CHAPTER 1 Introduction

This chapter introduces informally the concepts and technical material developed in the rest of the book. It discusses in particular the notion of *deliberation*, which is at the core of the interaction between planning and acting. Section 1.1 motivates our study of deliberation from a computational viewpoint and delineates the scope of the book. We then introduce a conceptual view of an artificial entity, called an *actor*, capable of acting deliberately on its environment, and discuss our main assumptions. Deliberation models and functions are presented next. Section 1.4 describes two application domains that will be simplified into illustrative examples of the techniques covered in rest of the book.

1.1 PURPOSE AND MOTIVATIONS

1.1.1 First Intuition

What is deliberative acting? That is the question we are studying in this book. We address it by investigating the computational reasoning principles and mechanisms supporting how to choose and perform actions.

We use the word *action* to refer to something that an agent does, such as exerting a force, a motion, a perception or a communication, in order to make a change in its environment and own state. An agent is any entity capable of interacting with its environment. An agent acting deliberately is motivated by some intended objective. It performs one or several actions that are justifiable by sound reasoning with respect to this objective.

Deliberation for acting consists of deciding which actions to undertake and how to perform them to achieve an objective. It refers to a *reasoning process*, both before and during acting, that addresses questions such as the following:

- If an agent performs an action, what will the result be?
- Which actions should an agent undertake, and how should the agent perform the chosen actions to produce a desired effect?

https://doi.org/10.1017/CBO9781139583923.003 Published online by Cambridge University Press

Such reasoning allows the agent to predict, to decide what to do and how do it, and to combine several actions that contribute jointly to the objective. The reasoning consists in using predictive models of the agent's environment and capabilities to simulate what will happen if the agent performs an action. Let us illustrate these abstract notions intuitively.

Example 1.1. Consider a bird in the following three scenes:

- To visually track a target, the bird moves its eyes, head, and body.
- To get some food that is out of reach, the bird takes a wire rod, finds a wedge to bend the wire into a hook, uses the hook to get the food.
- To reach a worm floating in a pitcher, the bird picks up a stone and drops it into the pitcher, repeats with other stones until the water has risen to a reachable level, and then picks up the worm. □

Example 1.1 mentions actions such as moving, sensing, picking, bending and throwing. The first scene illustrates a precise coordination of motion and sensing that is called visual servoing. This set of coordinated actions is certainly purposeful: it aims at keeping the target in the field of view. But it is more *reactive* than deliberative. The other two scenes are significantly more elaborate: they demand reasoning about causal relations among interdependent actions that transform objects, and the use of these actions to achieve an objective. They illustrate our intuitive notion of acting deliberately.

The mechanisms for acting deliberately have always been of interest to philosophy.[1] They are a subject of intense research in several scientific disciplines, including biology, neuroscience, psychology, and cognitive sciences. The deliberative bird behaviors of Example 1.1 have been observed and studied from the viewpoint of how deliberative capabilities are developed, in species of corvids such as crows [597] or rooks [71, 70]. Numerous other animal species have the ability to simulate their actions and deliberate on the basis of such simulations.[2] The sophisticated human deliberation faculties are the topic of numerous research, in particular regarding their development in infants and babies, starting from the work of Piaget (as in [478, 479]) to the recent diversity of more formal psychology models (e.g., [563, 19, 461]).

We are interested here in the study of computational deliberation capabilities that allow an *artificial* agent to reason about its actions, choose them, organize them purposefully, and act deliberately to achieve an objective. We call this artificial agent an *actor*. This is to underline the acting functions on which we are focusing and to differentiate them from the broader meaning of the word "agent." We consider *physical actors* such as robots, as well as *abstract actors* that act in simulated or virtual environments, for example, through graphic animation or electronic Web transactions. For both kinds of actors, sensory-motor functions designate in a broad sense the low-level functions that implement the execution of actions.

[1] In particular, the branch of philosophy called *action theory*, which explores questions such as, "What is left over if I subtract the fact that my arm goes up from the fact that I raise my arm?" [610].

[2] In the interesting classification of Dennett [150], these species are called *Popperian*, in reference to the epistemologist Karl Popper.

1.1.2 Motivations

We address the issue of how an actor acts deliberately by following the approaches and methods of *artificial intelligence* (AI). Our purpose proceeds from the usual motivations of AI research, namely:

- To understand, through effective formal models, the cognitive capabilities that correspond to acting deliberately.
- To build actors that exhibit these capabilities.
- To develop technologies that address socially useful needs.

Understanding deliberation is an objective for most cognitive sciences. The specifics of AI are to model deliberation through computational approaches that allow us to explain as well as to generate the modeled capabilities. Furthermore, the investigated capabilities are better understood by mapping concepts and theories into designed systems and experiments to test empirically, measure, and qualify the proposed models. The technological motivation for endowing an artificial actor with deliberation capabilities stems from two factors:

- *autonomy* – that is, the actor performs its intended functions without being directly operated by a person and
- *diversity* in the tasks it can perform and the environments in which it can operate.

Without autonomy, a directly operated or teleoperated device does not usually need to deliberate. It simply extends the acting and sensing capabilities of a human operator who is in charge of understanding and decision making, possibly with the support of advice and planning tools, for example, as in surgical robotics and other applications of teleoperation.

An autonomous system may not need deliberation if it operates only in the fully specified environment for which it has been designed. Manufacturing robots autonomously perform tasks such as painting, welding, assembling, or servicing a warehouse without much deliberation. Similarly, a vending machine or a driverless train operates autonomously without a need for deliberation. For these and similar examples of automation, deliberation is performed by the designer. The system and its environment are engineered so that the only variations that can occur are those accounted for at the design stage in the system's predefined *functioning envelope*. Diversity in the environment is not expected. A state outside of the functioning envelope puts the system into a failure mode in which a person takes deliberate actions.

Similarly, a device designed for a unique specialized task may perform it autonomously without much deliberation, as long the variations in its environment are within its designed range. For example, a vacuum-cleaning or lawn-mowing robot does not deliberate yet can cope autonomously with its specialized tasks in a reasonable range of lawns or floors. But it may cease to function properly when it encounters a slippery floor, a steep slope, or any condition outside of the range for which it was designed.

When a designer can account, within some functioning envelope, for all the environments and tasks a system will face and when a person can be in charge of deliberating outside of this envelope, by means of teleoperation or reprogramming, then deliberation generally is not needed in the system itself. Such a system will be endowed with a library of *reactive* behaviors (e.g., as the bird's visual target tracking in Example 1.1) that cover efficiently its functioning envelope. However, when an autonomous actor has to face a diversity of tasks, environments and interactions, then achieving its purpose will require some degree of deliberation. This is the case in many robotics applications, such as service and personal robots, rescue and exploration robots, autonomous space stations and satellites, or even driverless cars. This holds also for complex simulation systems used in entertainment (e.g., video games) or educational applications (serious games). It is equally applicable to many control systems that manage complex infrastructures such as industrial or energy plants, transportation networks, and urban facilities (smart cities).

Autonomy, diversity in tasks and environments, and the need for deliberation are not binary properties that are either true or false. Rather, the higher the need for autonomy and diversity, the higher the need for deliberation. This relationship is not restricted to artificial systems. Numerous natural species (plants and some invertebrates such as sponges or worms) have been able to evolve to fit into stable ecological niches, apparently without much deliberation. Species that had to face rapid changes in their environment and to adapt to a wide range of living conditions had to develop more deliberation capabilities.

1.1.3 Focus and Scope

We address deliberation from an AI viewpoint. Our focus is on the *reasoning functions* required for acting deliberately. This focus involves two restrictions:

- We are not interested in actions that consists solely of internal computations, such as adding "2 + 3" or deducing that "Socrates is mortal." These computations are not actions that change the state of the world.[3] They can be used as part of the actor's deliberation, but we take them as granted and outside of our scope.
- We are not concerned with techniques for designing the sensing, actuation, and sensory-motor control needed for the low-level execution of actions. Sensory-motor control (e.g., the visual servoing of Example 1.1) can be essential for acting, but its study is not within our scope. We assume that actions are performed with a set of primitives, which we will call *commands*, that implement sensory-motor control. The actor performs its actions by executing commands. To deliberate, it relies on models of how these commands work.

The scope of this book is not limited to the most studied deliberation function, which is *planning* what actions to perform. Planning consists in choosing and organizing the actions that can achieve a given objective. In many situations, there is not much need for planning: the actions to perform are known. But there is a need for significant deliberation in deciding *how* to perform each action, given the context and changes in the environment. We develop the view that planning can be needed for deliberation but is seldom sufficient. We argue that acting goes beyond the execution of low-level commands.

[3] The borderline between computational operations and actions that change the external world is not as sharp for an abstract actor as for a physical one.

Example 1.2. Dana finishes breakfast in a hotel restaurant, and starts going back to his room. On the way, he notices that the elevator is not on his floor and decides to walk up the stairs. After a few steps he becomes aware that he doesn't have his room key, which he left on the breakfast table. He goes back to pick it up. □

In this example, the actor does not need to plan the simple task of going to his room. He *continually* deliberates while acting: he makes opportunistic choices, simulates in advance and monitors his actions, stops when needed and decides on alternate actions.

Deliberation consists of reasoning with predictive models as well as *acquiring* these models. An actor may have to *learn* how to adapt to new situations and tasks, as much as to use the models it knows about for its decision making. Further, even if a problem can be addressed with the actor's generic models, it can be more efficient to transform the explicit computations with these models into low-level sensory-motor functions. Hence, it is natural to consider learning to act as a deliberation function. Section 7.3 offers a brief survey on learning and model acquisition for planning and acting. However, our focus is on deliberation techniques using predefined models.

1.2 CONCEPTUAL VIEW OF AN ACTOR

1.2.1 A Simple Architecture

An actor interacts with the external environment and with other actors. In a simplified architecture, depicted in Figure 1.1(a), the actor has two main components: a set of *deliberation functions* and an *execution platform*.

The actor's sensory-motor functions are part of its execution platform. They transform the actor's commands into actuations that execute its actions (e.g., the movement of a limb or a virtual character). The execution platform also transforms sensed signals into features of the world (e.g., to recognize a physical or virtual object, or to query information from the Web). The capabilities of the platform are explicitly described as models of the available commands.

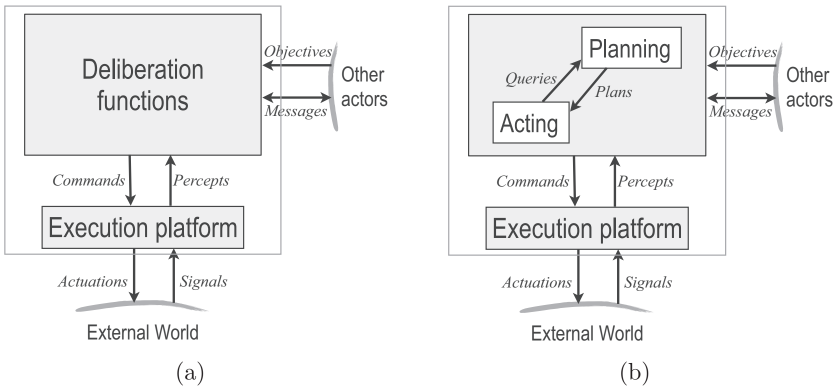

Figure 1.1. Conceptual view of an actor (a); its restriction to planning and acting (b).

Deliberation functions implement the reasoning needed to choose, organize, and perform actions that achieve the actor's objectives, to react adequately to changes in the environment, and to interact with other actors, including human operators. To choose and execute commands that ultimately achieve its objectives, the actor needs to perform a number of deliberation functions. For example, the actor must commit to intermediate goals, plan for those goals, refine each planned action into commands, react to events, monitor its activities to compare the predicted and observed changes, and decide whether recovery actions are needed. These deliberation functions are depicted in Figure 1.1(b) as two main components: planning and acting. The acting component is in charge of refining actions into commands, reacting to events, and monitoring.

1.2.2 Hierarchical and Continual Online Deliberation

The view presented in Section 1.2.1 can be a convenient first approach for describing an actor, but one must keep in mind that it is an oversimplification.

Example 1.3. To respond to a user's request, a robot has to bring an object o7 to a location room2 (see Figure 1.2). To do that, it plans a sequence of abstract actions such as "navigate to," "fetch," and "deliver." One of these refines into "move to door," "open door," "get out," and "close door." Once the robot is at the door, it refines the "open door" action appropriately for how it perceives that particular door.

The robot's deliberation can be accomplished by a collection of hierarchically organized components. In such a hierarchy, a component receives tasks from the component

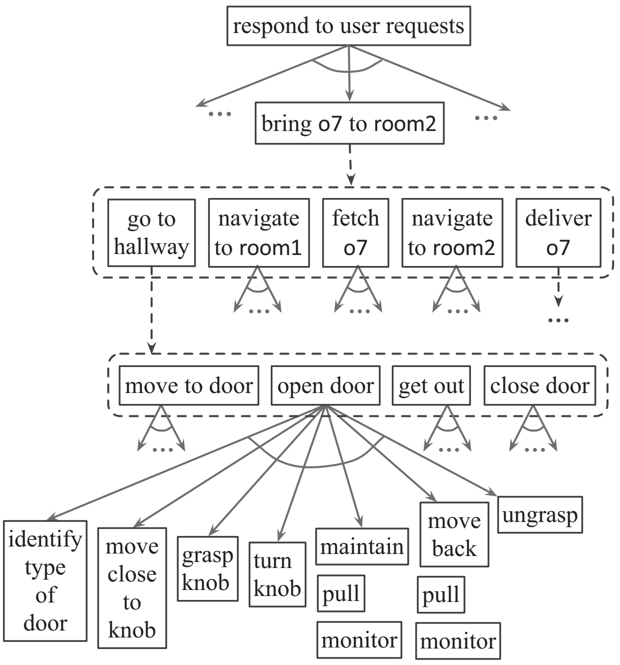

Figure 1.2. Multiple levels of abstraction in deliberative acting. Each solid arrow indicates a refinement of an abstract action into more concrete ones. Each dashed arrow maps a task into a plan of actions.

above it, and decides what activities need to be performed to carry out those tasks. Performing a task may involve refining it into lower-level steps, issuing subtasks to other components below it in the hierarchy, issuing commands to be executed by the platform, and reporting to the component that issued the task. In general, tasks in different parts of the hierarchy may involve concurrent use of different types of models and specialized reasoning functions. □

This example illustrates two important principles of deliberation: hierarchical organization and continual online processing.

- *Hierarchically organized deliberation.* Some of the actions the actor wishes to perform do not map directly into a command executable by its platform. An action may need further refinement and planning. This is done online and may require different representations, tools, and techniques from the ones that generated the task. A hierarchized deliberation process is not intended solely to reduce the search complexity of offline plan synthesis. It is needed mainly to address the heterogeneous nature of the actions about which the actor is deliberating, and the corresponding heterogeneous representations and models that such deliberations require.
- *Continual online deliberation.* Only in exceptional circumstances will the actor do all of its deliberation offline before executing any of its planned actions. Instead, the actor generally deliberates at runtime about how to carry out the tasks it is currently performing. The deliberation remains partial until the actor reaches its objective, including through flexible modification of its plans and retrials. The actor's predictive models are often limited. Its capability to acquire and maintain a broad knowledge about the current state of its environment is very restricted. The cost of minor mistakes and retrials are often lower than the cost of extensive modeling, information gathering, and thorough deliberation. Throughout the acting process, the actor refines and monitors its actions; reacts to events; and extends, updates, and repairs its plan on the basis of its perception focused on the relevant part of the environment.

Different parts of the actor's hierarchy often use different representations of the state of the actor and its environment. These representations may correspond to different amounts of detail in the description of the state and different mathematical constructs. In Figure 1.2, a graph of discrete locations may be used at the upper levels, while the lower levels may use vectors of continuous configuration variables for the robot limbs.

Finally, because complex deliberations can be compiled down by learning into low-level commands, the frontier between deliberation functions and the execution platform is not rigid; it evolves with the actor's experience.

1.2.3 Assumptions

We are not seeking knowledge representation and reasoning approaches that are effective across every kind of deliberation problem and at every level of a hierarchically organized actor. Neither are we interested in highly specialized actors tailored for a single niche, because deliberation is about facing diversity. Instead, we are proposing a few generic approaches that can be adapted to different classes of environments and, for a given actor, to different levels of its deliberation. These approaches rely on restrictive assumptions that are needed from a computational viewpoint, and that are acceptable for the class of environments and tasks in which we are interested.

Deliberation assumptions are usually about how variable, dynamic, observable, and predictable the environment is, and what the actor knows and perceives about it while acting. We can classify them into assumptions related to the dynamics of the environment, its observability, the uncertainty managed in models, and how time and concurrency are handled.

- *Dynamics* of the environment. An actor may assume to be in a static world except for its own actions, or it may take into account exogenous events and changes that are expected and/or observed. In both cases the dynamics of the world may be described using discrete, continuous or hybrid models. Of these, hybrid models are the most general. Acting necessarily involves discontinuities in the interaction with the environment,[4] and these are best modeled discretely. But a purely discrete model abstracts away continuous processes that may also need to be modeled.
- *Observability* of the environment. It is seldom the case that all the information needed for deliberation is permanently known to the actor. Some facts or parameters may be always known, others may be observable if specific sensing actions are performed, and others will remain hidden. The actor may have to act on the basis of reasonable assumptions or beliefs regarding the latter.
- *Uncertainty* in knowledge and predictions. No actor is omniscient. It may or may not be able to extend its knowledge with specific actions. It may or may not be able to reason about the uncertainty regarding the current state of the world and the predicted future (e.g., with nondeterministic or probabilistic models). Abstracting away uncertainty during a high-level deliberation can be legitimate if the actor can handle it at a lower level and correct its course of action when needed.
- *Time and concurrency*. Every action consumes time. But deliberation may or may not need to model it explicitly and reason about its flow for the purpose of meeting deadlines, synchronizing, or handling concurrent activities.

Different chapters of the book make different assumptions about time, concurrency, and uncertainty. Except for Section 7.4 on hybrid models, we'll restrict ourself to discrete approaches. This is consistent with the focus and scope discussed in Section 1.1.3, because it is primarily in sensory-motor functions and commands that continuous models are systematically needed.

1.3 DELIBERATION MODELS AND FUNCTIONS

1.3.1 Descriptive and Operational Models of Actions

An actor needs predictive models of its actions to decide *what* actions to do and *how* to do them. These two types of knowledge are expressed with, respectively, descriptive and operational models.

[4] Think of the phases in a walking or grasping action.

- *Descriptive models* of actions specify the actor's "know what." They describe which state or set of possible states may result from performing an action or command. They are used by the actor to reason about what actions may achieve its objectives.
- *Operational models* of actions specify the actor's "know how." They describe how to perform an action, that is, what commands to execute in the current context, and how organize them to achieve the action's intended effects. The actor relies on operational models to perform the actions that it has decided to perform.

In general, descriptive models are more abstract than operational models. Descriptive models abstract away the details, and focus on the main effects of an action; they are useful at higher levels of a deliberation hierarchy. This abstraction is needed because often it is too difficult to develop very detailed predictive models, and because detailed models require information that is unknown at planning time. Furthermore, reasoning with detailed models is computationally very complex. For example, if you plan to take a book from a bookshelf, at planning time you will not be concerned with the available space on the side or on the top of the book to insert your fingers and extract the book from the shelf. The descriptive model of the action will abstract away these details. It will focus on where the book is, whether it is within your reach, and whether you have a free hand to pick it up.

The simplifications allowed in a descriptive model are not possible in an operational model. To actually pick up the book, you will have to determine precisely where the book is located in the shelf, which positions of your hand and fingers are feasible, and which sequences of precise motions and manipulations will allow you to perform the action.

Furthermore, operational models may need to include ways to respond to *exogenous* events, that is, events that occur because of external factors beyond the actor's control. For example, someone might be standing in front of the bookshelf, the stool that you intended to use to reach the book on a high shelf might be missing, or any of a potentially huge number of other possibilities might interfere with your plan.

In principle, descriptive models can take into account the uncertainty caused by exogenous events, for example, through nondeterministic or probabilistic models (see Chapters 5 and 6), but the need to handle exogenous events is much more compelling for operational models. Indeed, exogenous events are often ignored in descriptive models because it is impractical to try to model all of the possible joint effects of actions and exogenous events, or to plan in advance for all of the contingencies. But operational models must have ways to respond to such events if they happen because they can interfere with the execution of an action. In the library example, you might need to ask someone to move out of the way, or you might have to stand on a chair instead of the missing stool.

Finally, an actor needs descriptive models of the available commands in order to use them effectively, but in general it does not need their operational models. Indeed, commands are the lower-level sensory-motor primitives embedded in the execution platform; their operational models correspond to what is implemented in these primitives. Taking this remark to the extreme, if one assumes that every known action corresponds to an executable command, then all operational models are embedded in the execution platform and can be ignored at the deliberation level. This assumption seldom holds.

1.3.2 Description of States for Deliberation

To specify both descriptive and operational models of actions, we will use representational primitives that define the state of an actor and its environment; these are called *state variables*. A state variable associates a value, which changes over time, to a relevant attribute of the world. The definition of a state with state variables needs to include enough details for the actor's deliberations, but it does not need to be, nor can it be, exhaustive.

In a hierarchically organized actor, different deliberative activities may need different amounts of detail in the state description. For example, in actions such as "grasp knob" and "turn knob" at the bottom of Figure 1.2, to choose the commands for grasping the handle and operating it, the actor needs to reason about detailed parameters such as the robot's configuration coordinates and the position and shape of the door handle. Higher up, where the actor refines "bring o7 to room2" into actions such as "go to hallway" and "navigate to room1," such details are not needed. It is more convenient there to reason about the values of more abstract variables, such as $\mathsf{location(robot)} = \mathsf{room1}$ or $\mathsf{position(door)} = \mathsf{closed}$. To establish correspondences between these abstract variables and the detailed ones, the actor could have definitions saying, for example, that $\mathsf{location(robot)} = \mathsf{room1}$ corresponds to a particular area in an Euclidean reference frame.

The precise organization of a hierarchy of data structures and state representations is a well-known area in computer science (e.g., [522]). It may take different forms in application domains such as robotics, virtual reality, or geographic information systems. Here, we'll keep this point as simple as possible and assume that at each part of an actor's deliberation hierarchy, the state representation includes not only the variables used in that part of the hierarchy (e.g., the robot's configuration coordinates at the bottom of Figure 1.2), but also the variables used higher up in the hierarchy (e.g., location(robot)).

An important issue is the distinction and correspondence between *predicted* states and *observed* states. When an actor reasons about what might happen and simulates changes of state to assess how desirable a course of action is, it uses predicted states. When it reasons about how to perform actions in some context, it relies on observed states; it may contrast its observations with its expectations. Predicted states are in general less detailed than the observed one; they are obtained as a result of one or several predictions starting from an abstraction of the current observed state. To keep the distinction clear, we'll use different notations:

- $s \in S$ is a predicted state;
- $\xi \in \Xi$ is an observed state.

Because of partial and inaccurate observations, there can be uncertainty about the *present observed* state as well as about the *future predicted* states. Furthermore, information in a dynamic environment is ephemeral. Some of the values in ξ may be out-of-date: they may refer to things that the actor previously observed but that it cannot currently observe. Thus, ξ is the state of the actor's knowledge, rather than the true state of the world. In general, the actor should be endowed with appropriate means to manage the uncertainty and temporality of the data in ξ.

Observability is an additional issue. As underlined in Section 1.2.3, some information relevant to the actor's behavior can be momentarily or permanently hidden; it has to be indirectly inferred. In the general case, the design of an actor should include the following distinctions among state variables:

- A variable is *invisible* if it is not observable but can only be estimated from observations and a priori information.
- A variable is *observable* if its value can be obtained by performing appropriate actions. At various points, it may be either *visible* if its value is known to the actor, or *hidden* if the actor must perform an observation action to get its value.

For simplicity, we'll start out by assuming that the values of all state variables are precisely known at every moment while acting. Later in the book, we'll consider more realistically that some state variables are observable but can only be observed by performing some specific actions. In Chapter 5, we deal with a specific case of partial observability: in Section 5.8.4, we transform a partially observable domain into an abstracted domain whose states are sets of states. We also examine (in Chapter 6) the case in which some state variables are permanently or momentarily observable but others remain hidden. The class of models known as partially observable models, in which every state variable is assumed to be always known or always hidden, is discussed in Section 6.8.3.

1.3.3 Planning Versus Acting

The simple architecture of Figure 1.1(b) introduces planning and acting as respectively finding what actions to perform and how to refine chosen actions into commands. Here, we further discuss these two functions, how they differ, and how they can be associated in the actor's deliberation.

The purpose of *planning* is to synthesize an organized set of actions to carry out some activity. For instance, this can be done by a lookahead procedure that combines prediction steps (Figure 1.3: when in state s, action a is predicted to produce state s') within a search through alternative sets of actions for a set that leads to a desired goal state.

Planning problems vary in the kinds of actions to be planned for, the kinds of predictive models that are needed, and the kinds of plans that are considered satisfactory.

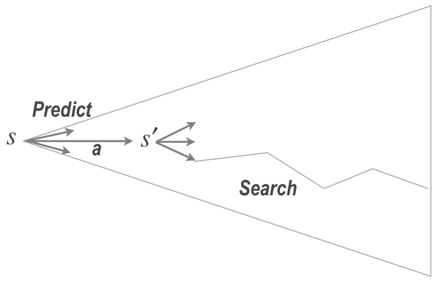

Figure 1.3. Planning as a combination of prediction steps and a search mechanism.

For some kinds of problems, *domain-specific* planning methods have been developed that are tailor-made for that kind of problem. For instance, motion planning synthesizes a geometric and kinematic trajectory for moving a mobile system (e.g., a truck, a robot, or a virtual character); perception planning synthesizes an organized set of sensing and interpretation actions to recognize an object or to build a three-dimensional model of a scene; infrastructure planning synthesizes plans to deploy and organize facilities, such as a public transportation infrastructure, to optimize their usage or to meet the needs of a community. Many other such examples can be given, such as flight navigation planning, satellite configuration planning, logistics planning, or industrial process planning.

There are, however, commonalities to many forms of planning. *Domain-independent* planning tries to grasp these commonalities at an abstract level, in which actions are generic state transformation operators over a widely applicable representation of states as relations among objects.

Domain-independent and domain-specific planning complement each other. In a hierarchically organized actor, planning takes place at multiple levels of the hierarchy. At high levels, abstract descriptions of a problem can be tackled using domain-independent planning techniques. The example shown in Figure 1.2 may require a path planner (for moving to locations), a manipulation planner (for grasping the door handle), and a domain-independent planner at the higher levels of the hierarchy.

Acting involves deciding how to perform the chosen actions (with or without the help of a planner) while reacting to the context in which the activity takes place. Each action is considered as an abstract task to be refined, given the current context, progressively into more concrete actions or commands. Whereas planning is a search over predicted states, acting requires a *continual assessment of the current state* ξ, to contrast it with a predicted state s and adapt accordingly. Consequently, acting also includes *reacting* to unexpected changes and exogenous events, which are independent from the actor's activity.

The techniques used in planning and acting can be compared as follows. Planning can be organized as an open-loop search, whereas acting needs to be a closed-loop process. Planning relies on descriptive models (know-what); acting uses mostly operational models (know-how). Domain-independent planners can be developed to take advantage of commonalities among different forms of planning problems, but this is less true for acting systems, which require more domain-specific programming.

The relationship between planning and acting is more complex than a simple linear sequence of "plan then act." Seeking a complete plan before starting to act is not always feasible, and not always needed. It is feasible when the environment is predictable and well modeled, for example, as for a manufacturing production line. It is needed when acting has a high cost or risk, and when actions are not reversible. Often in such applications, the designer has to engineer out the environment to reduce diversity as much as possible beyond what is modeled and can be predicted.

In dynamic environments where exogenous events can take place and are difficult to model and predict beforehand, plans should be expected to fail if carried out blindly until the end. Their first steps are usually more reliable than the rest and steer toward the objectives. Plan modification and replanning are normal and should be embedded

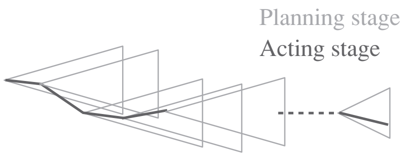

Figure 1.4. Receding horizon scheme for planning and acting.

in the design of an actor. Metaphorically, planning is useful to shed light on the road ahead, not to lay an iron rail all the way to the goal.

The interplay between acting and planning can be organized in many ways, depending on how easy it is to plan and how quickly the environment changes. A general paradigm is the *receding horizon* scheme, which is illustrated in Figure 1.4. It consists of repeating the two following steps until the actor has accomplished its goal:

(i) Plan from the current state toward the goal, but not necessarily all the way to the goal.
(ii) Act by refining one or a few actions of the synthesized plan into commands to be executed.

A receding horizon approach can be implemented in many ways. Options include various planning horizon, number of actions to perform at each planning stage, and what triggers replanning. Furthermore, the planning and acting procedures can be run either sequentially or in parallel with synchronization.

Suppose an actor does a depth-first refinement of the hierarchy in Figure 1.2. Depending on the actor's planning horizon, it may execute each command as soon as one is planned or wait until the planning proceeds a bit farther. Recall from Section 1.3.2 that the observed state ξ may differ from the predicted one. Furthermore, ξ may evolve even when no commands are being executed. Such situations may invalidate what is being planned, necessitating replanning.

The interplay between acting and planning is relevant even if the planner synthesizes alternative courses of action for different contingencies (see Chapters 5 and 6). Indeed, it may not be worthwhile to plan for all possible contingencies, or the planner may not know in advance what all of them are.

1.3.4 Other Deliberation Functions

We have mentioned deliberation functions other than planning and acting: perceiving, monitoring, goal reasoning, communicating, and learning. These functions (surveyed in Chapter 7) are briefly described here.

Perceiving goes beyond sensing, even with elaborate signal processing and pattern matching methods. Deliberation is needed in bottom-up processes for getting meaningful data from sensors, and in top-down activities such as focus-of-attention mechanisms, reasoning with sensor models, and planning how to do sensing and information gathering. Some of the issues include how to maintain a mapping between sensed data and deliberation symbols, where and how to use the platform sensors, or how to recognize actions and plans of other actors.

Monitoring consists of comparing observations of the environment with what the actor's deliberation has predicted. It can be used to detect and interpret discrepancies, perform diagnosis, and trigger initial recovery actions when needed. Monitoring may require planning what observation actions to perform, and what kinds of diagnosis tests to perform. There is a strong relationship between planning techniques and diagnosis techniques.

Goal reasoning is monitoring of the actor's objectives or mission, to keep the actor's commitments and goals in perspective. It includes assessing their relevance, given the observed evolutions, new opportunities, constraints or failures, using this assessment to decide whether some commitments should be abandoned, and if so, when and how to update the current goals.

Communicating and *interacting* with other actors open numerous deliberation issues such as communication planning, task sharing and delegation, mixed initiative planning, and adversarial interaction.

Learning may allow an actor to acquire, adapt, and improve through experience the models needed for deliberation and to acquire new commands to extend and improve the actor's execution platform. Conversely, techniques such as active learning may themselves require acting for the purpose of better learning.

1.4 ILLUSTRATIVE EXAMPLES

To illustrate particular representations and algorithms, we'll introduce a variety of examples inspired by two application domains: robotics and operations management. We'll use highly simplified views of these applications to include only the features that are relevant for the issue we're trying to illustrate. In this section, we provide summaries of the real-world context in which our simple examples might occur.

1.4.1 A Factotum Service Robot

We will use the word *factotum* to mean a general-purpose service robot that consists of a mobile platform equipped with several sensors (lasers, cameras, etc.) and actuators (wheels, arms, forklift) [329]. This robot operates in structured environments such as a mall, an office building, a warehouse or a harbor. It accomplishes transportation and logistics tasks autonomously, (e.g., fetching objects, putting them into boxes, assembling boxes into containers, move them around, delivering them or piling them up in storage areas).

This robot platform can execute parameterized commands, such as localize itself in the map, move along a path, detect and avoid obstacles, identify and locate items, and grasp, ungrasp and push items. It knows about a few actions using these commands, for example, map the environment (extend or update the map), goto a destination, open a door, search for or fetch an item.

These actions and commands are specified with descriptive and operational models. For example, move works if it is given waypoints in free space or an obstacle-free path that meet kinematics and localization constraints; the latter are, for example, visual landmarks required by action localize. These conditions need to be checked and monitored by the robot while performing the actions. Concurrency has to be managed. For example, goto should run in parallel with detect, avoid, and localize.

Factotum needs domain-specific planners, for example, a motion planner for move, a manipulation planner for grasp (possibly using locate, push, and move actions). Corresponding plans are more than a sequence or a partially ordered set of commands; they require closed-loop control and monitoring.

At the mission-preparation stage (the upper levels in Figure 1.2), it is legitimate to view a logistics task as an organized set of abstract subtasks for collecting, preparing, conveying, and delivering the goods. Each subtask may be further decomposed into a sequence of still abstract actions such as goto, take, and put. Domain-independent task planning techniques are needed here.

However, deliberation does not end with the mission preparation stage. A goto action can be performed in many ways depending on the environment properties: it may or may not require a planned path; it may use different localization, path following, motion control, detection, and avoidance methods (see the "goto" node in Figure 1.2). A goto after a take is possibly different from the one before because of the held object. To perform a goto action in different contexts, the robot relies on a collection of *skills* defined formally by *methods*. A method specifies a way to refine an action into commands. The same goto may start with a method (e.g., follow GPS waypoints) but may be pursued with more adapted methods when required by the environment (indoor without GPS signal) or the context. Such a change between methods may be a normal progression of the goto action or a retrial due to complications. The robot also has methods for take, put, open, close, and any other actions it may need to perform. These methods endow the robot with operational models (its know-how) and knowledge about how to choose the most adapted method with the right parameters.

The methods for performing actions may use complex control constructs with concurrent processes (loops, conditionals, semaphores, multithread and real-time locks). They can be developed from formal specifications in some representation and/or with plan synthesis techniques. Different representations may be useful to cover the methods needed by the factotum robot. Machine learning techniques can be used for improving the methods, acquiring their models, and adapting the factotum to a new trade.

In addition to acting with the right methods, the robot has to monitor its activity at every level, including possibly at the goal level. Prediction of what is needed to correctly perform and monitor foreseen activities should be made beforehand. Making the right predictions from the combined models of actions and models of the environment is a difficult problem that involves heterogeneous representations.

Finally, the robot requires extended *perception* capabilities: reasoning on what is observable and what is not, integrating knowledge-gathering actions to environment changing actions, acting to maintain sufficient knowledge for the task at hand with a consistent interpretation of self and the world.

1.4.2 A Complex Operations Manager

A *Harbor Operations Manager (HOM)* is a system that supervises and controls all the tasks performed in a harbor.[5] Examples of such tasks include unloading cars from ships, parking them in storage areas, moving them to a repair area, performing the repair,

[5] Example inspired from a facility developed for the port of Bremen, Germany [76, 100].

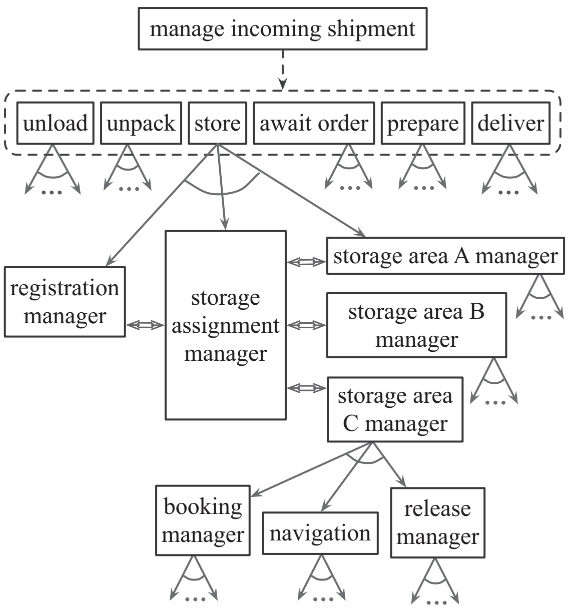

Figure 1.5. Deliberation components for a Harbor Operation Manager.

preparing the delivery of cars according to orders, and loading them onto trucks when the trucks arrive at the harbor. Some of these operations are performed by human workers, others automatically by machines such as the factotum robot of previous section. This complex environment has several features that require deliberation:

- It is *customizable*: for example, delivery procedures can be customized according to the car brand, model, or retailer-specific requirements.
- It is *variable*: procedures for unloading/loading cars depend on the car brands; storage areas have different parking procedures, for example.
- It is *dynamic*: ships, cars, trucks, and orders arrive dynamically.
- It is partially *predictable* and *controllable*: cars may be damaged and need repair, storage areas may not be available, orders have unpredictable requirements, ships and trucks have random delays, for example.

At a high level, an HOM has to carry out a simple sequence of abstract tasks: ⟨unload, unpack, store, wait-for-order, treatment, delivery⟩ (see Figure 1.5). This invariant plan is easily specified by hand. The deliberation problem in an HOM is not in the synthesis of this plan but in the dynamic refinement of its tasks in more concrete subtasks. For example, an HOM refines the abstract task store of Figure 1.5 into subtasks for registering a car to be stored, moving it, and other tasks, down to executable commands.

Moreover, the tasks to be refined and controlled are carried out by different *components*, for example, ships, gates, and storage or repair areas. Each ship has its own procedure to unload cars to a gate. A gate has its own procedure to accept cars that are unloaded to the deck. A natural design option is therefore to model the HOM in a distributed way, as a set of interacting deliberation components. The interactions between ships and gates, gates and trucks, and trucks and storage areas must be controlled with respect to the global constraints and objectives of the system. To do that, HOM must deal with uncertainty and nondeterminism due to exogenous events, and to the fact that each component may – from the point of view of the management facility – behave nondeterministically. For instance, in the task to synchronize a ship with a gate to unload cars, the ship may send a request for unloading cars to the unloading manager, and the gate may reply either that the request meets its requirements and the unloading operation can proceed according to some unloading specifications, or that the request cannot be handled. The management facility may not know a priori what the request, the unloading specifications, and reply will be.

In summary, HOM relies on a collection of interacting components, each implementing its own procedures. It refines the abstract tasks of the high-level plan into a composition of these procedures to address each new object arrival and adapt to each exogenous event. The refinement and adaptation mechanisms can be designed through an approach in which the HOM is an actor organized into a hierarchy of components, each abstract action is a task to be further refined and planned for, and where online planning and acting are performed continually to adapt and repair plans. The approach embeds one or several planners within these components, which are called at run-time, when the system has to refine an abstract action to adapt to a new context. It relies on refinement mechanisms that can be triggered at run-time whenever an abstract action in a procedure needs to be refined or an adaptation needs to be taken into account.

1.5 OUTLINE OF THE BOOK

This chapter has provided a rather abstract and broad introduction. Chapter 2 offers more concrete material regarding deliberation with deterministic models and full knowledge about a static environment. It covers the "classical planning" algorithms and heuristics, with state-space search, forward and backward, and plan-space search. It also presents how these planning techniques can be integrated online with acting.

Chapter 3 is focused on refinement methods for acting and planning. It explores how a unified representation can be used for both functions, at different levels of the deliberation hierarchy, and in different ways. It also discusses how the integration of planning and acting can be performed.

Chapter 4 is about deliberation with explicit time models using a representation with timelines and chronicles. A temporal planner, based on refinement methods, is presented together with the constraint management techniques needed for handling temporal data. Using the techniques from Chapter 3, we also discuss the integration of planning and acting with temporal models.

Uncertainty in deliberation is addressed in Chapters 5 and 6. The main planning techniques in nondeterministic search spaces are covered in Chapter 5, together with model checking and determinization approaches. In this chapter, we present online lookahead methods for the interleaving of planning and acting. We also show how nondeterministic models can be used with refinements techniques that intermix plans, actions, and goals. We discuss the integration of planning and acting with input/output automata to cover cases such as the distributed deliberation in the HOM example.

We cover probabilistic models in Chapter 6. We develop heuristic search techniques for stochastic shortest path problems. We present online approaches for planning and acting, discuss refinement methods for acting with probabilistic models, and analyze the specifics of descriptive models of actions in the probabilistic case together with several practical issues for modeling probabilistic domains.

Chapters 2 through 6 are devoted to planning and acting. Chapter 7 briefly surveys the other deliberation functions introduced in Section 1.3.4, – perceiving, monitoring, goal reasoning, interacting, and learning. It also discusses hybrid models and ontologies for planning and acting.