous tasks and methods, while the image-taking part differs slightly. Nevertheless, the extensions that allow rovers to obtain assistance from space drones when transmitting data to the lander involve similar tasks and methods to those used in soil sampling. In the domain model formulation and encoding, we explicitly implement the alternative in which space drone assistance is used during data transmission to the lander. The remaining alternatives, where drones collect data independently or jointly with rovers, can be formulated and modelled using the same approach.

The domain includes three top-level tasks, `get_soil_data`, `get_rock_data`, and `get_image_data` to obtain soil data, rock data, and take images, respectively. The `get_soil_data` compound task is decomposed using method $v_{m_0}$ into four totally ordered tasks; `navigate_abs`, `empty_store`, `sample_soil`, and `send_soil_data`. The `navigate_abs` compound tasks is further decomposed using $v_{m_1}$ into the primitive task `visit`, which marks the waypoints that the rover visits as visited, the compound task `navigate_abs1` to navigate through waypoints to the data location, and `unvisit` to mark the waypoints that the rover is not at anymore as unvisited when the rover arrives at the data location. The `navigate_abs1` compound task is further decomposed by three methods $v_{m_2}$, $v_{m_3}$, and $v_{m_4}$. $v_{m_2}$ decomposes the task into an empty task network when the rover is already at the destination. $v_{m_3}$ decomposes the task into a single primitive task `navigate` if the rover can directly traverse to the destination waypoint and that waypoint is visible from the rover's location. $v_{m_4}$ decomposes the task into four further tasks: `navigate` to move to the next waypoint, `visit` to

mark the current location as visited, `navigate_abs1` which is recursively decomposed to traverse all waypoints until reaching the destination, and lastly, `unvisit` to mark the traversed waypoint as unvisited. The `empty_store` task is decomposed by two methods; $v_{m_5}$, which results in an empty task network if the store is already empty, or $v_{m_6}$ if the store is full and needs to be emptied by executing the primitive task `drop`. The task `sample_soil` is primitive and can be executed if the rover is equipped for soil sampling and the rover's store is empty. Lastly, the `send_soil_data` task is a compound task for sending the data to the lander and can be decomposed by two alternative methods that the rover can choose from $v_{m_7}$ and $v_{m_8}$. The first method is for the rover to send the data to the lander without the space drone and it decomposes the task into two tasks `navigate_abs` for the rover to navigate to a location from which the lander is visible, and `communicate_soil_data`, which is a primitive task for sending the data to the lander. Method $v_{m_8}$ is for getting help from the space drone to send the data to the lander. This method decomposes the task into 4 tasks. The first task is `navigate_abs_drone`, which is for the drone to go to the location of the rover, and it involves similar tasks and methods as the `navigate_abs`, with the exception that the space drone, unlike the rover, can traverse all locations without restrictions. This task is followed by the `transfer_soil_data_to_drone` primitive task to transfer the data from the rover to the space drone and `navigate_abs_drone`, where the drone navigates to a location from which the lander is visible. Lastly, the primitive task `communicate_soil_data_drone` is performed to transmit the data to the lander.

The `get_rock_data` task is decomposed similarly to the `get_soil_data` task. However, `get_image_data` has slight differences. This task is decomposed by a single method to the tasks `calibrate_abs`, `navigate_abs`, `take_image`, and `send_image_data`. To calibrate the camera, the `calibrate_abs` task is decomposed into two tasks `navigate_abs` to navigate to the calibration target, and `calibrate` to calibrate the camera. After calibrating, the `navigate_abs`, `take_image`, and `send_image_data` tasks are performed to navigate to the location from which the targeted object is visible, take an image, and send the image data to the lander similar to the tasks for sending the soil and rock data, respectively.

In Appendix A-A, we provide an example of possible planning problems in this domain.

### C. Domain Model Encoding

To implement the proposed extensions, we encode the extended Rover domain in HDDL, the same language used in the original domain specification. Listing 1 shows the method that can be used to communicate soil data to the lander by the drone. The method decomposes the `send_soil_data` task into the following subtasks: the drone navigates to the rover's location, the rover transfers the data to the drone, the drone navigates to a location from which the lander is visible, and lastly, it communicates the data to the lander. This method requires that the drone belongs to the rover and that the lander is visible from the location to which the drone should navigate.

Listing 2 shows the method used to decompose the task of navigating the drone to a desired location. This method is modelled similarly to the navigation method of the rover. In particular, it is defined as a recursive method that eventually decomposes `navigate_drone_abs` into primitive tasks for navigating one step at a time between connected locations. However, unlike rovers, which may be restricted by terrain types, a drone can move to a location provided that the location is visible from its current position. Finally, Listing 3 shows the action used to communicate soil data from a drone to a lander. The action requires that the location of the lander is visible from the drone's location and that the drone has the collected soil data. As an effect, the soil data has been transmitted, the drone no longer holds the data, and the communication channel of the lander becomes available again.

Listing 1: Method for sending soil data by drone.

```
(:method m-send_soil_data_drone
 :parameters (?rover - rover
              ?waypoint ?x ?y ?z - waypoint
              ?l - lander
              ?drone - drone)
 :task (send_soil_data ?rover ?waypoint)
 :precondition (and (at_lander ?l ?y)
                (visible ?x ?y)
                (belongsTo ?drone ?rover)
                (at ?rover ?z))
 :ordered-subtasks (and
           (navigate_abs_drone ?drone ?z)
           (transfer_soil_data_to_drone
                 ?rover ?drone ?waypoint)
           (navigate_abs_drone ?drone ?x)
           (communicate_soil_data_drone
             ?drone ?l ?waypoint ?x ?y)))
```

Listing 2: Recursive method for navigating the drone to the desired location.

```
(:method m-navigate_abs_drone-1
 :parameters (?drone - drone
              ?from ?to - waypoint)
 :task (navigate_abs_drone ?drone ?to)
 :precondition (at_drone ?drone ?from)
 :ordered-subtasks (and (visit ?from)
            (navigate_abs_drone-1
                    ?drone ?from ?to)
            (unvisit ?from)))
```

Listing 3: Action for communicating soil data from the drone to the lander.

```
(:action communicate_soil_data_drone
 :parameters (?r - drone
              ?l - lander
              ?p ?x ?y - waypoint)
 :precondition (and (at_drone ?r ?x)
            (at_lander ?l ?y)
            (have_soil_analysis_drone ?r ?p)
            (visible ?x ?y)
            (channel_free ?l))
  :effect (and (channel_free ?l)
```

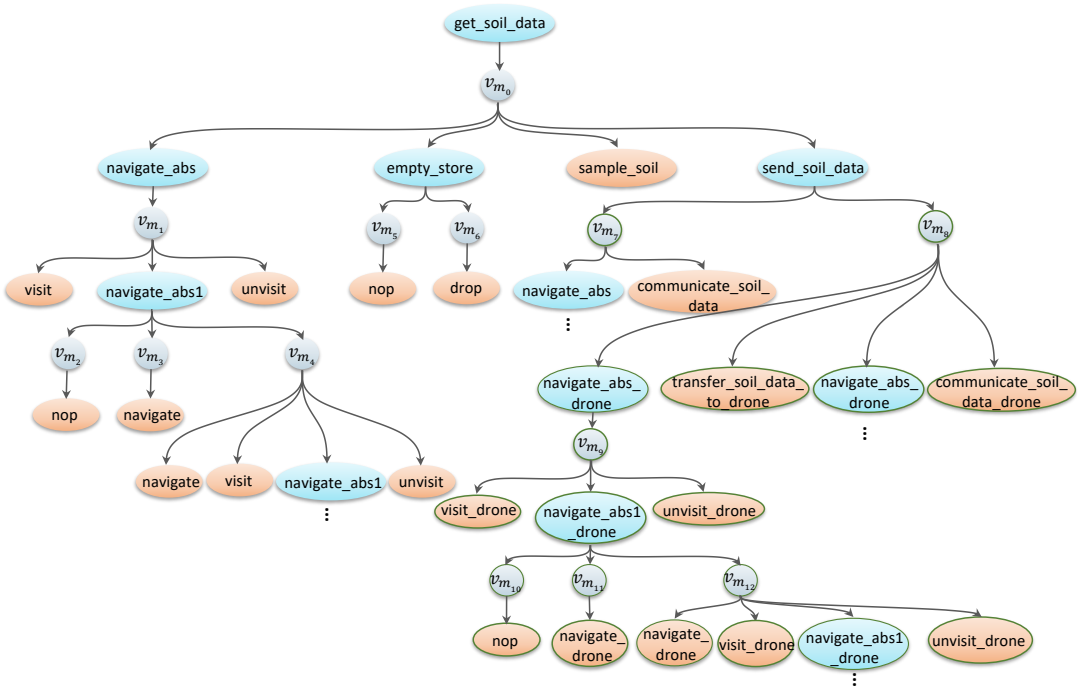



Figure 3. HTN model for the Rover domain.

```
            (communicated_soil_data ?p)
(not (have_soil_analysis_drone ?r ?p))))
```

## V. Class 1: Satellite Domain with Risk Awareness

The Satellite application domain involves space applications with autonomous orbiting spacecraft that perform tasks such as imaging, data collection, navigation, and scientific research. Figure 4 shows an example of such a spacecraft, which is the Ocean Surface Topography Mission/Jason 2 Earth satellite designed to make observations of ocean topography for investigations into sea-level rise and the relationship between ocean circulation and climate change.

We enrich the Satellite domain by introducing risk awareness. Accordingly, our analysis focuses on aspects related to task structure and decomposition, namely **A1**, **A2**, **A3**, **A4**, and **A7**, and aspects related to non-determinism and quantities, namely **A5** and **A6**). We additionally discuss the domain with respect to the foundational aspects **A8** and **A9**. The rest of the section follows the same structure as for the Rover domain.

### *A. Identification of Relevant Real-World Aspects and Analysis of the Existing Domain*

The objective of satellites is to perform stellar observations while optimising factors such as mission time and power consumption (**A1**). This objective must be achieved while considering aspects, such as risk and uncertainty, and the states of satellites and their instruments. Achieving this objective requires satellites to perform multiple activities that have different abstraction levels and causal relationships (**A2**).

Satellites are equipped with several observation instruments, each with specific operation modes such as infrared, spectrograph, X-ray, and thermography, and have defined calibration targets (directions). Conducting a space observation mission is a complex task that involves preparing a satellite and then capturing an image of the target phenomenon. Preparing the satellite requires routing energy to the instrument, properly calibrating it, and orienting the satellite toward the phenomenon to be captured. Once these steps are completed, the satellite can capture the desired image. These activities must follow a specific execution order (**A3**). Note that the satellite itself represents the main agent in this domain (**A4**).

Figure 4. The Ocean Surface Topography Mission/Jason 2 Earth satellite. Courtesy NASA/JPL-Caltech.

In practice, multiple instruments often need to be activated simultaneously due to time and resource constraints [30]. Consequently, the satellite must be capable of powering on several instruments at once and choosing how many instruments to activate, while accounting for faults, such as

power failures. Thus, extending the Satellite domain to support the selection of the number of instruments to power on reflects additional characteristics of the application domain and enables the modelling of realistic alternatives for performing stellar observation tasks (**A2**).

The complexity of satellite missions also arises from the inherent uncertainty of space environments and the technologies involved in satellite operations and instrumentation (**A5**). Planning stellar observations, therefore, requires reasoning about uncertainty and its inherent randomness, specifically the distinction between information available during planning and information revealed only during execution. These uncertainties directly affect resource consumption, such as time and power required to perform actions (**A6**).

Uncertainty in satellite missions comes from both internal and external sources. *Internal sources* are related to the satellite platform and instruments. For example, limitations of physical sensors may affect data quality (e.g., resolution, accuracy, noise), introducing uncertainty into action costs when these are tied to the required data quality [31]. Other internal uncertainties arise from the potential of power system failures, such as battery or solar array malfunctions [32]. This can be caused by power overload if the satellite's energy source (e.g., solar panels, radioisotope thermoelectric generators, or helium cells) cannot meet the demand [33]. Such power overload and potential power loss can occur when powering several instruments in parallel, ultimately increasing execution time and delaying recovery.

*External sources* stem from the satellite's operating environment. Space weather events may cause anomalies such as temporary outages, power failures, and solar cell degradation [34]. Similarly, solar energetic particles or protons can penetrate satellite electronics and cause electrical failures [35].

The Satellite domain is structurally diverse and includes tasks that capture the specific characteristics of satellite operations, such as routing energy to instruments and calibrating them (**A7**). Thus, similarly to the Rover domain, it goes beyond merely restating common or generic tasks that appear across many planning domains. Adding the ability to power multiple instruments at once, however, is a step forward in exploiting these application-specific characteristics.

The domain satisfies **A8**, as it was originally modelled on the basis of an actual application of planning technology at NASA. It also reflects emerging characteristics of contemporary stellar observation missions (**A9**). In particular, modern satellites support multiple observation instruments operating in different modes may activate several instruments simultaneously to maximise observation opportunities.

### *B. Domain Model Formulation*

Next, we formulate the gathered knowledge as a prerequisite to model the Satellite domain. An abstract graphical overview of this HTN domain model is shown in Figure 5. The graphical notation follows the same conventions as in Figure 3, where colours distinguish compound tasks, primitive tasks, methods, and extensions.

Knowledge about stellar observation is represented as compound and primitive tasks. The domain contains a single top-level compound task, `do_observation`, which represents the execution of a stellar observation mission. This task consists of preparing an instrument and taking an image of the target phenomenon. It can be decomposed using four methods, $v_{m_0}$ to $v_{m_3}$, which represent different situational contexts rather than alternative strategies of performing the observation task. Their main differences lie in the preparation phase. Specifically, $v_{m_0}$ performs the full preparation sequence consisting of `activate_instrument`, `turn_to`, and `take_image`. Method $v_{m_1}$ omits instrument activation, assuming that the instrument is pre-calibrated, for example, from previous observations. Method $v_{m_2}$ skips the turning step, while method $v_{m_3}$ performs only `take_image`.

While a satellite can have multiple instruments onboard, the original domain allows only one instrument to be powered at a time per satellite. This constraint is enforced by switching off any active instrument before powering on another and by making the power unavailable once an instrument has been activated. This behaviour is modelled through methods $v_{m_4}$ and $v_{m_7}$. $v_{m_4}$ decomposes `activate_instrument` into three subtasks: `switch_off`, `switch_on`, and `auto_calibrate`. The first two tasks switch off an already powered instrument and activate the instrument corresponding to the required observation mode. The `auto_calibrate` is further decomposed by $v_{m_{10}}$ and $v_{m_{11}}$. Method $v_{m_{10}}$ decomposes the task into `turn_to` and `calibrate` to slew the satellite toward the calibration direction and calibrate the instrument, respectively. Method $v_{m_{11}}$ skips turning, assuming the satellite is already aligned with the calibration target. Method $v_{m_7}$ applies when no instrument is currently powered on, thus it skips switching off.

However, real-world satellite missions often require activating multiple instruments simultaneously due to time and resource constraints [30]. To capture this behaviour, we extend the domain introducing new methods for the `activate_instrument` task that allow satellites to power multiple instruments concurrently and incorporate permissive decomposition.

To model the situation of powering several instruments under the risk of power overload and recovery delays, we introduce methods $v_{m_8}$, $v_{m_9}$, $v_{m_5}$, and $v_{m_6}$. Methods $v_{m_8}$ and $v_{m_9}$ allow additional instruments to be powered by decomposing into `overload` or `superload`, which switch on a second or third instrument, respectively, followed by `auto_calibrate`. Methods $v_{m_5}$ and $v_{m_6}$ instead switch of multiple instruments at once by decomposing into `switch_off_overload` or `switch_off_superload`, followed by `switch_on` and `auto_calibrate`.

These extensions introduce alternative ways of activating instruments. For example, to power a second instrument, the planner may either switch off the first instrument using $v_{m_4}$ or overload the satellite using $v_{m_8}$. Likewise, $v_{m_5}$ and $v_{m_9}$ offer a choice between switching off two instruments or activating a third, resulting in a "superload" state with three active

instruments.

To model non-determinism and action costs, we analyse how internal and external uncertainty sources affect actions that rely on a stable satellite power system. We focus on the cost variability of actions `overload` and `superload`, which place higher demands on power. Specifically, considering that costs are indicative of the time required for actions to be completed, activating a second instrument via the `overload` action has a 99% probability of taking 15 units of time and a 1% probability of taking 100 units of time due to power failure. Action `superload` is riskier as the unfavourable outcome incurs much higher cost, although it has a lower likelihood of occurrence, with a 95% probability of taking 15 units of time and a 5% probability of taking 400 units of time. All other actions are assigned a fixed cost of 15 units with 100% probability, enabling a clear comparison of planner behaviour under different risk attitudes concerning the simultaneous or sequential activation and deactivation of instruments. Here, we use the constant cost function type $c^{ic}(a)$. This means that the action costs are defined within the domain model and remain consistent across different problem instances. However, other types of cost functions and alternative cost distributions can also be considered, as the statistical inferences used to derive them may vary across different satellite domains.

In Appendix A-B, we give examples of possible planning problems in this domain.

### C. Domain Model Encoding

We encode the planning knowledge formulated earlier directly into a risk-aware HTN planning domain model using rHDDL. Listing 4 shows a fragment of the domain model, including the actions `switch_on`, `overload`, and `superload`. The action `switch_on` can be executed only when the satellite's power is available and makes it unavailable once an instrument is activated. In contrast, actions `overload` and `superload` can be executed when the power is already unavailable, but they introduce a risk of power failure. In this model, we use a predicate `(power_avail)` that represents a safety constraint on the maximum power usage. This constraint can be deliberately ignored when choosing to overload or superload the satellite. The potential recovery time from such failures is represented using probabilistic cost distributions via the : $costdist$ construct of rHDDL. A cost distribution specifies alternative recovery costs together with their associated probabilities in the form $(or(p_1(c_1)(p_2(c_2))\dots(p_n(c_n))$, where each expression $(p_i(c_i))$ denotes that cost $c_i$ occurs with probability $p_i$.

Listing 4: Switch on, overload, and superload actions in the Satellite domain.

```
(:action switch_on
 :parameters (?so_i - instrument ?so_s -
    satellite)
 :precondition (and (on_board ?so_i ?so_s)(
    power_avail ?so_s))
 :effect (and (power_on ?so_i)(not(calibrated
    ?so_i))(not(power_avail ?so_s)))
 :costdist (or (1(15))))

(:action overload
 :parameters (?so_i - instrument ?so_s -
    satellite)
 :precondition (and (on_board ?so_i ?so_s)(not
    (overloaded ?so_s))(not(power_avail ?so_s)
    ))
 :effect (and (power_on ?so_i)(overloaded ?
    so_s)(overloads ?so_i ?so_s)(not(
    calibrated ?so_i)))
 :costdist (or (0.99(15)) (0.01(100))))

(:action superload
 :parameters (?so_i - instrument ?so_s -
    satellite)
 :precondition (and (on_board ?so_i ?so_s)(
    overloaded ?so_s)(not(power_avail ?so_s))(
    not(superloaded ?so_s)))
 :effect (and (power_on ?so_i)(superloaded ?
    so_s)(superloads ?so_i ?so_s)(not(
    calibrated ?so_i)))
 :costdist (or (0.95(15)) (0.05(400))))
```

## VI. Class 2: Self-Driving Car with Alternative Decompositions and Risk Awareness

We now turn from extending existing domains to engineering a new planning domain from scratch. To demonstrate the generality of our approach across different operational environments, we consider the application domain of self-driving cars, for which, to our knowledge, no corresponding (HTN) planning domain model currently exists.

Self-driving cars are autonomous transportation systems that navigate with little or no direct human control, typically transporting passengers or performing tasks under human supervision. They operate in environments that are inherently complex and uncertain, characterised by factors such as varying road conditions, road incidents, risk, diverse driving tasks, and resource constraints, including travel time and fuel or battery levels. Safe operation, therefore, requires continuously tracking both the vehicle's internal state and its surrounding environment.

We engineer an HTN planning domain model for this application, which we call *Self-Driving Car*. We analyse all aspects (**A1**-**A9**), with a particular focus on task alternatives and risk awareness.

### A. Identification of Relevant Real-world Aspects

The objective of the Self-Driving Car domain is to transport passengers or goods between locations while satisfying operational constraints such as travel time, energy consumption, and safety requirements (**A1**). Achieving this objective requires considering the vehicle's overall state (e.g., current location), the states of its components (e.g., headlights), and various environmental factors (e.g., weather and road conditions). It also involves a variety of driving activities organised across different levels of abstraction, where complex tasks are realised through the execution of multiple subtasks such as route planning, lane following, obstacle avoidance, and parking (**A2**).

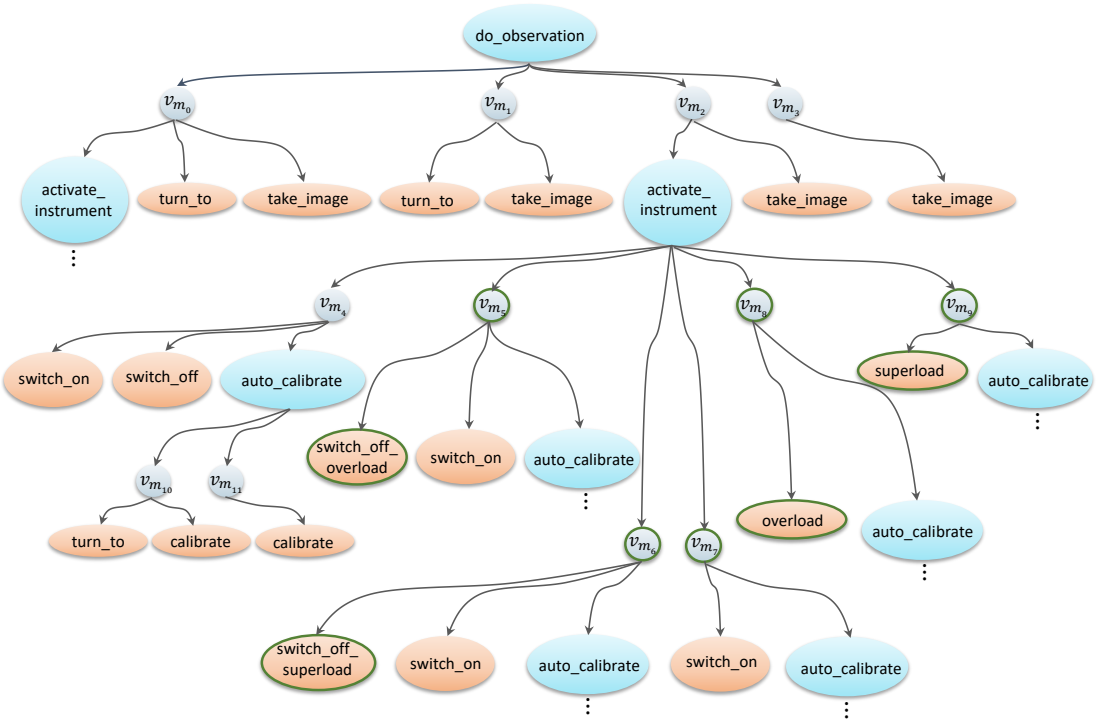



Figure 5. HTN model for the Satellite domain.

For example, reaching a destination may require navigating through intermediate locations, stopping or starting the engine, and managing turn signals. During operation, the vehicle must also handle road contingencies, such as pedestrians or construction zones, which may require actions like stopping, dodging the incident, or restarting the engine. Environmental factors, such as slippery roads, introduce additional complexity and may require actions like activating the Electronic Stability Program (ESP) and adjusting the car's speed.

These activities must also respect ordering constraints arising from traffic regulations and the dynamics of road environments (**A3**). Additionally, many driving activities can be accomplished in multiple ways (i.e., alternatives). For example, a car may address poor road conditions by adjusting speed, with or without activating ESP, and may choose from several alternative routes to reach the same destination. The self-driving car itself serves as the primary agent in this domain, performing all actions (**A4**).

The complexity of self-driving tasks largely arises from the dynamic and uncertain environments in which cars operate (**A5**). Factors such as driver behaviour, vehicle control variability, traffic conditions, and environmental changes introduce uncertainty into the execution of driving tasks. Following our previous work [8][11], we categorise uncertainty sources according to the level of vehicle autonomy: (1) non-autonomous (human-driven), (2) fully autonomous (no human intervention), and (3) semi-autonomous (shared control). These uncertainties may be internal, originating from the agent performing the driving tasks, or external, coming from the environment.

For human-driven cars, internal regular uncertainties often arise from differences in driving skills, habits, intentions, tactics, and speed preferences. As a result, travel time and energy consumption may vary depending on how a driver behaves. Additional uncertainties may arise from driver fatigue, which can lead to accidents, such as falling asleep at the wheel. The consequences of driver drowsiness have been studied statistically [36]. Unlike regular sources, which can be statistically anticipated, some internal uncertainty sources are rare and unpredictable, such as a driver experiencing a medical emergency (e.g., a stroke), making them difficult to anticipate.

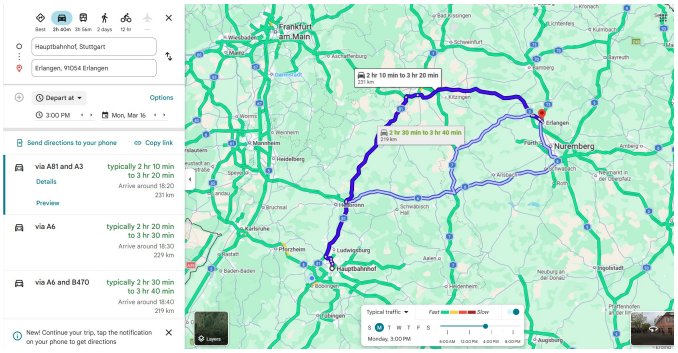
For fully autonomous cars, internal regular uncertainties may stem from control variability. For instance, maintaining stability on slippery roads may require adjustments in control behaviour, which can lead to variations in travel time or energy consumption. Random internal sources of uncertainty may also exist, leading to unpredictable outcomes. For example, mechanical failures (e.g., a flat tyre) can interrupt the car's operation (e.g., the car stops).

Semi-autonomous vehicles inherit elements of both cases, as the car's behaviour results from interactions between the human driver and the autonomous control system. Consequently, both human-related and system-related uncertainties may affect task execution.

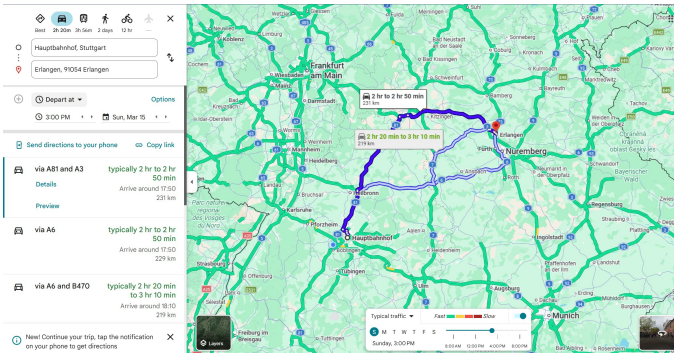
External sources of uncertainty can also influence cost variability (**A6**). These sources may likewise be regular or random. Rare events, such as a vehicle being unable to charge due to an unexpected charging-station malfunction or encountering unexpected roadblocks caused by accidents, are difficult to predict during planning, making the outcomes of actions uncertain.

Now consider regular external sources of uncertainty that arise from environmental conditions that vary over time. The weather is a prominent example: due to the chaotic nature of the atmosphere and limitations in observation and modelling, forecasts inherently involve uncertainty [37]. Consequently, weather conditions are often reported using probabilistic forecasts in which variables such as temperature, wind, and precipitation are quantified probabilistically [38].

Traffic conditions represent another major source of regular external uncertainty. Route planning systems, such as Google Maps, illustrate this effect by providing ranges of estimated travel times for the same route depending on expected traffic conditions. As shown in Figure 6, these estimates of travel times vary between weekdays and weekends, with shorter times typically observed on weekends due to lighter traffic. Similarly, queues at charging stations introduce additional regular external uncertainty. Studies such as [39] estimate the probability of waiting times at charging stations, which directly affect charging duration and overall trip time.




(a) Travelling times on a workday [40].




(b) Travelling times on a weekend [41].

Figure 6. Variability of travelling times in Google Maps.

In addition to the factors discussed above, it is important to account for resource consumption, i.e., the costs associated with performing actions. These costs may include time, money, energy, effort, or even potential risks to human lives (**A6**). In the presence of uncertainty, such costs become variable and may differ across executions of the same action. Therefore, understanding and modelling cost variability is essential for effective planning.

For instance, uncertainty in weather conditions may significantly affect driving costs. Travelling between two locations in winter may take longer and require more energy if snow falls. The extent to which such variability can be characterised depends on the available knowledge about the underlying uncertainty. When uncertainty can be described probabilistically, such as through weather forecasts, the associated action costs, such as travel delays, can also be represented probabilistically. This situation corresponds to decision-making under risk. Conversely, random sources such as unpredictable accidents introduce uncertainty that leads to costs, e.g., delays, injuries, financial loss, and environmental impact, that are difficult to quantify probabilistically. Such events can reduce road capacity, increase congestion and travel time, and potentially cause further incidents [42].

Figure 7. Autonomous Self-Driving Car on Urban Street. Free for use from Pexels.

The domain also exhibits structural diversity, as it includes tasks specific to autonomous driving applications, such as activating the ESP, decelerating, and accelerating (**A7**). Moreover, the domain represents a realistic planning application, as shown in Figure 7 (**A8**), and reflects contemporary technological advancements in modern vehicles, such as stability control systems and sensory data (**A9**).

### *B. Domain Model Formulation*

Having identified the relevant aspects of the self-driving car domain, we now formulate them as HTN planning constructs to build the corresponding domain model.

Driving activities are constructed as compound and primitive tasks in a hierarchy that captures how high-level driving objectives are achieved through decompositions to low-level actions. An abstract graphical overview of the resulting HTN domain model is shown in Figure 8. At the top of the hierarchy is the compound task `drive`, which represents travelling between two locations. This task can be decomposed using three methods, denoted as $v_{m_1}$, $v_{m_2}$, and $v_{m_3}$, which capture different driving situations. Method $v_{m_1}$ is applicable when the vehicle does not have enough energy to reach the next location. In this case, the vehicle must first drive to a charging station (`drive`), recharge its battery (`recharge`), and then resume its trip by driving from the charging station to the intended destination (`drive`).

The second method, $v_{m_2}$, applies when the vehicle has sufficient energy but has not yet reached its destination. In this case, before moving, the vehicle prepares for driving

by cranking the engine if it is not already running (`start`) and turning on the lights if they are off and it is nighttime (`turnon`). The vehicle then moves one step to the next location (`move_step`). After each movement, the `drive` task is recursively decomposed, allowing the vehicle to continue travelling through intermediate locations until it reaches the destination.

The third method, $v_{m_3}$, becomes applicable when the vehicle has reached its destination. In this case, the `drive` task is decomposed into a single compound task `stop_vehilce`, which handles the termination of the trip. The task `stop_vehilce` can itself be decomposed by two methods, $v_{m_4}$ and $v_{m_5}$. The former stops the engine using a primitive task, while the latter applies when the engine is already off ahd therefore performs no further action, indicated by the `nop` primitive task. This latter case corresponds to the phantomisation pattern commonly encountered in HTN domain models, where tasks may be reduced to an empty task network [43].

The `move_step` compound task mentioned above is decomposed by two methods $v\_m_{10}$ and $v\_m_{11}$, depending on whether the road segment ahead is free of any complexity. When the road is clear, $v_{m_{10}}$ decomposes into a single primitive task `accelerate`, representing a straightforward movement to the next location. When the road involves additional complications, $v_{m_{11}}$ decomposes into two consecutive compound tasks: `handle_incidents` and `handle_road_conditions`. The `handle_incidents` task is decomposed by two methods, $v_{m_{13}}$ and $v_{m_{12}}$, depending on the type of incident encountered. For still incidents, such as rocks or construction work, the vehicle must decelerate, dodge the incident, and then accelerate again. For moving incidents, such as pedestrians crossing a street, the vehicle decelerates, brakes to allow the moving incident to pass, and accelerates again.

The `handle_road_conditions` task can be decomposed by four methods, $v_{m_{15}}$, $v_{m_{16}}$, $v_{m_{17}}$ and $v_{m_{18}}$. All these methods are applicable when the road conditions are abnormal, for example, when the road is icy, slippery, under construction, or covered with loose gravel. The methods correspond to different driving strategies: accelerating without adjusting for the road condition, decelerating only, activating the ESP while decelerating, and activating the ESP but accelerating. Since these methods share the same precondition, they are all applicable at the same time during planning assuming the planning state allows for it. This allows an HTN planner to choose among several alternatives, a modelling patter that we refer to as *permissive decomposition* or *non-exclusive decomposition*.

Both `handle_incidents` and `handle_road_conditions` also include an additional method ($v_{m_{14}}$ and $v_{m_{19}}$, respectively) that decomposes the corresponding task into an empty task network when there are no incidents or abnormal road conditions, respectively. These cases represent another instance of phantomisation. This modelling choice allows an HTN planner to handle situations in which incidents and adverse road conditions occur simultaneously between two connected locations.

We now formulate the aspects related to non-determinism and resource quantities. In particular, the domain model considers the effects of three sources of uncertainty. The first source is the speed at which pedestrians walk across the pedestrian crossing, which represents a regular external source of uncertainty. The second source is traffic on roads, which also constitutes an external regular source of uncertainty. The third source is the ability of the autonomous vehicle to stabilise on slippery roads, which represents an internal regular source of uncertainty related to vehicle control.

When cost variability comes from regular uncertainty sources, such as traffic jams, it can be described by a probability distribution that is either known or statistically inferred. In such cases, the corresponding actions are considered risk-inducing. Within this domain model, we define driving costs as the time required to traverse roads while handling the various road complexities. Traffic conditions, such as traffic jams, during planning make the estimation of travel time variable (as shown in Figure 6). Although we focus on risk-inducing actions and travel time as the primary cost metrics, other forms of uncertainty and cost measures could be considered. Examples include uncertainty-inducing actions or costs related to fuel or battery consumption, ride comfort, or road characteristics such as curvature.

Travel costs depend both on external factors, such as traffic conditions, and on properties of the road itself, such as length and surface conditions. To capture this dependency, we use a hybrid external and state-dependent cost function $c^{ies}(a)$ (see Section II). Estimation of uncertain traffic conditions is obtained through an external function, while road properties are represented by a state-dependent cost function defined with respect to the grounded planning problem. These two components are combined into a hybrid function that produces a probability distribution over travel times. Figure 9 shows example probability distributions over travel times for a planning problem in this domain. For instance, accelerating or decelerating on road $(l_4, E)$ leads to variable travel times depending on whether the ESP is activated (state-dependent) and on external factors such as traffic and vehicle stability. When the ESP is active, deceleration guarantees arrival at destination $E$ in ten hours. Without ESP, arrival may take eleven hours with probability 80% or six hours with probability 20%.

A detailed discussion of the planning problems in this domain, including a detailed reasoning behind the assigned cost probability distributions in a specific problem instance, is provided in Appendix A-C.

### C. Domain Model Encoding

We encode the domain and problem instances using rHDDL. Compound tasks are modelled by specifying the task name together with its parameter list. For example, Listing 5 shows the definition of the compound task `handle_incidents`. Methods are modelled by specifying their parameters, the compound task they decompose, the corresponding preconditions, and the resulting task network. For example, `handle_incidents` can

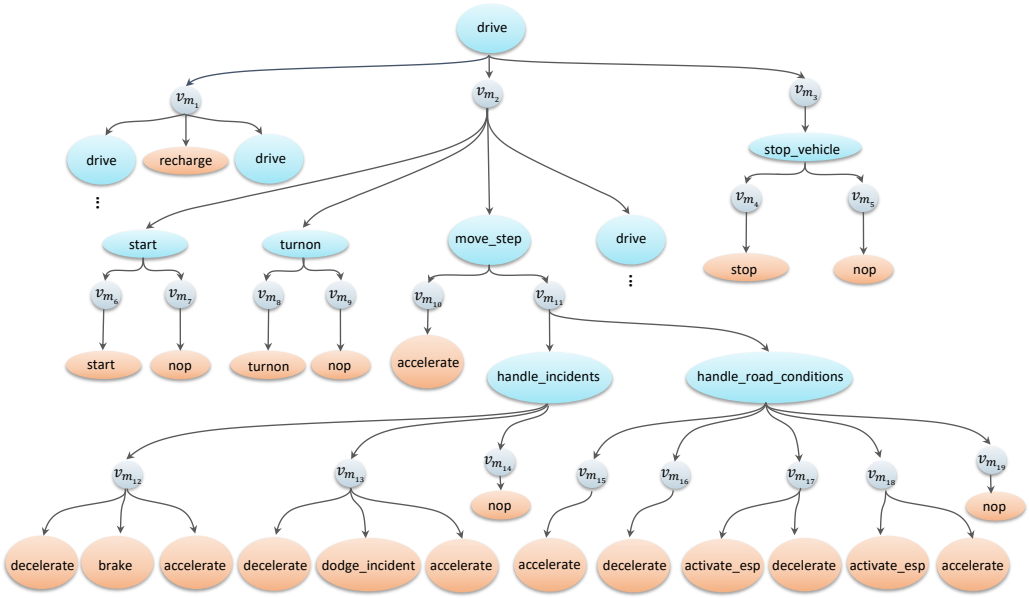



Figure 8. HTN model for the Self-Driving Car domain.

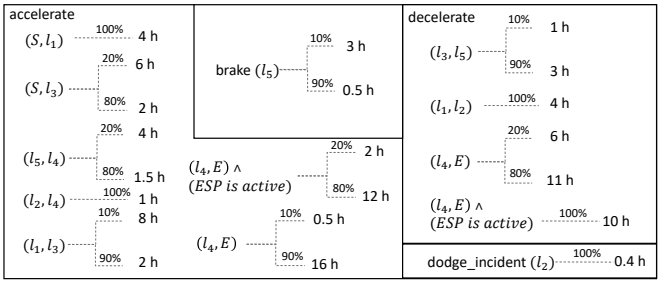



Figure 9. Actions with the corresponding possible bindings and probability distribution of costs (time).

be decomposed by three methods. Two of these methods apply depending on whether the encountered incident is static or moving, as specified in their preconditions. The third method represents a phantomisation case, allowing the task to be reduced to an empty task network when no incident is present.

Listing 5: Methods for handling incidents in the Self-Driving Car domain model

```
(:method handle_incidents_0
 :parameters(?v - vehicle ?l1 ?l2 ?l3 - loc ?
    movinc - movingobs)
 :task(handle_incidents ?v ?l1 ?l2)
 :precondition(and (connected ?l1 ?l3)(
    connected ?l3 ?l2)(in-inc ?movinc ?l3))
 :ordered-subtasks(and (decelerate_incident ?v
     ?l1 ?l3)(brake ?v ?l3)(
    accelerate_incident ?v ?l3 ?l2)))

(:method handle_incidents_1
 :parameters(?v - vehicle ?l1 ?l2 ?l3 - loc ?
    stillinc - stillobs)
 :task(handle_incidents ?v ?l1 ?l2)
 :precondition(and(connected ?l1 ?l3)(
    connected ?l3 ?l2)(in-inc ?stillinc ?l3))
 :ordered-subtasks(and(
    decelerate_still_incident ?v ?l1 ?l3)(
    dodge_incident ?v ?l3 ?l2)(
    accelerate_still_incident ?v ?l3 ?l2)))

(:method handle_incidents_2
 :parameters(?v - vehicle ?l1 ?l2 - loc)
 :task(handle_incidents ?v ?l1 ?l2)
 :precondition(clearroad ?l1 ?l2)
 :ordered-subtasks())
```

Actions are modelled with parameters, preconditions, and effects. Risk is represented using the `:costdist` construct. Listing 6 shows four actions related to decelerating and accelerating when handling bad road conditions. Each action is associated with cost distributions shown in Figure 9.

Since the cost functions are hybrid, the actions can be preprocessed and expanded into multiple variants corresponding to different road conditions. These variants directly reference the appropriate hybrid cost functions within the domain model.

Listing 6: Accelerating and decelerating actions of the Self-Driving Car domain model that deal with bad road conditions.

```
(:action accelerate_bad_road
  :parameters (?v - vehicle ?l1 ?l2 - loc)
  :precondition (and (not (in ?v ?l2))(not(
    activated_esp ?v)))
  :effect (and (in ?v ?l2)(highspeed ?v))
  :costdist (or (0.9(16)) (0.1(0.5))))

(:action accelerate_after_esp
  :parameters (?v - vehicle ?l1 ?l2 - loc)
  :precondition (and (not(in ?v ?l2))(
    activated_esp ?v))
  :effect (and (in ?v ?l2)(highspeed ?v))
  :costdist (or (0.8(12)) (0.2(2))))

(:action decelerate_after_esp
```

```
  :parameters (?v - vehicle ?l1 ?l2 - loc)
  :precondition (and (not(in ?v ?l2))(
    activated_esp ?v))
  :effect (and (in ?v ?l2) (not(highspeed ?v))
    )
  :costdist (or (1(10))))

(:action decelerate_bad_road
  :parameters (?v - vehicle ?l1 ?l2 - loc)
  :precondition (and (not(in ?v ?l2))(not(
    activated_esp ?v)))
  :effect (and (in ?v ?l2)(not(highspeed ?v)))
  :costdist (or (0.2(6)) (0.8(11))))
```

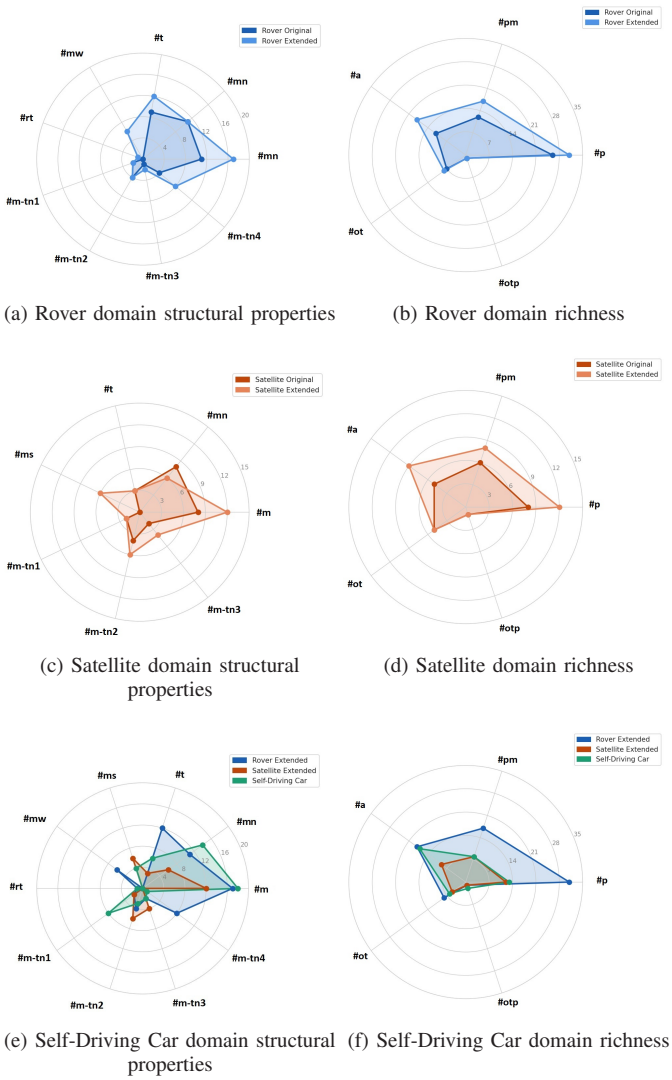
## VII. Quantitative Analysis of the Developed HTN Domains

The developed HTN planning domains are developed following the systematic domain-engineering approach presented in Section III, which focuses on incorporating realistic, application-inherent aspects. As a result, the developed HTN domains include features that the original domains miss, making them qualitatively more advanced. However, quantitative analysis of these domains can provide additional insights into their structural properties. The results of the quantitative analysis of the analysed HTN planning domains are shown in Table I. The table compares the structural properties and domain richness of the original and extended versions of the Satellite and Rover domains. It also reports the same metrics for the newly engineered Self-Driving Car domain.

Regarding structural properties, we highlight that in the original versions of both the Satellite and Rover domains, no alternatives are available for accomplishing tasks. After the proposed extensions, the Satellite domain includes six strict alternative methods that can accomplish the instrument activation task, whereas the Rover domain includes six weak alternative methods that enrich data transmission to the lander offered using space drone's assistance. Since the extended Rover domain is modelled under the assumption that each rover is associated with a drone, these weak alternative methods are guaranteed to provide an option to perform the task. Thus, in this specific domain, these six methods can be interpreted as strict alternatives.

The Self-Driving Car domain has the largest number of methods overall. Among these, four are strict alternatives that offer the choice of how to perform driving tasks on roads with abnormal conditions. Both the Rover and Self-Driving Car domains include one recursive task related to navigation or driving. Such recursion is common in domains involving continuous behaviour, such as movement between locations until reaching a destination.

As expected, the introduction of additional alternatives increases the total number of methods in the extended Rover and Satellite domains. In the Satellite domain, the extension introduces additional methods with two and three subtasks, while in the Rover domain, the number of methods with three or four subtasks increases. However, the maximum number of subtasks across all methods remains unchanged in both domains.




(a) Rover domain structural properties (b) Rover domain richness

(c) Satellite domain structural properties (d) Satellite domain richness

(e) Self-Driving Car domain structural properties (f) Self-Driving Car domain richness

Figure 10. Comparison of the structural and domain-richness of the original, extended, and newly engineered domains.

Regarding the domains' richness, Table I shows that the number of actions, predicates, and multi-object predicates increases in both the Rover and Satellite domains after the proposed extensions. The values observed in the newly developed Self-Driving Car domain lie between those of the two extended domains. A larger number of objects and richer relationships between them provide more detailed information about the application domain. Similarly, a greater number of actions expands the range of achievable behaviours. Taken together, these metrics indicate that the developed HTN domains exhibit increased knowledge richness and more closely reflect the complexity of real-world scenarios.

The radar charts in Figure 10 provide a visual comparison of the structural and domain-richness profiles of the analysed HTN planning domains across their original, extended, and newly engineered configurations. Overall, the radar charts show that the proposed domain engineering process yields distinct struc-

TABLE I. STRUCTURAL AND DOMAIN-RICHNESS METRICS OF THE ORIGINAL AND EXTENDED VERSIONS OF THE SATELLITE AND ROVER DOMAIN AND THE DEVELOPED SELF-DRIVING CAR DOMAIN. THE METRICS ARE EXPLAINED IN SECTION III-E

| Domain | | Structural metrics | | | | | | | | | | Domain-richness metrics | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | #m | #ms | #mw | #mn | #t | #rt | #m-tn | | | | #p | #pm | #ot | #otp | #a |
| | | | | | | | | 1 | 2 | 3 | 4 | | | | | |
| Rover | Original | 11 | 0 | 0 | 11 | 9 | 0 | 2 | 4 | 1 | 4 | 26 | 12 | 7 | 1 | 11 |
| | Extended | 17 | 0 | **6** | 11 | **12** | **1** | 2 | 4 | 2 | **8** | **31** | **17** | **8** | 1 | **18** |
| Satellite | Original | 8 | 0 | 0 | 8 | 3 | 0 | 2 | 4 | 2 | 0 | 8 | 6 | 5 | 1 | 5 |
| | Extended | 12 | **6** | 0 | 6 | 3 | 0 | 2 | **6** | **4** | 0 | 12 | 8 | 5 | 1 | 9 |
| Self-Driving Car | Original | **18** | 4 | 0 | **14** | 6 | **1** | **8** | 3 | 2 | 1 | 13 | 8 | 6 | **2** | 17 |

tural outcomes across domains. The Rover domain primarily grows through additional methods, tasks, predicates, and actions (Figures 10a and 10b). The growth in the number of predicates and the growth in predicates that define relations between two or more objects appear to be comparable (Figure 10b). This suggests that the extended domain introduced additional predicates primarily to capture more complex relationships among the objects, rather than merely increasing the number of stand-alone properties. The Satellite domain expands through the addition of new methods, predicates, and actions, while the set of tasks remains unchanged (Figures 10c and 10d). This indicates that the extended domain introduces alternative ways to accomplish the existing tasks, providing greater flexibility in how the available tasks can be executed. The Satellite and Rover domains introduce new alternatives through restructuring of task decompositions, as shown in Figures 10c and 10a. The Self-Driving Car domain exhibits comparatively high procedural complexity due to the diversity of driving behaviours represented in the model, as shown in Figure 10e.

For the Rover domain, the transition from original to extended is characterised by a radial expansion across all axes, as shown in Figure 10a and 10b, with the most pronounced gains observed in the number of methods, particularly alternative methods and those with four subtasks in their task network, actions, predicates, and task arity. This indicates that the extension introduces enrichment of both procedural and operational expressivity while preserving the overall structure of the domain.

The Satellite domain, by contrast, exhibits a different extension pattern. While the total number of methods and predicates increases moderately compared to the Rover domain, the number of non-alternative methods decreases, as shown in Figure 10c. This reflects a deliberate structural reorganisation of the domain model in which some previously flat task decompositions are replaced by alternative hierarchical decompositions, thereby increasing the availability of alternatives. In general, Figures 10c and 10d show that the transition from the original to the extended version of the domain exhibits a radial expansion across all axes, except for the non-alternative methods.

The radar comparison across all three engineered domains (Figures 10e and 10f) shows that the Self-Driving Car domain has the largest number of methods and a relatively high number of actions. This indicates a comparatively high level of procedural complexity in the domain model. At the same time, the number of predicates remains lower than in the extended Rover domain and comparable to the extended Satellite domain, suggesting that the Self-Driving Car domain encodes much of its complexity through task decompositions and actions rather than through a large set of predicates.

Across all domains, the number of object types remains relatively stable. This observation suggests that the object-type hierarchy, once defined during the initial modelling stage, tends to remain largely unchanged during subsequent domain extensions.

## VIII. RELATED WORK

Most existing AI planning domains have been developed either as benchmarks for IPCs or as illustrative examples for evaluating planning techniques. In the context of IPCs, domains are often intentionally simplified in order to facilitate systematic evaluation and comparison of planners, e.g., [6]. While this simplification is necessary for benchmarking purposes, it can limit the suitability for modelling actual planning problems. This limitation has been characterised as the *sanitised benchmark domain barrier*, referring to the limited utility of simplified benchmark domains for real-world environments [12].

Other domains are introduced to demonstrate the capabilities of domain modelling languages (e.g., [44]) or to illustrate or evaluate specific planning approaches. As a result, many planning domains are designed with particular experimental objectives in mind rather than through a systematic domain engineering process [4]. Some works also propose domains that address aspects different from those considered in our domains, e.g., [45][46].

In the following, we review existing work that models the Rover, Satellite, and Self-Driving Car domains, or closely related planning domains.

### A. Rover Domains

Research on AI planning for planetary rover missions is closely related to our work. However, most existing rover domain models generally do not reflect recent technological developments, such as the use of drones as assistants in planetary exploration. The Rover domain we extend was used in the HTN planning track of IPC-2020 [21]. While it provides a useful benchmark, it does not incorporate recent advancements in exploration technologies or capture several realistic aspects considered in our extension.

Several studies have modelled rover domains for AI planning using techniques other than HTN planning and with a focus on incorporating aspects different from those we focus on. In [47][48], for example, a rover domain is formulated as a hybrid domain exhibiting both continuous and discrete behaviour. The work aligns with the goal of providing more options for achieving tasks by allowing alternative methods for energy generation and management. To transmit data, the rover can either draw power from its batteries or generate energy via solar panels. The domain is modelled in PDDL+ for hybrid planning rather than HTN planning and incorporates different alternatives from those introduced in our extension. Other works focus on incorporating temporal and uncertainty aspects into rover-related planning domains. In [44], a rover domain is presented using PDDL2.1, with particular emphasis on temporal constraints. Similarly, in [49], a model of the rover domain is proposed, which integrates temporal and uncertainty aspects within a framework combining HTN and timeline-based planning. In [45], an Europa lander domain is developed for HTN planning. Although it shares similarities with rover missions in terms of planetary exploration tasks, it focuses primarily on uncertainty in energy consumption and resource quantities, representing these through action costs related to battery life and predefined task utilities. Overall, while these studies provide valuable insights into modelling rover-related planning domains, they address different planning techniques or incorporate a distinct set of aspects from those considered in our work. In particular, most existing HTN or HTN-like Rover domains were developed without following a clear methodological domain-engineering process. As a result, they often overlook several realistic aspects that we explicitly target in our extensions, including risk awareness and the systematic modelling of alternative task decompositions.

### B. Satellite Domains

Existing Satellite planning domains lack several aspects that we incorporate in our work, such as explicit modelling of risk and uncertainty. A version of the Satellite domain was used in [22][24][50]. This version is developed on the basis of its classical planning counterparts, which were originally introduced for earlier IPC benchmarks. As a result, it inherits classical planning assumptions, thereby simplifying several characteristics of real-world satellite operations. In our work, we build on this benchmark version by analysing sources of uncertainty and their effects on action costs, enabling the incorporation of risk-aware decision-making. In addition, we expand the domain by introducing additional tasks and alternative methods for completing space missions.

In [46], a Satellite domain for onboard and online planning is presented using a language based on an extended HTN representation. The model captures several operational aspects, including unexpected events and resource consumption, but does not incorporate risk. Another line of work on satellite planning focuses on broadcasting and communication satellites, e.g., [51]. These domains address scheduling and resource allocation problems that differ semantically from the space observation and exploration scenarios considered in our work.

### C. Self-Driving Car Domains

Existing AI planning works that model self-driving car domains are rather scarce. Several studies have explored traffic control and multi-vehicle navigation problems using classical planning and its extensions, focusing on coordinating vehicles to improve traffic flow and management [52]–[56]. While these works consider vehicle movement and routing tasks, they primarily focus on traffic optimisation rather than modelling the behaviour of individual autonomous vehicles.

Several works use probabilistic planning to model the uncertainty of the self-driving car environments, e.g., [57]–[59]. These works account for uncertainty through Partially Observable Markov Decision Processes (POMDPs), thereby relying on a different planning technique than our approach. Moreover, they do not model risk.

Research on route planning in AI planning is also relevant to self-driving car applications. For example, several works investigate route planning under uncertainty in marine environments, such as those involving autonomous underwater vehicles [60]. Although such vehicles share some navigation tasks with autonomous terrestrial vehicles, their operational environments and mission objectives differ significantly. Other works address route planning within a temporal HTN planning framework in the aerial domain, e.g., [61], focusing on challenges specific to aerial vehicles, such as flight dynamics and airspace constraints, which differ significantly from those encountered in autonomous ground vehicle scenarios.

## IX. Conclusions

Developing operational AI planning systems requires domain models that capture the characteristics of real-world environments. However, many existing planning domains remain simplified and omit important aspects such as uncertainty, risk, and diverse ways of achieving tasks. We addressed this limitation by focusing on the systematic engineering and analysis of realistic HTN planning domains.

Our contributions are twofold. First, we proposed a systematic domain-engineering approach for developing realistic HTN planning domains. The approach integrates the identification of relevant domain aspects, their formulation using HTN constructs, their encoding in a domain model using a planning language, and a quantitative analysis of the resulting domain models. Second, we applied this approach to develop and analyse three HTN planning domains: two extended benchmark domains, Rover and Satellite, and a newly engineered domain named Self-Driving Car. In these domains, we incorporated several aspects that are often absent from existing domain models, including richer alternative task decompositions, explicit representations of uncertainty and risk, and considerations of recent technological developments in the respective application areas.

Our analysis shows that incorporating these aspects leads to domains with increased structural diversity and richer representations of application knowledge. The quantitative evaluation

demonstrates that the developed domain models contain more alternative methods, more expressive task structures, and richer object and predicate relationships compared to their original counterparts. These properties allow the domain models to better reflect the complexity of real-world environments and provide a more informative basis for evaluating planning systems.

The quantitative analysis also reveals recurring structural patterns in how HTN domains evolve when enriched with realistic aspects. Extensions of existing domains manifest either as *uniform growth in structural constructs* or as *structural reorganisation* that introduces additional alternatives and hierarchical relationships. In contrast, engineering domains from scratch may exhibit *high procedural complexity* from the outset through richer task decompositions and action sets. These patterns suggest that systematic domain engineering not only increases domain richness but also influences how planning knowledge is conceptualised and structured.

Finally, richer and more realistic domains may also stimulate advances in planning technology itself. As demonstrated by the historical influence of benchmark domains introduced through IPCs, the availability of more representative domains can drive the development of planners capable of handling increasingly complex planning problems.

As future work, we plan to engineer more HTN planning domains by incorporating realistic aspects, considering also aspects not addressed in this treatment. We also aim to explore systematic methods for evaluating realism in planning domains and to investigate how richer domains influence the design and performance of HTN planning systems. Together, these directions will contribute to bridging the gap between domain models and operational planning applications.

## Acknowledgements

During the preparation of this work, the authors used ChatGPT to polish self-authored text and Claude to visualise the radar charts. After using the tool, the authors reviewed and edited the content as needed, and take full responsibility for the content of the work.

## References

[1] E. Alnazer, I. Georgievski, and M. Aiello, "Risk-Aware HTN Planning Domain Models for Autonomous Vehicles and Satellites", in *International Conference on AI-based Systems and Services (AISyS)*, 2025, pp. 36–45.

[2] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. Elsevier, 2004.

[3] I. Georgievski and M. Aiello, "HTN Planning: Overview, Comparison, and Beyond", *Artificial Intelligence (AIJ)*, vol. 222, pp. 124–156, 2015.

[4] T. L. McCluskey, T. S. Vaquero, and M. Vallati, "Engineering Knowledge for Automated Planning: Towards a Notion of Quality", in *International Conference on Knowledge Capture (K-CAP)*, Association for Computing Machinery, 2017, pp. 1–8.

[5] I. Georgievski, "Software Development Life Cycle for Engineering AI Planning Systems", in *International Conference on Software Technologies (ICSOFT)*, 2023, pp. 751–760.

[6] J. Hoffmann et al., "Engineering Benchmarks for Planning: The Domains Used in the Deterministic Part of IPC-4", *Journal of Artificial Intelligence Research (JAIR)*, vol. 26, pp. 453–541, 2006.

[7] E. Alnazer, I. Georgievski, and M. Aiello, "On Bringing HTN Domains Closer to Reality-The Case of Satellite and Rover Domains", in *International Conference on Automated Planning Systems (ICAPS) Workshop on Scheduling and Planning Applications (SPARK)*, 2022.

[8] E. Alnazer, I. Georgievski, and M. Aiello, "Risk Awareness in HTN Planning", *arXiv preprint arXiv:2204.10669v2*, 2025.

[9] M. Vallati and L. McCluskey, "A Quality Framework for Automated Planning Knowledge Models", in *International Conference on Agents and Artificial Intelligence (ICAART)*, SciTePress, 2021, pp. 635–644.

[10] I. Georgievski, "Conceptualising Software Development Lifecycle for Engineering AI Planning Systems", in *International Conference on AI Engineering – Software Engineering for AI (CAIN)*, IEEE, 2023, pp. 88–89.

[11] E. Alnazer and I. Georgievski, "Understanding Real-World AI Planning Domains: A Conceptual Framework", in *Symposium and Summer School on Service-Oriented Computing (SummerSoC)*, Springer, 2023, pp. 3–23.

[12] I. Georgievski, *Engineering AI Planning Systems*, Habilitation thesis, University of Stuttgart, 2025.

[13] E. Turan, S. Speretta, and E. Gill, "Autonomous Navigation for Deep Space Small Satellites: Scientific and Technological Advances", *Acta Astronautica*, vol. 193, pp. 56–74, 2022.

[14] D. Long and M. Fox, "The 3rd International Planning Competition: Results and Analysis", *Journal of Artificial Intelligence Research (JAIR)*, vol. 20, pp. 1–59, 2003.

[15] M. Weser, D. Off, and J. Zhang, "HTN Robot Planning in Partially Observable Dynamic Environments", in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2010, pp. 1505–1510.

[16] J. Fdez-Olivares, L. Castillo, J. A. Cózar, and O. Garcia Perez, "Supporting Clinical Processes and Decisions by Hierarchical Planning and Scheduling", *Computational Intelligence (CI)*, vol. 27, no. 1, pp. 103–122, 2011.

[17] I. Georgievski et al., "Planning Meets Activity Recognition: Service Coordination for Intelligent Buildings", *Pervasive and Mobile Computing (PMC)*, vol. 38, no. 1, pp. 110–139, 2017.

[18] J. R. Silva, J. M. Silva, and T. S. Vaquero, "Formal Knowledge Engineering for Planning: Pre and Post-Design Analysis", *Knowledge Engineering Tools and Techniques for AI Planning*, pp. 47–65, 2020.

[19] I. Georgievski, "Hierarchical Planning Definition Language", University of Groningen, Tech. Rep. JBI 2013-12-3, 2013.

[20] D. Höller et al., "HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems", in *AAAI Conference on Artificial Intelligence*, AAAI Press, 2020, pp. 9883–9891.

[21] G. Behnke, D. Höller, and P. Bercher, Eds., *Proceedings of the 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*, 2021.

[22] P. Bercher, G. Behnke, D. Höller, and S. Biundo, "An Admissible HTN Planning Heuristic.", in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2017, pp. 480–488.

[23] G. Behnke, D. Höller, and S. Biundo, "totSAT-Totally-ordered Hierarchical Planning Through SAT", in *AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[24] B. Schattenberg, "Hybrid Planning & Scheduling", Ph.D. dissertation, Ulm, Univ., Diss., 2009, 2009.

[25] R. C. Anderson et al., "Collecting Samples in Gale Crater, Mars; An Overview of the Mars Science Laboratory Sample Acquisition, Sample Processing and Handling System", *Space*

*Science Reviews (Space Sci. Rev.)*, vol. 170, no. 1, pp. 57–75, 2012.

[26] D. Höller, P. Bercher, G. Behnke, and S. Biundo, "A Generic Method to Guide HTN Progression Search with Classical Heuristics", in *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 28, 2018, pp. 114–122.

[27] D. Pellier and H. Fiorino, "From Classical to Hierarchical: Benchmarks for the HTN Track of the International Planning Competition", in *International Planning Competition (IPC)*, 2020.

[28] C. Carr et al., "Space Drones: An Opportunity to Include, Engage, Accelerate, and Advance", in *Bulletin of the American Astronomical Society*, vol. 53, 2021.

[29] N. Potter, "A Mars Helicopter Preps for Launch: The First Drone to Fly on Another Planet will Hitch a Ride on NASA's Perseverance Rover", *IEEE Spectrum*, vol. 57, no. 7, pp. 06–07, 2020.

[30] C. Powell, C. S. Ruf, S. Gleason, and S. C. Rafkin, "Sampled Together: Assessing the Value of Simultaneous Collocated Measurements for Optimal Satellite Configurations", *Bulletin of the American Meteorological Society (BAMS)*, vol. 105, no. 1, E285–E296, 2024.

[31] V. Maggioni, C. Massari, and C. Kidd, "Errors and Uncertainties Associated with Quasiglobal Satellite Precipitation Products", in *Precipitation Science*, Elsevier, 2022, pp. 377–390.

[32] G. A. Landis, S. G. Bailey, and R. Tischler, "Causes of Power-Related Satellite Failures", in *World Conference on Photovoltaic Energy Conference (EU PVSEC)*, IEEE, 2006, pp. 1943–1945.

[33] P. S. Morgan, "Fault Protection Techniques in JPL Spacecraft", in *Integrated System Health Engineering and Management in Aerospace (ISHEM)*, 2005.

[34] H.-S. Choi et al., "Analysis of GEO Spacecraft Anomalies: Space Weather Relationships", *Space Weather*, vol. 9, no. 6, 2011.

[35] R. Horne et al., "Space Weather Impacts on Satellites and Forecasting the Earth's Electron Radiation Belts with SPACECAST", *Space Weather*, vol. 11, no. 4, pp. 169–186, 2013.

[36] C. C. Liu, S. G. Hosking, and M. G. Lenné, "Predicting Driver Drowsiness Using Vehicle Measures: Recent Insights and Future Challenges", *Journal of Safety Research*, vol. 40, no. 4, pp. 239–245, 2009.

[37] N. R. Council et al., *Completing the Forecast: Characterizing and Communicating Uncertainty for Better Decisions Using Weather and Climate Forecasts*. National Academies Press, 2006.

[38] *Enhancing Weather Information with Probability Forecasts*, https://shorturl.at/RFuna, Accessed: 2026-05-15, 2026.

[39] J. Antoun, M. E. Kabir, R. F. Atallah, and C. Assi, "A Data Driven Performance Analysis Approach for Enhancing the QoS of Public Charging Stations", *Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 23, no. 8, pp. 11 116–11 125, 2021.

[40] *Google Maps Directions for Driving from Stuttgart to Erlangen*, https://shorturl.at/O2jdL, Accessed: 2026-03-15, 2026.

[41] *Google Maps Directions for Driving from Stuttgart to Erlangen*, https://shorturl.at/H7riv, Accessed: 2026-03-15, 2026.

[42] P. Farradyne, "Traffic Incident Management Handbook", *Prepared for Federal Highway Administration, Office of Travel Management*, 2000.

[43] I. Georgievski and M. Aiello, "Phantomisation in State-Based HTN Planning", in *International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI)*, 2019, pp. 39–44.

[44] M. Fox and D. Long, "PDDL2. 1: An Extension to PDDL for Expressing Temporal Planning Domains", *Journal of Artificial Intelligence Research (JAIR)*, vol. 20, pp. 61–124, 2003.

[45] D. Wang, J. A. Russino, C. Basich, and S. Chien, "Analyzing the Efficacy of Flexible Execution, Replanning, and Plan Optimization for a Planetary Lander", in *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 32, 2022, pp. 518–526.

[46] F. D. S. Cividanes, M. G. V. Ferreira, and F. de Novaes Kucinskis, "An Extended HTN Language for Onboard Planning and Acting Applied to a Goal-Based Autonomous Satellite", *Aerospace and Electronic Systems Magazine*, vol. 36, no. 8, pp. 32–50, 2021.

[47] W. M. Piotrowski, M. Fox, D. Long, D. Magazzeni, F. Mercorio, et al., "Heuristic Planning for PDDL+ Domains", in *AAAI Workshop: Planning for Hybrid Systems*, vol. 16, 2016, p. 12.

[48] W. M. Piotrowski, "Heuristics for AI Planning in Hybrid Systems", Ph.D. dissertation, King's College London, 2018.

[49] J. D. Victoria, N. Policella, Y. Gao, and O. von Stryk, "Design Concepts for a New Temporal Planning Paradigm", in *International Conference on Automated Planning and Scheduling (ICAPS) Workshop on Planning and Scheduling with Timelines (PSTL)*, 2012, p. 16.

[50] P. Bercher, S. Keen, and S. Biundo, "Hybrid Planning Heuristics Based on Task Decomposition Graphs", in *International Symposium on Combinatorial Search (SOCS)*, vol. 5, 2014, pp. 35–43.

[51] M. D. Rodríguez-Moreno, D. Borrajo, and D. Meziat, "An AI Planning-based Tool for Scheduling Satellite Nominal Operations", *AI Magazine*, vol. 25, no. 4, pp. 9–9, 2004.

[52] F. Jimoh, L. Chrpa, T. L. McCluskey, and S. Shah, "Towards Application of Automated Planning in Urban Traffic Control", in *International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2013, pp. 985–990.

[53] M. Vallati, D. Magazzeni, B. De Schutter, L. Chrpa, and T. McCluskey, "Efficient Macroscopic Urban Traffic Models for Reducing Congestion: A PDDL+ Planning Approach", in *AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[54] M. Gulić, R. Olivares, and D. Borrajo, "Using Automated Planning for Traffic Signals Control", *PROMET-Traffic&Transportation*, vol. 28, no. 4, pp. 383–391, 2016.

[55] T. McCluskey and M. Vallati, "Embedding Automated Planning Within Urban Traffic Management Operations", in *International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 27, 2017, pp. 391–399.

[56] F. Ivankovic and M. Roveri, "Planning with Global State Constraints for Urban Traffic Control", in *CEUR WORKSHOP PROCEEDINGS*, CEUR-WS, vol. 2987, 2021, pp. 1–5.

[57] X. Dong et al., "Why Autonomous Vehicles Are Not Ready Yet: A Multi-Disciplinary Review of Problems, Attempted Solutions, and Future Directions", *Journal of Field Robotics*, 2025.

[58] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic Decision-making Under Uncertainty for Autonomous Driving Using Continuous POMDPs", in *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2014, pp. 392–399.

[59] S. Ulbrich and M. Maurer, "Probabilistic Online POMDP Decision Making for Lane Changes in Fully Automated Driving", in *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2013, pp. 2063–2067.

[60] T. X. Lin, M. Hou, C. R. Edwards, M. Cox, and F. Zhang, "Bounded Cost HTN Planning for Marine Autonomy", in *Global Oceans 2020: Singapore–US Gulf Coast*, IEEE, 2020, pp. 1–6.

[61] J. J. Kiam, P. Bercher, and A. Schulte, "Temporal Hierarchical Task Network Planning with Nested Multi-Vehicle Routing Problems — A Challenge to be Resolved", in *International Conference on Automated Planning and Scheduling (ICAPS) Workshop on Hierarchical Planning (HPlan)*, 2021, pp. 71–75.

## APPENDIX A
## PROBLEM INSTANCES

### A. Rover Domain

Planning problems in the Rover domain represent the exploration environment through the specification of the initial state, the initial task network, and the domain objects. The initial state captures both static and dynamic knowledge of the environment. Static knowledge describes properties that remain unchanged during execution. These include the visibility relationships of the waypoints, the visibility of calibration objectives from the waypoints, the locations of rock, soil, and image samples, and the capabilities of each rover. The rover capabilities specify whether a rover can sample soil, rock, or image data, and which terrain types it can traverse between waypoints. Additional static knowledge includes the lander's location, the cameras mounted on each rover, and the calibration targets and operating modes supported by these cameras. Dynamic knowledge describes the evolving state of the system during execution. This includes the current locations of rovers and drones and the status of the rover's storage units, i.e., whether they are empty or contain collected samples. The initial task network includes tasks for sampling rock and soil data from specified locations and for taking images in a specific mode. The objects of the planning problem include various waypoints, drones, available cameras and their operating modes, rovers and their storage units, the lander, and the objectives for taking images.

### B. Satellite Domain

Planning problems in the Satellite domain represent the observation environment, where the initial state captures both static and dynamic knowledge of the environment. Static knowledge includes properties about each satellite's instrument, supported modes, and the calibration target of each instrument. Dynamic knowledge includes properties about the available power and current pointing direction of individual satellites. The initial task network includes all required observation missions, each targeting a phenomenon with a specified mode. The objects of the planning problem include the satellite, instruments, calibration directions, modes, and image directions.

### C. Self-Driving Car Domain

Knowledge about planning problems is formulated such that the initial state contains the knowledge about the static and the dynamic states of the environment. The static state includes the road network, i.e., locations and routes connecting them, the location of still and moving incidents (e.g., construction works and pedestrians), the conditions of roads (e.g., slippery roads, roads with gravel, and normal roads), and the length of the roads. The dynamic states include the location of the car. The initial task network in the planning problem is to move the car from one location to another. The objects are the locations that the car can navigate to and the various incidents.

Let us consider an example of a planning problem with seven different locations denoted as $S$, $l_1$, $l_2$, $l_3$, $l_4$, $l_5$, and $E$, depicted in Figure 11. The car is initially at $S$ and should navigate to $E$ while handling the various road complexities. We assume that the car has enough power to navigate all the roads, and it is nighttime, so the car has to turn on the lights. The roads between $S$ and $l_1$ and between $S$ and $l_3$ are complexity-free. However, the first road is longer than the second. The road between $l_1$ and $l_4$ has constructions at location $l_2$. The road between $l_3$ and $l_4$ has a school area that is very crowded with pedestrians and traffic jams, i.e., moving incidents, at location $l_5$. The road between $l_1$ and $l_3$ is a highway free from any complexities, but has a variable level of traffic congestion. Finally, the road between $l_4$ and $E$ is icy and slippery.

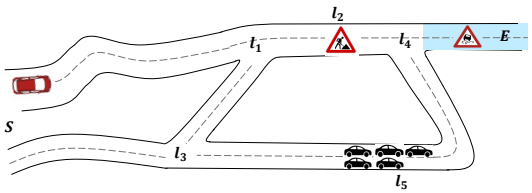



Figure 11. A problem instance in the Self-Driving Car domain with seven locations $S$, $l_1$, $l_2$, $l_3$, $l_4$, $l_5$, and $E$, and various road complexities.

Figure 9 shows some possible bindings of the actions with the corresponding probability distribution of costs with respect to the ground planning problem and external traffic as sources of uncertainty. Note that we only show feasible bindings and ground actions that can be part of the computed plans. For example, the action of accelerating has a probability distribution of costs when the car is accelerating on the road between $S$ and $l_1$, different from the probability distribution of costs when accelerating on the road between $S$ and $l_3$, since the length and traffic jams of these roads are different. The logic behind our assignments of the action variable costs is as follows. The road between $S$ and $l_1$ is complexity-free and free of traffic jams. Thus, the time needed to drive through this road is certain, i.e., driving through this road takes four hours with 100% probability. On the other hand, although shorter, the road between $S$ and $l_3$ has a variable traffic jam throughout the day. Driving through this road can take six hours with 20% probability, or two hours, in the best case, with 80% probability. Thus, taking the short road is riskier than taking the long road. The road between $l_1$ and $l_4$ has construction work at location $l_2$. Since the construction is considered a still incident, passing through this road, i.e., decelerating, dodging the constructions, and accelerating again, is done in a certain time under the assumption that this road does not include any traffic jam. In particular, cumulative deceleration on the road between $l_1$ and $l_2$ requires four hours, dodging the incident requires 0.4 hours, and accelerating on the road between $l_2$ and $l_4$ takes one hour. Unlike the road between $l_1$ and $l_4$, the road between $l_3$ and $l_4$ has multiple schools and heavy traffic at location $l_5$, which can lead to long waiting times. This is an external source of uncertainty since pedestrians have uncertain times and speeds at which they cross the road, which can make this area congested. Additionally, the roads between $l_3$ and $l_5$, and $l_5$ and $l_4$ have an uncertain level of traffic congestion. Thus, driving from $l_3$ to $l_4$, i.e., decelerating, braking, and

accelerating, requires a variable amount of time, as shown in Figure 9. In particular, decelerating on the road between $l_3$ and $l_5$ takes one hour with 10% probability and three hours with 90% probability. Consider the school area is very crowded, and the car might need to brake for a long time, waiting for the pedestrians to walk. Thus, braking before this area can take half an hour with a high probability of 90% and can, in the worst case, take three hours with a probability of 10%.

The road between $l_1$ and $l_3$ is a highway that is complexity-free. However, it has an uncertain level of traffic congestion. Thus, although the car can accelerate on this road, with a small probability of 10%, it can take eight hours to reach $l_3$ when there is a high traffic congestion. With a high probability of 90%, the car can travel from $l_1$ to $l_3$ within two hours since the highway is mostly free of traffic jams. Lastly, the road between $l_4$ and $E$ is icy and slippery. If the agent chooses to decelerate without activating the ESP, the time needed to reach $E$ will be variable and is based on the ability of the car to stabilise on this slippery road. This choice can lead to six hours of driving with 20% probability and eleven hours of driving with 80% probability. If the agent chooses to decelerate after activating the ESP, it will need 10 hours, i.e., a known and certain amount of time, to reach $E$ since this is the safest and most guaranteed option to choose. However, suppose the agent accelerates after activating the ESP. In that case, it will need an uncertain amount of time, depending on the car's stability. This option is very risky since it might require 12 hours in the worst case with an 80% probability as the car will probably lose stability. At the same time, there is a 20% probability that the car will have good stability and reach its destination in two hours since it is accelerating. An even riskier option is to accelerate on this road without activating the ESP. In that case, the car might need sixteen hours to reach location $E$ with a probability of 90%, and, in the best case, it needs half an hour with a probability of 10%. Comparing the option of decelerating after activating the ESP with the option of decelerating without activating the ESP, the first option has a more guaranteed outcome, i.e., execution time, although both options have the same expected value, which is 10 hours. We can also see that the option of accelerating after activating the ESP is riskier than decelerating without activating the ESP, since, with an 80% probability, it might lead to a higher time than the outcome of only decelerating. Lastly, when comparing the options of decelerating without activating the ESP and only decelerating, we see that both options involve risk. However, unless the agent is highly risk-seeking, it is less likely that the first option is preferable since it has the probability of 20% of costing six hours compared to the second option, which has the probability of costing four hours less with the same 20% probability. At the same time, the first option has an 80% probability of costing eleven hours compared to twelve hours for the second option, with the same probability, which means a one-hour difference only. Note that, despite the differences in the risk level, all three options have the same expected value, which is ten hours. The only option that has a higher expected travelling time compared to all other options is accelerating without activating the ESP. That is why this option is the riskiest one and is only chosen if the agent is extremely risk-seeking. Additionally, we extend the possible roads that the car can take by adding the possibility of taking a shortcut road that has a 90% probability of taking two hours and a 10% probability of taking eight hours. This extension increases the number of choices the agent should make according to its risk attitude, since taking the shortcut road can have a high risk of incurring long travelling times compared to taking several longer roads.

We model the objects existing in the planning problem, the initial task network is to drive to a destination, and the initial state is a list of predicates. Listing 7 shows the initial state of the problem instance illustrated in Figure 11.

Listing 7: Initial state of the problem instance illustrated in Figure 11.

```
(:init (connected s l1)(connected s l3)(
    connected l1 l4)(connected l3 l4)(
    connected l4 e)(connected l1 l2)(connected
     l2 l4)(connected l3 l5) (connected l5 l4)
    (clearroadlong s l1)(clearroad s l1)(
    clearroadshort s l3)(clearroad s l3)(
    in-inc construction l2)(in-inc bottleneck
    l5)(clearroad l4 e)(badroad l4 e)(in v0 s)
    )
```