Chapter 11

Sensors and Information Spaces

Up until now it has been assumed everywhere that the current state is known. What if the state is not known? In this case, information regarding the state is obtained from sensors during the execution of a plan. This situation arises in most applications that involve interaction with the physical world. For example, in robotics it is virtually impossible for a robot to precisely know its state, except in some limited cases. What should be done if there is limited information regarding the state? A classical approach is to take all of the information available and try to estimate the state. In robotics, the state may include both the map of the robot's environment and the robot configuration. If the estimates are sufficiently reliable, then we may safely pretend that there is no uncertainty in state information. This enables many of the planning methods introduced so far to be applied with little or no adaptation.

The more interesting case occurs when state estimation is altogether avoided. It may be surprising, but many important tasks can be defined and solved without ever requiring that specific states are sensed, even though a state space is defined for the planning problem. To achieve this, the planning problem will be expressed in terms of an *information space*. Information spaces serve the same purpose for sensing problems as the configuration spaces of Chapter 4 did for problems that involve geometric transformations. Each information space represents the place where a problem that involves sensing uncertainty naturally lives. Successfully formulating and solving such problems depends on our ability to manipulate, simplify, and control the information space. In some cases elegant solutions exist, and in others there appears to be no hope at present of efficiently solving them. There are many exciting open research problems associated with information spaces and sensing uncertainty in general.

Recall the situation depicted in Figure 11.1, which was also shown in Section 1.4. It is assumed that the state of the environment is not known. There are three general sources of information regarding the state:

1. The *initial conditions* can provide powerful information before any actions are applied. It might even be the case that the initial state is given. At the other extreme, the initial conditions might contain no information.

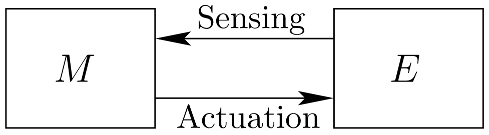


Figure 11.1: The state of the environment is not known. The only information available to make inferences is the history of sensor observations, actions that have been applied, and the initial conditions. This history becomes the *information state*.

2. The *sensor observations* provide measurements related to the state during execution. These measurements are usually incomplete or involve disturbances that distort their values.

3. The *actions* already executed in the plan provide valuable information regarding the state. For example, if a robot is commanded to move east (with no other uncertainties except an unknown state), then it is expected that the state is further east than it was previously. Thus, the applied actions provide important clues for deducing possible states.

Keep in mind that there are generally two ways to use the information space:

1. *Take all of the information available, and try to estimate the state.* This is the classical approach. Pretend that there is no longer any uncertainty in state, but prove (or hope) that the resulting plan works under reasonable estimation error. A plan is generally expressed as $\pi : X \rightarrow U$.

2. *Solve the task entirely in terms of an information space.* Many tasks may be achieved without ever knowing the exact state. The goals and analysis are formulated in the information space, without the need to achieve particular states. For many problems this results in dramatic simplifications. A plan is generally expressed as $\pi : \mathcal{I} \rightarrow U$ for an information space, $\mathcal{I}$.

The first approach may be considered somewhat traditional and can be handled by the concepts of Chapter 8 once a good estimation technique is defined. Most of the focus of the chapter is on the second approach, which represents a powerful way to express and solve planning problems.

For brevity, "information" will be replaced by "I" in many terms. Hence, information spaces and information states become I-spaces and I-states, respectively. This is similar to the shortening of configuration spaces to C-spaces.

Sections 11.1 to 11.3 first cover information spaces for discrete state spaces. This case is much easier to formulate than information spaces for continuous spaces. In Sections 11.4 to 11.6, the ideas are extended from discrete state spaces to continuous state spaces. It is helpful to have a good understanding of the discrete case before proceeding to the continuous case. Section 11.7 extends the formulation of information spaces to game theory, in which multiple players interact over the same state space. In this case, each player in the game has its own information space over which it makes decisions.

11.1 Discrete State Spaces

11.1.1 Sensors

As the name suggests, *sensors* are designed to sense the state. Throughout all of this section it is assumed that the state space, X, is finite or countably infinite, as in Formulations 2.1 and 2.3. A *sensor* is defined in terms of two components: 1) an *observation space*, which is the set of possible readings for the sensor, and 2) a *sensor mapping*, which characterizes the readings that can be expected if the current state or other information is given. Be aware that in the planning model, the state is not really given; it is only assumed to be given when modeling a sensor. The sensing model given here generalizes the one given in Section 9.2.3. In that case, the sensor provided information regarding θ instead of x because state spaces were not needed in Chapter 9.

Let Y denote an *observation space*, which is a finite or countably infinite set. Let h denote the *sensor mapping*. Three different kinds of sensor mappings will be considered, each of which is more complicated and general than the previous one:

1. **State sensor mapping:** In this case, $h : X \rightarrow Y$, which means that given the state, the observation is completely determined.

2. **State-nature sensor mapping:** In this case, a finite set, $\Psi(x)$, of *nature sensing actions* is defined for each $x \in X$. Each nature sensing action, $\psi \in \Psi(x)$, interferes with the sensor observation. Therefore, the state-nature mapping, h, produces an observation, $y = h(x, \psi) \in Y$, for every $x \in X$ and $\psi \in \Psi(x)$. The particular ψ chosen by nature is assumed to be unknown during planning and execution. However, it is specified as part of the sensing model.

3. **History-based sensor mapping:** In this case, the observation could be based on the current state or any previous states. Furthermore, a nature sensing action could be applied. Suppose that the current stage is k. The set of nature sensing actions is denoted by $\Psi_k(x)$, and the particular nature sensing action is $\psi_k \in \Psi_k(x)$. This yields a very general sensor mapping,

$$y_k = h_k(x_1, \ldots, x_k, \psi_k), \tag{11.1}$$

in which y_k is the observation obtained in stage k. Note that the mapping is denoted as h_k because the domain is different for each k. In general, any of the sensor mappings may be stage-dependent, if desired.

Many examples of sensors will now be given. These are provided to illustrate the definitions and to provide building blocks that will be used in later examples of I-spaces. Examples 11.1 to 11.6 all involve state sensor mappings.

Example 11.1 (Odd/Even Sensor) Let $X = \mathbb{Z}$, the set of integers, and let $Y = \{0, 1\}$. The sensor mapping is

$$y = h(x) = \begin{cases} 0 & \text{if } x \text{ is even} \\ 1 & \text{if } x \text{ is odd.} \end{cases} \tag{11.2}$$

The limitation of this sensor is that it only tells whether $x \in X$ is odd or even. When combined with other information, this might be enough to infer the state, but in general it provides incomplete information. ■

Example 11.2 (Mod Sensor) Example 11.1 can be easily generalized to yield the remainder when x is divided by k for some fixed integer k. Let $X = \mathbb{Z}$, and let $Y = \{0, 1, \ldots, k-1\}$. The sensor mapping is

$$y = h(x) = x \bmod k. \tag{11.3}$$

■

Example 11.3 (Sign Sensor) Let $X = \mathbb{Z}$, and let $Y = \{-1, 0, 1\}$. The sensor mapping is

$$y = h(x) = \operatorname{sgn} x. \tag{11.4}$$

This sensor provides very limited information because it only indicates on which side of the boundary $x = 0$ the state may lie. It can, however, precisely determine whether $x = 0$. ■

Example 11.4 (Selective Sensor) Let $X = \mathbb{Z} \times \mathbb{Z}$, and let $(i, j) \in X$ denote a state in which $i, j \in \mathbb{Z}$. Suppose that only the first component of (i, j) can be observed. This yields the sensor mapping

$$y = h(i, j) = i. \tag{11.5}$$

An obvious generalization can be made for any state space that is formed from Cartesian products. The sensor may reveal the values of one or more components, and the rest remain hidden. ■

Example 11.5 (Bijective Sensor) Let X be any state space, and let $Y = X$. Let the sensor mapping be any bijective function $h : X \rightarrow Y$. This sensor provides information that is equivalent to knowing the state. Since h is bijective, it can be inverted to obtain $h^{-1} : Y \rightarrow X$. For any $y \in Y$, the state can be determined as $x = h^{-1}(y)$.

A special case of the *bijective sensor* is the *identity sensor*, for which h is the identity function. This was essentially assumed to exist for all planning problems covered before this chapter because it immediately yields the state. However, any bijective sensor could serve the same purpose. ■

Example 11.6 (Null Sensor) Let X be any state space, and let $Y = \{0\}$. The *null sensor* is obtained by defining the sensor mapping as $h(x) = 0$. The sensor reading remains fixed and hence provides no information regarding the state. ■

From the examples so far, it is tempting to think about partitioning X based on sensor observations. Suppose that in general a state mapping, h, is not bijective, and let $H(y)$ denote the following subset of X:

$$H(y) = \{x \in X \mid y = h(x)\}, \tag{11.6}$$

which is the *preimage* of y. The set of preimages, one for each $y \in Y$, forms a partition of X. In some sense, this indicates the "resolution" of the sensor. A bijective sensor partitions X into singleton sets because it contains perfect information. At the other extreme, the null sensor partitions X into a single set, X itself. The sign sensor appears slightly more useful because it partitions X into three sets: $H(1) = \{1, 2, \ldots\}$, $H(-1) = \{\ldots, -2, -1\}$, and $H(0) = \{0\}$. The preimages of the selective sensor are particularly interesting. For each $i \in \mathbb{Z}$, $H(i) = \mathbb{Z}$. The partitions induced by the preimages may remind those with an algebra background of the construction of quotient groups via homomorphisms [769].

Next consider some examples that involve a state-action sensor mapping. There are two different possibilities regarding the model for the nature sensing action:

1. **Nondeterministic:** In this case, there is no additional information regarding which $\psi \in \Psi(x)$ will be chosen.
2. **Probabilistic:** A probability distribution is known. In this case, the probability, $P(\psi|x)$, that ψ will be chosen is known for each $\psi \in \Psi(x)$.

These two possibilities also appeared in Section 10.1.1, for nature actions that interfere with the state transition equation.

It is sometimes useful to consider the state-action sensor model as a probability distribution over Y for a given state. Recall the conversion from $P(\psi|\theta)$ to $P(y|\theta)$ in (9.28). By replacing Θ by X, the same idea can be applied here. Assume that if the domain of h is restricted to some $x \in X$, it forms an injective (one-to-one) mapping from Ψ to Y. In this case,

$$P(y|x) = \begin{cases} P(\psi|x) & \text{for the unique } \psi \text{ such that } y = h(x, \psi). \\ 0 & \text{if no such } \psi \text{ exists.} \end{cases} \tag{11.7}$$

If the injective assumption is lifted, then $P(\psi|x)$ is replaced by a sum over all ψ for which $y = h(x, \psi)$.

Example 11.7 (Sensor Disturbance) Let $X = \mathbb{Z}$, $Y = \mathbb{Z}$, and $\Psi = \{-1, 0, 1\}$. The idea is to construct a sensor that would be the identity sensor if it were not for the interference of nature. The sensor mapping is

$$y = h(x, \psi) = x + \psi. \tag{11.8}$$

It is always known that $|x-y| \leq 1$. Therefore, if y is received as a sensor reading, one of the following must be true: $x = y-1$, $x = y$, or $x = y+1$. ■

Example 11.8 (Disturbed Sign Sensor) Let $X = \mathbb{Z}$, $Y = \{-1, 0, 1\}$, and $\Psi = \{-1, 0, 1\}$. Let the sensor mapping be

$$y = h(x, \psi) = \operatorname{sgn}(x + \psi). \tag{11.9}$$

In this case, if $y = 0$, it is no longer known for certain whether $x = 0$. It is possible that $x = -1$ or $x = 1$. If $x = 0$, then it is possible for the sensor to read -1, 0, or 1. ■

Example 11.9 (Disturbed Odd/Even Sensor) It is not hard to construct examples for which some mild interference from nature destroys all of the information. Let $X = \mathbb{Z}$, $Y = \{0, 1\}$, and $\Psi = \{0, 1\}$. Let the sensor mapping be

$$y = h(x, \psi) = \begin{cases} 0 & \text{if } x + \psi \text{ is even.} \\ 1 & \text{if } x + \psi \text{ is odd.} \end{cases} \tag{11.10}$$

Under the nondeterministic model for the nature sensing action, the sensor provides no useful information regarding the state. Regardless of the observation, it is never known whether x is even or odd. Under a probabilistic model, however, this sensor may provide some useful information. ■

It is once again informative to consider preimages. For a state-action sensor mapping, the preimage is

$$H(y) = \{x \in X \mid \; \exists \psi \in \Psi(x) \text{ for which } y = h(x, \psi)\}. \tag{11.11}$$

In comparison to state sensor mappings, the preimage sets are larger for state-action sensor mappings. Also, they do not generally form a partition of X. For example, the preimages of Example 11.8 are $H(1) = \{0, 1, \ldots\}$, $H(0) = \{-1, 0, 1\}$, and $H(-1) = \{\ldots, -2, -1, 0\}$. This is not a partition because every preimage contains 0. If desired, $H(y)$ can be directly defined for each $y \in Y$, instead of explicitly defining nature sensing actions.

Finally, one example of a history-based sensor mapping is given.

Example 11.10 (Delayed-Observation Sensor) Let $X = Y = \mathbb{Z}$. A *delayed-observation sensor* can be defined for some fixed positive integer i as $y_k = x_{k-i}$. It indicates what the state was i stages ago. In this case, it gives a perfect measurement of the old state value. Many other variants are possible. For example, it might only give the sign of the state from i stages ago. ■

$$\begin{tikzcd}
y_1 & & y_2 & & y_3 & \\
x_1 \arrow[u] \arrow[r] & u_1 \arrow[r] & x_2 \arrow[u] \arrow[r] & u_2 \arrow[r] & x_3 \arrow[u] \arrow[r] & \cdots
\end{tikzcd}$$

Figure 11.2: In each stage, k, an observation, $y_k \in Y$, is received and an action $u_k \in U$ is applied. The state, x_k, however, is hidden from the decision maker.

11.1.2 Defining the History Information Space

This section defines the most basic and natural I-space. Many others will be derived from it, which is the topic of Section 11.2. Suppose that X, U, and f have been defined as in Formulation 10.1, and the notion of stages has been defined as in Formulation 2.2. This yields a state sequence x_1, x_2, ..., and an action sequence u_1, u_2, ..., during the execution of a plan. However, in the current setting, the state sequence is not known. Instead, at every stage, an observation, y_k, is obtained. The process depicted in Figure 11.2.

In previous formulations, the action space, $U(x)$, was generally allowed to depend on x. Since x is unknown in the current setting, it would seem strange to allow the actions to depend on x. This would mean that inferences could be made regarding the state by simply noticing which actions are available.[1] Instead, it will be assumed by default that U is fixed for all $x \in X$. In some special contexts, however, $U(x)$ may be allowed to vary.

Initial conditions As stated at the beginning of the chapter, the initial conditions provide one of the three general sources of information regarding the state. Therefore, three alternative types of initial conditions will be allowed:

1. **Known State:** The initial state, $x_1 \in X$, is given. This initializes the problem with perfect state information. Assuming nature actions interfere with the state transition function, f, uncertainty in the current state will generally develop.

2. **Nondeterministic:** A set of states, $X_1 \subset X$, is given. In this case, the initial state is only known to lie within a particular subset of X. This can be considered as a generalization of the first type, which only allowed singleton subsets.

3. **Probabilistic:** A probability distribution, $P(x_1)$, over X is given.

In general, let η_0 denote the initial condition, which may be any one of the three alternative types.

[1]Such a problem could be quite interesting to study, but it will not be considered here.

History Suppose that the kth stage has passed. What information is available? It is assumed that at every stage, a sensor observation is made. This results in a *sensing history*,

$$\tilde{y}_k = (y_1, y_2, \ldots, y_k). \tag{11.12}$$

At every stage an action can also be applied, which yields an *action history*,

$$\tilde{u}_{k-1} = (u_1, u_2, \ldots, u_{k-1}). \tag{11.13}$$

Note that the action history only runs to u_{k-1}; if u_k is applied, the state x_{k+1} and stage $k+1$ are obtained, which lie beyond the current stage, k. By combining the sensing and action histories, the *history* at stage k is $(\tilde{u}_{k-1}, \tilde{y}_k)$.

History information states The history, $(\tilde{u}_{k-1}, \tilde{y}_k)$, in combination with the initial condition, η_0, yields the *history I-state*, which is denoted by η_k. This corresponds to all information that is known up to stage k. In spite of the fact that the states, x_1, ..., x_k, might not be known, the history I-states are always known because they are defined directly in terms of available information. Thus, the history I-state is

$$\eta_k = (\eta_0, \tilde{u}_{k-1}, \tilde{y}_k). \tag{11.14}$$

When representing I-spaces, we will generally ignore the problem of nesting parentheses. For example, (11.14) is treated a single sequence, instead of a sequence that contains two sequences. This distinction is insignificant for the purposes of decision making.

The history I-state, η_k, can also be expressed as

$$\eta_k = (\eta_{k-1}, u_{k-1}, y_k), \tag{11.15}$$

by noticing that the history I-state at stage k contains all of the information from the history I-state at stage $k-1$. The only new information is the most recently applied action, u_{k-1}, and the current sensor observation, y_k.

The history information space The history I-space is simply the set of all possible history I-states. Although the history I-states appear to be quite complicated, it is helpful to think of them abstractly as points in a new space. To define the set of all possible history I-states, the sets of all initial conditions, actions, and observations must be precisely defined.

The set of all observation histories is denoted as $\tilde{Y}_k$ and is obtained by a Cartesian product of k copies of the observation space:

$$\tilde{Y}_k = \underbrace{Y \times Y \ldots \times Y}_{k}. \tag{11.16}$$

Similarly, the set of all action histories is $\tilde{U}_{k-1}$, the Cartesian product of $k-1$ copies of the action space U.

It is slightly more complicated to define the set of all possible initial conditions because three different types of initial conditions are possible. Let $\mathcal{I}_0$ denote the *initial condition space.* Depending on which of the three types of initial conditions are used, one of the following three definitions of $\mathcal{I}_0$ is used:

1. **Known State:** If the initial state, x_1, is given, then $\mathcal{I}_0 \subseteq X$. Typically, $\mathcal{I}_0 = X$; however, it might be known in some instances that certain initial states are impossible. Therefore, it is generally written that $\mathcal{I}_0 \subseteq X$.

2. **Nondeterministic:** If X_1 is given, then $\mathcal{I}_0 \subseteq \text{pow}(X)$ (the power set of X). Again, a typical situation is $\mathcal{I}_0 = \text{pow}(x)$; however, it might be known that certain subsets of X are impossible as initial conditions.

3. **Probabilistic:** Finally, if $P(x)$ is given, then $\mathcal{I}_0 \subseteq \mathcal{P}(X)$, in which $\mathcal{P}(x)$ is the set of all probability distributions over X.

The *history I-space at stage k* is expressed as

$$\mathcal{I}_k = \mathcal{I}_0 \times \tilde{U}_{k-1} \times \tilde{Y}_k. \tag{11.17}$$

Each $\eta_k \in \mathcal{I}_k$ yields an initial condition, an action history, and an observation history. It will be convenient to consider I-spaces that do not depend on k. This will be defined by taking a union (be careful not to mistakenly think of this construction as a Cartesian product). If there are K stages, then the *history I-space* is

$$\mathcal{I}_{hist} = \mathcal{I}_0 \cup \mathcal{I}_1 \cup \mathcal{I}_2 \cup \cdots \cup \mathcal{I}_K. \tag{11.18}$$

Most often, the number of stages is not fixed. In this case, $\mathcal{I}_{hist}$ is defined to be the union of $\mathcal{I}_k$ over all $k \in \{0\} \cup \mathbb{N}$:

$$\mathcal{I}_{hist} = \mathcal{I}_0 \cup \mathcal{I}_1 \cup \mathcal{I}_2 \cup \cdots . \tag{11.19}$$

This construction is related to the state space obtained for time-varying motion planning in Section 7.1. The history I-space is stage-dependent because information accumulates over time. In the discrete model, the reference to time is only implicit through the use of stages. Therefore, stage-dependent I-spaces are defined. Taking the union of all of these is similar to the state space that was formed in Section 7.1 by making time be one axis of the state space. For the history I-space, $\mathcal{I}_{hist}$, the stage index k can be imagined as an "axis."

One immediate concern regarding the history I-space $\mathcal{I}_{hist}$ is that its I-states may be arbitrarily long because the history grows linearly with the number of stages. For now, it is helpful to imagine $\mathcal{I}_{hist}$ abstractly as another kind of state space, without paying close attention to how complicated each $\eta \in \mathcal{I}_{hist}$ may be to represent. In many contexts, there are ways to simplify the I-space. This is the topic of Section 11.2.

11.1.3 Defining a Planning Problem

Planning problems will be defined directly on the history I-space, which makes it appear as an ordinary state space in many ways. Keep in mind, however, that it was derived from another state space for which perfect state observations could not be obtained. In Section 10.1, a feedback plan was defined as a function of the state. Here, a feedback plan is instead a function of the I-state. Decisions cannot be based on the state because it will be generally unknown during the execution of the plan. However, the I-state is always known; thus, it is logical to base decisions on it.

Let π_K denote a *K-step information-feedback plan*, which is a sequence (π_1, π_2, ..., π_K) of K functions, $\pi_k : \mathcal{I}_k \to U$. Thus, at every stage k, the I-state $\eta_k \in \mathcal{I}_k$ is used as a basis for choosing the action $u_k = \pi_k(\eta_k)$. Due to interference of nature through both the state transition equation and the sensor mapping, the action sequence $(u_1, \ldots, u_K)$ produced by a plan, π_K, will not be known until the plan terminates.

As in Formulation 2.3, it will be convenient to assume that U contains a *termination action*, u_T. If u_T is applied at stage k, then it is repeatedly applied forever. It is assumed once again that the state x_k remains fixed after the termination condition is applied. Remember, however, x_k is still unknown in general; it becomes fixed but unknown. Technically, based on the definition of the history I-space, the I-state must change after u_T is applied because the history grows. These changes can be ignored, however, because no new decisions are made after u_T is applied. A plan that uses a termination condition can be specified as $\pi = (\pi_1, \pi_2, \ldots)$ because the number of stages may vary each time the plan is executed. Using the history I-space definition in (11.19), an *information-feedback plan* is expressed as

$$\pi : \mathcal{I}_{hist} \to U. \tag{11.20}$$

We are almost ready to define the planning problem. This will require the specification of a cost functional. The cost depends on the histories of states $\tilde{x}$ and actions $\tilde{u}$ as in Section 10.1. The planning formulation involves the following components, summarizing most of the concepts introduced so far in Section 11.1 (see Formulation 10.1 for similarities):

Formulation 11.1 (Discrete Information Space Planning)

1. A nonempty *state space* X that is either finite or countably infinite.

2. A nonempty, finite *action space* U. It is assumed that U contains a special *termination action*, which has the same effect as defined in Formulation 2.3.

3. A finite *nature action space* $\Theta(x, u)$ for each $x \in X$ and $u \in U$.

4. A *state transition function* f that produces a state, $f(x, u, \theta)$, for every $x \in X$, $u \in U$, and $\theta \in \Theta(x, u)$.

5. A finite or countably infinite *observation space* Y.
6. A finite *nature sensing action space* $\Psi(x)$ for each $x \in X$.
7. A *sensor mapping* h which produces an observation, $y = h(x, \psi)$, for each $x \in X$ and $\psi \in \Psi(x)$. This definition assumes a state-nature sensor mappings. A state sensor mapping or history-based sensor mapping, as defined in Section 11.1.1, could alternatively be used.
8. A set of *stages*, each denoted by k, which begins at $k = 1$ and continues indefinitely.
9. An *initial condition* η_0, which is an element of an *initial condition space*, $\mathcal{I}_0$.
10. A *history I-space* $\mathcal{I}_{hist}$ which is the union of $\mathcal{I}_0$ and $\mathcal{I}_k = \mathcal{I}_0 \times \tilde{U}_{k-1} \times \tilde{Y}_k$ for every stage $k \in \mathbb{N}$.
11. Let L denote a stage-additive cost functional, which may be applied to any pair $(\tilde{x}_{K+1}, \tilde{u}_K)$ of state and action histories to yield

$$L(\tilde{x}_{K+1}, \tilde{u}_K) = \sum_{k=1}^{K} l(x_k, u_k) + l_F(x_{K+1}). \tag{11.21}$$

If the termination action u_T is applied at some stage k, then for all $i \geq k$, $u_i = u_T$, $x_i = x_k$, and $l(x_i, u_T) = 0$. Either a feasible or optimal planning problem can be defined, as in Formulation 10.1; however, the plan here is specified as $\pi : \mathcal{I} \rightarrow U$.

A *goal set* may be defined as $X_G \subset X$. Alternatively, the goal could be expressed as a desirable set of history I-states. After Section 11.2, it will be seen that the goal can be expressed in terms of I-states that are derived from histories.

Some immediate extensions of Formulation 11.1 are possible, but we avoid them here simplify notation in the coming concepts. One extension is to allow different action sets, $U(x)$, for each $x \in X$. Be careful, however, because information regarding the current state can be inferred if the action set $U(x)$ is given, and it varies depending on x. Another extension is to allow the costs to depend on nature, to obtain $l(x_k, u_k, \theta_k)$, instead of $l(x_k, u_k)$ in (11.21).

The cost of a plan The next task is to extend the definition of the cost-to-go under a fixed plan, which was given in Section 10.1.3, to the case of imperfect state information. Consider evaluating the quality of a plan, so that the "best" one might be selected. Suppose that the nondeterministic uncertainty is used to model nature and that a nondeterministic initial condition is given. If a plan π is fixed, some state and action trajectories are possible, and others are not. It is impossible to know in general what histories will occur; however, the plan constrains the choices substantially. Let $\mathcal{H}(\pi, \eta_0)$ denote the set of state-action histories that could arise from π applied to the initial condition η_0.

The cost of a plan π from an initial condition η_0 is measured using *worst-case analysis* as

$$G_\pi(\eta_0) = \max_{(\tilde{x},\tilde{u}) \in \mathcal{H}(\pi,\eta_0)} \Big\{ L(\tilde{x},\tilde{u}) \Big\}. \tag{11.22}$$

Note that $\tilde{x}$ includes x_1, which is usually not known. It may be known only to lie in X_1, as specified by η_0. Let Π denote the set of all possible plans. An optimal plan using worst-case analysis is any plan for which (11.22) is minimized over all $\pi \in \Pi$ and $\eta_0 \in \mathcal{I}_0$. In the case of feasible planning, there are usually numerous equivalent alternatives.

Under probabilistic uncertainty, the cost of a plan can be measured using *expected-case analysis* as

$$G_\pi(\eta_0) = E_{\mathcal{H}(\pi,\eta_0)} \Big[L(\tilde{x},\tilde{u}) \Big], \tag{11.23}$$

in which E denotes the mathematical expectation of the cost, with the probability distribution taken over $\mathcal{H}(\pi,\eta_0)$. The task is to find a plan $\pi \in \Pi$ that minimizes (11.23).

The information space is just another state space It will become important throughout this chapter and Chapter 12 to view the I-space as an ordinary state space. It only seems special because it is derived from another state space, but once this is forgotten, it exhibits many properties of an ordinary state space in planning. One nice feature is that the state in this special space is always known. Thus, by converting from an original state space to its I-space, we also convert from having imperfect state information to always knowing the state, albeit in a larger state space.

One important consequence of this interpretation is that the state transition equation can be lifted into the I-space to obtain an *information transition function*, $f_\mathcal{I}$. Suppose that there are no sensors, and therefore no observations. In this case, future I-states are predictable, which leads to

$$\eta_{k+1} = f_\mathcal{I}(\eta_k, u_k). \tag{11.24}$$

The function $f_\mathcal{I}$ generates η_{k+1} by concatenating u_k onto η_k.

Now suppose that there are observations, which are generally unpredictable. In Section 10.1, the nature action $\theta_k \in \Theta(x,u)$ was used to model the unpredictability. In terms of the information transition equation, y_{k+1} serves the same purpose. When the decision is made to apply u_k, the observation y_{k+1} is not yet known (just as θ_k is unknown in Section 10.1). In a sequential game against nature with perfect state information, x_{k+1} is directly observed at the next stage. For the information transition equation, y_{k+1} is instead observed, and η_{k+1} can be determined. Using the history I-state representation, (11.14), simply concatenate u_k and y_{k+1} onto the histories in η_k to obtain η_{k+1}. The information transition equation is expressed as

$$\eta_{k+1} = f_\mathcal{I}(\eta_k, u_k, y_{k+1}), \tag{11.25}$$

$$\begin{tikzcd}
\mathcal{I}_{hist} \arrow[rr, "\kappa_{ndet}"] \arrow[drr, "\kappa_{prob}"] \arrow[dd, "\kappa_{est}"] \arrow[ddr, "\kappa_1"] & & \mathcal{I}_{ndet} \arrow[r, "\kappa_2"] & \mathcal{I}_2 \\
 & & \mathcal{I}_{prob} \arrow[r, "\kappa_3"] & \mathcal{I}_3 \\
X & \mathcal{I}_1 & &
\end{tikzcd}$$

Figure 11.3: Many alternative information mappings may be proposed. Each leads to a derived information space.

with the understanding that y_{k+1} plays the same role as θ_k in the case of perfect state information and unpredictable future states. Even though nature causes future I-states to be unpredictable, the current I-state is always known. A plan, $\pi : \mathcal{I} \rightarrow U$, now seems like a state-feedback plan, if the I-space is viewed as a state space. The transitions are all specified by $f_{\mathcal{I}}$.

The costs in this new state space can be derived from the original cost functional, but a maximization or expectation is needed over all possible states given the current information. This will be covered in Section 12.1.

11.2 Derived Information Spaces

The history I-space appears to be quite complicated. Every I-state corresponds to a history of actions and observations. Unfortunately, the length of the I-state sequence grows linearly with the number of stages. To overcome this difficultly, it is common to map history I-states to some simpler space. In many applications, the ability to perform this simplification is critical to finding a practical solution. In some cases, the simplification fully preserves the history I-space, meaning that completeness, and optimality if applicable, is not lost. In other cases, we are willing to tolerate a simplification that destroys much of the structure of the history I-space. This may be necessary to obtain a dramatic reduction in the size of the I-space.

11.2.1 Information Mappings

Consider a function that maps the history I-space into a space that is simpler to manage. Formally, let $\kappa : \mathcal{I}_{hist} \rightarrow \mathcal{I}_{der}$ denote a function from a history I-space, $\mathcal{I}_{hist}$, to a *derived I-space*, $\mathcal{I}_{der}$. The function, κ, is called an *information mapping*, or *I-map*. The derived I-space may be any set; hence, there is great flexibility in defining an I-map.[2] Figure 11.3 illustrates the idea. The starting place is $\mathcal{I}_{hist}$, and mappings are made to various derived I-spaces. Some generic mappings, κ_1,

[2]Ideally, the mapping should be *onto* $\mathcal{I}_{der}$; however, to facilitate some definitions, this will not be required.

κ_2, and κ_3, are shown, along with some very important kinds, $\mathcal{I}_{est}$, $\mathcal{I}_{ndet}$ and $\mathcal{I}_{prob}$. The last two are the subjects of Sections 11.2.2 and 11.2.3, respectively. The other important I-map is κ_{est}, which uses the history to estimate the state; hence, the derived I-space is X (see Example 11.11). In general, an I-map can even map any derived I-space to another, yielding $\kappa : \mathcal{I}_{der} \rightarrow \mathcal{I}'_{der}$, for any I-spaces $\mathcal{I}_{der}$ and $\mathcal{I}'_{der}$. Note that any composition of I-maps yields an I-map. The derived I-spaces $\mathcal{I}_2$ and $\mathcal{I}_3$ from Figure 11.3 are obtained via compositions.

Making smaller information-feedback plans The primary use of an I-map is to simplify the description of a plan. In Section 11.1.3, a plan was defined as a function on the history I-space, $\mathcal{I}_{hist}$. Suppose that an I-map, κ, is introduced that maps from $\mathcal{I}_{hist}$ to $\mathcal{I}_{der}$. A feedback plan on $\mathcal{I}_{der}$ is defined as $\pi : \mathcal{I}_{der} \rightarrow U$. To execute a plan defined on $\mathcal{I}_{der}$, the derived I-state is computed at each stage k by applying κ to η_k to obtain $\kappa(\eta_k) \in \mathcal{I}_{der}$. The action selected by π is $\pi(\kappa(\eta_k)) \in U$.

To understand the effect of using $\mathcal{I}_{der}$ instead of $\mathcal{I}_{hist}$ as the domain of π, consider the set of possible plans that can be represented over $\mathcal{I}_{der}$. Let Π_{hist} and Π_{der} be the sets of all plans over $\mathcal{I}_{hist}$ and $\mathcal{I}_{der}$, respectively. Any $\pi \in \Pi_{der}$ can be converted into an equivalent plan, $\pi' \in \Pi_{hist}$, as follows: For each $\eta \in \mathcal{I}_{hist}$, define $\pi'(\eta) = \pi(\kappa(\eta))$.

It is not always possible, however, to construct a plan, $\pi \in \Pi_{der}$, from some $\pi' \in \mathcal{I}_{hist}$. The problem is that there may exist some $\eta_1, \eta_2 \in \mathcal{I}_{hist}$ for which $\pi'(\eta_1) \neq \pi'(\eta_2)$ and $\kappa(\eta_1) = \kappa(\eta_2)$. In words, this means that the plan in Π_{hist} requires that two histories cause different actions, but in the derived I-space the histories cannot be distinguished. For a plan in Π_{der}, both histories must yield the same action.

An I-map κ has the potential to collapse $\mathcal{I}_{hist}$ down to a smaller I-space by inducing a partition of $\mathcal{I}_{hist}$. For each $\eta_{der} \in \mathcal{I}_{der}$, let the preimage $\kappa^{-1}(\eta_{der})$ be defined as

$$\kappa^{-1}(\eta_{der}) = \{\eta \in \mathcal{I}_{hist} \mid \eta_{der} = \kappa(\eta)\}. \tag{11.26}$$

This yields the set of history I-states that map to η_{der}. The induced partition can intuitively be considered as the "resolution" at which the history I-space is characterized. If the sets in (11.26) are large, then the I-space is substantially reduced. The goal is to select κ to make the sets in the partition as large as possible; however, one must be careful to avoid collapsing the I-space so much that the problem can no longer be solved.

Example 11.11 (State Estimation) In this example, the I-map is the classical approach that is conveniently taken in numerous applications. Suppose that a technique has been developed that uses the history I-state $\eta \in \mathcal{I}_{hist}$ to compute an estimate of the current state. In this case, the I-map is $\kappa_{est} : \mathcal{I}_{hist} \rightarrow X$. The derived I-space happens to be X in this case! This means that a plan is specified as $\pi : X \rightarrow U$, which is just a state-feedback plan.

Consider the partition of $\mathcal{I}_{hist}$ that is induced by κ_{est}. For each $x \in X$, the set $\kappa_{est}^{-1}(x)$, as defined in (11.26), is the set of all histories that lead to the same state estimate. A plan on X can no longer distinguish between various histories that led to the same state estimate. One implication is that the ability to encode the amount of uncertainty in the state estimate has been lost. For example, it might be wise to make the action depend on the covariance in the estimate of x; however, this is not possible because decisions are based only on the estimate itself. ■

Example 11.12 (Stage Indices) Consider an I-map, κ_{stage}, that returns only the current stage index. Thus, $\kappa_{stage}(\eta_k) = k$. The derived I-space is the set of stages, which is $\mathbb{N}$. A feedback plan on the derived I-space is specified as $\pi : \mathbb{N} \rightarrow U$. This is equivalent to specifying a plan as an action sequence, $(u_1, u_2, \ldots,)$, as in Section 2.3.2. Since the feedback is trivial, this is precisely the original case of planning without feedback, which is also refereed to as an open-loop plan. ■

Constructing a derived information transition equation As presented so far, the full history I-state is needed to determine a derived I-state. It may be preferable, however, to discard histories and work entirely in the derived I-space. Without storing the histories on the machine or robot, a derived information transition equation needs to be developed. The important requirement in this case is as follows:

> *If η_k is replaced by $\kappa(\eta_k)$, then $\kappa(\eta_{k+1})$ must be correctly determined using only $\kappa(\eta_k)$, u_k, and y_{k+1}.*

Whether this requirement can be met depends on the particular I-map. Another way to express the requirement is that if $\kappa(\eta_k)$ is given, then the full history η does not contain any information that could further constrain $\kappa(\eta_{k+1})$. The information provided by κ is *sufficient* for determining the next derived I-states. This is similar to the concept of a *sufficient statistic*, which arises in decision theory [89]. If the requirement is met, then κ is called a *sufficient I-map*. One peculiarity is that the sufficiency is relative to $\mathcal{I}_{der}$, as opposed to being absolute in some sense. For example, any I-map that maps onto $\mathcal{I}_{der} = \{0\}$ is sufficient because $\kappa(\eta_{k+1})$ is always known (it remains fixed at 0). Thus, the requirement for sufficiency depends strongly on the particular derived I-space.

For a sufficient I-map, a *derived information transition equation* is determined as

$$\kappa(\eta_{k+1}) = f_{\mathcal{I}_{der}}(\kappa(\eta_k), u_k, y_{k+1}). \tag{11.27}$$

The implication is that $\mathcal{I}_{der}$ is the new I-space in which the problem "lives." There is no reason for the decision maker to consider histories. This idea is crucial to the success of many planning algorithms. Sections 11.2.2 and 11.2.3 introduce nondeterministic I-spaces and probabilistic I-spaces, which are two of the most important derived I-spaces and are obtained from sufficient I-maps. The I-map

$$\begin{tikzcd}
\mathcal{I}_{hist} \arrow[r, "f_{\mathcal{I}}"] \arrow[d, "\kappa"'] & \mathcal{I}_{hist} \arrow[d, "\kappa"'] \\
\mathcal{I}_{der} \arrow[r, dashed, "f_{\mathcal{I}_{der}}?"] & \mathcal{I}_{der} \\
\text{Stage } k & \text{Stage } k+1
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
\mathcal{I}_{hist} \arrow[rr, "f_{\mathcal{I}}"] & & \mathcal{I}_{hist} \arrow[dd, "\kappa"'] \\
 & \text{singleton?} \arrow[dr] & \\
\mathcal{I}_{der} \arrow[uu, "\kappa^{-1}"] & & \mathcal{I}_{der} \\
\text{Stage } k & & \text{Stage } k+1
\end{tikzcd}$$

(b)

Figure 11.4: (a) For an I-map to be sufficient, the same result must be reached in the lower right, regardless of the path taken from the upper left. (b) The problem is that κ images may contain many histories, which eventually map to multiple derived I-states.

κ_{stage} from Example 11.12 is also sufficient. The estimation I-map from Example 11.11 is usually not sufficient because some history is needed to provide a better estimate.

The diagram in Figure 11.4a indicates the problem of obtaining a sufficient I-map. The top of the diagram shows the history I-state transitions before the I-map was introduced. The bottom of the diagram shows the attempted derived information transition equation, $f_{\mathcal{I}_{der}}$. The requirement is that the derived I-state obtained in the lower right must be the same regardless of which path is followed from the upper left. Either $f_{\mathcal{I}}$ can be applied to η, followed by κ, or κ can be applied to η, followed by some $f_{\mathcal{I}_{der}}$. The problem with the existence of $f_{\mathcal{I}_{der}}$ is that κ is usually not invertible. The preimage $\kappa^{-1}(\eta_{der})$ of some derived I-state $\eta_{der} \in \mathcal{I}_{der}$ yields a set of histories in $\mathcal{I}_{hist}$. Applying $f_{\mathcal{I}}$ to all of these yields a set of possible next-stage history I-states. Applying κ to these may yield a set of derived I-states because of the ambiguity introduced by κ^{-1}. This chain of mappings is shown in Figure 11.4b. If a singleton is obtained under all circumstances, then this yields the required values of $f_{\mathcal{I}_{der}}$. Otherwise, new uncertainty arises about the current derived I-state. This could be handled by defining an information space over the information space, but this nastiness will be avoided here.

Since I-maps can be defined from any derived I-space to another, the concepts presented in this section do not necessarily require $\mathcal{I}_{hist}$ as the starting point. For example, an I-map, $\kappa : \mathcal{I}_{der} \rightarrow \mathcal{I}'_{der}$, may be called *sufficient with respect to* $\mathcal{I}_{der}$ rather than with respect to $\mathcal{I}_{hist}$.

11.2.2 Nondeterministic Information Spaces

This section defines the I-map κ_{ndet} from Figure 11.3, which converts each history I-state into a subset of X that corresponds to all possible current states. Nature is modeled nondeterministically, which means that there is no information about what actions nature will choose, other than that they will be chosen from Θ and Ψ. Assume that the state-action sensor mapping from Section 11.1.1 is used. Consider what inferences may be drawn from a history I-state, $\eta_k = (\eta_0, \tilde{u}_{k-1}, \tilde{y}_k)$. Since the model does not involve probabilities, let η_0 represent a set $X_1 \subseteq X$. Let $\kappa_{ndet}(\eta_k)$ be the minimal subset of X in which x_k is known to lie given η_k. This subset is referred to as a *nondeterministic I-state.* To remind you that $\kappa_{ndet}(\eta_k)$ is a subset of X, it will now be denoted as $X_k(\eta_k)$. It is important that $X_k(\eta_k)$ be as small as possible while consistent with η_k.

Recall from (11.6) that for every observation y_k, a set $H(y_k) \subseteq X$ of possible values for x_k can be inferred. This could serve as a crude estimate of the nondeterministic I-state. It is certainly known that $X_k(\eta_k) \subseteq H(y_k)$; otherwise, x_k, would not be consistent with the current sensor observation. If we carefully progress from the initial conditions while applying constraints due to the state transition equation, the appropriate subset of $H(y_k)$ will be obtained.

From the state transition function f, define a set-valued function F that yields a subset of X for every $x \in X$ and $u \in U$ as

$$F(x,u) = \{x' \in X \mid \; \exists \theta \in \Theta(x,u) \text{ for which } x' = f(x,u,\theta)\}. \tag{11.28}$$

Note that both F and H are set-valued functions that eliminate the direct appearance of nature actions. The effect of nature is taken into account in the set that is obtained when these functions are applied. This will be very convenient for computing the nondeterministic I-state.

An inductive process will now be described that results in computing the nondeterministic I-state, $X_k(\eta_k)$, for any stage k. The base case, $k = 1$, of the induction proceeds as

$$X_1(\eta_1) = X_1(\eta_0, y_1) = X_1 \cap H(y_1). \tag{11.29}$$

The first part of the equation replaces η_1 with (η_0, y_1), which is a longer way to write the history I-state. There are not yet any actions in the history. The second part applies set intersection to make consistent the two pieces of information: 1) The initial state lies in X_1, which is the initial condition, and 2) the states in $H(y_1)$ are possible given the observation y_1.

Now assume inductively that $X_k(\eta_k) \subseteq X$ has been computed and the task is to compute $X_{k+1}(\eta_{k+1})$. From (11.15), $\eta_{k+1} = (\eta_k, u_k, y_{k+1})$. Thus, the only new pieces of information are that u_k was applied and y_{k+1} was observed. These will be considered one at a time.

Consider computing $X_{k+1}(\eta_k, u_k)$. If x_k was known, then after applying u_k, the state could lie anywhere within $F(x_k, u_k)$, using (11.28). Although x_k is actually not known, it is at least known that $x_k \in X_k(\eta_k)$. Therefore,

$$X_{k+1}(\eta_k, u_k) = \bigcup_{x_k \in X_k(\eta_k)} F(x_k, u_k). \tag{11.30}$$

This can be considered as the set of all states that can be reached by starting from some state in $X_k(\eta_k)$ and applying any actions $u_k \in U$ and $\theta_k \in \Theta(x_k, u_k)$. See Figure 11.5.

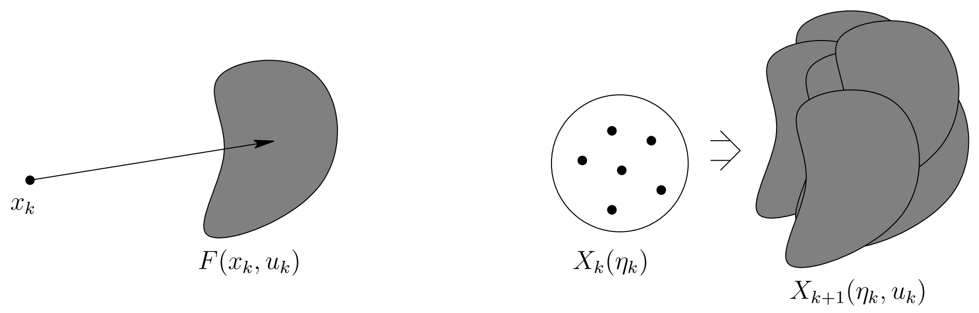


Figure 11.5: The first step in computing the nondeterministic I-state is to take the union of $F(x_k, u_k)$ over all possible $x_k \in X_k(\eta_k)$.

The next step is to take into account the observation y_{k+1}. This information alone indicates that x_{k+1} lies in $H(y_{k+1})$. Therefore, an intersection is performed to obtain the nondeterministic I-state,

$$X_{k+1}(\eta_{k+1}) = X_{k+1}(\eta_k, u_k, y_{k+1}) = X_{k+1}(\eta_k, u_k) \cap H(y_{k+1}). \tag{11.31}$$

Thus, it has been shown how to compute $X_{k+1}(\eta_{k+1})$ from $X_k(\eta_k)$. After starting with (11.29), the nondeterministic I-states at any stage can be computed by iterating (11.30) and (11.31) as many times as necessary.

Since the nondeterministic I-state is always a subset of X, the *nondeterministic I-space*, $\mathcal{I}_{ndet} = \mathrm{pow}(X)$, is obtained (shown in Figure 11.3). If X is finite, then $\mathcal{I}_{ndet}$ is also finite, which was not the case with $\mathcal{I}_{hist}$ because the histories continued to grow with the number of stages. Thus, if the number of stages is unbounded or large in comparison to the size of X, then nondeterministic I-states seem preferable. It is also convenient that κ_{ndet} is a sufficient I-map, as defined in Section 11.2.1. This implies that a planning problem can be completely expressed in terms of $\mathcal{I}_{ndet}$ without maintaining the histories. The goal region, X_G, can be expressed directly as a nondeterministic I-state. In this way, the planning task is to terminate in a nondeterministic I-state, $X_k(\eta_k)$, for which $X_k(\eta_k) \subseteq X_G$.

The sufficiency of κ_{ndet} is obtained because (11.30) and (11.31) show that $X_{k+1}(\eta_{k+1})$ can be computed from $X_k(\eta_k)$, u_k, and y_{k+1}. This implies that a derived information transition equation can be formed. The nondeterministic I-space can also be treated as "just another state space." Although many history I-states may map to the same nondeterministic I-state, it has been assumed for decision-making purposes that particular history is irrelevant, once $X_k(\eta_k)$ is given.

The following example is not very interesting in itself, but it is simple enough to illustrate the concepts.

Example 11.13 (Three-State Example) Let $X = \{0, 1, 2\}$, $U = \{-1, 0, 1\}$, and $\Theta(x, u) = \{0, 1\}$ for all $x \in X$ and $u \in U$. The state transitions are given by $f(x,u,\theta) = (x + u + \theta) \bmod 3$. Regarding sensing, $Y = \{0,1,2,3,4\}$ and $\Psi(x) = \{0,1,2\}$ for all $x \in X$. The sensor mapping is $y = h(x,\psi) = x + \psi$.

The history I-space appears very cumbersome for this example, which only involves three states. The nondeterministic I-space for this example is

$$\mathcal{I}_{ndet} = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0,1\}, \{1,2\}, \{0,2\}, \{0,1,2\}\}, \tag{11.32}$$

which is the power set of $X = \{0,1,2\}$. Note, however, that the empty set, $\emptyset$, can usually be deleted from $\mathcal{I}_{ndet}$.[3] Suppose that the initial condition is $X_1 = \{0,2\}$ and that the initial state is $x_1 = 0$. The initial state is unknown to the decision maker, but it is needed to ensure that valid observations are made in the example.

Now consider the execution over a number of stages. Suppose that the first observation is $y_1 = 2$. Based on the sensor mapping, $H(y_1) = H(2) = \{0,1,2\}$, which is not very helpful because $H(2) = X$. Applying (11.29) yields $X_1(\eta_1) = \{0,2\}$. Now suppose that the decision maker applies the action $u_1 = 1$ and nature applies $\theta_1 = 1$. Using f, this yields $x_2 = 2$. The decision maker does not know θ_1 and must therefore take into account any nature action that could have been applied. It uses (11.30) to infer that

$$X_2(\eta_1, u_1) = F(2,1) \cup F(0,1) = \{0,1\} \cup \{1,2\} = \{0,1,2\}. \tag{11.33}$$

Now suppose that $y_2 = 3$. From the sensor mapping, $H(3) = \{1,2\}$. Applying (11.31) yields

$$X_2(\eta_2) = X_2(\eta_1, u_1) \cap H(y_2) = \{0,1,2\} \cap \{1,2\} = \{1,2\}. \tag{11.34}$$

This process may be repeated for as many stages as desired. A path is generated through $\mathcal{I}_{ndet}$ by visiting a sequence of nondeterministic I-states. If the observation $y_k = 4$ is ever received, the state, x_k, becomes immediately known because $H(4) = \{2\}$. ■

11.2.3 Probabilistic Information Spaces

This section defines the I-map κ_{prob} from Figure 11.3, which converts each history I-state into a probability distribution over X. A Markov, probabilistic model is assumed in the sense that the actions of nature only depend on the current state and action, as opposed to state or action histories. The set union and intersection of (11.30) and (11.31) are replaced in this section by marginalization and Bayes' rule, respectively. In a sense, these are the probabilistic equivalents of union and intersection. It will be very helpful to compare the expressions from this section to those of Section 11.2.2.

[3]One notable exception is in the theory of nondeterministic finite automata, in which it is possible that all copies of the machine die and there is no possible current state [891].

Rather than write $\kappa_{prob}(\eta)$, standard probability notation will be applied to obtain $P(x|\eta)$. Most expressions in this section of the form $P(x_k|\cdot)$ have an analogous expression in Section 11.2.2 of the form $X_k(\cdot)$. It is helpful to recognize the similarities.

The first step is to construct probabilistic versions of H and F. These are $P(x_k|y_k)$ and $P(x_{k+1}|x_k, u_k)$, respectively. The latter term was given in Section 10.1.1. To obtain $P(x_k|y_k)$, recall from Section 11.1.1 that $P(y_k|x_k)$ is easily derived from $P(\psi_k|x_k)$. To obtain $P(x_k|y_k)$, Bayes' rule is applied:

$$P(x_k|y_k) = \frac{P(y_k|x_k)P(x_k)}{P(y_k)} = \frac{P(y_k|x_k)P(x_k)}{\displaystyle\sum_{x_k \in X} P(y_k|x_k)P(x_k)}. \tag{11.35}$$

In the last step, $P(y_k)$ was rewritten using marginalization, (9.8). In this case x_k appears as the sum index; therefore, the denominator is only a function of y_k, as required. Bayes' rule requires knowing the prior, $P(x_k)$. In the coming expressions, this will be replaced by a probabilistic I-state.

Now consider defining probabilistic I-states. Each is a probability distribution over X and is written as $P(x_k|\eta_k)$. The initial condition produces $P(x_1)$. As for the nondeterministic case, probabilistic I-states can be computed inductively. For the base case, the only new piece of information is y_1. Thus, the probabilistic I-state, $P(x_1|\eta_1)$, is $P(x_1|y_1)$. This is computed by letting $k = 1$ in (11.35) to yield

$$P(x_1|\eta_1) = P(x_1|y_1) = \frac{P(y_1|x_1)P(x_1)}{\displaystyle\sum_{x_1 \in X} P(y_1|x_1)P(x_1)}. \tag{11.36}$$

Now consider the inductive step by assuming that $P(x_k|\eta_k)$ is given. The task is to determine $P(x_{k+1}|\eta_{k+1})$, which is equivalent to $P(x_{k+1}|\eta_k, u_k, y_{k+1})$. As in Section 11.2.2, this will proceed in two parts by first considering the effect of u_k, followed by y_{k+1}. The first step is to determine $P(x_{k+1}|\eta_k, u_k)$ from $P(x_k|\eta_k)$. First, note that

$$P(x_{k+1}|\eta_k, x_k, u_k) = P(x_{k+1}|x_k, u_k) \tag{11.37}$$

because η_k contains no additional information regarding the prediction of x_{k+1} once x_k is given. Marginalization, (9.8), can be used to eliminate x_k from $P(x_{k+1}|x_k, u_k)$. This must be eliminated because it is not given. Putting these steps together yields

$$\begin{aligned} P(x_{k+1}|\eta_k, u_k) &= \sum_{x_k \in X} P(x_{k+1}|x_k, u_k, \eta_k)P(x_k|\eta_k) \\ &= \sum_{x_k \in X} P(x_{k+1}|x_k, u_k)P(x_k|\eta_k), \end{aligned} \tag{11.38}$$

which expresses $P(x_{k+1}|\eta_k, u_k)$ in terms of given quantities. Equation (11.38) can be considered as the probabilistic counterpart of (11.30).

The next step is to take into account the observation y_{k+1}. This is accomplished by making a version of (11.35) that is conditioned on the information accumulated so far: η_k and u_k. Also, k is replaced with $k+1$. The result is

$$P(x_{k+1}|y_{k+1}, \eta_k, u_k) = \frac{P(y_{k+1}|x_{k+1}, \eta_k, u_k) P(x_{k+1}|\eta_k, u_k)}{\displaystyle\sum_{x_{k+1} \in X} P(y_{k+1}|x_{k+1}, \eta_k, u_k) P(x_{k+1}|\eta_k, u_k)}. \tag{11.39}$$

This can be considered as the probabilistic counterpart of (11.31). The left side of (11.39) is equivalent to $P(x_{k+1}|\eta_{k+1})$, which is the probabilistic I-state for stage $k+1$, as desired. There are two different kinds of terms on the right. The expression for $P(x_{k+1}|\eta_k, u_k)$ is given in (11.38). Therefore, the only remaining term to calculate is $P(y_{k+1}|x_{k+1}, \eta_k, u_k)$. Note that

$$P(y_{k+1}|x_{k+1}, \eta_k, u_k) = P(y_{k+1}|x_{k+1}) \tag{11.40}$$

because the sensor mapping depends only on the state (and the probability model for the nature sensing action, which also depends only on the state). Since $P(y_{k+1}|x_{k+1})$ is specified as part of the sensor model, we have now determined how to obtain $P(x_{k+1}|\eta_{k+1})$ from $P(x_k|\eta_k)$, u_k, and y_{k+1}. Thus, $\mathcal{I}_{prob}$ is another I-space that can be treated as just another state space.

The probabilistic I-space $\mathcal{I}_{prob}$ (shown in Figure 11.3) is the set of all probability distributions over X. The update expressions, (11.38) and (11.39), establish that the I-map κ_{prob} is sufficient, which means that the planning problem can be expressed entirely in terms of $\mathcal{I}_{prob}$, instead of maintaining histories. A goal region can be specified as constraints on the probabilities. For example, from some particular $x \in X$, the goal might be to reach any probabilistic I-state for which $P(x_k|\eta_k) > 1/2$.

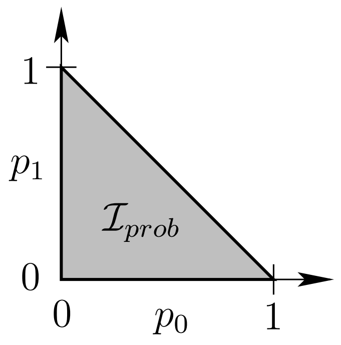


Figure 11.6: The probabilistic I-space for the three-state example is a 2-simplex embedded in $\mathbb{R}^3$. This simplex can be projected into $\mathbb{R}^2$ to yield the depicted triangular region in $\mathbb{R}^2$.

Example 11.14 (Three-State Example Revisited) Now return to Example 11.13, but this time use probabilistic models. For a probabilistic I-state, let p_i

denote the probability that the current state is $i \in X$. Any probabilistic I-state can be expressed as $(p_0, p_1, p_2) \in \mathbb{R}^3$. This implies that the I-space can be nicely embedded in $\mathbb{R}^3$. By the axioms of probability (given in Section 9.1.2), $p_0+p_1+p_2 = 1$, which can be interpreted as a plane equation in $\mathbb{R}^3$ that restricts $\mathcal{I}_{prob}$ to a 2D set. Also following the axioms of probability, for each $i \in \{0, 1, 2\}$, $0 \leq p_i \leq 1$. This means that $\mathcal{I}_{prob}$ is restricted to a triangular region in $\mathbb{R}^3$. The vertices of this triangular region are $(0, 0, 1)$, $(0, 1, 0)$, and $(1, 0, 0)$; these correspond to the three different ways to have perfect state information. In a sense, the distance away from these points corresponds to the amount of uncertainty in the state. The uniform probability distribution $(1/3, 1/3, 1/3)$ is equidistant from the three vertices. A projection of the triangular region into $\mathbb{R}^2$ is shown in Figure 11.6. The interpretation in this case is that p_0 and p_1 specify a point in $\mathbb{R}^2$, and p_2 is automatically determined from $p_2 = 1 - p_0 - p_1$.

The triangular region in $\mathbb{R}^3$ is an uncountably infinite set, even though the history I-space is countably infinite for a fixed initial condition. This may seem strange, but there is no mistake because for a fixed initial condition, it is generally impossible to reach all of the points in $\mathcal{I}_{prob}$. If the initial condition can be any point in $\mathcal{I}_{prob}$, then all of the probabilistic I-space is covered because $\mathcal{I}_0 = \mathcal{I}_{prob}$, in which $\mathcal{I}_0$ is the initial condition space.. ■

11.2.4 Limited-Memory Information Spaces

Limiting the amount of memory provides one way to reduce the sizes of history I-states. Except in special cases, this usually does not preserve the feasibility or optimality of the original problem. Nevertheless, such I-maps are very useful in practice when there appears to be no other way to reduce the size of the I-space. Furthermore, they occasionally do preserve the desired properties of feasibility, and sometimes even optimality.

Previous i stages Under this model, the history I-state is truncated. Any actions or observations received earlier than i stages ago are dropped from memory. An I-map, κ_i, is defined as

$$\kappa_i(\eta_k) = (u_{k-i}, \ldots, u_{k-1}, y_{k-i+1}, \ldots, y_k), \tag{11.41}$$

for any integer $i > 0$ and $k > i$. If $i \leq k$, then the derived I-state is the full history I-state, (11.14). The advantage of this approach, if it leads to a solution, is that the length of the I-state no longer grows with the number of stages. If X and U are finite, then the derived I-space is also finite. Note that κ_i is sufficient in the sense defined in Section 11.2.1 because enough history is passed from stage to stage to determine the derived I-states.

Sensor feedback An interesting I-map is obtained by removing all but the last sensor observation from the history I-state. This yields an I-map, $\kappa_{sf} : \mathcal{I}_{hist} \rightarrow Y$, which is defined as $\kappa_{sf}(\eta_k) = y_k$. The model is referred to as *sensor feedback*. In this case, all decisions are made directly in terms of the current sensor observation. The derived I-space is Y, and a plan on the derived I-space is $\pi : Y \to U$, which is called a *sensor-feedback plan*. In some literature, this may be referred to as a purely *reactive plan*. Many problems for which solutions exist in the history I-space cannot be solved using sensor feedback. Neglecting history prevents the complicated deductions that are often needed regarding the state. In some sense, sensor feedback causes short-sightedness that could unavoidably lead to repeating the same mistakes indefinitely. However, it may be worth determining whether such a sensor-feedback solution plan exists for some particular problem. Such plans tend to be simpler to implement in practice because the actions can be connected directly to the sensor output. Certainly, if a sensor-feedback solution plan exists for a problem, and feasibility is the only concern, then it is pointless to design and implement a plan in terms of the history I-space or some larger derived I-space. Note that this I-map is sufficient, even though it ignores the entire history.

11.3 Examples for Discrete State Spaces

11.3.1 Basic Nondeterministic Examples

First, we consider a simple example that uses the sign sensor of Example 11.3.

Example 11.15 (Using the Sign Sensor) This example is similar to Example 10.1, except that it involves sensing uncertainty instead of prediction uncertainty. Let $X = \mathbb{Z}$, $U = \{-1, 1, u_T\}$, $Y = \{-1, 0, 1\}$, and $y = h(x) = \mathrm{sgn}\, x$. For the state transition equation, $x_{k+1} = f(x_k, u_k) = x_k + u_k$. No nature actions interfere with the state transition equation or the sensor mapping. Therefore, future history I-states are predictable. The information transition equation is $\eta_{k+1} = f_{\mathcal{I}}(\eta_k, u_k)$. Suppose that initially, $\eta_0 = X$, which means that any initial state is possible. The goal is to terminate at $0 \in X$.

The general expression for a history I-state at stage k is

$$\eta_k = (X, u_1, \ldots, u_{k-1}, y_1, \ldots, y_k). \tag{11.42}$$

A possible I-state is $\eta_5 = (X, -1, 1, 1, -1, 1, 1, 1, 1, 0)$. Using the nondeterministic I-space from Section 11.2.2, $\mathcal{I}_{ndet} = \mathrm{pow}(X)$, which is uncountably infinite. By looking carefully at the problem, however, it can be seen that most of the nondeterministic I-states are not reachable. If $y_k = 0$, it is known that $x_k = 0$; hence, $X_k(\eta_k) = \{0\}$. If $y_k = 1$, it will always be the case that $X_k(\eta_k) = \{1, 2, \ldots\}$ unless 0 is observed. If $y_k = -1$, then $X_k(\eta_k) = \{\ldots, -2, -1\}$. From this a plan, π, can be specified over the three nondeterministic I-states mentioned above. For the first one, $\pi(X_k(\eta_k)) = u_T$. For the other two, $\pi(X_k(\eta_k)) = -1$ and $\pi(X_k(\eta_k)) = 1$, respectively. Based on the sign, the plan tries to move toward 0. If different initial conditions are allowed, then more nondeterministic I-states can be reached, but this was not required as the problem was defined. Note that optimal-length solutions are produced by the plan.

The solution can even be implemented with sensor feedback because the action depends only on the current sensor value. Let $\pi : Y \to U$ be defined as

$$\pi(y) = \begin{cases} -1 & \text{if } y = 1 \\ 1 & \text{if } y = -1 \\ u_T & \text{if } y = 0. \end{cases} \tag{11.43}$$

This provides dramatic memory savings over defining a plan on $\mathcal{I}_{hist}$. ■

The next example provides a simple illustration of solving a problem without ever knowing the current state. This leads to the *goal recognizability* problem [659] (see Section 12.5.1).

Example 11.16 (Goal Recognizability) Let $X = \mathbb{Z}$, $U = \{-1, 1, u_T\}$, and $Y = \mathbb{Z}$. For the state transition equation, $x_{k+1} = f(x_k, u_k) = x_k + u_k$. Now suppose that a variant of Example 11.7 is used to model sensing: $y = h(x, \psi) = x + \psi$ and $\Psi = \{-5, -4, \ldots, 5\}$. Suppose that once again, $\eta_0 = X$. In this case, it is impossible to guarantee that a goal, $X_G = \{0\}$, is reached because of the goal recognizability problem. The disturbance in the sensor mapping does not allow precise enough state measurements to deduce the precise achievement of the state. If the goal region, X_G, is enlarged to $\{-5, -4, \ldots, 5\}$, then the problem can be solved. Due to the disturbance, the nondeterministic I-state is always a subset of a consecutive sequence of 11 states. It is simple to derive a plan that moves this interval until the nondeterministic I-state becomes a subset of X_G. When this occurs, then the plan applies u_T. In solving this problem, the exact state never had to be known. ■

The problem shown in Figure 11.7 serves two purposes. First, it is an example of *sensorless planning* [321, 394], which means that there are no observations (see Sections 11.5.4 and 12.5.2). This is an interesting class of problems because it appears that no information can be gained regarding the state. Contrary to intuition, it turns out for this example and many others that plans can be designed that estimate the state. The second purpose is to illustrate how the I-space can be dramatically collapsed using the I-map concepts of Section 11.2.1. The standard nondeterministic I-space for this example contains 2^{19} I-states, but it can be mapped to a much smaller derived I-space that contains only a few elements.

Example 11.17 (Moving in an L-shaped Corridor) The state space X for the example shown in Figure 11.7 has 19 states, each of which corresponds to a location on one of the white tiles. For convenience, let each state be denoted by (i, j). There are 10 *bottom states*, denoted by $(1, 1)$, $(2, 1)$, …, $(10, 1)$, and 10 *left states*, denoted by $(1, 1)$, $(1, 2)$, …, $(1, 10)$. Since $(1, 1)$ is both a bottom state and a left state, it is called the *corner state*.

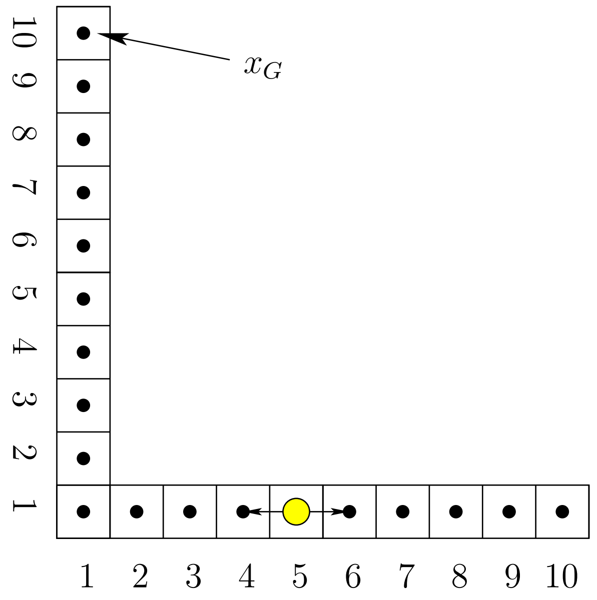


Figure 11.7: An example that involves 19 states. There are no sensor observations; however, actions can be chosen that enable the state to be estimated. The example provides an illustration of reducing the I-space via I-maps.

There are no sensor observations for this problem. However, nature interferes with the state transitions, which leads to a form of nondeterministic uncertainty. If an action is applied that tries to take one step, nature may cause two or three steps to be taken. This can be modeled as follows. Let

$$U = \{(1,0), (-1,0), (0,1), (0,-1)\} \tag{11.44}$$

and let $\Theta = \{1, 2, 3\}$. The state transition equation is defined as $f(x, u, \theta) = x + \theta u$ whenever such motion is not blocked (by hitting a dead end). For example, if $x = (5, 1)$, $u = (-1, 0)$, and $\theta = 2$, then the resulting next state is $(5, 1) + 2(-1, 0) = (3, 1)$. If blocking is possible, the state changes as much as possible until it becomes blocked. Due to blocking, it is even possible that $f(x, u, \theta) = x$.

Since there are no sensor observations, the history I-state at stage k is

$$\eta_k = (\eta_0, u_1, \ldots, u_{k-1}). \tag{11.45}$$

Now use the nondeterministic I-space, $\mathcal{I}_{ndet} = \text{pow}(X)$. The initial state, $x_1 = (10, 1)$, is given, which means that the initial I-state, η_0, is $\{(10, 1)\}$. The goal is to arrive at the I-state, $\{(1, 10)\}$, which means that the task is to design a plan that moves from the lower right to the upper left.

With perfect information, this would be trivial; however, without sensors the uncertainty may grow very quickly. For example, after applying the action $u_1 = (-1, 0)$ from the initial state, the nondeterministic I-state becomes $\{(7, 1), (8, 1), (9, 1)\}$. After $u_2 = (-1, 0)$ it becomes $\{(4, 1), \ldots, (8, 1)\}$. A nice feature of this problem, however, is that uncertainty can be reduced without sensing. Suppose that for 100 stages, we repeatedly apply $u_k = (-1, 0)$. What is the resulting I-state? As the corner state is approached, the uncertainty is reduced because the state cannot be further changed by nature. It is known that each action, $u_k = (-1, 0)$, decreases the X coordinate by at least one each time. Therefore, after nine or more stages, it is known that $\eta_k = \{(1, 1)\}$. Once this is known, then the action $(0, 1)$ can be applied. This will again increase uncertainty as the state moves through the set of left states. If $(0, 1)$ is applied nine or more times, then it is known for certain that $x_k = (1, 10)$, which is the required goal state.

A successful plan has now been obtained: 1) Apply $(-1, 0)$ for nine stages, 2) then apply $(0, 1)$ for nine stages. This plan could be defined over $\mathcal{I}_{ndet}$; however, it is simpler to use the I-map κ_{stage} from Example 11.12 to define a plan as $\pi : \mathbb{N} \to U$. For k such that $1 \leq k \leq 9$, $\pi(k) = (-1, 0)$. For k such that $10 \leq k \leq 18$, $\pi(k) = (0, 1)$. For $k > 18$, $\pi(k) = u_T$. Note that the plan works even if the initial condition is any subset of X. From this point onward, assume that any subset may be given as the initial condition.

Some alternative plans will now be considered by making other derived I-spaces from $\mathcal{I}_{ndet}$. Let κ_3 be an I-map from $\mathcal{I}_{ndet}$ to a set $\mathcal{I}_3$ of three derived I-states. Let $\mathcal{I}_3 = \{g, l, a\}$, in which g denotes "goal," l denotes "left," and a denotes "any." The I-map, κ_3, is

$$X(\eta) = \begin{cases} g & \text{if } X(\eta) = \{(1, 10)\} \\ l & \text{if } X(\eta) \text{ is a subset of the set of left states} \\ a & \text{otherwise.} \end{cases} \tag{11.46}$$

Based on the successful plan described so far, a solution on $\mathcal{I}_3$ is defined as $\pi(g) = u_T$, $\pi(l) = (0, 1)$, and $\pi(a) = (-1, 0)$. This plan is simpler to represent than the one on $\mathbb{N}$; however, there is one drawback. The I-map κ_3 is not sufficient. This implies that more of the nondeterministic I-state needs to be maintained during execution. Otherwise, there is no way to know when certain transitions occur. For example, if $(-1, 0)$ is applied from a, how can the robot determine whether l or a is reached in the next stage? This can be easily determined from the nondeterministic I-state.

To address this problem, consider a new I-map, $\kappa_{19} : \mathcal{I}_{ndet} \to \mathcal{I}_{19}$, which is sufficient. There are 19 derived I-states, which include g as defined previously, l_i for $1 \leq j \leq 9$, and a_i for $2 \leq i \leq 10$. The I-map is defined as $\kappa_{19}(X(\eta)) = g$ if $X(\eta) = \{(1, 10)\}$. Otherwise, $\kappa_{19}(X(\eta)) = l_i$ for the smallest value of i such that $X(\eta)$ is a subset of $\{(1, i), \ldots, (1, 10)\}$. If there is no such value for i, then $\kappa_{19}(X(\eta)) = a_i$, for the smallest value of i such that $X(\eta)$ is a subset of $\{(1, 1), \ldots, (1, 10), (2, 1), \ldots, (i, 1)\}$. Now the plan is defined as $\pi(g) = u_T$, $\pi(l_i) = (0, 1)$, and $\pi(a_i) = (-1, 0)$. Although the plan is larger, the robot does not need to represent the full nondeterministic I-state during execution. The correct transitions occur. For example, if $u_k = (-1, 0)$ is applied at a_5, then a_4 is obtained. If $u = (-1, 0)$ is applied at a_2, then l_1 is obtained. From there, $u = (0, 1)$ is applied to yield l_2. These actions can be repeated until eventually l_9 and g are reached.

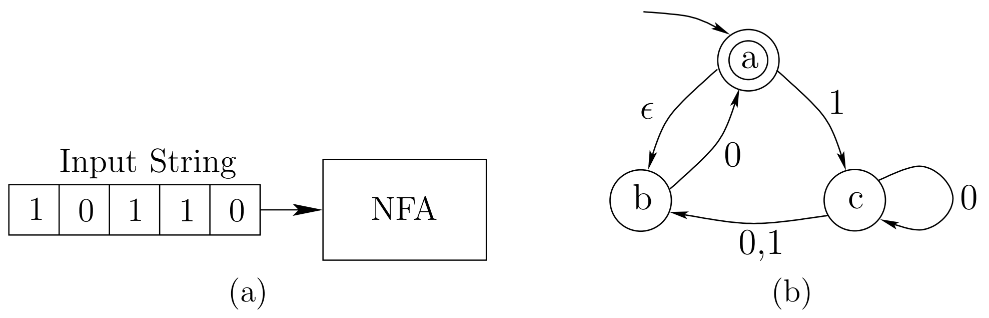


Figure 11.8: (a) An nondeterministic finite automaton (NFA) is a state machine that reads an input string and decides whether to accept it. (b) A graphical depiction of an NFA.

The resulting plan, however, is not an improvement over the original open-loop one. ■

11.3.2 Nondeterministic Finite Automata

An interesting connection lies between the ideas of this chapter and the theory of finite automata, which is part of the theory of computation (see [462, 891]). In Section 2.1, it was mentioned that determining whether there exists some string that is accepted by a DFA is equivalent to a discrete feasible planning problem. If unpredictability is introduced into the model, then a *nondeterministic finite automaton* (NFA) is obtained, as depicted in Figure 11.8. This represents one of the simplest examples of nondeterminism in theoretical computer science. Such nondeterministic models serve as a powerful tool for defining models of computation and their associated complexity classes. It turns out that these models give rise to interesting examples of information spaces.

An NFA is typically described using a directed graph as shown in Figure 11.8b, and is considered as a special kind of finite state machine. Each vertex of the graph represents a state, and edges represent possible transitions. An *input string* of finite length is read by the machine. Typically, the input string is a binary sequence of 0's and 1's. The initial state is designated by an inward arrow that has no source vertex, as shown pointing into state a in Figure 11.8b. The machine starts in this state and reads the first symbol of the input string. Based on its value, it makes appropriate transitions. For a DFA, the next state must be specified for each of the two inputs 0 and 1 from each state. From a state in an NFA, there may be any number of outgoing edges (including zero) that represent the response to a single symbol. For example, there are two outgoing edges if 0 is read from state c (the arrow from c to b actually corresponds to two directed edges, one for 0 and the other for 1). There are also edges designated with a special ϵ symbol. If a state has an outgoing ϵ, the state may immediately transition along the edge without reading another symbol. This may be iterated any number of times, for any outgoing ϵ edges that may be encountered, without reading the next input symbol. The nondeterminism arises from the fact that there are multiple choices for possible next states due to multiple edges for the same input and ϵ transitions. There is no sensor that indicates which state is actually chosen.

The interpretation often given in the theory of computation is that when there are multiple choices, the machine clones itself and one copy runs each choice. It is like having multiple universes in which each different possible action of nature is occurring simultaneously. If there are no outgoing edges for a certain combination of state and input, then the clone dies. Any states that are depicted with a double boundary, such as state a in Figure 11.8, are called *accept states*. When the input string ends, the NFA is said to *accept* the input string if there exists at least one alternate universe in which the final machine state is an accept state.

The formulation usually given for NFAs seems very close to Formulation 2.1 for discrete feasible planning. Here is a typical NFA formulation [891], which formalizes the ideas depicted in Figure 11.8:

Formulation 11.2 (Nondeterministic Finite Automaton)

1. A finite state space X.
2. A finite *alphabet* Σ which represents the possible input symbols. Let $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$.
3. A *transition function*, $\delta : X \times \Sigma_\epsilon \rightarrow \mathrm{pow}(X)$. For each state and symbol, a set of outgoing edges is specified by indicating the states that are reached.
4. A *start state* $x_0 \in X$.
5. A set $A \subseteq X$ of *accept states*.

Example 11.18 (Three-State NFA) The example in Figure 11.8 can be expressed using Formulation 11.2. The components are $X = \{a, b, c\}$, $\Sigma = \{0, 1\}$, $\Sigma_\epsilon = \{0, 1, \epsilon\}$, $x_0 = a$, and $A = \{a\}$. The state transition equation requires the specification of a state for every $x \in X$ and symbol in Σ_ϵ:

$$\begin{array}{c|ccc} & 0 & 1 & \epsilon \\ \hline a & \emptyset & \{c\} & \{b\} \\ b & \{a\} & \emptyset & \emptyset \\ c & \{b,c\} & \{b\} & \emptyset \ . \end{array} \qquad (11.47)$$

■

Now consider reformulating the NFA and its acceptance of strings as a kind of planning problem. An input string can be considered as a plan that uses no form of feedback; it is a fixed sequence of actions. The feasible planning problem is to determine whether any string exists that is accepted by the NFA. Since there is no feedback, there is no sensing model. The initial state is known, but subsequent states cannot be measured. The history I-state η_k at stage k reduces to $\eta_k = \tilde{u}_{k-1} = (u_1, \ldots, u_{k-1})$, the action history. The nondeterminism can be accounted for by defining nature actions that interfere with the state transitions. This results in the following formulation, which is described in terms of Formulation 11.2.

Formulation 11.3 (An NFA Planning Problem)

1. A finite state space X.
2. An action space $U = \Sigma \cup \{u_T\}$.
3. A *state transition function*, $F : X \times U \to \text{pow}(X)$. For each state and symbol, a set of outgoing edges is specified by indicating the states that are reached.
4. An *initial state* $x_0 = x_1$.
5. A set $X_G = A$ of *goal states.*

The history I-space $\mathcal{I}_{hist}$ is defined using

$$\mathcal{I}_k = \tilde{U}_{k-1} \qquad (11.48)$$

for each $k \in \mathbb{N}$ and taking the union as defined in (11.19). Assume that the initial state of the NFA is always fixed; therefore, it does not appear in the definition of $\mathcal{I}_{hist}$.

For expressing the planning task, it is best to use the nondeterministic I-space $\mathcal{I}_{ndet} = \text{pow}(X)$ from Section 11.2.2. Thus, each nondeterministic I-state, $X(\eta) \in \mathcal{I}_{ndet}$, is the subset of X that corresponds to the possible current states of the machine. The initial condition could be any subset of X because ϵ transitions can occur from x_1. Subsequent nondeterministic I-states follow directly from F. The task is to compute a plan of the form

$$\pi = (u_1, u_2, \ldots, u_K, u_T), \qquad (11.49)$$

which results in $X_{K+1}(\eta_{K+1}) \in \mathcal{I}_{ndet}$ with $X_{K+1}(\eta_{K+1}) \cap X_G \neq \emptyset$. This means that at least one possible state of the NFA must lie in X_G after the termination action is applied. This condition is much weaker than a typical planning requirement. Using worst-case analysis, a typical requirement would instead be that *every* possible NFA state lies in X_G.

The problem given in Formulation 11.3 is not precisely a specialization of Formulation 11.1 because of the state transition function. For convenience, F was directly defined, instead of explicitly requiring that f be defined in terms of nature actions, $\Theta(x, u)$, which in this context depend on both x and u for an NFA. There is one other small issue regarding this formulation. In the planning problems considered in this book, it is always assumed that there is a current state.

For an NFA, it was already mentioned that if there are no outgoing edges for a certain input, then the clone of the machine dies. This means that potential current states cease to exist. It is even possible that every clone dies, which leaves no current state for the machine. This can be easily enabled by directly defining F; however, planning problems must always have a current state. To resolve this issue, we could augment X in Formulation 11.3 to include an extra *dead* state, which signifies the death of a clone when there are no outgoing edges. A dead state can never lie in X_G, and once a transition to a dead state occurs, the state remains dead for all time. In this section, the state space will not be augmented in this way; however, it is important to note that the NFA formulation can easily be made consistent with Formulation 11.3.

The planning model can now be compared to the standard use of NFAs in the theory of computation. A *language* of an NFA is defined to be the set of all input strings that it accepts. The planning problem formulated here determines whether there exists a string (which is a plan that ends with termination actions) that is accepted by the NFA. Equivalently, a planning algorithm determines whether the language of an NFA is empty. Constructing the set of all successful plans is equivalent to determining the language of the NFA.

Example 11.19 (Planning for the Three-State NFA) The example in Figure 11.8 can be expressed using Formulation 11.2. The components are $X = \{a, b, c\}$, $\Sigma = \{0, 1\}$, $\Sigma_\epsilon = \{0, 1, \epsilon\}$, $x_0 = a$, and $F = \{a\}$. The function $F(x, u)$ is defined as

$$\begin{array}{c|cc} & 0 & 1 \\ \hline a & \emptyset & \{c\} \\ b & \{a,b\} & \emptyset \\ c & \{b,c\} & \{b\}. \end{array} \tag{11.50}$$

The nondeterministic I-space is

$$X(\eta) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}, \tag{11.51}$$

in which the initial condition is $\eta_0 = \{a, b\}$ because an ϵ transition occurs immediately from a. An example plan that solves the problem is $(1, 0, 0, u_T, \ldots)$. This corresponds to sending an input string "100" through the NFA depicted in Figure 11.8. The sequence of nondeterministic I-states obtained during the execution of the plan is

$$\{a,b\} \xrightarrow{1} \{c\} \xrightarrow{0} \{b,c\} \xrightarrow{0} \{a,b,c\} \xrightarrow{u_T} \{a,b,c\}. \tag{11.52}$$

■

A basic theorem from the theory of finite automata states that for the set of strings accepted by an NFA, there exists a DFA (deterministic) that accepts the same set [891]. This is proved by constructing a DFA directly from the nondeterministic I-space. Each nondeterministic I-state can be considered as a state of a DFA. Thus, the DFA has 2^n states, if the original NFA has n states. The state transitions of the DFA are derived directly from the transitions between nondeterministic I-states. When an input (or action) is given, then a transition occurs from one subset of X to another. A transition is made between the two corresponding states in the DFA. This construction is an interesting example of how the I-space is a new state space that arises when the states of the original state space are unknown. Even though the I-space is usually larger than the original state space, its states are always known. Therefore, the behavior appears the same as in the case of perfect state information. This idea is very general and may be applied to many problems beyond DFAs and NFAs; see Section 12.1.2

11.3.3 The Probabilistic Case: POMDPs

Example 11.14 generalizes nicely to the case of n states. In operations research and artificial intelligence literature, these are generally referred to as *partially observable Markov decision processes* or *POMDPs* (pronounced "pom dee peez"). For the case of three states, the probabilistic I-space, $\mathcal{I}_{prob}$, is a 2-simplex embedded in $\mathbb{R}^3$. In general, if $|X| = n$, then $\mathcal{I}_{prob}$ is an $(n-1)$-simplex embedded in $\mathbb{R}^n$. The coordinates of a point are expressed as $(p_0, p_1, \ldots, p_{n-1}) \in \mathbb{R}^n$. By the axioms of probability, $p_0 + \cdots + p_{n-1} = 1$, which implies that $\mathcal{I}_{prob}$ is an $(n-1)$-dimensional subspace of $\mathbb{R}^n$. The vertices of the simplex correspond to the n cases in which the state is known; hence, their coordinates are $(0, 0, \ldots, 0, 1)$, $(0, 0, \ldots, 0, 1, 0)$, $\ldots$, $(1, 0, \ldots, 0)$. For convenience, the simplex can be projected into $\mathbb{R}^{n-1}$ by specifying a point in $\mathbb{R}^{n-1}$ for which $p_1 + \cdots + p_{n-2} \leq 1$ and then choosing the final coordinate as $p_{n-1} = 1 - p_1 + \cdots + p_{n-2}$. Section 12.1.3 presents algorithms for planning for POMDPs.

11.4 Continuous State Spaces

This section takes many of the concepts that have been developed in Sections 11.1 and 11.2 and generalizes them to continuous state spaces. This represents an important generalization because the configuration space concepts, on which motion planning was defined in Part II, are all based on continuous state spaces. In this section, the state space might be a configuration space, $X = \mathcal{C}$, as defined in Chapter 4 or any other continuous state space. Since it may be a configuration space, many interesting problems can be drawn from robotics.

During the presentation of the concepts of this section, it will be helpful to recall analogous concepts that were already developed for discrete state spaces. In many cases, the formulations appear identical. In others, the continuous case is more complicated, but it usually maintains some of the properties from the discrete case. It will be seen after introducing continuous sensing models in Section 11.5.1 that some problems formulated in continuous spaces are even more elegant and easy to understand than their discrete counterparts.

11.4.1 Discrete-Stage Information Spaces

Assume here that there are discrete stages. Let $X \subseteq \mathbb{R}^m$ be an n-dimensional manifold for $n \leq m$ called the *state space*.[4] Let $Y \subseteq \mathbb{R}^m$ be an n_y-dimensional manifold for $n_y \leq m$ called the *observation space*. For each $x \in X$, let $\Psi(x) \subseteq \mathbb{R}^m$ be an n_n-dimensional manifold for $n_n \leq m$ called the set of *nature sensing actions*. The three kinds of sensors mappings, h, defined in Section 11.1.1 are possible, to yield either a *state mapping*, $y = h(x)$, a *state-nature mapping* $y = h(x, \psi)$, or a *history-based*, $y = h_k(x_1, \ldots, x_k, y)$. For the case of a state mapping, the preimages, $H(y)$, once again induce a partition of X. Preimages can also be defined for state-action mappings, but they do not necessarily induce a partition of X.

Many interesting sensing models can be formulated in continuous state spaces. Section 11.5.1 provides a kind of sensor catalog. There is once again the choice of nondeterministic or probabilistic uncertainty if nature sensing actions are used. If nondeterministic uncertainty is used, the expressions are the same as the discrete case. Probabilistic models are defined in terms of a probability density function, $p : \Psi \to [0, \infty)$,[5] in which $p(\psi)$ denotes the continuous-time replacement for $P(\psi)$. The model can also be expressed as $p(y|x)$, in that same manner that $P(y|x)$ was obtained for discrete state spaces.

The usual three choices exist for the initial conditions: 1) Either $x_1 \in X$ is given; 2) a subset $X_1 \in X$ is given; or 3) a probability density function, $p(x)$, is given. The initial condition spaces in the last two cases can be enormous. For example, if $X = [0, 1]$ and any subset is possible as an initial condition, then $\mathcal{I}_0 = \text{pow}(\mathbb{R})$, which has higher cardinality than $\mathbb{R}$. If any probability density function is possible, then $\mathcal{I}_0$ is a space of functions.[6]

The I-space definitions from Section 11.1.2 remain the same, with the understanding that all of the variables are continuous. Thus, (11.17) and (11.19) serve as the definitions of $\mathcal{I}_k$ and $\mathcal{I}$. Let $U \subseteq \mathbb{R}^m$ be an n_u-dimensional manifold for $n_u \leq m$. For each $x \in X$ and $u \in U$, let $\Theta(x, u)$ be an n_θ-dimensional manifold for $n_\theta \leq m$. A discrete-stage I-space planning problem over continuous state spaces can be easily formulated by replacing each discrete variable in Formulation 11.1 by its continuous counterpart that uses the same notation. Therefore, the full formulation is not given.

[4]If you did not read Chapter 4 and are not familiar with manifold concepts, then assume $X = \mathbb{R}^n$; it will not make much difference. Make similar assumptions for Y, $\Psi(x)$, U, and $\Theta(x, u)$.

[5]Assume that all continuous spaces are measure spaces and all probability density functions are measurable functions over these spaces.

[6]To appreciate of the size of this space, it can generally be viewed as an infinite-dimensional vector space (recall Example 8.5). Consider, for example, representing each function with a series expansion. To represent any analytic function exactly over $[0, 1]$, an infinite sequence of real-valued coefficients may be needed. Each sequence can be considered as an infinitely long vector, and the set of all such sequences forms an infinite-dimensional vector space. See [346, 836] for more background on function spaces and functional analysis.

11.4.2 Continuous-Time Information Spaces

Now assume that there is a continuum of stages. Most of the components of Section 11.4.1 remain the same. The spaces X, Y, $\Psi(x)$, U, and $\Theta(x,u)$ remain the same. The sensor mapping also remains the same. The main difference occurs in the state transition equation because the effect of nature must be expressed in terms of velocities. This was already introduced in Section 10.6. In that context, there was only uncertainty in predictability. In the current context there may be uncertainties in both predictability and in sensing the current state.

For the discrete-stage case, the history I-states were based on action and observation sequences. For the continuous-time case, the history instead becomes a function of time. As defined in Section 7.1.1, let T denote a *time interval*, which may be bounded or unbounded. Let $\tilde{y}_t : [0,t] \rightarrow Y$ be called the *observation history* up to time $t \in T$. Similarly, let $\tilde{u}_t : [0,t) \rightarrow U$ and $\tilde{x}_t : [0,t] \rightarrow X$ be called the *action history* and *state history*, respectively, up to time $t \in T$.

Thus, the three kinds of sensor mappings in the continuous-time case are as follows:

1. A *state-sensor mapping* is expressed as $y(t) = h(x(t))$, in which $x(t)$ and $y(t)$ are the state and observation, respectively, at time $t \in T$.
2. A *state-nature mapping* is expressed as $y(t) = h(x(t), \psi(t))$, which implies that nature chooses some $\psi(t) \in \Psi(x(t))$ for each $t \in T$.
3. A *history-based sensor mapping*, which could depend on all of the states obtained so far. Thus, it depends on the entire function $\tilde{x}_t$. This could be denoted as $y(t) = h(\tilde{x}_t, \psi(t))$ if nature can also interfere with the observation.

If $\tilde{u}_t$ and $\tilde{y}_t$ are combined with the initial condition η_0, the *history I-state at time t* is obtained as

$$\eta_t = (\eta_0, \tilde{u}_t, \tilde{y}_t). \tag{11.53}$$

The *history I-space at time t* is the set of all possible η_t and is denoted as $\mathcal{I}_t$. Note that $\mathcal{I}_t$ is a space of functions because each $\eta_t \in \mathcal{I}_t$ is a function of time. Recall that in the discrete-stage case, every $\mathcal{I}_k$ was combined into a single history I-space, $\mathcal{I}_{hist}$, using (11.18) or (11.19). The continuous-time analog is obtained as

$$\mathcal{I}_{hist} = \bigcup_{t \in T} \mathcal{I}_t, \tag{11.54}$$

which is an irregular collection of functions because they have different domains; this irregularity also occurred in the discrete-stage case, in which $\mathcal{I}_{hist}$ was composed of sequences of varying lengths.

A continuous-time version of the cost functional in Formulation 11.1 can be given to evaluate the execution of a plan. Let L denote a cost functional that may be applied to any state-action history $(\tilde{x}_t, \tilde{u}_t)$ to yield

$$L(\tilde{x}_t, \tilde{u}_t) = \int_0^t l(x(t'), u(t'))dt' + l_F(x(t)), \tag{11.55}$$

in which $l(x(t'), u(t'))$ is the instantaneous cost and $l_F(x(t))$ is a final cost.

11.4.3 Derived Information Spaces

For continuous state spaces, the motivation to construct derived I-spaces is even stronger than in the discrete case because the I-space quickly becomes unwieldy.

11.4.3.1 Nondeterministic and probabilistic I-spaces for discrete stages

The concepts of I-maps and derived I-spaces from Section 11.2 extend directly to continuous spaces. In the nondeterministic case, κ_{ndet} once again transforms the initial condition and history into a subset of X. In the probabilistic case, κ_{prob} yields a probability density function over X. First, consider the discrete-stage case.

The nondeterministic I-states are obtained exactly as defined in Section 11.2.2, except that the discrete sets are replaced by their continuous counterparts. For example, $F(x, u)$ as defined in (11.28) is now a continuous set, as are X and $\Theta(x, u)$. Since probabilistic I-states are probability density functions, the derivation in Section 11.2.3 needs to be modified slightly. There are, however, no important conceptual differences. Follow the derivation of Section 11.2.3 and consider which parts need to be replaced.

The replacement for (11.35) is

$$p(x_k|y_k) = \frac{p(y_k|x_k)p(x_k)}{\displaystyle\int_X p(y_k|x_k)p(x_k)dx_k}, \tag{11.56}$$

which is based in part on deriving $p(y_k|x_k)$ from $p(\psi_k|x_k)$. The base of the induction, which replaces (11.36), is obtained by letting $k = 1$ in (11.56). By following the explanation given from (11.37) to (11.40), but using instead probability density functions, the following update equations are obtained:

$$\begin{aligned} p(x_{k+1}|\eta_k, u_k) &= \int_X p(x_{k+1}|x_k, u_k, \eta_k)p(x_k|\eta_k)dx_k \\ &= \int_X p(x_{k+1}|x_k, u_k)p(x_k|\eta_k)dx_k, \end{aligned} \tag{11.57}$$

and

$$p(x_{k+1}|y_{k+1}, \eta_k, u_k) = \frac{p(y_{k+1}|x_{k+1})p(x_{k+1}|\eta_k, u_k)}{\displaystyle\int_X p(y_{k+1}|x_{k+1})p(x_{k+1}|\eta_k, u_k)dx_{k+1}}. \tag{11.58}$$

11.4.3.2 Approximating nondeterministic and probabilistic I-spaces

Many other derived I-spaces extend directly to continuous spaces, such as the limited-memory models of Section 11.2.4 and Examples 11.11 and 11.12. In the present context, it is extremely useful to try to collapse the I-space as much as possible because it tends to be unmanageable in most practical applications. Recall that an I-map, $\kappa : \mathcal{I}_{hist} \rightarrow \mathcal{I}_{der}$, partitions $\mathcal{I}_{hist}$ into sets over which a constant action must be applied. The main concern is that restricting plans to $\mathcal{I}_{der}$ does not inhibit solutions.

Consider making derived I-spaces that approximate nondeterministic or probabilistic I-states. Approximations make sense because X is usually a metric space in the continuous setting. The aim is to dramatically simplify the I-space while trying to avoid the loss of critical information. A trade-off occurs in which the quality of the approximation is traded against the size of the resulting derived I-space. For the case of nondeterministic I-states, *conservative approximations* are formulated, which are sets that are guaranteed to contain the nondeterministic I-state. For the probabilistic case, *moment-based approximations* are presented, which are based on general techniques from probability and statistics to approximate probability densities. To avoid unnecessary complications, the presentation will be confined to the discrete-stage model.

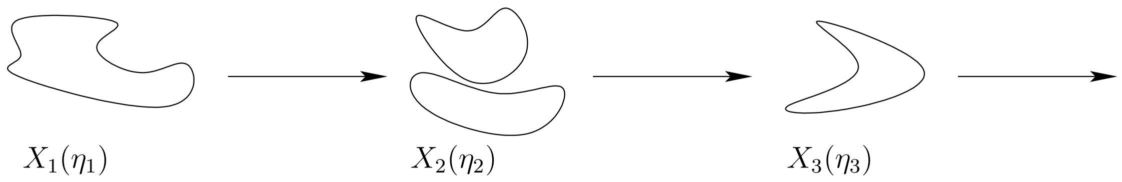


Figure 11.9: The nondeterministic I-states may be complicated regions that are difficult or impossible to compute.

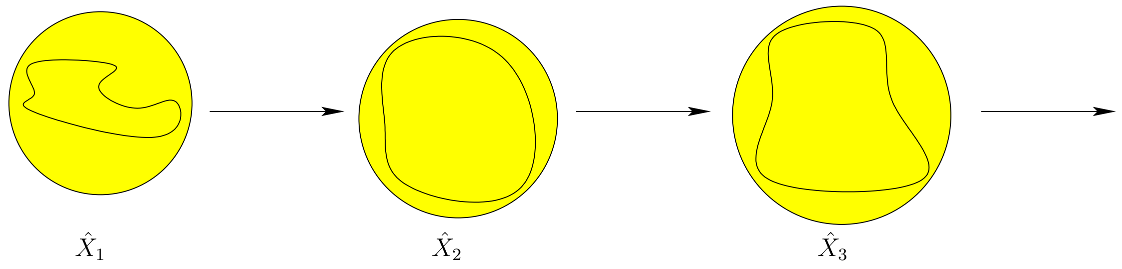


Figure 11.10: The nondeterministic I-states can be approximated by bounding spheres.

Conservative approximations Suppose that nondeterministic uncertainty is used and an approximation is made to the nondeterministic I-states. An I-map, $\kappa_{app} : \mathcal{I}_{ndet} \rightarrow \mathcal{I}_{app}$, will be defined in which $\mathcal{I}_{app}$ is a particular family of subsets of X. For example, $\mathcal{I}_{app}$ could represent the set of all ball subsets of X. If $X = \mathbb{R}^2$, then the balls become discs, and only three parameters (x, y, r) are needed to parameterize $\mathcal{I}_{app}$ (x, y for the center and r for the radius). This implies that $\mathcal{I}_{app} \subset \mathbb{R}^3$; this appears to be much simpler than $\mathcal{I}_{ndet}$, which could be a complicated collection of regions in $\mathbb{R}^2$. To make $\mathcal{I}_{app}$ even smaller, it could be required that x, y, and r are integers (or are sampled with some specified dispersion, as defined in Section 5.2.3). If $\mathcal{I}_{app}$ is bounded, then the number of derived I-states would become finite. Of course, this comes an at expense because $\mathcal{I}_{ndet}$ may be poorly approximated.

For a fixed sequence of actions $(u_1, u_2, \ldots)$ consider the sequence of nondeterministic I-states:

$$X_1(\eta_1) \xrightarrow{u_1,y_2} X_2(\eta_2) \xrightarrow{u_2,y_3} X_3(\eta_3) \xrightarrow{u_3,y_4} \cdots , \tag{11.59}$$

which is also depicted in Figure 11.9. The I-map $\mathcal{I}_{app}$ must select a bounding region for every nondeterministic I-state. Starting with a history I-state, η, the nondeterministic I-state $X_k(\eta_k)$ can first be computed, followed by applying $\mathcal{I}_{app}$ to yield a bounding region. If there is a way to efficiently compute $X_k(\eta_k)$ for any η_k, then a plan on $\mathcal{I}_{app}$ could be much simpler than those on $\mathcal{I}_{ndet}$ or $\mathcal{I}_{hist}$.

If it is difficult to compute $X_k(\eta_k)$, one possibility is to try to define a derived information transition equation, as discussed in Section 11.2.1. The trouble, however, is that $\mathcal{I}_{app}$ is usually not a sufficient I-map. Imagine wanting to compute $\kappa_{app}(X_{k+1}(\eta_{k+1}))$, which is a bounding approximation to $X_{k+1}(\eta_{k+1})$. This can be accomplished by starting with $X_k(\eta_k)$, applying the update rules (11.30) and (11.31), and then applying κ_{app} to $X_{k+1}(\eta_{k+1})$. In general, this does not produce the same result as starting with the bounding volume $\mathcal{I}_{app}(X_k(\eta_k))$, applying (11.30) and (11.31), and then applying κ_{app}.

Thus, it is not possible to express the transitions entirely in $\mathcal{I}_{app}$ without some further loss of information. However, if this loss of information is tolerable, then an information-destroying approximation may nevertheless be useful. The general idea is to make a bounding region for the nondeterministic I-state in each iteration. Let $\hat{X}_k$ denote this bounding region at stage k. Be careful in using such approximations. As depicted in Figures 11.9 and 11.10, the sequences of derived I-states diverge. The sequence in Figure 11.10 is *not* obtained by simply bounding each calculated I-state by an element of $\mathcal{I}_{app}$; the entire sequence is different.

Initially, $\hat{X}_1$ is chosen so that $X_1(\eta_1) \subseteq \hat{X}_1$. In each inductive step, $\hat{X}_k$ is treated as if it were the true nondeterministic I-state (not an approximation). Using (11.30) and (11.31), the update for considering u_k and y_{k+1} is

$$\hat{X}'_{k+1} = \left(\bigcup_{x_k \in \hat{X}_k} F(x_k, u_k) \right) \cap H(y_{k+1}). \tag{11.60}$$

In general, $\hat{X}'_{k+1}(\eta_{k+1})$ might not lie in $\mathcal{I}_{app}$. Therefore, a bounding region, $\hat{X}_{k+1} \in \mathcal{I}_{app}$, must be selected to approximate $\hat{X}'$ under the constraint that $\hat{X}'_{k+1} \subseteq \hat{X}_{k+1}$. This completes the inductive step, which together with the base case yields a sequence

$$\hat{X}_1 \xrightarrow{u_1,y_2} \hat{X}_2 \xrightarrow{u_2,y_3} \hat{X}_3 \xrightarrow{u_3,y_4} \cdots , \tag{11.61}$$

which is depicted in Figure 11.10.

Both a plan, $\pi : \mathcal{I}_{app} \rightarrow U$, and information transitions can now be defined over $\mathcal{I}_{app}$. To ensure that a plan is sound, the approximation must be conservative. If in some iteration, $\hat{X}_{k+1}(\eta_{k+1}) \subset \hat{X}'_{k+1}(\eta_{k+1})$, then the true state may not necessarily be included in the approximate derived I-state. This could, for example, mean that a robot is in a collision state, even though the derived I-state indicates that this is impossible. This bad behavior is generally avoided by keeping conservative approximations. At one extreme, the approximations can be made very conservative by always assigning $\hat{X}_{k+1}(\eta_{k+1}) = X$. This, however, is useless because the only possible plans must apply a single, fixed action for every stage. Even if the approximations are better, it might still be impossible to cause transitions in the approximated I-state. To ensure that solutions exist to the planning problem, it is therefore important to make the bounding volumes as close as possible to the derived I-states.

This trade-off between the simplicity of bounding volumes and the computational expense of working with them was also observed in Section 5.3.2 for collision detection. Dramatic improvement in performance can be obtained by working with simpler shapes; however, in the present context this could come at the expense of failing to solve the problem. Using balls as described so far might not seem to provide very tight bounds. Imagine instead using solid ellipsoids. This would provide tighter approximations, but the dimension of $\mathcal{I}_{app}$ grows quadratically with the dimension of X. A sphere equation generally requires $n + 1$ parameters, whereas the ellipsoid equation requires $\binom{n}{2} + 2n$ parameters. Thus, if the dimension of X is high, it may be difficult or even impossible to use ellipsoid approximations. Nonconvex bounding shapes could provide even better approximations, but the required number of parameters could easily become unmanageable, even if $X = \mathbb{R}^2$. For very particular problems, however, it may be possible to design a family of shapes that is both manageable and tightly approximates the nondeterministic I-states. This leads to many interesting research issues.

Moment-based approximations Since the probabilistic I-states are functions, it seems natural to use function approximation methods to approximate $\mathcal{I}_{prob}$. One possibility might be to use the first m coefficients of a Taylor series expansion. The derived I-space then becomes the space of possible Taylor coefficients. The quality of the approximation is improved as m is increased, but also the dimension of the derived I-space rises.

Since we are working with probability density functions, it is generally preferable to use moments as approximations instead of Taylor series coefficients or other generic function approximation methods. The first and second moments are the familiar *mean* and *covariance*, respectively. These are preferable over other approximations because the mean and covariance exactly represent the Gaussian density, which is the most basic and fundamental density in probability theory. Thus, approximating the probabilistic I-space with first and second moments is equivalent to assuming that the resulting probability densities are always Gaussian. Such approximations are frequently made in practice because of the convenience of working with Gaussians. In general, higher order moments can be used to obtain higher quality approximations at the expense of more coefficients. Let $\kappa_{mom} : \mathcal{I}_{prob} \rightarrow \mathcal{I}_{mom}$ denote a moment-based I-map.

The same issues arise for κ_{mom} as for κ_{app}. In most cases, κ_{mom} is not a sufficient I-map. The moments are computed in the same way as the conservative approximations. The update equations (11.57) and (11.58) are applied for probabilistic I-states; however, after each step, κ_{mom} is applied to the resulting probability density function. This traps the derived I-states in $\mathcal{I}_{mom}$. The moments could be computed after each of (11.57) and (11.58) or after both of them have been applied (different results may be obtained). The later case may be more difficult to compute, depending on the application.

First consider using the mean (first moment) to represent some probabilistic I-state, $p(x|\eta)$. Let x_i denote the ith coordinate of x. The mean, $\bar{x}_i$, with respect to x_i is generally defined as

$$\bar{x}_i = \int_X x_i \, p(x|\eta) dx. \tag{11.62}$$

This leads to the vector mean $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)$. Suppose that we would like to construct $\mathcal{I}_{mom}$ using only the mean. Since there is no information about the covariance of the density, working with $\bar{x}$ is very similar to estimating the state. The mean value serves as the estimate, and $\mathcal{I}_{mom} = X$. This certainly helps to simplify the I-space, but there is no way to infer the amount of uncertainty associated with the I-state. Was the probability mass concentrated greatly around $\bar{x}$, or was the density function very diffuse over X?

Using second moments helps to alleviate this problem. The covariance with respect to two variables, x_i and x_i, is

$$\sigma_{i,j} = \int_X x_i x_j \, p(x|\eta) dx. \tag{11.63}$$

Since $\sigma_{ij} = \sigma_{ji}$, the second moments can be organized into a symmetric *covariance matrix*,

$$\Sigma = \begin{pmatrix} \sigma_{1,1} & \sigma_{1,2} & \cdots & \sigma_{1,n} \\ \sigma_{2,1} & \sigma_{2,2} & \cdots & \sigma_{2,n} \\ \vdots & \vdots & & \vdots \\ \sigma_{n,1} & \sigma_{n,2} & \cdots & \sigma_{n,n} \end{pmatrix} \tag{11.64}$$

for which there are $\binom{n}{2} + n$ unique elements, corresponding to every $x_{i,i}$ and every way to pair x_i with x_j for each distinct i and j such that $1 \leq i, j \leq n$. This implies that if first and second moments are used, then the dimension of $\mathcal{I}_{mom}$ is $\binom{n}{2} + 2n$. For some problems, it may turn out that all probabilistic I-states are indeed Gaussians. In this case, the mean and covariance exactly capture the probabilistic I-space. The I-map in this case is sufficient. This leads to a powerful tool called the Kalman filter, which is the subject of Section 11.6.1.

Higher quality approximations can be made by taking higher order moments. The *rth moment* is defined as

$$\int_X x_{i_1} x_{i_2} \cdots x_{i_r} \, p(x|\eta) dx, \tag{11.65}$$

in which i_1, i_2, ..., i_r are r integers chosen with replacement from $\{1, \ldots, n\}$.

The moment-based approximation is very similar to the conservative approximations for nondeterministic uncertainty. The use of mean and covariance appears very similar to using ellipsoids for the nondeterministic case. The level sets of a Gaussian density are ellipsoids. These level sets generalize the notion of confidence intervals to confidence ellipsoids, which provides a close connection between the nondeterministic and probabilistic cases. The domain of a Gaussian density is $\mathbb{R}^n$, which is not bounded, contrary to the nondeterministic case. However, for a given confidence level, it can be approximated as a bounded set. For example, an elliptical region can be computed in which 99.9% of the probability mass falls. In general, it may be possible to combine the idea of moments and bounding volumes to construct a derived I-space for the probabilistic case. This could yield the guaranteed correctness of plans while also taking probabilities into account. Unfortunately, this would once again increase the dimension of the derived I-space.

11.4.3.3 Derived I-spaces for continuous time

The continuous-time case is substantially more difficult, both to express and to compute in general forms. In many special cases, however, there are elegant ways to compute it. Some of these will be covered in Section 11.5 and Chapter 12. To help complete the I-space framework, some general expressions are given here. In general, I-maps and derived I-spaces can be constructed following the ideas of Section 11.2.1.

Since there are no discrete transition rules, the derived I-states cannot be expressed in terms of simple update rules. However, they can at least be expressed as a function that indicates the state $x(t)$ that will be obtained after $\tilde{u}_t$ and $\tilde{\theta}_t$ are applied from an initial state $x(0)$. Often, this is obtained via some form of integration (see Section 14.1), although this may not be explicitly given. In general, let $X_t(\eta_t) \subset X$ denote a nondeterministic I-state at time t; this is the replacement for X_k from the discrete-stage case. The initial condition is denoted as X_0, as opposed to X_1, which was used in the discrete-stage case.

More definitions are needed to precisely characterize $X_t(\eta_t)$. Let $\tilde{\theta}_t : [0, t) \rightarrow \Theta$ denote the history of nature actions up to time t. Similarly, let $\tilde{\psi}_t : [0, t] \rightarrow \Psi$ denote the history of nature sensing actions. Suppose that the initial condition is $X_0 \subset X$. The nondeterministic I-state is defined as

$$\begin{split} X_t(\eta_t) = \{x \in X \mid \exists x' \in X_0, \;\; \exists \tilde{\theta}_t, \text{ and } \exists \tilde{\psi}_t \text{ such that} \\ x = \Phi(x', \tilde{u}_t, \tilde{\theta}_t) \text{ and } \forall t' \in [0, t], \; y(t') = h(x(t'), \psi(t'))\}. \end{split} \tag{11.66}$$

In words, this means that a state $x(t)$ lies in $X_t(\eta_t)$ if and only if there exists an initial state $x' \in X_0$, a nature history $\tilde{\theta}_t$, and a nature sensing action history, $\tilde{\psi}_t$ such that the transition equation causes arrival at $x(t)$ and the observation history $\tilde{y}_t$ agrees with the sensor mapping over all time from 0 to t.

It is also possible to derive a probabilistic I-state, but this requires technical details from continuous-time stochastic processes and stochastic differential equations. In some cases, the resulting expressions work out very nicely; however, it is difficult to formulate a general expression for the derived I-state because it depends on many technical assumptions regarding the behavior of the stochastic processes. For details on such systems, see [567].

11.5 Examples for Continuous State Spaces

11.5.1 Sensor Models

A remarkable variety of sensing models arises in applications that involve continuous state spaces. This section presents a catalog of different kinds of sensor models that is inspired mainly by robotics problems. The models are gathered together in one place to provide convenient reference. Some of them will be used in upcoming sections, and others are included to help in the understanding of I-spaces. For each sensor, there are usually two different versions, based on whether nature sensing actions are included.

Linear sensing models Developed mainly in control theory literature, *linear sensing models* are some of the most common and important. For all of the sensors in this family, assume that $X = Y = \mathbb{R}^n$ (nonsingular linear transformations allow the sensor space to effectively have lower dimension, if desired). The simplest case in this family is the *identity sensor*, in which $y = x$. In this case, the state is immediately known. If this sensor is available at every stage, then the I-space collapses to X by the I-map $\kappa_{sf} : \mathcal{I}_{hist} \rightarrow X$.

Now nature sensing actions can be used to corrupt this perfect state observation to obtain $y = h(x, \psi) = x + \psi$. Suppose that y is an estimate of x, the current state, with error bounded by a constant $r \in (0, \infty)$. This can be modeled by assigning for every $x \in X$, $\Psi(x)$ as a closed ball of radius r, centered at the origin:

$$\Psi = \{\psi \in \mathbb{R}^n \mid \|\psi\| \leq r\}. \tag{11.67}$$

Figure 11.11 illustrates the resulting nondeterministic sensing model. If the observation y is received, then it is known that the true state lies within a ball in X of radius r, centered at y. This ball is the preimage, $H(y)$, as defined in (11.11). To make the model probabilistic, a probability density function can be defined over Ψ. For example, it could be assumed that $p(\psi)$ is a uniform density (although this model is not very realistic in many applications because there is a boundary at which the probability mass discontinuously jumps to zero).

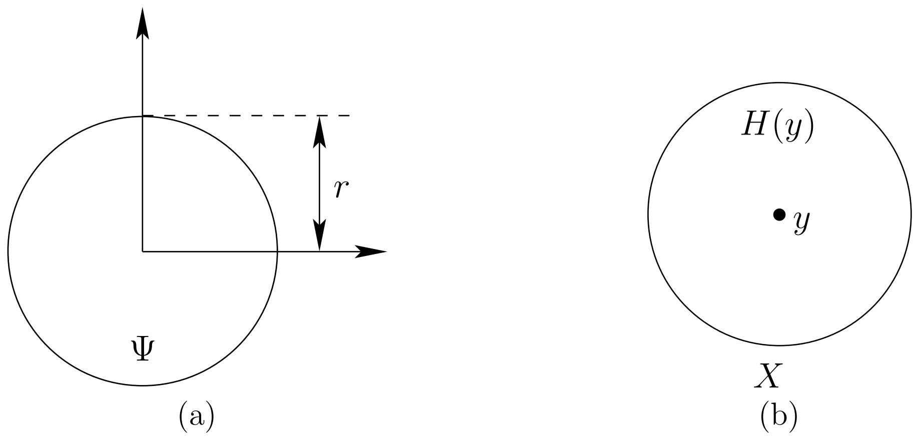


Figure 11.11: A simple sensing model in which the observation error is no more than r: (a) the nature sensing action space; (b) the preimage in X based on observation y.

A more typical probabilistic sensing model can be made by letting $\Psi(x) = \mathbb{R}^n$ and defining a probability density function over all of $\mathbb{R}^n$. (Note that the nondeterministic version of this sensor is completely useless.) One of the easiest choices to work with is the multivariate Gaussian probability density function,

$$p(\psi) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}\psi^T \Sigma \psi}, \tag{11.68}$$

in which Σ is the covariance matrix (11.64), $|\Sigma|$ is its determinant, and $\psi^T\Sigma\psi$ is a quadratic form, which multiplies out to yield

$$\psi^T \Sigma \psi = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{i,j} \psi_i \psi_j. \tag{11.69}$$

If $p(x)$ is a Gaussian and y is received, then $p(y|x)$ must also be Gaussian under this model. This will become very important in Section 11.6.1.

The sensing models presented so far can be generalized by applying linear transformations. For example, let C denote a nonsingular $n \times n$ matrix with real-valued entries. If the sensor mapping is $y = h(x) = Cx$, then the state can still be determined immediately because the mapping $y = Cx$ is bijective; each $H(y)$ contains a unique point of X. A linear transformation can also be formed on the nature sensing action. Let W denote an $n \times n$ matrix. The sensor mapping is

$$y = h(x) = Cx + W\psi. \tag{11.70}$$

In general, C and W may even be singular, and a linear sensing model is still obtained. Suppose that $W = 0$. If C is singular, however, it is impossible to infer the state directly from a single sensor observation. This generally corresponds to a projection from an n-dimensional state space to a subset of Y whose dimension is the rank of C. For example, if

$$C = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \tag{11.71}$$

then $y = Cx$ yields $y_1 = x_2$ and $y_2 = 0$. Only x_2 of each $(x_1, x_2) \in X$ can be observed because C has rank 1. Thus, for some special cases, singular matrices can measure some state variables while leaving others invisible. For a general singular matrix C, the interpretation is that X is projected into some k-dimensional subspace by the sensor, in which k is the rank of C. If W is singular, this means that the effect of nature is limited. The degrees of freedom with which nature can distort the sensor observations is the rank of W. These concepts motivate the next set of sensor models.

Simple projection sensors Several common sensor models can be defined by observing particular coordinates of X while leaving others invisible. This is the continuous version of the selective sensor from Example 11.4. Imagine, for example, a mobile robot that rolls in a 2D world, $\mathcal{W} = \mathbb{R}^2$, and is capable of rotation. The state space (or configuration space) is $X = \mathbb{R}^2 \times \mathbb{S}^1$. For visualization purposes, it may be helpful to imagine that the robot is very tiny, so that it can be interpreted as a point, to avoid the complicated configuration space constructions of Section 4.3.[7] Let $p = (p_1, p_2)$ denote the coordinates of the point, and let $s \in \mathbb{S}^1$ denote its orientation. Thus, a state in $\mathbb{R}^2 \times \mathbb{S}^1$ is specified as (p_1, p_2, s) (rather than (x, y, θ), which may cause confusion with important spaces such as X, Y, and Θ).

Suppose that the robot can estimate its position but does not know its orientation. This leads to a *position sensor* defined as $Y = \mathbb{R}^2$, with $y_1 = p_1$ and $y_2 = p_2$ (also denoted as $y = h(x) = p$). The third state variable, s, of the state remains unknown. Of course, any of the previously considered nature sensing action models can be added. For example, nature might cause errors that are modeled with Gaussian probability densities.

A *compass* or *orientation sensor* can likewise be made by observing only the final state variable, s. In this case, $Y = \mathbb{S}^1$ and $y = s$. Nature sensing actions can be included. For example, the sensed orientation may be y, but it is only known that $|s - y| \leq \epsilon$ for some constant ϵ, which is the maximum sensor error. A Gaussian model cannot exactly be applied because its domain is unbounded and $\mathbb{S}^1$ is bounded. This can be fixed by truncating the Gaussian or by using a more appropriate distribution.

The position and orientation sensors generalize nicely to a 3D world, $\mathcal{W} = \mathbb{R}^3$. Recall from Section 4.2 that in this case the state space is $X = SE(3)$, which can be represented as $\mathbb{R}^3 \times \mathbb{RP}^3$. A position sensor measures the first three coordinates, whereas an orientation sensor measures the last three coordinates. A physical

[7]This can also be handled, but it just adds unnecessary complication to the current discussion.

sensor that measures orientation in $\mathbb{R}^3$ is often called a *gyroscope*. These are usually based on the principle of precession, which means that they contain a spinning disc that is reluctant to change its orientation due to angular momentum. For the case of a linkage of bodies that are connected by revolute joints, a point in the state space gives the angles between each pair of attached links. A *joint encoder* is a sensor that yields one of these angles.

Dynamics of mechanical systems will not be considered until Part IV; however, it is worth pointing out several sensors. In these problems, the state space will be expanded to include velocity parameters and possibly even acceleration parameters. In this case, a *speedometer* can sense a velocity vector or a scalar speed. Sensors even exist to measure *angular velocity*, which indicates the speed with which rotation occurs. Finally, an *accelerometer* can be used to sense acceleration parameters. With any of these models, nature sensing actions can be used to account for measurement errors.

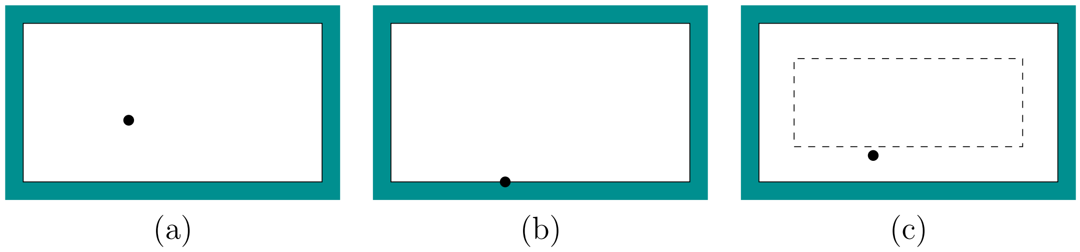


Figure 11.12: Boundary sensors indicate whether contact with the boundary has occurred. In the latter case, a proximity sensor may indicate whether the boundary is within a specified distance.

Boundary sensors If the state space has an interesting boundary, as in the case of $\mathcal{C}_{free}$ for motion planning problems, then many important *boundary sensors* can be formulated based on the detection of the boundary. Figure 11.12 shows several interesting cases on which sensors are based.

Suppose that the state space is a closed set with some well-defined boundary. To provide a connection to motion planning, assume that $X = \mathrm{cl}(\mathcal{C}_{free})$, the closure of $\mathcal{C}_{free}$. A *contact sensor* determines whether the boundary is being contacted. In this case, $Y = \{0, 1\}$ and h is defined as $h(x) = 1$ if $x \in \partial X$, and $h(x) = 0$ otherwise. These two cases are shown in Figures 11.12a and 11.12b, respectively. Using this sensor, there is no information regarding where along the boundary the contact may be occurring. In mobile robotics, it may be disastrous if the robot is in contact with obstacles. Instead, a *proximity sensor* is often used, which yields $h(x) = 1$ if the state or position is within some specified constant, r, of ∂X, and $h(x) = 0$ otherwise. This is shown in Figure 11.12.

In robot manipulation, haptic interfaces, and other applications in which physical interaction occurs between machines and the environment, a *force sensor* may be used. In addition to simply indicating contact, a force sensor can indicate the magnitude and direction of the force. The robot model must be formulated so that it is possible to derive the anticipated force value from a given state.

Landmark sensors Many important sensing models can be defined in terms of *landmarks*. A landmark is a special point or region in the state space that can be detected in some way by the sensor. The measurements of the landmark can be used to make inferences about the current state. An ancient example is using stars to navigate on the ocean. Based on the location of the stars relative to a ship, its orientation can be inferred. You may have found landmarks useful for trying to find your way through an unfamiliar city. For example, mountains around the perimeter of Mexico City or the Eiffel Tower in Paris might be used to infer your heading. Even though the streets of Paris are very complicated, it might be possible to walk to the Eiffel Tower by walking toward it whenever it is visible. Such models are common in the *competitive ratio* framework for analyzing on-line algorithms [674].

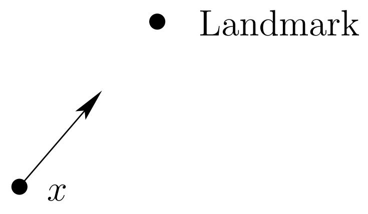


Figure 11.13: The most basic landmark sensor indicates only its direction.

In general, a set of states may serve as landmarks. A common model is to make x_G a single landmark. In robotics applications, these landmarks may be instead considered as points in the world, $\mathcal{W}$. Generalizations from points to landmark regions are also possible. The ideas, here, however, will be kept simple to illustrate the concept. Following this presentation, you can imagine a wide variety of generalizations. Assume for all examples of landmarks that $X = \mathbb{R}^2$, and let a state be denoted by $x = (x_1, x_2)$.

For the first examples, suppose there is only one landmark, $l \in X$, with coordinates (l_1, l_2). A *homing sensor* is depicted in Figure 11.13 and yields values in $Y = \mathbb{S}^1$. The sensor mapping is $h(x) = \operatorname{atan2}(l_1 - x_1, l_2 - x_2)$, in which atan2 gives the angle in the proper quadrant.

Another possibility is a *Geiger counter sensor* (radiation level), in which $Y = [0, \infty)$ and $h(x) = \|x - l\|$. In this case, only the distance to the landmark is reported, but there is no directional information.

A contact sensor could also be combined with the landmark idea to yield a sensor called a *pebble*. This sensor reports 1 if the pebble is "touched"; otherwise, it reports 0. This idea can be generalized nicely to regions. Imagine that there is a *landmark region*, $X_l \subset X$. If $x \in X_l$, then the *landmark region detector* reports 1; otherwise, it reports 0.

Many useful and interesting sensing models can be formulated by using the ideas explained so far with multiple landmarks. For example, using three homing sensors that are not collinear, it is possible to reconstruct the exact state. Many interesting problems can be made by populating the state space with landmark regions and their associated detectors. In mobile robotics applications, this can be implemented by placing stationary cameras or other sensors in an environment. The sensors can indicate which cameras can currently view the robot. They might also provide the distance from each camera.

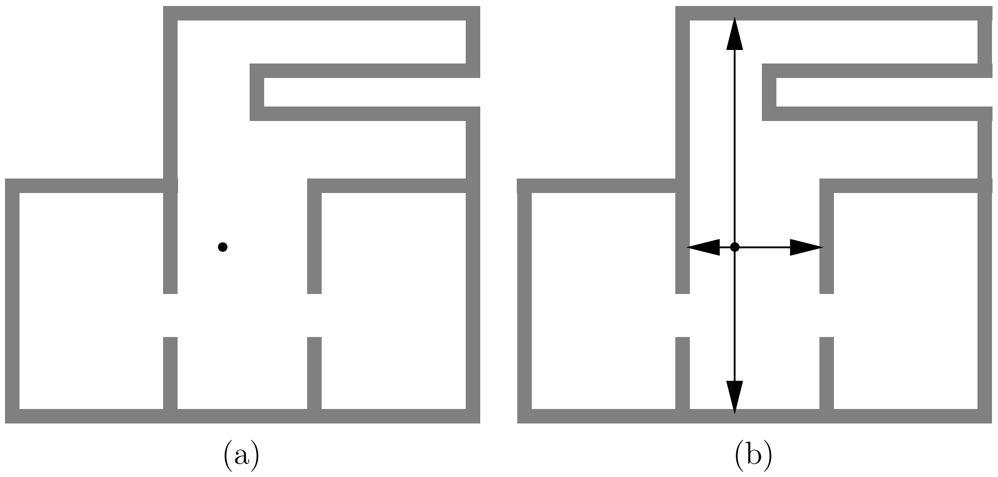


Figure 11.14: (a) A mobile robot is dropped into an unknown environment. (b) Four sonars are used to measure distances to the walls.

Depth-mapping sensors In many robotics applications, the robot may not have a map of the obstacles in the world. In this case, sensing is used to both learn the environment and to determine its position and orientation within the environment. Suppose that a robot is dropped into an environment as shown in Figure 11.14a. For problems such as this, the state represents both the position of the robot and the obstacles themselves. This situation is explained in further detail in Section 12.3. Here, some sensor models for problems of this type are given. These are related to the boundary and proximity sensors of Figure 11.12, but they yield more information when the robot is not at the boundary.

One of the oldest sensors used in mobile robotics is an acoustic *sonar*, which emits a high-frequency sound wave in a specific direction and measures the time that it takes for the wave to reflect off a wall and return to the sonar (often the sonar serves as both a speaker and a microphone). Based on the speed of sound and the time of flight, the distance to the wall can be estimated. Sometimes, the wave never returns; this can be modeled with nature. Also, errors in the distance estimate can be modeled with nature. In general, the observation space Y for a single sonar is $[0, \infty]$, in which ∞ indicates that the wave did not return. The interpretation of Y could be the time of flight, or it could already be transformed into estimated distance. If there are k sonars, each pointing in a different direction, then $Y = [0, \infty]^k$, which indicates that one reading can be obtained for each sonar.

For example, Figure 11.14b shows four sonars and the distances that they can measure. Each observation therefore yields a point in $\mathbb{R}^4$.

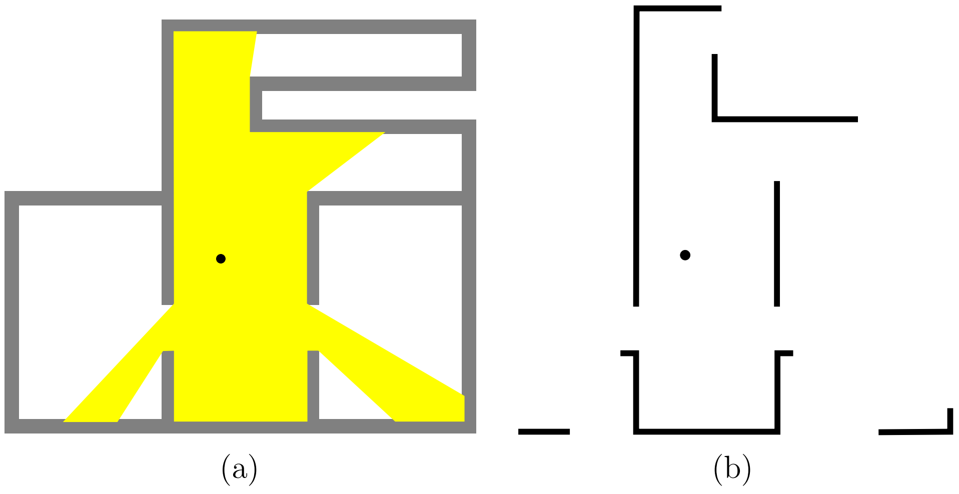


Figure 11.15: A range scanner or visibility sensor is like having a continuum of sonars, even with much higher accuracy. A distance value is provided for each $s \in \mathbb{S}^1$.

Modern laser scanning technology enables very accurate distance measurements with very high angular density. For example, the SICK LMS-200 can obtain a distance measurement for at least every 1/2 degree and sweep the full 360 degrees at least 30 times a second. The measurement accuracy in an indoor environment is often on the order of a few millimeters. Imagine the limiting case, which is like having a continuum of sonars, one for every angle in $\mathbb{S}^1$. This results in a sensor called a *range scanner* or *visibility sensor*, which provides a distance measurement for each $s \in \mathbb{S}^1$, as shown in Figure 11.15.

A weaker sensor can be made by only indicating points in $\mathbb{S}^1$ at which discontinuities (or gaps) occur in the depth scan. Refer to this as a *gap sensor*; an example is shown in Figure 11.16. It might even be the case that only the circular

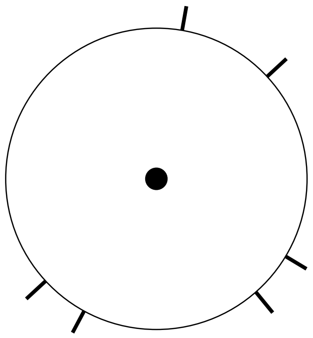


Figure 11.16: A gap sensor indicates only the directions at which discontinuities in depth occur, instead of providing distance information.

ordering of these gaps is given around $\mathbb{S}^1$, without knowing the relative angles between them, or the distance to each gap. A planner based on this sensing model is presented in Section 12.3.4.

Odometry sensors A final category will be given, which provides interesting examples of history-based sensor mappings, as defined for discrete state spaces in Section 11.1.1. Mobile robots often have *odometry sensors*, which indicate how far the robot has traveled, based on the amount that the wheels have turned. Such measurements are often inaccurate because of wheel slippage, surface imperfections, and small modeling errors. For a given state history, $\tilde{x}_t$, a sensor can estimate the total distance traveled. For this model, $Y = [0, \infty)$ and $y = h(\tilde{x}_t)$, in which the argument, $\tilde{x}_t$, to h is the entire state history up to time t. Another way to model odometry is to have a sensor indicate the estimated distance traveled since the last stage. This avoids the dependency on the entire history, but it may be harder to model the resulting errors in distance estimation.

In some literature (e.g., [350]) the action history, $\tilde{u}_k$, is referred to as odometry. This interpretation is appropriate in some applications. For example, each action might correspond to turning the pedals one full revolution on a bicycle. The number of times the pedals have been turned could serve as an odometry reading. Since this information already appears in η_k, it is not modeled in this book as part of the sensing process. For the bicycle example, there might be an odometry sensor that bases its measurements on factors other than the pedal motions. It would be appropriate to model this as a history-based sensor.

Another kind of history-based sensor is to observe a *wall clock* that indicates how much time has passed since the initial stage. This, in combination with other information, such as the speed of the robot, could enable strong inferences to be made about the state.

11.5.2 Simple Projection Examples

This section gives examples of I-spaces for which the sensor mapping is $y = h(x)$ and h is a projection that reveals some of the state variables, while concealing others. The examples all involve continuous time, and the focus is mainly on the nondeterministic I-space $\mathcal{I}_{ndet}$. It is assumed that there are no actions, which means that $U = \emptyset$. Nature actions, $\Theta(x)$, however, will be allowed. Since there are no robot actions and no nature sensing actions, all of the uncertainty arises from the fact that h is a projection and the nature actions that affect the state transition equation are not known. This is a very important and interesting class of problems in itself. The examples can be further complicated by allowing some control from the action set, U; however, the purpose here is to illustrate I-space concepts. Therefore, it will not be necessary.

Example 11.20 (Moving on a Sine Curve) Suppose that the state space is

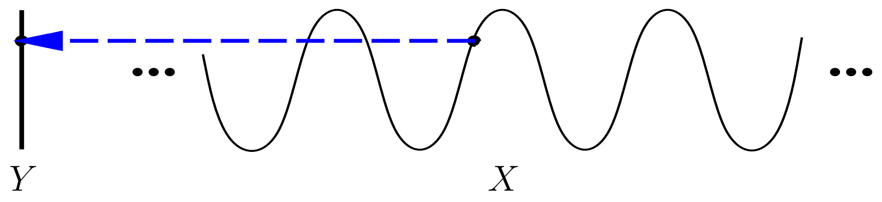


Figure 11.17: The state space is the set of points traced out by a sine curve in $\mathbb{R}^2$.

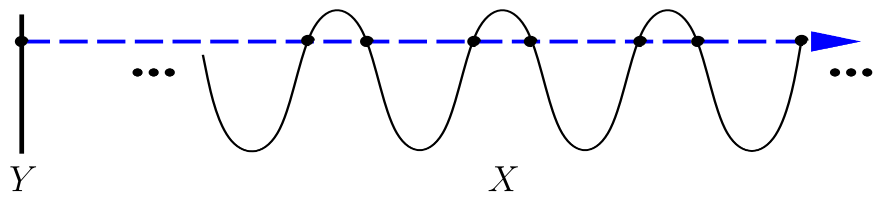


Figure 11.18: The preimage, $H(y)$, of an observation y is a countably infinite set of points along X.

the set of points that lie on the sine curve in the plane:

$$X = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_2 = \sin x_1\}. \tag{11.72}$$

Let $U = \emptyset$, which results in no action history. The observation space is $Y = [-1, 1]$ and the sensor mapping yields $y = h(x) = x_2$, the height of the point on the sine curve, as shown in Figure 11.17.

The nature action space is $\Theta = \{-1, 1\}$, in which -1 means to move at unit speed in the $-x_1$ direction along the sine curve, and 1 means to move at unit speed in the x_1 direction along the curve. Thus, for some nature action history $\tilde{\theta}_t$, a state trajectory $\tilde{x}_t$ that moves the point along the curve can be determined by integration.

A history I-state takes the form $\eta_t = (X_0, \tilde{y}_t)$, which includes the initial condition $X_0 \subseteq X$ and the observation history $\tilde{y}_t$ up to time t. The nondeterministic I-states are very interesting for this problem. For each observation y, the preimage $H(y)$ is a countably infinite set of points that corresponds to the intersection of X with a horizontal line at height y, as shown in Figure 11.18.

The uncertainty for this problem is always characterized by the number of intersection points that might contain the true state. Suppose that $X_0 = X$. In this case, there is no state trajectory that can reduce the amount of uncertainty. As the point moves along X, the height is always known because of the sensor, but the x_1 coordinate can only be narrowed down to being any of the intersection points.

Suppose instead that $X_0 = \{x_0\}$, in which x_0 is some particular point along X. If y remains within $(0, 1)$ over some any period of time starting at $t = 0$, then $x(t)$ is known because the exact segment of the sine curve that contains the state is known. However, if the point reaches an extremum, which results in $y = 0$ or $y = 1$, then it is not known which way the point will travel. From this point, the

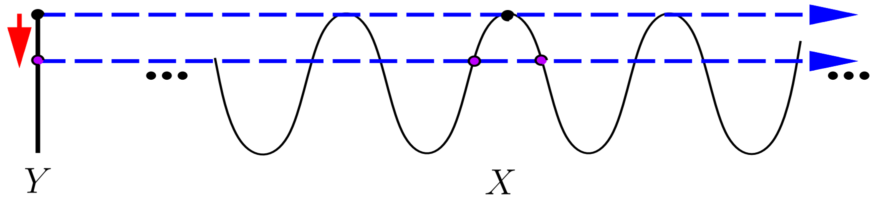


Figure 11.19: A bifurcation occurs when $y = 1$ or $y = -1$ is received. This irreversibly increases the amount of uncertainty in the state.

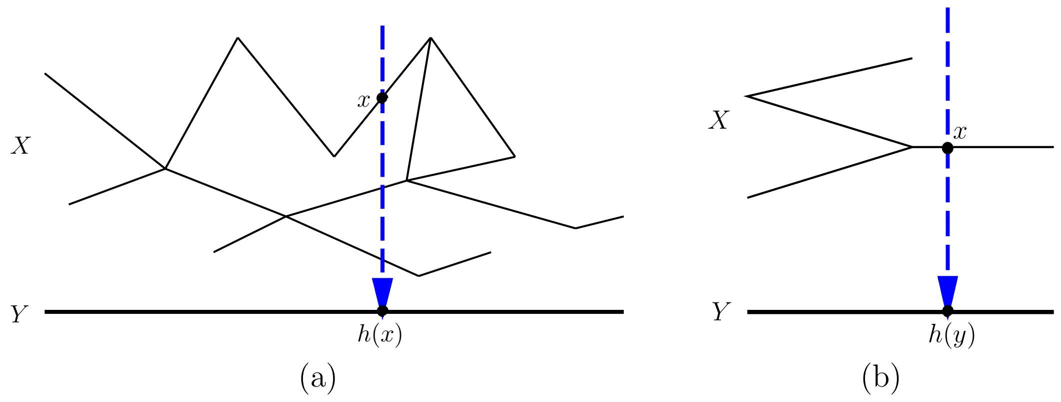


Figure 11.20: (a) Imagine trying to infer the location of a point on a planar graph while observing only a single coordinate. (b) This simple example involves a point moving along a graph that has four edges. When the point is on the rightmost edge, there is no uncertainty; however, uncertainty exists when the point travels along the other edges.

sensor cannot disambiguate moving in the $-x_1$ direction from the x_1 direction. Therefore, the uncertainty grows, as shown in Figure 11.19. After the observation $y = 1$ is obtained, there are two possibilities for the current state, depending on which action was taken by nature when $y = 1$; hence, the nondeterministic I-state contains two states. If the motion continues until $y = -1$, then there will be four states in the nondeterministic I-state. Unfortunately, the uncertainty can only grow in this example. There is no way to use the sensor to reduce the size of the nondeterministic I-states. ■

The previous example can be generalized to observing a single coordinate of a point that moves around in a planar topological graph, as shown in Figure 11.20a. Most of the model remains the same as for Example 11.20, except that the state space is now a graph. The set of nature actions, $\Theta(x)$, needs to be extended so that if x is a vertex of the graph, then there is one input for each incident edge. These are the possible directions along which the point could move.

Example 11.21 (Observing a Point on a Graph) Consider the graph shown

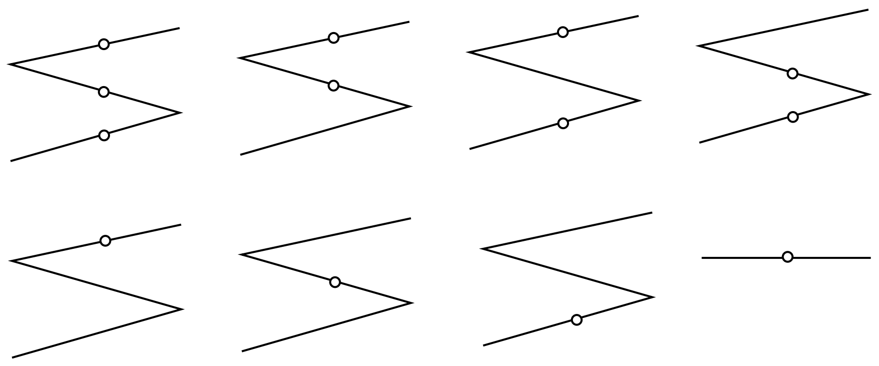


Figure 11.21: Pieces of the nondeterministic I-space $\mathcal{I}_{ndet}$ are obtained by the different possible sets of edges on which the point may lie.

in Figure 11.20b, in which there are four edges.[8] When the point moves on the interior of the rightmost edge of the graph, then the state can be inferred from the sensor. The set $H(y)$ contains a single point on the rightmost edge. If the point moves in the interior of one of the other edges, then $H(y)$ contains three points, one for each edge above y. This leads to seven possible cases for the nondeterministic I-state, as shown in Figure 11.21. Any subset of these edges may be possible for the nondeterministic I-state, except for the empty set.

The eight pieces of $\mathcal{I}_{ndet}$ depicted in Figure 11.21 are connected together in an interesting way. Suppose that the point is on the rightmost edge and moves left. After crossing the vertex, the I-state must be the case shown in the upper right of Figure 11.21, which indicates that the point could be on one of two edges. If the point travels right from one of the I-states of the left edges, then the I-state shown in the bottom right of Figure 11.20 is always reached; however, it is not necessarily possible to return to the same I-state on the left. Thus, in general, there are directional constraints on $\mathcal{I}_{ndet}$. Also, note that from the I-state on the lower left of Figure 11.20, it is impossible to reach the I-state on the lower right by moving straight right. This is because it is known from the structure of the graph that this is impossible. ■

The graph example can be generalized substantially to reflect a wide variety of problems that occur in robotics and other areas. For example, Figure 11.22 shows a polygon in which a point can move. Only one coordinate is observed, and the resulting nondeterministic I-space has layers similar to those obtained for Example 11.21. These ideas can be generalized to any dimension. Interesting models can be constructed using the simple projection sensors, such as a position sensor or compass, from Section 11.5.1. In Section 12.4, such layers will appear in a pursuit-evasion game that uses visibility sensors to find moving targets.

[8]This example was significantly refined after a helpful discussion with Rob Ghrist.

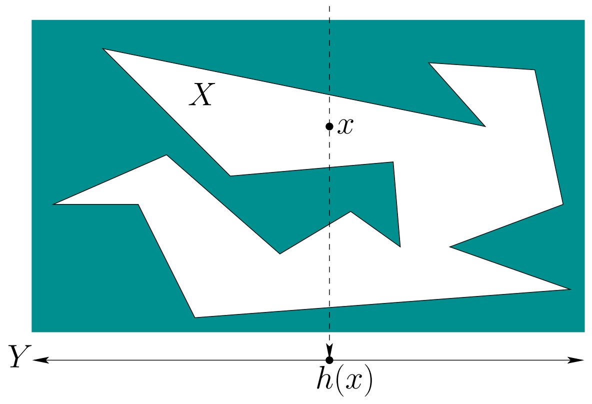


Figure 11.22: The graph can be generalized to a planar region, and layers in the nondeterministic I-space will once again be obtained.

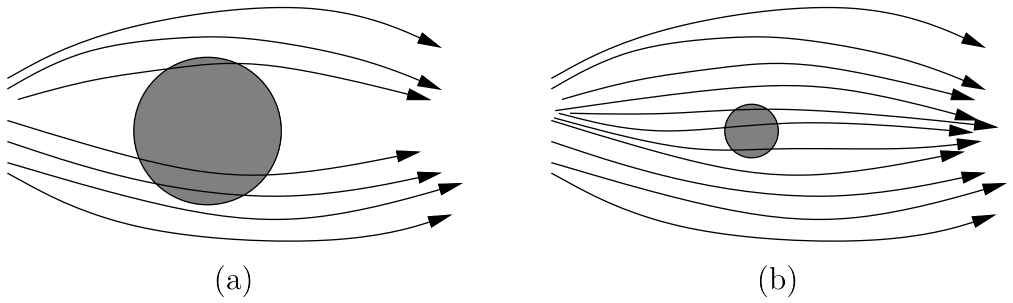


Figure 11.23: (a) It is always possible to determine whether the state trajectory went above or below the designated region. (b) Now the ability to determine whether the trajectory went above or below the hole depends on the particular observations. In some cases, it may not be possible.

11.5.3 Examples with Nature Sensing Actions

This section illustrates the effect of nature sensing actions, but only for the nondeterministic case. General methods for computing probabilistic I-states are covered in Section 11.6.

Example 11.22 (Above or Below Disc?) This example involves continuous time. Suppose that the task is to gather information and determine whether the state trajectory travels above or below some designated region of the state space, as shown in Figure 11.23.

Let $X = \mathbb{R}^2$. Motions are generated by integrating the velocity $(\dot{x}, \dot{y})$, which is expressed as $\dot{x} = \cos(u(t) + \theta(t))$ and $\dot{y} = \sin(u(t) + \theta(t))$. For simplicity, assume $u(t) = 0$ is applied for all time, which is a command to move right. The nature action $\theta(t) \in \Theta = [-\pi/4, \pi/4]$ interferes with the outcome. The robot tries to make progress by moving in the positive x_1 direction; however, the interference of nature makes it difficult to predict the x_2 direction. Without nature, there

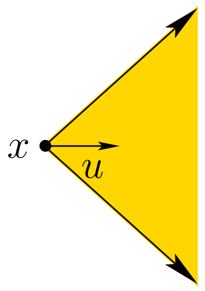


Figure 11.24: Nature interferes with the commanded direction, so that the true state could be anywhere within a circular section.

should be no change in the x_2 coordinate; however, with nature, the error in the x_2 direction could be as much as t, after t seconds have passed. Figure 11.24 illustrates the possible resulting motions.

Sensor observations will be made that alleviate the growing cone of uncertainty; use the sensing model from Figure 11.11, and suppose that the measurement error r is 1. Suppose there is a disc in $\mathbb{R}^2$ of radius larger than 1, as shown in Figure 11.23a. Since the true state is never further than 1 from the measured state, it is always possible to determine whether the state passed above or below the disc. Multiple possible observation histories are shown in Figure 11.23a. The observation history need not even be continuous, but it is drawn that way for convenience. For a disc with radius less than 1, there may exist some observation histories for which it is impossible to determine whether the true state traveled above or below the disc; see Figure 11.23b. For other observation histories, it may still be possible to make the determination; for example, from the uppermost trajectory shown in Figure 11.23b it is known for certain that the true state traveled above the disc. ■

Example 11.23 (A Simple Mobile Robot Model) In this example, suppose that a robot is modeled as a point that moves in $X = \mathbb{R}^2$. The sensing model is the same as in Example 11.22, except that discrete stages are used instead of continuous time. It can be imagined that each stage represents a constant interval of time (e.g., 1 second).

To control the robot, a motion command is given in the form of an action $u_k \in U = \mathbb{S}^1$. Nature interferes with the motions in two ways: 1) The robot tries to travel some distance d, but there is some error $\epsilon_d > 0$, for which the true distance traveled, d', is known satisfy $|d' - d| < \epsilon_d$; and 2) the robot tries to move in a direction u, but there is some error, $\epsilon_u > 0$, for which the true direction u' is known to satisfy $|u - u'| < \epsilon_u$. These two independent errors can be modeled by defining a 2D nature action set, $\Theta(x)$. The transition equation is then defined so that the forward projection $F(x, u)$ is as shown in Figure 11.25.

Some nondeterministic I-states will now be constructed. Suppose that the

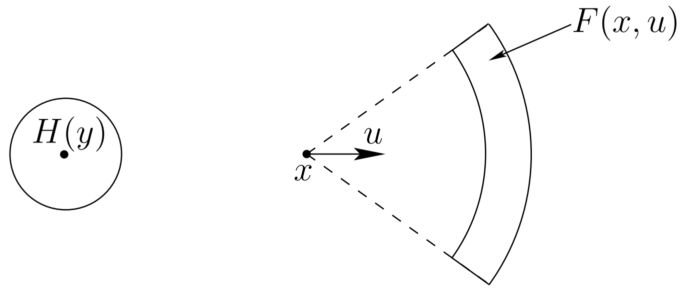


Figure 11.25: A simple mobile robot motion model in which the sensing model is as given in Figure 11.11 and then nature interferes with commanded motions to yield an uncertainty region that is a circular ring.

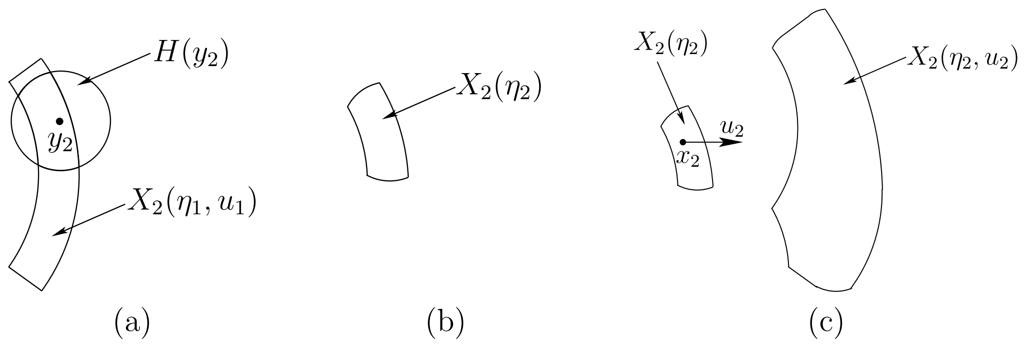


Figure 11.26: (a) Combining information from $X_2(\eta_1, u_1)$ and the observation y_2; (b) the intersection must be taken between $X_2(\eta_1, u_1)$ and $H(y_2)$. (c) The action u_2 leads to a complicated nondeterministic I-state that is the union of $F(x_2, u_2)$ over all $x_2 \in X_2(\eta_2)$.

initial state x_1 is known, and history I-states take the form

$$\eta_k = (x_1, u_1, \ldots, u_{k-1}, y_1, \ldots, y_k). \tag{11.73}$$

The first sensor observation, y_1, is useless because the initial state is known. Equation (11.29) is applied to yield $H(y_1) \cap \{x_1\} = \{x_1\}$. Suppose that the action $u_1 = 0$ is applied, indicating that the robot should move horizontally to the right. Equation (11.30) is applied to yield $X_2(\eta_1, u_1)$, which looks identical to the $F(x, u)$ shown in Figure 11.25. Suppose that an observation y_2 is received as shown in Figure 11.26a. Using this, $X_2(\eta_2)$ is computed by taking the intersection of $H(y_2)$ and $X_2(\eta_1, u_1)$, as shown in Figure 11.26b.

The next step is considerably more complicated. Suppose that $u_2 = 0$ and that (11.30) is applied to compute $X_3(\eta_2, u_2)$ from $X_2(\eta_2)$. The shape shown in Figure 11.26c is obtained by taking the union of $F(x_2, u_2)$ for all possible $x_2 \in X_2(\eta_2)$. The resulting shape is composed of circular arcs and straight line segments (see Exercise 13). Once y_3 is obtained, an intersection is taken once again to yield $X_3(\eta_3) = X_3(\eta_2, u_2) \cap H(y_3)$, as shown in Figure 11.27. The process repeats in the

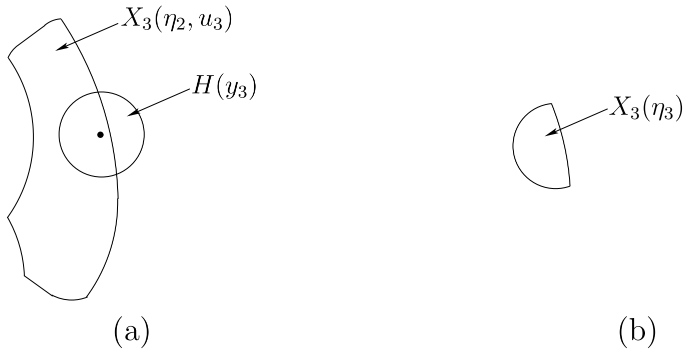


Figure 11.27: After the sensor observation, y_3, the intersection must be taken between $X_3(\eta_2, u_2)$ and $H(y_3)$.

same way for the desired number of stages. The complexity of the region in Figure 11.26c provides motivation for the approximation methods of Section 11.4.3. For example, the nondeterministic I-states could be nicely approximated by ellipsoidal regions. ■

11.5.4 Gaining Information Without Sensors

For some problems, it is remarkable that uncertainty may be reduced without even using sensors. Recall Example 11.17. This is counterintuitive because it seems that information regarding the state can only be gained from sensing. It is possible, however, to also gain information from the knowledge that some actions have been executed and the effect that should have in terms of the state transitions. The example presented in this section is inspired by work on *sensorless manipulation planning* [321, 396], which is covered in more detail in Section 12.5.2. This topic underscores the advantages of reasoning in terms of an I-space, as opposed to requiring that accurate state estimates can be made.

Example 11.24 (Tray Tilting) The state space, $X \subset \mathbb{R}^2$, indicates the position of a ball that rolls on a flat surface, as shown Figure 11.28. The ball is confined to roll within the polygonal region shown in the figure. It can be imagined that the ball rolls in a tray on which several barriers have been glued to confine its motion (try this experiment at home!). If the tray is tilted, it is assumed that the ball rolls in a direction induced by gravity (in the same way that a ball rolls to the bottom of a pinball machine).

The tilt of the tray is considered as an action that can be chosen by the robot. It is assumed that the initial position of the ball (initial state) is unknown and there are no sensors that can be used to estimate the state. The task is to find some tilting motions that are guaranteed to place the ball in the position shown in Figure 11.28, regardless of its initial position.

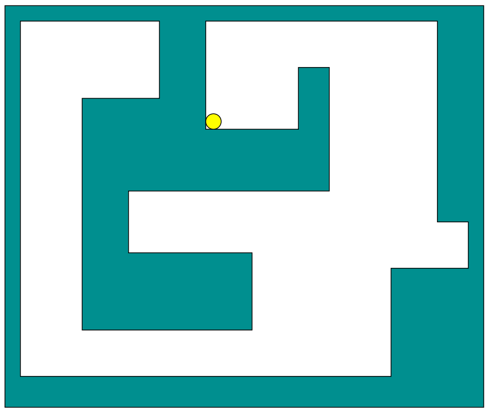


Figure 11.28: A top view of a tray that must be tilted to roll the ball into the desired corner.

The problem could be modeled with continuous time, but this complicates the design. If the tray is tilted in a particular orientation, it is assumed that the ball rolls in a direction, possibly following the boundary, until it comes to rest. This can be considered as a discrete-stage transition: The ball is in some rest state, a tilt action is applied, and a then it enters another rest state. Thus, a discrete-stage state transition equation, $x_{k+1} = f(x_k, u_k)$, is used.

To describe the tilting actions, we can formally pick directions for the upward normal vector to the tray from the upper half of $\mathbb{S}^2$; however, this can be reduced to a one-dimensional set because the steepness of the tilt is not important, as long as the ball rolls to its new equilibrium state. Therefore, the set of actions can be considered as $U = \mathbb{S}^1$, in which a direction $u \in \mathbb{S}^1$ indicates the direction that the ball rolls due to gravity. Before any action is applied, it is assumed that the tray is initially level (its normal is parallel to the direction of gravity). In practice, one should be more careful and model the motion of the tray between a pair of actions; this is neglected here because the example is only for illustrative purposes. This extra level of detail could be achieved by introducing new state variables that indicate the orientation of the tray or by using continuous-time actions. In the latter case, the action is essentially providing the needed state information, which means that the action function would have to be continuous. Here it is simply assumed that a sequence of actions from $\mathbb{S}^1$ is applied.

The initial condition is $X_1 = X$ and the history I-state is

$$\eta_k = (X_1, u_1, u_2, \ldots, u_{k-1}). \tag{11.74}$$

Since there are no observations, the path through the I-space is predictable. Therefore, a plan, π, is simply an action sequence, $\pi = (u_1, u_2, \ldots, u_K)$, for any desired K.

It is surprisingly simple to solve this task by reasoning in terms of nondeterministic I-states, each of which corresponds to a set of possible locations for the ball. A sequence of six actions, as shown in Figure 11.29, is sufficient to guarantee

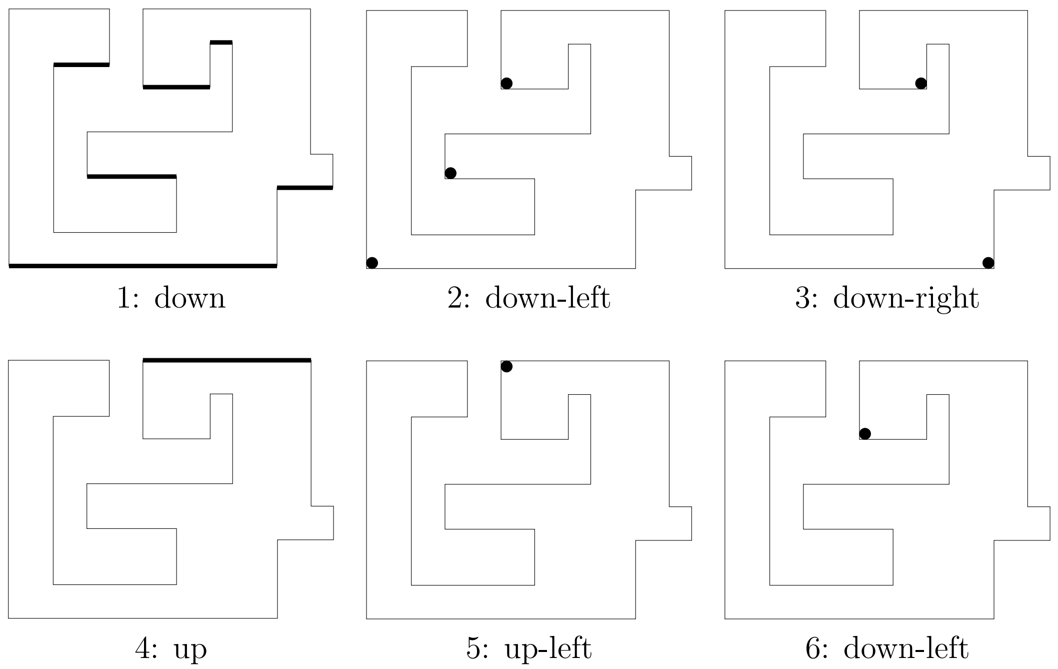


Figure 11.29: A plan is shown that places the ball in the desired location using a sequence of six tilts, regardless of its initial position and in spite of the fact that there are no sensors. The thickened black lines and black dots indicate the possible locations for the ball: the nondeterministic I-states. Under each picture, the direction that the ball rolls due to the action is written.

that the ball will come to rest at the goal position, regardless of its initial position. ■

11.6 Computing Probabilistic Information States

The probabilistic I-states can be quite complicated in practice because each element of $\mathcal{I}_{prob}$ is a probability distribution or density function. Therefore, substantial effort has been invested in developing efficient techniques for computing probabilistic I-states efficiently. This section can be considered as a continuation of the presentations in Sections 11.2.3 (and part of Section 11.4, for the case of continuous state spaces). Section 11.6.1 covers Kalman filtering, which provides elegant computations of probabilistic I-states. It is designed for problems in which the state transitions and sensor mapping are linear, and all acts of nature are modeled by multivariate Gaussian densities. Section 11.6.2 covers a general sampling-based planning approach, which is approximate but applies to a broader class of problems. One of these methods, called *particle filtering*, has become very popular in recent years for mobile robot localization.

11.6.1 Kalman Filtering

This section covers the most successful and widely used example of a derived I-space that dramatically collapses the history I-space. In the special case in which both f and h are linear functions, and $p(\theta)$, $p(\psi)$, and $p(x_1)$ are Gaussian, all probabilistic I-states become Gaussian. This means that the probabilistic I-space, $\mathcal{I}_{prob}$, does not need to represent every conceivable probability density function. The probabilistic I-state is always trapped in the subspace of $\mathcal{I}_{prob}$ that corresponds only to Gaussians. The subspace is denoted as $\mathcal{I}_{gauss}$. This implies that an I-map, $\kappa_{mom} : \mathcal{I}_{prob} \rightarrow \mathcal{I}_{gauss}$, can be applied without any loss of information.

The model is called *linear-Gaussian* (or *LG*). Each Gaussian density on $\mathbb{R}^n$ is fully specified by its n-dimensional mean vector μ and an $n \times n$ symmetric covariance matrix, Σ. Therefore, $\mathcal{I}_{gauss}$ can be considered as a subset of $\mathbb{R}^m$ in which $m = 2n + \binom{n}{2}$. For example, if $X = \mathbb{R}^2$, then $\mathcal{I}_{gauss} \subset \mathbb{R}^5$, because two independent parameters specify the mean and three independent parameters specify the covariance matrix (not four, because of symmetry). It was mentioned in Section 11.4.3 that moment-based approximations can be used in general; however, for an LG model it is important to remember that $\mathcal{I}_{gauss}$ is an *exact* representation of $\mathcal{I}_{prob}$.

In addition to the fact that the $\mathcal{I}_{prob}$ collapses nicely, κ_{mom} is a sufficient I-map, and convenient expressions exist for incrementally updating the derived I-states entirely in terms of the computed means and covariance. This implies that we can work directly with $\mathcal{I}_{gauss}$, without any regard for the original histories or even the general formulas for the probabilistic I-states from Section 11.4.1. The update expressions are given here without the full explanation, which is lengthy but not difficult and can be found in virtually any textbook on stochastic control (e.g., [95, 564]).

For Kalman filtering, all of the required spaces are Euclidean, but they may have different dimensions. Therefore, let $X = \mathbb{R}^n$, $U = \Theta = \mathbb{R}^m$, and $Y = \Psi = \mathbb{R}^r$. Since Kalman filtering relies on linear models, everything can be expressed in terms of matrix transformations. Let A_k, B_k, C_k, G_k, and H_k each denote a matrix with constant real-valued entries and which may or may not be singular. The dimensions of the matrices will be inferred from the equations in which they will appear (the dimensions have to be defined correctly to make the multiplications work out right). The k subscript is used to indicate that a different matrix may be used in each stage. In many applications, the matrices will be the same in each stage, in which case they can be denoted by A, B, C, G, and H. Since Kalman filtering can handle the more general case, the subscripts are included (even though they slightly complicate the expressions).

In general, the state transition equation, $x_{k+1} = f_k(x_k, u_k, \theta_k)$, is defined as

$$x_{k+1} = A_k x_k + B_k u_k + G_k \theta_k, \tag{11.75}$$

in which the matrices A_k, B_k, and G_k are of appropriate dimensions. The notation f_k is used instead of f, because the Kalman filter works even if f is different in every stage.

Example 11.25 (Linear-Gaussian Example) For a simple example of (11.75), suppose $X = \mathbb{R}^3$ and $U = \Theta = \mathbb{R}^2$. A particular instance is

$$x_{k+1} = \begin{pmatrix} 0 & \sqrt{2} & 1 \\ 1 & -1 & 4 \\ 2 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} u_k + \begin{pmatrix} 1 & 1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \theta_k. \tag{11.76}$$

■

The general form of the sensor mapping $y_k = h_k(x_k, \psi_k)$ is

$$y_k = C_k x_k + H_k \psi_k, \tag{11.77}$$

in which the matrices C_k and H_k are of appropriate dimension. Once again, h_k is used instead of h because a different sensor mapping can be used in every stage.

So far the linear part of the model has been given. The next step is to specify the Gaussian part. In each stage, both nature actions θ_k and ψ_k are modeled with zero-mean Gaussians. Thus, each has an associated covariance matrix, denoted by Σ_θ and Σ_ψ, respectively. Using the model given so far and starting with an initial Gaussian density over X, all resulting probabilistic I-states will be Gaussian [564].

Every derived I-state in $\mathcal{I}_{gauss}$ can be represented by a mean and covariance. Let μ_k and Σ_k denote the mean and covariance of $P(x_k|\eta_k)$. The expressions given in the remainder of this section define a derived information transition equation that computes μ_{k+1} and Σ_{k+1}, given μ_k, Σ_k, u_k, and y_{k+1}. The process starts by computing μ_1 and Σ_1 from the initial conditions.

Assume that an initial condition is given that represents a Gaussian density over $\mathbb{R}^n$. Let this be denoted by μ_0, and Σ_0. The first I-state, which incorporates the first observation y_1, is computed as $\mu_1 = \mu_0 + L_1(y_1 - C_1\mu_0)$ and

$$\Sigma_1 = (I - L_1 C_1)\Sigma_0, \tag{11.78}$$

in which I is the identity matrix and

$$L_1 = \Sigma_0 C_1^T \big(C_1 \Sigma_0 C_1^T + H_1 \Sigma_\psi H_1\big)^{-1}. \tag{11.79}$$

Although the expression for L_1 is complicated, note that all matrices have been specified as part of the model. The only unfortunate part is that a matrix inversion is required, which sometimes leads to numerical instability in practice; see [564] or other sources for an alternative formulation that alleviates this problem.

Now that μ_1 and Σ_1 have been expressed, the base case is completed. The next part is to give the iterative updates from stage k to stage $k+1$. Using μ_k, the mean at the next stage is computed as

$$\mu_{k+1} = A_k\mu_k + B_k u_k + L_{k+1}(y_{k+1} - C_{k+1}(A_k\mu_k + B_k u_k)), \tag{11.80}$$

in which L_{k+1} will be defined shortly. The covariance is computed in two steps; one is based on applying u_k, and the other arises from considering y_{k+1}. Thus, after u_k is applied, the covariance becomes

$$\Sigma'_{k+1} = A_k \Sigma_k A_k^T + G_k \Sigma_\theta G_k^T. \tag{11.81}$$

After y_{k+1} is received, the covariance Σ_{k+1} is computed from Σ'_{k+1} as

$$\Sigma_{k+1} = (I - L_{k+1} C_{k+1}) \Sigma'_{k+1}. \tag{11.82}$$

The expression for L_k is

$$L_k = \Sigma'_k C_k^T \big(C_k \Sigma'_k C_k^T + H_k \Sigma_\psi H_k \big)^{-1}. \tag{11.83}$$

To obtain L_{k+1}, substitute $k+1$ for k in (11.83). Note that to compute μ_{k+1} using (11.80), Σ'_{k+1} must first be computed because (11.80) depends on L_{k+1}, which in turn depends on Σ'_{k+1}.

The most common use of the Kalman filter is to provide reliable estimates of the state x_k by using μ_k. It turns out that the optimal expected-cost feedback plan for a cost functional that is a quadratic form can be obtained for LG systems in a closed-from expression; see Section 15.2.2. This model is often called LQG, to reflect the fact that it is linear, quadratic-cost, and Gaussian. The optimal feedback plan can even be expressed directly in terms of μ_k, without requiring Σ_k. This indicates that the I-space may be collapsed down to X; however, the corresponding I-map is not sufficient. The covariances are still needed to compute the means, as is evident from (11.80) and (11.83). Thus, an optimal plan can be specified as $\pi : X \to U$, but the derived I-states in $\mathcal{I}_{gauss}$ need to be represented for the I-map to be sufficient.

The Kalman filter provides a beautiful solution to the class of linear Gaussian models. It is even successfully applied quite often in practice for problems that do not even satisfy these conditions. This is called the *extended Kalman filter*. The success may be explained by recalling that the probabilistic I-space may be approximated by mean and covariance in a second-order moment-based approximation. In general, such an approximation may be inappropriate, but it is nevertheless widely used in practice.

11.6.2 Sampling-Based Approaches

Since probabilistic I-space computations over continuous spaces involve the evaluation of complicated, possibly high-dimensional integrals, there is strong motivation for using sampling-based approaches. If a problem is nonlinear and/or non-Gaussian, such approaches may provide the only practical way to compute probabilistic I-states. Two approaches are considered here: grid-based sampling and particle filtering. One of the most common applications of the techniques described here is mobile robot localization, which is covered in Section 12.2.

A grid-based approach Perhaps the most straightforward way to numerically compute probabilistic I-states is to approximate probability density functions over a grid and use numerical integration to evaluate the integrals in (11.57) and (11.58).

A grid can be used to compute a discrete probability distribution that approximates the continuous probability density function. Consider, for example, using the Sukharev grid shown in Figure 5.5a, or a similar grid adapted to the state space. Consider approximating some probability density function $p(x)$ using a finite set, $S \subset X$. The Voronoi region surrounding each point can be considered as a "bucket" that holds probability mass. A probability is associated with each sample and is defined as the integral of $p(x)$ over the Voronoi region associated with the point. In this way, the samples S and their discrete probability distribution, $P(s)$ for all $s \in S$ approximate $p(x)$ over X. Let $P(s_k)$ denote the probability distribution over S_k, the set of grid samples at stage k.

In the initial step, $P(s)$ is computed from $p(x)$ by numerically evaluating the integrals of $p(x_1)$ over the Voronoi region of each sample. This can alternatively be estimated by drawing random samples from the density $p(x_1)$ and then recording the number of samples that fall into each bucket (Voronoi region). Normalizing the counts for the buckets yields a probability distribution, $P(s_1)$. Buckets that have little or no points can be eliminated from future computations, depending on the desired accuracy. Let S_1 denote the samples for which nonzero probabilities are associated.

Now suppose that $P(s_k|\eta_k)$ has been computed over S_k and the task is to compute $P(s_{k+1}|\eta_{k+1})$ given u_k and y_{k+1}. A discrete approximation, $P(s_{k+1}|s_k, u_k)$, to $p(x_{k+1}|x_k, u_k)$ can be computed using a grid and buckets in the manner described above. At this point the densities needed for (11.57) have been approximated by discrete distributions. In this case, (11.38) can be applied over S_k to obtain a grid-based distribution over S_{k+1} (again, any buckets that do not contain enough probability mass can be discarded). The resulting distribution is $P(s_{k+1}|\eta_k, u_k)$, and the next step is to consider y_{k+1}. Once again, a discrete distribution can be computed; in this case, $p(x_{k+1}|y_{k+1})$ is approximated by $P(s_{k+1}|y_{k+1})$ by using the grid samples. This enables (11.58) to be replaced by the discrete counterpart (11.39), which is applied to the samples. The resulting distribution, $P(s_{k+1}|\eta_{k+1})$, represents the approximate probabilistic I-state.

Particle filtering As mentioned so far, the discrete distributions can be estimated by using samples. In fact, it turns out that the Voronoi regions over the samples do not even need to be carefully considered. One can work directly with a collection of samples drawn randomly from the initial probability density, $p(x_1)$. The general method is referred to as *particle filtering* and has yielded good performance in applications to experimental mobile robotics. Recall Figure 1.7 and see Section 12.2.3.

Let $S \subset X$ denote a finite collection of samples. A probability distribution is defined over S. The collection of samples, together with its probability distribution, is considered as an approximation of a probability density over X. Since S is used to represent probabilistic I-states, let P_k denote the probability distribution over S_k, which is computed at stage k using the history I-state η_k. Thus, at every stage, there is a new sample set, S_k, and probability distribution, P_k.

The general method to compute the probabilistic I-state update proceeds as follows. For some large number, m, of iterations, perform the following:

1. Select a state $x_k \in S_k$ according to the distribution P_k.

2. Generate a new sample, x_{k+1}, for S_{k+1} by generating a single sample according to the density $p(x_{k+1}|x_k, u_k)$.

3. Assign the weight, $w(x_{k+1}) = p(y_{k+1}|x_{k+1})$.

After the m iterations have completed, the weights over S_{k+1} are normalized to obtain a valid probability distribution, P_{k+1}. It turns out that this method provides an approximation that converges to the true probabilistic I-states as m tends to infinity. Other methods exist, which provide faster convergence [536]. One of the main difficulties with using particle filtering is that for some problems it is difficult to ensure that a sufficient concentration of samples exists in the places where they are needed the most. This is a general issue that plagues many sampling-based algorithms, including the motion planning algorithms of Chapter 5.

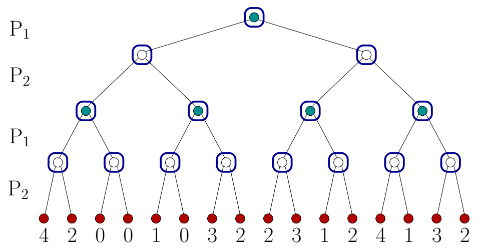
11.7 Information Spaces in Game Theory

This section unifies the sequential game theory concepts from Section 10.5 with the I-space concepts from this chapter. Considerable attention is devoted to the modeling of information in game theory. The problem is complicated by the fact that each player has its own frame of reference, and hence its own I-space. Game solution concepts, such as saddle points or Nash equilibria, depend critically on the information available to each player as it makes it decisions. Paralleling Section 10.5, the current section first covers I-states in game trees, followed by I-states for games on state spaces. The presentation in this section will be confined to the case in which the state space and stages are finite. The formulation of I-spaces extends naturally to countably infinite or continuous state spaces, action spaces, and stages [59].

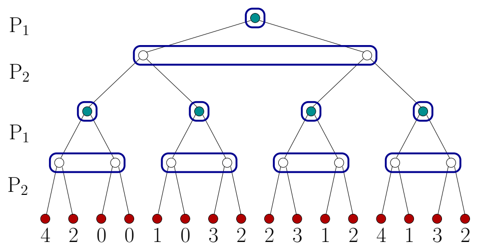
11.7.1 Information States in Game Trees

Recall from Section 10.5.1 that an important part of formulating a sequential game in a game tree is specifying the information model. This was described in Step 4 of Formulation 10.3. Three information models were considered in Section 10.5.1: alternating play, stage-by-stage, and open loop. These and many other information models can be described using I-spaces.

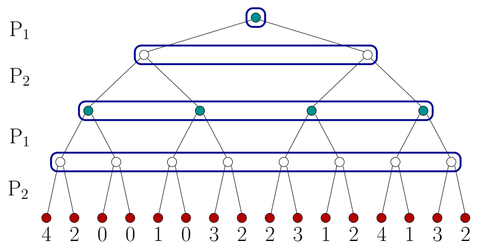
From Section 11.1, it should be clear that an I-space is always defined with respect to a state space. Even though Section 10.5.1 did not formally introduce a



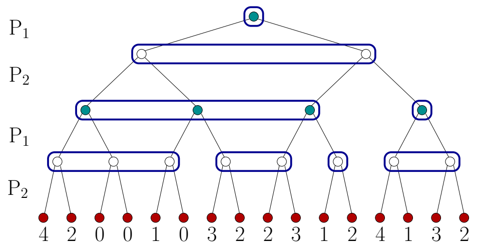
(a) Alternating play



(b) Stage-by-stage



(c) Open loop



(d) Something else

Figure 11.30: Several different information models are illustrated for the game in Figure 10.13.

state space, it is not difficult to define one. Let the state space X be N, the set of all vertices in the game tree. Assume that two players are engaged in a sequential zero-sum game. Using notation from Section 10.5.1, N_1 and N_2 are the decision vertices of P$_1$ and P$_2$, respectively. Consider the nondeterministic I-space $\mathcal{I}_{ndet}$ over N. Let η denote a nondeterministic I-state; thus, each $\eta \in \mathcal{I}_{ndet}$ is a subset of N.

There are now many possible ways in which the players can be confused while making their decisions. For example, if some η contains vertices from both N_1 and N_2, the player does not know whether it is even its turn to make a decision. If η additionally contains some leaf vertices, the game may be finished without a player even being aware of it. Most game tree formulations avoid these strange situations. It is usually assumed that the players at least know when it is their turn to make a decision. It is also usually assumed that they know the stage of the game. This eliminates many sets from $\mathcal{I}_{ndet}$.

While playing the game, each player has its own nondeterministic I-state because the players may hide their decisions from each other. Let η_1 and η_2 denote the nondeterministic I-states for P$_1$ and P$_2$, respectively. For each player, many sets in $\mathcal{I}_{ndet}$ are eliminated. Some are removed to avoid the confusions mentioned above. We also impose the constraint that $\eta_i \subseteq N_i$ for $i = 1$ and $i = 2$. We only care about the I-state of a player when it is that player's turn to make a decision. Thus, the nondeterministic I-state should tell us which decision vertices in N_i are possible as P$_i$ faces a decision. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ represent the nondeterministic I-spaces for P$_1$ and P$_2$, respectively, with all impossible I-states eliminated.

The I-spaces $\mathcal{I}_1$ and $\mathcal{I}_2$ are usually defined directly on the game tree by circling vertices that belong to the same I-state. They form a partition of the vertices in each level of the tree (except the leaves). In fact, $\mathcal{I}_i$ even forms a partition of N_i for each player. Figure 11.30 shows four information models specified in this way for the example in Figure 10.13. The first three correspond directly to the models allowed in Section 10.5.1. In the alternating-play model, each player always knows the decision vertex. This corresponds to a case of perfect state information. In the stage-by-stage model, P_1 always knows the decision vertex; P_2 knows the decision vertex from which P_1 made its last decision, but it does not know which branch was chosen. The open-loop model represents the case that has the poorest information. Only P_1 knows its decision vertex at the beginning of the game. After that, there is no information about the actions chosen. In fact, the players cannot even remember their own previous actions. Figure 11.30d shows an information model that does not fit into any of the three previous ones. In this model, very strange behavior results. If P_1 and P_2 initially choose right branches, then the resulting decision vertex is known; however, if P_2 instead chooses the left branch, then P_1 will forget which action it applied (as if the action of P_2 caused P_1 to have amnesia!). Here is a single-stage example:

Example 11.26 (An Unusual Information Model) Figure 11.31 shows a game that does not fit any of the information models in Section 10.5.1. It is actually a variant of the game considered before in Figure 10.12. The game is a kind of hybrid that partly looks like the alternating-play model and partly like the stage-by-stage model. This particular problem can be solved in the usual way, from the bottom up. A value is computed for each of the nondeterministic I-states, for the level in which P_2 makes a decision. The left I-state has value 5, which corresponds to P_1 choosing 1 and P_2 responding with 3. The right I-state has value 4, which results from the deterministic saddle point in a 2×3 matrix game played between P_1 and P_2. The overall game has a deterministic saddle point in which P_1 chooses 3 and P_2 chooses 3. This results in a value of 4 for the game. ■

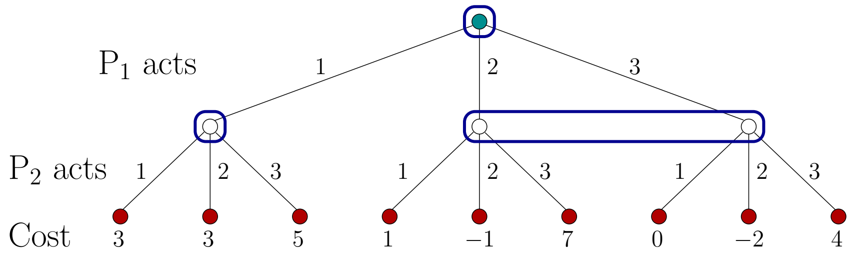


Figure 11.31: A single-stage game that has an information model unlike those in Section 10.5.1.

Plans are now defined directly as functions on the I-spaces. A *(deterministic) plan for* P_1 is defined as a function π_1 on $\mathcal{I}_1$ that yields an action $u \in U(\eta_1)$ for each $\eta_1 \in \mathcal{I}_1$, and $U(\eta_1)$ is the set of actions that can be inferred from the I-state η_1; assume that this set is the same for all decision vertices in η_1. Similarly, a *(deterministic) plan for* P_2 is defined as a function π_2 on $\mathcal{I}_2$ that yields an action $v \in V(\eta_2)$ for each $\eta_2 \in \mathcal{I}_2$.

There are generally two alternative ways to define a randomized plan in terms of I-spaces. The first choice is to define a *globally randomized plan*, which is a probability distribution over the set of all deterministic plans. During execution, this means that an entire deterministic plan will be sampled in advance according to the probability distribution. An alternative is to sample actions as they are needed at each I-state. This is defined as follows. For the randomized case, let $W(\eta_1)$ and $Z(\eta_2)$ denote the sets of all probability distributions over $U(\eta_1)$ and $V(\eta_2)$, respectively. A *locally randomized plan for* P_1 is defined as a function that yields some $w \in W(\eta_1)$ for each $\eta_1 \in \mathcal{I}_1$. Likewise, a *locally randomized plan for* P_2 is a function that maps from $\mathcal{I}_2$ into $Z(\eta_2)$. Locally randomized plans expressed as functions of I-states are often called *behavioral strategies* in game theory literature.

A randomized saddle point on the space of locally randomized plans does not exist for all sequential games [59]. This is unfortunate because this form of randomization seems most natural for the way decisions are made during execution. At least for the stage-by-stage model, a randomized saddle point always exists on the space of locally randomized plans. For the open-loop model, randomized saddle points are only guaranteed to exist using a globally randomized plan (this was actually done in Section 10.5.1). To help understand the problem, suppose that the game tree is a balanced, binary tree with k stages (hence, $2k$ levels). For each player, there are 2^k possible deterministic plans. This means that $2^k - 1$ probability values may be assigned independently (the last one is constrained to force them to sum to 1) to define a globally randomized plan over the space of deterministic plans. Defining a locally randomized plan, there are k I-states for each player, one for each search stage. At each stage, a probability distribution is defined over the action set, which contains only two elements. Thus, each of these distributions has only one independent parameter. A randomized plan is specified in this way using $k - 1$ independent parameters. Since $k - 1$ is much less than $2^k - 1$, there are many globally randomized plans that cannot be expressed as a locally randomized plan. Unfortunately, in some games the locally randomized representation removes the randomized saddle point.

This strange result arises mainly because players can forget information over time. A player with *perfect recall* remembers its own actions and also never forgets any information that it previously knew. It was shown by Kuhn that the space of all globally randomized plans is equivalent to the space of all locally randomized plans if and only if the players have perfect memory [562]. Thus, by sticking to games in which all players have perfect recall, a randomized saddle point always exists in the space locally randomized plans. The result of Kuhn even holds for the more general case of the existence of randomized Nash equilibria on the space of locally randomized plans.

The nondeterministic I-states can be used in game trees that involve more players. Accordingly, deterministic, globally randomized, and locally randomized plans can be defined. The result of Kuhn applies to any number of players, which ensures the existence of a randomized Nash equilibrium on the space of locally randomized strategies if (and only if) the players have perfect recall. It is generally preferable to exploit this fact and decompose the game tree into smaller matrix games, as described in Section 10.5.1. It turns out that the precise condition that allows this is that it must be *ladder-nested* [59]. This means that there are decision vertices, other than the root, at which 1) the player that must make a decision knows it is at that vertex (the nondeterministic I-state is a singleton set), and 2) the nondeterministic I-state will not leave the subtree rooted at that vertex (vertices outside of the subtree cannot be circled when drawing the game tree). In this case, the game tree can be decomposed at these special decision vertices and replaced with the game value(s). Unfortunately, there is still the nuisance of multiple Nash equilibria.

It may seem odd that nondeterministic I-states were defined without being derived from a history I-space. Without much difficulty, it is possible to define a sensing model that leads to the nondeterministic I-states used in this section. In many cases, the I-state can be expressed using only a subset of the action histories. Let $\tilde{u}_k$ and $\tilde{v}_k$ denote the action histories of P_1 and P_2, respectively. The history I-state for the alternating-play model at stage k is $(\tilde{u}_{k-1}, \tilde{v}_{k-1})$ for P_1 and $(\tilde{u}_k, \tilde{v}_{k-1})$ for P_2. The history I-state for the stage-by-stage model is $(\tilde{u}_{k-1}, \tilde{v}_{k-1})$ for both players. The nondeterministic I-states used in this section can be derived from these histories. For other models, such as the one in Figure 11.31, a sensing model is additionally needed because only partial information regarding some actions appears. This leads into the formulation covered in the next section, which involves both sensing models and a state space.

11.7.2 Information Spaces for Games on State Spaces

I-space concepts can also be incorporated into sequential games that are played over state spaces. The resulting formulation naturally extends Formulation 11.1 of Section 11.1 to multiple players. Rather than starting with two players and generalizing later, the full generality of having n players is assumed up front. The focus in this section is primarily on *characterizing* I-spaces for such games, rather than solving them. Solution approaches depend heavily on the particular information models; therefore, they will not be covered here.

As in Section 11.7.1, each player has its own frame of reference and therefore its own I-space. The I-state for each player indicates its information regarding a common game state. This is the same state as introduced in Section 10.5; however, each player may have different observations and may not know the actions of others. Therefore, the I-state is different for each decision maker. In the case of perfect state sensing, these I-spaces all collapse to X.

Suppose that there are n players. As presented in Section 10.5, each player has its own action space, U^i; however, here it is not allowed to depend on x, because the state may generally be unknown. It can depend, however, on the I-state. If nature actions may interfere with the state transition equation, then (10.120) is used (if there are two players); otherwise, (10.121) is used, which leads to predictable future states if the actions of all of the players are given. A single nature action, $\theta \in \Theta(x, u^1, u^2, \ldots, u^n)$, is used to model the effect of nature across all players when uncertainty in prediction exists.

Any of the sensor models from Section 11.1.1 may be defined in the case of multiple players. Each has its own observation space Y^i and sensor mapping h^i. For each player, nature may interfere with observations through nature sensing actions, $\Psi^i(x)$. A state-action sensor mapping appears as $y^i = h^i(x, \psi^i)$; state sensor mappings and history-based sensor mappings may also be defined.

Consider how the game appears to a single player at stage k. What information might be available for making a decision? Each player produces the following in the most general case: 1) an initial condition, η^i_0; 2) an action history, $\tilde{u}^i_{k-1}$; and 3) and an observation history, $\tilde{y}^i_k$. It must be specified whether one player knows the previous actions that have been applied by other players. It might even be possible for one player to receive the observations of other players. If P$_i$ receives all of this information, its history I-state at stage k is

$$\eta^i_k = (\eta^i_0, \tilde{u}^1_{k-1}, \tilde{u}^2_{k-1}, \ldots, \tilde{u}^n_{k-1}, \tilde{y}^1_k, \tilde{y}^2_k, ..., \tilde{y}^n_k). \tag{11.84}$$

In most situations, however, η^i_k only includes a subset of the histories from (11.84). A typical situation is

$$\eta^i_k = (\eta^i_0, \tilde{u}^i_{k-1}, \tilde{y}^i_k), \tag{11.85}$$

which means that P$_i$ knows only its own actions and observations. Another possibility is that all players know all actions that have been applied, but they do not receive the observations of other players. This results in

$$\eta^i_k = (\eta^i_0, \tilde{u}^1_{k-1}, \tilde{u}^2_{k-1}, \ldots, \tilde{u}^n_{k-1}, \tilde{y}^i_k). \tag{11.86}$$

Of course, many special cases may be defined by generalizing many of the examples in this chapter. For example, an intriguing sensorless game may be defined in which the history I-state consists only of actions. This could yield

$$\eta^i_k = (\eta^i_0, \tilde{u}^1_{k-1}, \tilde{u}^2_{k-1}, \ldots, \tilde{u}^n_{k-1}), \tag{11.87}$$

or even a more secretive game in which the actions of other players are not known:

$$\eta^i_k = (\eta^i_0, \tilde{u}^i_{k-1}). \tag{11.88}$$

Once the I-state has been decided upon, a history I-space $\mathcal{I}^i_{hist}$ for each player is defined as the set of all history I-states. In general, I-maps and derived I-spaces can be defined to yield alternative simplifications of each history I-space.

Assuming all spaces are finite, the concepts given so far can be organized into a sequential game formulation that is the imperfect state information counterpart of Formulation 10.4:

Formulation 11.4 (Sequential Game with I-Spaces)

1. A set of n players, P_1, P_2, ..., P_n.
2. A nonempty, finite *state space* X.
3. For each P_i, a finite *action space* U^i. We also allow a more general definition, in which the set of available choices depends on the history I-state; this can be written as $U^i(\eta^i)$.
4. A finite *nature action space* $\Theta(x, u^1, \ldots, u^n)$ for each $x \in X$, and $u^i \in U^i$ for each i such that $1 \leq i \leq m$.
5. A *state transition function* f that produces a state, $f(x, u^1, \ldots, u^n, \theta)$, for every $x \in X$, $\theta \in \Theta(x, u)$, and $u^i \in U^i$ for each i such that $1 \leq i \leq n$.
6. For each P_i, a finite *observation space* Y^i.
7. For each P_i, a finite *nature sensing action space* $\Psi^i(x)$ for each $x \in X$.
8. For each P_i, a *sensor mapping* h^i which produces an observation, $y = h^i(x, \psi^i)$, for each $x \in X$ and $\psi^i \in \Psi^i(x)$. This definition assumes a state-nature sensor mapping. A state sensor mapping or history-based sensor mapping, as defined in Section 11.1.1, may alternatively be used.
9. A set of K *stages*, each denoted by k, which begins at $k = 1$ and ends at $k = K$. Let $F = K + 1$.
10. For each P_i, an *initial condition* η_0^i, which is an element of an *initial condition space* $\mathcal{I}_0^i$.
11. For each P_i, a *history I-space* $\mathcal{I}_{hist}^i$ which is the set of all history I-states, formed from action and observation histories, and may include the histories of other players.
12. For each P_i, let L^i denote a stage-additive cost functional,

$$L^i(\tilde{x}_F, \tilde{u}_K^1, \ldots, \tilde{u}_K^2) = \sum_{k=1}^{K} l(x_k, u_k^1, \ldots, u_k^n) + l_F(x_F). \tag{11.89}$$

Extensions exist for cases in which one or more of the spaces are continuous; see [59]. It is also not difficult to add goal sets and termination conditions and allow the stages to run indefinitely.

An interesting specialization of Formulation 11.4 is when all players have identical cost functions. This is not equivalent to having a single player because the players have different I-states. For example, a task may be for several robots to search for a treasure, but they have limited communication between them. This results in different I-states. They would all like to cooperate, but they are unable

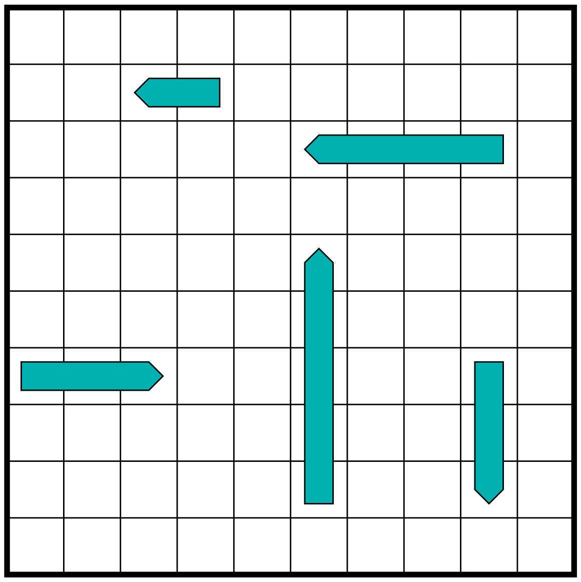


Figure 11.32: In the Battleship game, each player places several ships on a grid. The other player must guess the locations of ships by asking whether a particular tile is occupied.

to do so without knowing the state. Such problems fall under the subject of *team theory* [225, 450, 530].

As for the games considered in Formulation 10.4, each player has its own plan. Since the players do not necessarily know the state, the decisions are based on the I-state. The definitions of a deterministic plan, a globally randomized plan, and a locally randomized plan are essentially the same as in Section 11.7.1. The only difference is that more general I-spaces are defined in the current setting. Various kinds of solution concepts, such as saddle points and Nash equilibria, can be defined for the general game in Formulation 11.4. The existence of locally randomized saddle points and Nash equilibria depends on general on the particular information model [59].

Example 11.27 (Battleship Game) Many interesting I-spaces arise from classical board games. A brief illustration is provided here from Battleship, which is a sequential game under the alternating-turn model. Two players, P_1 and P_2, each having a collection of battleships that it arranges secretly on a 10×10 grid; see Figure 11.32.

A state is the specification of the exact location of all ships on each player's grid. The state space yields the set of all possible ship locations for both players. Each player always knows the location of its own ships. Once they are placed on the grid, they are never allowed to move.

The players take turns guessing a single grid tile, expressed as a row and column, that it suspects contains a ship. The possible observations are "hit" and "miss," depending on whether a ship was at that location. In each turn, a single guess is made, and the players continue taking turns until one player has observed a hit for every tile that was occupied by a ship.

This is an interesting game because once a "hit" is discovered, it is clear that a player should search for other hits in the vicinity because there are going to be several contiguous tiles covered by the same ship. The only problem is that the precise ship position and orientation are unknown. A good player essentially uses the nondeterministic I-state to improve the chances that a hit will occur next. ■

Example 11.28 (The Princess and the Monster) This is a classic example from game theory that involves no sensing. A princess and a monster move about in a 2D environment. A simple motion model is assumed; for example, they take single steps on a grid. The princess is trying not to be discovered by the monster, and the game is played in complete darkness. The game ends when the monster and the princess are on the same grid point. There is no form of feedback that can be used during the game; however, it is possible to construct nondeterministic I-states for the players. For most environments, it is impossible for the monster to be guaranteed to win; however, for some environments it is guaranteed to succeed. This example can be considered as a special kind of *pursuit-evasion game*. A continuous-time pursuit-evasion game that involves I-spaces is covered in Section 12.4. ■

Further Reading

The basic concept of an information space can be traced back to work of Kuhn [562] in the context of game trees. There, the nondeterministic I-state is referred to as an *information set*. After spreading throughout game theory, the concept was also borrowed into stochastic control theory (see [95, 564]). The term *information space* is used extensively in [59] in the context of sequential and differential game theory. For further reading on I-spaces in game theory, see [59, 759]. In artificial intelligence literature, I-states are referred to as *belief states* and are particularly important in the study of POMDPs; see the literature suggested at the end of Chapter 12. The *observability problem* in control theory also results in I-spaces [192, 308, 478, 912], in which *observers* are used to reconstruct the current state from the history I-state. In robotics literature, they have been called *hyperstates* [396] and *knowledge states* [315]. Concepts closely related to I-spaces also appear as *perceptual equivalence classes* in [287] and also appear in the *information invariants* framework of Donald [286]. I-spaces were proposed as a general way to represent planning under sensing uncertainty in [68, 604, 605]. For further reading on sensors in general, see [352].

The Kalman filter is covered in great detail in numerous other texts; see for example, [226, 564, 912]. The original reference is [500]. For more on particle filters, see [45, 293, 350, 536].

Exercises

1. Forward projections in $\mathcal{I}_{ndet}$:

   (a) Starting from a nondeterministic I-state, $X_k(\eta_k)$, and applying an action u_k, derive an expression for the nondeterministic one-stage forward projection by extending the presentation in Section 10.1.2.

   (b) Determine an expression for the two-stage forward projection starting from $X_k(\eta_k)$ and applying u_k and u_{k+1}.

2. Forward projections in $\mathcal{I}_{prob}$:

   (a) Starting from a probabilistic I-state, $P(x_k|\eta_k)$, and applying an action u_k, derive an expression for the probabilistic one-stage forward projection.

   (b) Determine an expression for the two-stage forward projection starting from $P(x_k|\eta_k)$ and applying u_k and u_{k+1}.

3. Determine the strong and weak backprojections on $\mathcal{I}_{hist}$ for a given history I-state, η_k. These should give sets of possible $\eta_{k-1} \in \mathcal{I}_{hist}$.

4. At the end of Section 11.3.2, it was mentioned that an equivalent DFA can be constructed from an NFA.

   (a) Give an explicit DFA that accepts the same set of strings as the NFA in Figure 11.8b.

   (b) Express the problem of determining whether the NFA in Figure 11.8b accepts any strings as a planning problem using Formulation 2.1.

5. This problem involves computing probabilistic I-states for Example 11.14. Let the initial I-state be
$$P(x_1) = [1/3 \;\; 1/3 \;\; 1/3], \tag{11.90}$$
in which the ith entry in the vector indicates $P(x_1 = i+1)$. Let $U = \{0, 1\}$. For each action, a state transition matrix can be specified, which gives the probabilities $P(x_{k+1}|x_k, u_k)$. For $u = 0$, let $P(x_{k+1}|x_k, u_k = 0)$ be
$$\begin{pmatrix} 4/5 & 1/5 & 0 \\ 1/10 & 4/5 & 1/10 \\ 0 & 1/5 & 4/5 \end{pmatrix}. \tag{11.91}$$
The jth entry of the ith row yields $P(x_{k+1} = i \mid x_k = j, u_k = 0)$. For $u = 1$, let $P(x_{k+1} \mid x_k, u_k = 1)$ be
$$\begin{pmatrix} 1/10 & 5/5 & 1/10 \\ 0 & 1/5 & 4/5 \\ 0 & 0 & 1 \end{pmatrix}. \tag{11.92}$$
The sensing model is specified by three vectors:
$$P(y_k|x_k = 0) = [4/5 \;\; 1/5], \tag{11.93}$$
$$P(y_k|x_k = 1) = [1/2 \;\; 1/2], \tag{11.94}$$

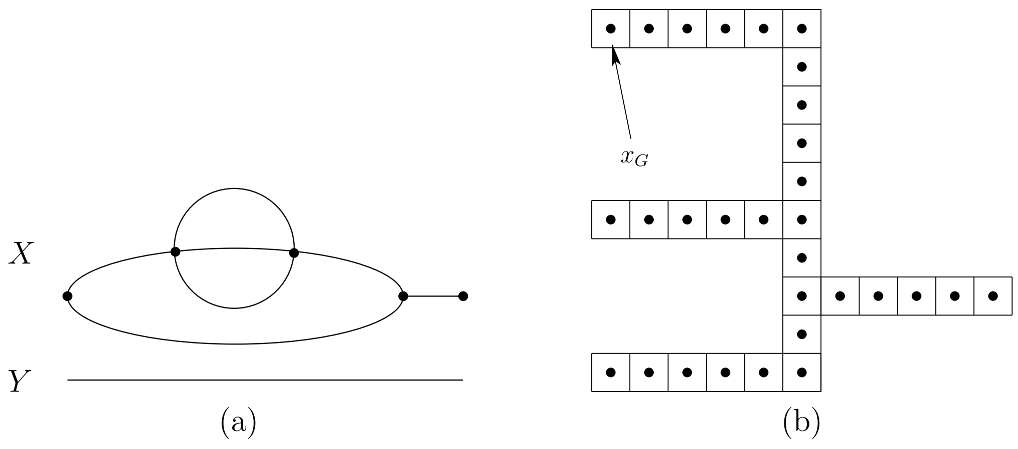


Figure 11.33: (a) A topological graph in which a point moves (note that two vertices are vertically aligned). (b) An exercise that is a variant of Example 11.17.

and

$$P(y_k|x_k = 2) = [1/5 \;\; 4/5], \tag{11.95}$$

in which the ith component yields $P(y_k = i \mid x_k)$. Suppose that $k = 3$ and the history I-state obtained so far is

$$(\eta_0, u_1, u_2, y_1, y_2, y_3) = (\eta_0, 1, 0, 1, 0, 0). \tag{11.96}$$

The task is to compute the probabilistic I-state. Starting from $P(x_1)$, compute the following distributions: $P(x_1|\eta_1)$, $P(x_2|\eta_1, u_1)$, $P(x_2|\eta_2)$, $P(x_3|\eta_2, u_2)$, $P(x_3|\eta_3)$.

6. Explain why it is not possible to reach every nondeterministic I-state from every other one for Example 11.7. Give an example of a nondeterministic I-state that cannot be reached from the initial I-state. Completely characterize the reachability of nondeterministic I-states from all possible initial conditions.

7. In the same spirit as Example 11.21, consider a point moving on the topological graph shown in Figure 11.33. Fully characterize the connectivity of $\mathcal{I}_{ndet}$ (you may exploit symmetries to simplify the answer).

8. Design an I-map for Example 11.17 that is not necessarily sufficient but leads to a solution plan defined over only three derived I-states.

9. Consider the discrete problem in Figure 11.33b, using the same sensing and motion model as in Example 11.17.

   (a) Develop a sufficient I-map and a solution plan that uses as few derived I-states as possible.

   (b) Develop an I-map that is not necessarily sufficient, and a solution plan that uses as few derived I-states as possible.

10. Suppose that there are two I-maps, $\kappa_1 : \mathcal{I}_1 \rightarrow \mathcal{I}_2$ and $\kappa_2 : \mathcal{I}_2 \rightarrow \mathcal{I}_3$, and it is given that κ_1 is sufficient with respect to $\mathcal{I}_1$, and κ_2 is sufficient with respect to $\mathcal{I}_2$. Determine whether the I-map $\kappa_2 \circ \kappa_1$ is sufficient with respect to $\mathcal{I}_1$, and prove your claim.

11. Propose a solution to Example 11.16 that uses fewer nondeterministic I-states.

12. Suppose that a point robot moves in $\mathbb{R}^2$ and receives observations from three homing beacons that are not collinear and originate from known locations. Assume that the robot can calibrate the three observations on $\mathbb{S}^1$.

    (a) Prove that the robot can always recover its position in $\mathbb{R}^2$.

    (b) What can the robot infer if there are only two beacons?

13. Nondeterministic I-state problems:

    (a) Prove that the nondeterministic I-states for Example 11.23 are always a single connected region whose boundary is composed only of circular arcs and line segments.

    (b) Design an algorithm for efficiently computing the nondeterministic I-states from stage to stage.

14. Design an algorithm that takes as input a simply connected rectilinear region (i.e., described by a polygon that has all right angles) and a goal state, and designs a sequence of tray tilts that guarantees the ball will come to rest at the goal. Example 11.24 provides an illustration.

15. Extend the game-theoretic formulation from Section 11.7.2 of history I-spaces to continuous time.

16. Consider the "where did I come from?" problem.

    (a) Derive an expression for $X_1(\eta_k)$.

    (b) Derive an expression for $P(x_1|\eta_k)$.

17. In the game of Example 11.27, could there exist a point in the game at which one player has not yet observed every possible "hit" yet it knows the state of the game (i.e., the exact location of all ships)? Explain.

18. When playing blackjack in casinos, many *card-counting* strategies involve remembering simple statistics of the cards, rather than the entire history of cards seen so far. Define a game of blackjack and card counting as an example of history I-states and an I-map that dramatically reduces the size of the I-space, and an information-feedback plan.

Implementations

19. Implement the Kalman filter for the case of a robot moving in the plane. Show the confidence ellipsoids obtained during execution. Be careful of numerical issues (see [564]).

20. Implement probabilistic I-state computations for a point robot moving in a 2D polygonal environment. Compare the efficiency and accuracy of grid-based approximations to particle filtering.

21. Design and implement an algorithm that uses nondeterministic I-states to play a good game of Battleship, as explained in Example 11.27.