Chapter 9

Multi-Agent Systems

Up to now, for the most part, we have taken the possible worlds to be black boxes, with no structure. The one exception was when we were thinking of a world as being characterized by a collection of random variables. Once there are a number of agents in the picture, it is useful to have even more structure. In this chapter, I present one framework that that gives some useful added structure to the worlds. In addition, it incorporates time in a natural way. This is important for providing a more accurate model of how agents' beliefs change over time. While the framework presented here is certainly not the only way of describing multi-agent systems, it does seem to be a useful approach, as I shall try to demonstrate by example.

9.1 The Basic Framework

Suppose that we want to analyze a multi-agent system. The phrase "system" is intended to be interpreted rather loosely here. Players in a poker game, agents conducting a bargaining session, robots interacting to clean a house, and processes in a computing system can all be viewed as multi-agent systems. The only assumption I shall make here about a system is that, at all times, each of the agents in the can be viewed as being in some *local* or *internal* state. Intuitively, the local state encapsulates all the relevant information to which the agent has access. For example, if we are modeling a poker game, a player's state might consist of the cards he currently holds, the bets made by the other players, any other cards he has seen, and any information he may have about the strategies of the other players (for example, Bob may know that Alice likes to bluff, while Charlie tends to bet conservatively). These states could have further structure (and typically will in most applications of interest). In particular, they can often be characterized by a set of random variables.

It is also useful to view the system as a whole as being in a state. The first thought might be to make the system's state be a tuple of the form $(s_1, \ldots, s_n)$, where s_i is agent i's state. But, in general, more than just the local states of the agents may be relevant to an analysis of the system. In a message-passing system where agents send messages back and forth along communication lines, the messages in transit and the status of each communication line (whether it is up or down) may also be relevant. In a system of sensors observing some terrain, features of the terrain may certainly be relevant. Thus, the system is conceptually divided into two components: the agents and the *environment*, where the environment can be thought of as "everything else that is relevant". In many ways the environment can be viewed as just another agent. A *global state* of a system with n agents or agents is an $(n+1)$-tuple of the form $(s_e, s_1, \ldots, s_n)$, where s_e is the state of the environment and s_i is the local state of agent i.

A global state describes the system at a given point in time. But a system is not a static entity. It is constantly changing over time. A *run* captures the dynamic aspects of a system. Intuitively, a run is a complete description of one possible way that the system's state can evolve over time. Formally, a run is a function from time to global states. For definiteness, I take time to range over the natural numbers. Thus, $r(0)$ describes the initial global state of the system in a possible execution, $r(1)$ describes the next global state, and so on. A pair (r, m) consisting of a run r and time m is called a *point*. If $r(m) = (s_e, s_1, \ldots, s_n)$, then define $r_e(m) = s_e$ and $r_i(m) = s_i$, $i = 1, \ldots, n$; thus, $r_i(m)$ is agent i's local state at the point (r, m).

Typically global states change as a result of actions. I think of an action performed at time m in run r as taking place between time m and time $m + 1$. The point (r, m) describes the situation just before the action takes place and the point $(r, m + 1)$ describes the situation just after the action takes place.

There are many possible executions of a system: there are a number of possible initial states and many things that could happen from each initial state. For example, in a draw poker game, the initial states could describe the possible deals of the hand, where s_i, player i's local state, describes the cards held by player i. For each fixed deal of the cards, there may still be many possible betting sequences, and thus many runs. Formally, a *system* is a nonempty set of runs. Intuitively, these runs describe all the possible sequences of events that could occur in the system. Thus, I am essentially identifying a system with its possible behaviors.

Example 9.1.1 Consider the situation in Example 8.1.1, where Alice chooses a number and then tosses a coin. Suppose that we want to think of this as happening over time: At time 0, Alice chooses the number and at time 1, she tosses the coin. The first step in representing this as a system is to decide what the local states look like. There is no "right" way of modeling the local states. What I am about to describe is one reasonable way of doing it, but there are clearly others.

The environment state will be used to model what actually happens. At time 0, it is $\langle\,\rangle$, the empty sequence, indicating that nothing has yet happened. At time 1, it is either $\langle 0\rangle$ or $\langle 1\rangle$, depending on Alice's choice. At time 2, it is either $\langle 0, H\rangle$, $\langle 0, T\rangle$, $\langle 1, H\rangle$, or $\langle 1, T\rangle$, depending on Alice's choice and the outcome of the coin toss. (Note that Alice's local state in this example is characterized by two random variables, describing Alice's chosen number and the outcome of the coin toss.) It seems reasonable to further assume that Alice knows her choice of number and learns the result of the outcome of the coin toss; thus, I take Alice's local state to be the same as the environment state at all times. However, note that the framework can distinguish what actually happens (which is encoded by the environment state) from Alice's information and beliefs about what has happened (which are encoded in Alice's state).

What about Bob's local state? Suppose that Bob does not know what number Alice chooses (and never finds out), but does learn the outcome of the coin toss. The first thought might then be to take his local states to have the form $\langle\,\rangle$ at time 0 and time 1 (since he does not know what number Alice chose) and either $\langle H\rangle$ or $\langle T\rangle$ at time 2. This would be all right if Bob cannot distinguish between time 0 and time 1; that is, if he cannot tell when Alice chooses the number. But if Bob is aware of the passage of time, then it may be important to keep track of the time in his local state. (We may still choose to ignore the time if we deem it irrelevant to the analysis. Recall that I said that the local state encapsulates all the *relevant* information to which the agent has access. Bob has all sorts of other information that I have chosen not to model—his sister's name, his age, the color of his car, and so on. It is up to the modeler to decide what is relevant here.) In any case, if the time is deemed to be relevant, then at time 1, Bob's state must somehow encode the fact that the time is 1. I do this by taking Bob's state at time 1 to be $\langle tick\rangle$, to denote that one time tick has passed. (Other ways of encoding the time are, of course, also possible.) Note that the time is already implicitly encoded in Alice's state: The time is 1 iff her state is either $\langle 0\rangle$ or $\langle 1\rangle$.

Under this representation of global states, there are seven possible global states:

- $(\langle\,\rangle, \langle\,\rangle, \langle\,\rangle)$ – the initial state,
- two time-1 states of the form $(\langle j\rangle, \langle j\rangle, \langle \mathit{tick}\rangle)$, for $j \in \{0, 1\}$,
- 4 time-2 states of the form $(\langle j, X\rangle, \langle j, X\rangle, \langle \mathit{tick}, X\rangle)$, for $j \in \{0, 1\}$ and $X \in \{H, T\}$.

In this simple case, the environment state determines the global state, but this is not always so.

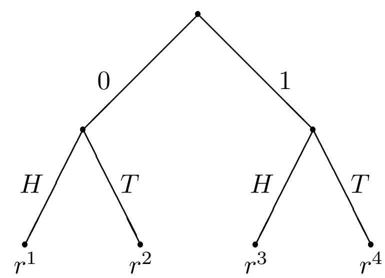

Figure 9.1: Choosing a number then tossing a coin

The system describing this situation has four runs, $r^1, \ldots, r^4$, one for each of the time-2 global states. The runs are perhaps best thought of as being the branches of the *computation tree* described in Figure 9.1. ∎

9.2 Knowledge and Time in Multi-Agent Systems

Consider the multi-agent system described in Example 9.1.1. In this system, Alice knows the outcome of of the coin toss at time 2, but not at time 1; Bob never knows the number that Alice chose. Intuitively, all this is encoded in the agents' local states. I want to relate this intuition to the formal definition of knowledge given in Chapter 2.

The basic idea is that a statement such as "agent i does not know φ" means that, as far as i is concerned, the system could be at a point where φ does not hold. The formalization of "as far as i is concerned, the system could be at a point where φ does not hold" is closely related to the notion of possible worlds in Kripke structures. Think of i's knowledge as being determined by its local state, so that i cannot distinguish between two points in the system in which it has the same local state, and it can distinguish points in which its local state differs. This means that a multi-agent system is essentially a *frame*, in the sense defined in Section 2.2.4. All that prevents it from being a Kripke structure is that there are no primitive propositions and no interpretation π assigning truth values to the primitive propositions. This problem is easily rectified.

Let Φ be a set of primitive propositions; these can be thought of as describing basic facts about the system. Φ might include such facts as "the coin landed heads" or "Alice chose 0". An *interpreted system* $\mathcal{I}$ consists of a pair $(\mathcal{R}, \pi)$, where $\mathcal{R}$ is a system and π associates with each point in $\mathcal{R}$ a truth assignment to the primitive propositions in Φ. For simplicity, I assume that π depends only on the global state, so that if $r(m) = r'(m')$, then $\pi(r, m) = \pi(r', m')$. Notice that Φ and π are not intrinsic to the system $\mathcal{R}$. They constitute additional structure on top of $\mathcal{R}$ that a modeler adds for convenience, to help in analyzing and understanding the system. If $r \in \mathcal{R}$, then I sometimes refer to r as a run in $\mathcal{I}$ and (r, m) as a point in $\mathcal{I}$.

Associate with an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ a Kripke structure $M_{\mathcal{I}} = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ as follows: The set S of states in $M_{\mathcal{I}}$ consists of the points in $\mathcal{I}$. If $s = (s_e, s_1, \ldots, s_n)$ and $s' = (s'_e, s'_1, \ldots, s'_n)$ are two global states in $\mathcal{R}$, then s and s' are *indistinguishable to agent* i, written $s \sim_i s'$, if i has the same state in both s and s', i.e., if $s_i = s'_i$. The indistinguishability relation $\sim_i$ can be extended to points. Two points (r, m) and (r', m') are *indistinguishable to* i, written $(r, m) \sim_i (r', m')$, if $r(m) \sim_i r'(m')$ (or, equivalently, if $r_i(m) = r'_i(m')$). Clearly $\sim_i$ is an equivalence relation on points; take the equivalence relation $\mathcal{K}_i$ of $M_{\mathcal{I}}$ to be $\sim_i$. Thus, $\mathcal{K}_i(r, m) = \{(r', m') : r_i(m) = r'_i(m')\}$, the set of points indistinguishable by i from (r, m).

The satisfaction relation $\models$ in an interpreted system $\mathcal{I}$ can now be defined by reducing it to the definition given of Chapter 2 for Kripke structures, applied to $M_{\mathcal{I}}$. Thus, $(\mathcal{I}, r, m) \models \varphi$ exactly if $(M_{\mathcal{I}}, s) \models \varphi$, where $s = (r, m)$. For example,

$$(\mathcal{I}, r, m) \models p \text{ (for } p \in \Phi\text{) iff } \pi(r, m)(p) = \mathbf{true},$$
$$(\mathcal{I}, r, m) \models K_i\varphi \text{ iff } (\mathcal{I}, r', m') \models \varphi \text{ for all } (r', m') \in \mathcal{K}_i(r, m).$$

This discussion shows that an interpreted system can be viewed as a special Kripke structure, whose worlds are global states. However, worlds in a Kripke structure do not in general have any special structure; global states do. That structure makes explicit exactly what information the agents have and what the temporal relationship is between worlds. The definition of knowledge in interpreted systems exploits one aspect of this structure—the fact that an agent's local state encapsulates his information. The fact that time is modeled explicitly in systems also allows for straightforward reasoning about time.

To reason about time, we need a language. There may be many things we want to say about temporal relationships, such as "φ happened before ψ", "φ happened immediately after ψ", or "φ holds throughout the run". The temporal reasoning we do in the examples in this book is relatively unsophisticated, so I consider a language with just two *temporal operators*, which are new modal operators for talking about time: $\Box$ ("always") and $\bigcirc$ ("next time"). Intuitively, $\Box\varphi$ is true if φ is true now and at all later points and $\bigcirc\varphi$ is true if φ is true at the next step. More formally, in interpreted systems,

$$\begin{array}{lll}(\mathcal{I}, r, m) \models \Box\varphi & \text{iff} & (\mathcal{I}, r, m') \models \varphi \text{ for all } m' \geq m,\\ (\mathcal{I}, r, m) \models \bigcirc\varphi & \text{iff} & (\mathcal{I}, r, m+1) \models \varphi.\end{array}$$

Note that the interpretation of $\bigcirc\varphi$ as "φ holds at the next step" makes sense because time is discrete; $\Box\varphi$ make perfect sense even for continuous notions of time. Define $\Diamond\varphi$ to be an abbreviation for $\neg\Box\neg\varphi$. It is easy to check that $(\mathcal{I}, r, m) \models \Diamond\varphi$ iff $(\mathcal{I}, r, m') \models \varphi$ for some $m' \geq m$ (Exercise 9.1).

Example 9.2.1 I now construct an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ based on the system $\mathcal{R}$ of Example 9.1.1. Just as with the analysis of Example 8.1.1, I take the primitive propositions to be c_0, c_1, h, and t. (I am ignoring the action **a**, since it plays no role in this discussion.) The interpretation π is defined so that c_0 is true at points where the environment state has the form $\langle 0, \ldots \rangle$ and c_1 is true at points where the environment state has the form $\langle 1, \ldots \rangle$. At points of the form $(r, 0)$, where the environment state has the form $\langle\,\rangle$, neither c_0 nor c_1 is true. Intuitively, this is because Alice has not yet chosen a number. Similarly, h is true at the two points where the environment state has the form $\langle i, H \rangle$ and t is true at the two points where the environment state has the form $\langle i, T \rangle$. Neither h nor t is true at points of the form $(r, 0)$ and $(r, 1)$; intuitively, this is because the outcome of the coin toss has not yet been decided.

It is now easy to check that, for example,

$$(\mathcal{I}, r^1, 0) \models \neg c_0 \wedge \neg c_1 \wedge \neg h \wedge \neg t \wedge \bigcirc(c_0 \wedge \neg h \wedge \neg t) \wedge \bigcirc\bigcirc(c_0 \wedge h)$$

(Exercise 9.2). ∎

It is easy to also add uncertainty to this framework as well. For definiteness, I use probability throughout this chapter, but everything I say works for other representations of uncertainty as well. Indeed, in Chapter **??**, plausibility is used. To reason about probability, we can use an *interpreted probability system*, which is a tuple $\mathcal{I} = (\mathcal{R}, \pi, \mathcal{PR}_1, \ldots, \mathcal{PR}_n)$, where $(\mathcal{R}, \pi)$ is an interpreted system and $\mathcal{PR}_1, \ldots, \mathcal{PR}_n$ are probability assignments; just as in the case of Kripke structures, the probability assignment $\mathcal{PR}_i$ associates with each point (r, m) a probability space $\mathcal{PR}_i(r, m) = (W_{r,m,i}, \mathcal{F}_{r,m,i}, \mu_{r,m,i})$.

While there is no problem adding probability (or any other method of uncertainty) to the framework in this way, it does not seem completely in keeping with the spirit of the framework to just assign an arbitrary probability space to each point in a system. Of course, constraints such as CONS, SDP, or UNIF could be imposed, just as in Chapter 8, but this does not seem to be enough. It is reasonable to expect, for example, that $\mathcal{PR}_i(r, m+1)$ would somehow incorporate whatever agent i learned at $(r, m+1)$, but otherwise involve minimal changes from $\mathcal{PR}_i(r, m)$. This suggests the use of conditioning—and, indeed, conditioning will essentially be used—but there are some subtleties involved. It may well be that $W_{r,m,i}$ and $W_{r,m+1,i}$ are disjoint sets. In that case, clearly $\mu_{r,m+1,i}$ cannot be the result of conditioning $\mu_{r,m,i}$ on some event. Nevertheless, as we shall see, there is a way of viewing $\mu_{r,m+1,i}$ as arising from $\mu_{r,m,i}$ by conditioning, by thinking in terms of a probability on runs, not points.

9.3 From Probability on Runs . . .

While it is not always straightforward to define a probability space for each agent at every point, it is quite often the case that a probability on the set of runs in a system can be defined with relative ease. Consider the following simple example.

Example 9.3.1 Suppose that Alice tosses first a biased coin (with heads

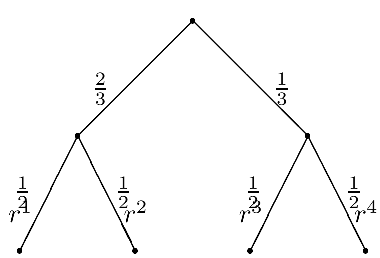

Figure 9.2: Tossing two coins

having probability 2/3) and then a fair coin. There are clearly four runs, depending on the outcome of the coin tosses. Ignoring for now the details of the global states, take the runs to be the paths in the tree drawn in Figure 9.2. If the coin tosses are independent, then the probability of the leftmost run, where Alice gets two heads, is just the product of the probability that Alice gets heads on each of the two tosses, namely, 1/3 for each toss. More generally, the probability of run is just the product of the probabilities labeling the edges that make up the run. ∎

There may not always be an obvious probability on the set of all runs. Consider Example 8.1.1, where Alice chooses a number and then tosses a coin. I argued that, for this example, it is unreasonable to assume that there is a probability on the event that Alice chooses 0. One solution to this problem is to partition the set of runs according to Alice's choice. Using the notation of Figure 9.1, it is easy to put a probability on $\{r_1, r_2\}$, the two runs where Alice chooses 0, and on $\{r_3, r_4\}$, the two runs where Alice chooses 1. In both spaces, it so happens that each run gets probability 1/2.

Although this example, where there are nonprobabilistic choices as well as probabilistic choices, may seem artificial, in fact, such situations are quite common, as the following example shows.

Example 9.3.2 Consider the problem of testing whether a number n is prime. There is a well-known deterministic algorithms for testing primality, which is often taught in elementary school: Test each number between 1 and n to see if it divides n evenly. If one of them does, then n is composite; otherwise, it is prime.

This algorithm can clearly be improved. For example, there is no need to check *all* the numbers up to $n-1$. Testing up to $\sqrt{n}$ suffices: if n is composite, its smaller factor is bound to be less than $\sqrt{n}$. Moreover, there is no need to check any even numbers besides 2. However, even with these improvement, if n is represented in binary (as a string of 0s and 1s), then there are still exponentially many numbers to check in the worst case as a function of the length of n. For example, if n is a 100-digit number, the square root of n is a 50-digit number, so there will still be roughly 2^{50} numbers to check to see if n is prime. This is infeasible on current computers (and likely to remain so for a while). While further improvements are possible, all known *deterministic* algorithms for checking if a number n is prime run, in the worst case, in time that is roughly exponential in the length of n. That means we cannot ever hope to use such a deterministic algorithm to test whether a 500-digit number is prime.

The problem of testing whether a number is prime is not just of academic interest. The well-known RSA algorithm for *public-key cryptography* depends on finding composite numbers that are the product of two large primes. To generate these primes, we need an efficient primality-testing algorithm. (Theorems of number theory assure us that there are roughly $n/\log n$ primes less than n, and that these are well distributed. Thus, if we simply test many 100-digit numbers generated at random, we will quickly find one that is prime, provided the primality test itself is fast.)

Fortunately, there is a very efficient *probabilistic* primality-testing algorithm. It is based on the existence of a predicate P that takes two arguments, n and a (think of n as the number whose primality we are trying to test and a as an arbitrary number between 1 and n) which has the following properties:

1. $P(n, a)$ is either 1 (true) or 0 (false); computing which it is can be done very rapidly (technically, in time polynomial in the length of n and a).

2. If n is composite, then $P(n, a)$ is true for at least $n/2$ possible choices of a in $\{1, \ldots, n-1\}$.

3. If n is prime, then $P(n, a)$ is false for all a.

The existence of such a predicate P can be proved using number-theoretic arguments. There is a straightforward algorithm for testing if n is prime using P. Choose, say, 100 different value for a at random less than n. Then compute $P(n, a)$ for each of these choices of a. If $P(n, a)$ is true for any of these choices of a, then the algorithm outputs "composite"; otherwise it outputs "prime". Property (1) guarantees that this algorithm is very efficient. Property (3) guarantees that if the algorithm outputs "composite", then n is definitely composite. If the algorithm outputs "prime", then there is a chance that n is not prime, but property (2) guarantees that this is very rarely the case: if n is indeed composite, then with high probability (probability at least $1 - (1/2)^{100}$) the algorithm outputs "composite".

Corresponding to this algorithm, there is a set of runs. The first step in these runs in nonprobabilistic: an input n is given (or chosen somehow). The correctness of the algorithm should not depend on how it is chosen. Moreover, in proving the correctness of this algorithm, we do not want to assume that there is a probability measure on the inputs. We want to prove that it is correct for all inputs. But what does "correct" even mean? The arguments above show that if the algorithm outputs "composite", then it is correct in the sense that n is really composite. On the other hand, if the algorithm outputs "prime", then n may not be prime. It might seem natural to say that n is then prime with high probability, but that is not quite right. The input n is either prime or it is not; it does not make sense to say that it is prime with high probability. But it does make sense to say that the algorithm gives the correct answer with high probability. (Moreover, if the algorithm says that n is prime, then it seems reasonable for an agent to ascribe a high subjective degree of belief to the proposition "n is prime".)

The natural way to make this statement precise is to partition the runs of the algorithm into a collection of subsystems, one for each possible input, and prove that the algorithm gives the right answer with high probability in each of these subsystems, where the probability on the runs in each subsystem is generated by the random choices for a. While for a fixed composite input n there may be a few runs where the algorithm incorrectly outputs "prime", in almost all runs it will give the correct output. The spirit of this argument is identical to that used in Example 8.1.1 to argue that the probability that Alice performs action **a** is 1/2, because she performs **a** with probability 1/2 whichever number she chooses in the first step. ∎

In Examples 9.1.1 and 9.3.2, the only nonprobabilistic event happens at the first step. In general, nonprobabilistic events may be interleaved with probabilistic events. However, the system can always be represented so that all nonprobabilistic events happen at the first step. The following example should help make this clear.

Example 9.3.3 Suppose that Alice has a fair coin and Bob has a biased coin, which lands heads with probability 2/3. There will be three coin tosses; Charlie gets to choose who tosses each time. That is, at steps 1, 3, and 5, Charlie chooses who will toss a coin at the next step; at steps 2, 4, and 6, the person who Charlie chose at the previous step tosses his or her coin. Notice this in this example, the probabilistic steps—the coin tosses—alternative with the nonprobabilistic steps—Charlie's choices.

One way of representing this situation is to take the local state of Alice, Bob, Charlie, and the environment to be identical at each step, and just to encode what has happened this far in the run: who Charlie chose and what the outcome of their coin toss was. It is easy to see that under this representation there are 64 ($= 2^6$) runs in the system, since there are two possibilities at each step. In one run, for example, Charlie chooses Alice, who tosses her coin and gets heads, then Charlie chooses Bob, who gets heads, and then Charlie chooses Alice again, who gets tails.

What is the probability of getting three heads on the three coin tosses? That depends, of course, on who Charlie chose at each step; we are not given probabilities for these choices. Intuitively, however, it should be somewhere between 1/8 (if Alice was chosen all the time) and 8/27 (if Bob was chosen all the time). Can we partition the set of runs as in Example 9.1.1 to make this precise?

When the only nonprobabilistic step occurs at the beginning, it is straightforward to partition the runs according to the outcome of the nonprobabilistic choice. Because the nonprobabilistic steps here are interspersed with the probabilistic steps, it is not so obvious how to do it. One approach to dealing with this problem is to convert this situation to one there is only one nonprobabilistic step, which happens at the beginning. The trick is to ask what Charlie's *strategy* is for deciding who goes next. Charlie may have a very simple strategy like "pick Alice at every step" or "pick Alice, then Bob, then Alice again". However, in general, Charlie's strategy may depend on what has happened thus far in the run (in this case, the outcome of the coin tosses). For example, Charlie's strategy may be the following: "First pick Alice. If she tosses tails, pick Alice again, otherwise pick Bob. If whoever was picked the second time tosses tails, then pick him/her again, otherwise pick the other person." Note that both this strategy and the strategy "pick Alice, then Bob, then Alice again" are consistent with the run described earlier. In general, more than one strategy is consistent with observed behavior.

In any case, notice that once Charlie chooses a strategy, then all his other choices are determined. Fixing a strategy factors out all the nondeterminism. The story in this example can then be captured by a multiagent system where, at the first step, Charlie chooses a strategy and from then on, picks Alice and Bob according to the strategy. In more detail, Charlie's local state now would include his choice of strategy together with what has happened thus far, while, as before, the local state of Alice, Bob, and the environment just describe the observable events (who Charlie chose up to the current time and the outcome of the coin tosses).

It can be shown that Charlie has 2^7 possible strategies (Exercise 9.3(a)), and for each of these strategies, there are eight runs, corresponding to the possible outcomes of the coin tosses. Thus, under this representation, we get a system with $2^7 \cdot 8 = 1024$ runs. These runs can be partitioned according to the initial choice of strategy; there is a natural probability measure on the eight runs that arise from any fixed strategy (Exercise 9.3(b)). With this representation, we can indeed say that the probability of getting three heads is somewhere between $1/8$ and $8/27$, since in the probability space corresponding to each strategy, the probability of the run with three heads is between $1/8$ and $8/27$.

Notice that each of the 64 runs in our original representation corresponds to 16 runs in this representation (Exercise 9.3(c)). Although, in some sense, the new representation can be viewed as "inefficient", it has the advantage of partitioning the system cleanly into subsystems, each of which gives rise to a natural probability space. ∎

As this example shows, in systems where there is possibly an initial nonprobabilistic step, and from then on all steps are probabilistic, we can partition the system into subsystems according to the initial step and then view each subsystem as a probability space in a natural way. This technique also works if, rather than there being an initial nonprobabilistic step, the initial nonprobabilistic choice was encoded into the initial global state. Then we could partition the set of runs according to their initial global state, and put a natural probability measure on each cell in this partition. This gives an essentially equivalent representation.

9.4 . . . To Probability on Points

As the title suggests, in this section I want to show how to go from a probability measure on runs to a probability assignment, which gives a probability measure on points. To do this, it is helpful to have some notation that relates sets of runs to sets of points. If $\mathcal{S}$ is a set of runs and U is a set of points, let $\mathcal{S}(U)$ be the set of runs in $\mathcal{S}$ going through some point in U and let $U(\mathcal{S})$ be the set of points in U that lie on some run in $\mathcal{S}$. That is,

$$\begin{aligned} \mathcal{S}(U) &= \{r \in \mathcal{S} : (r,m) \in U \text{ for some } m\} \text{ and} \\ U(\mathcal{S}) &= \{(r,m) \in U : r \in \mathcal{S}\}. \end{aligned}$$

Given a system $\mathcal{R}$, suppose that for each agent i and run r, there is a probability space $(\mathcal{R}_{r,i}, \mathcal{F}_{r,i}, \mu_{r,i})$, where $\mathcal{R}_{r,i} \subseteq \mathcal{R}$. Further suppose that Moreover, assume that $\mathcal{R}_{r,i}(\mathcal{K}_i(r,m)) \in \mathcal{F}_{r,i}$ and that $\mu_{r,i}(\mathcal{R}_j(\mathcal{K}_i(r,m))) > 0$ for each time m. That is, $\mu_{r,i}$ assigns positive probability to the set of runs in $\mathcal{R}_{r,i}$ compatible with what happens in run r up to time m, as far as agent i is concerned. Then it is straightforward to associate with each point (r,m) and agent i a probability space. Intuitively, this space is the result of conditioning the prior probability (on runs) on agent i's information at (r,m), as encoded by $\mathcal{K}_i(r,m)$. This does not quite work because $\mathcal{K}_i(r,m)$ is a set of points; it does not make sense to condition a probability measure defined on sets of runs on a set of points. This problem can be dealt with by shifting from sets of runs to sets of points.

Let $W_{r,m,i} = \mathcal{K}_i(r,m)(\mathcal{R}_{r,i})$; that is, $W_{r,m,i}$ consists of those points (r',m') indistinguishable from (r,m) such that $r' \in \mathcal{R}_{r,i}$. Let $\mathcal{F}_{r,m,i} = \{\mathcal{K}_i(r,m)(\mathcal{S}) : \mathcal{S} \in \mathcal{F}_{r,i}\}$. That is, the measurable subsets of points are those determined by the measurable sets in $\mathcal{F}_{r,i}$. Think of $\mathcal{K}_i(r,m)(\mathcal{S})$ as the result of "projecting" the runs in $\mathcal{S}$ onto the points in $\mathcal{K}_i(r,m)$. Finally, define $\mu_{r,m,i}$ so that $\mu_{r,m,i}(\mathcal{K}_i(r,m)(\mathcal{S})) = \mu_{r,i}(\mathcal{S}|\mathcal{R}_{r,i}(\mathcal{K}_i(r,m)))$ for all $\mathcal{S} \in \mathcal{F}_j$. (Recall that I assumed that $\mu_{r,i}(\mathcal{R}_{r,i}(\mathcal{K}_i(r,m))) > 0$, so this conditional probability is well defined.) Then define the probability assignment $\mathcal{PR}$ so that $\mathcal{PR}_i(r,m) = (W_{r,m,i}, \mathcal{F}_{r,m,i}, \mu_{r,m,i})$. This process is described in Figure 9.3. The main complications are due to the transition back and forth between entities defined over runs and ones defined over points.

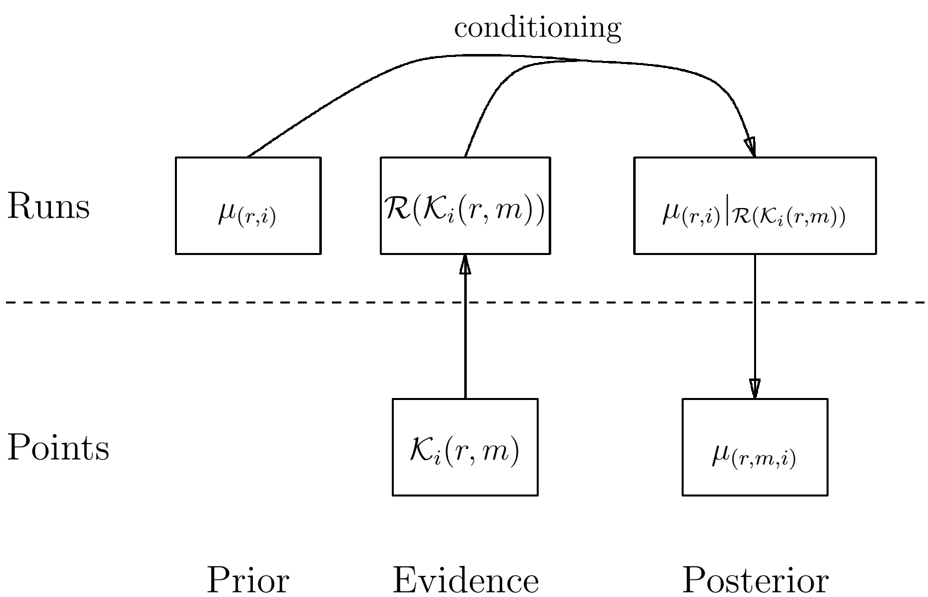

Figure 9.3: Schematic description of the entities involved in the definition of PRIOR.

A system $\mathcal{I} = (\mathcal{R}, \mathcal{PR}_1, \ldots, \mathcal{PR}_n, \pi)$ is said to satisfy the property *PRIOR* if there are probability spaces $(\mathcal{R}_{r,i}, \mathcal{F}_{r,i}, \mu_{r,i})$ such that $\mathcal{PR}_i(r,m)$ is defined by "conditioning" in this way.

A system satisfying PRIOR must satisfy CONS. It does not in general satisfy CONS or UNIF, but there are reasonable assumptions that guarantee that it does. PRIOR becomes more interesting if the prior probabilities $\mu_{r,i}$ are related in some way. The situation that arises most often is one where $\mathcal{R}$ can be partitioned into sets $\mathcal{R}_1, \ldots, \mathcal{R}_m$ and there is a probability space $(\mathcal{R}_j, \mathcal{F}_j, \mu_j)$, $j = 1, \ldots, m$, such that $(\mathcal{R}_{r,i}, \mathcal{F}_{r,i}, \mu_{r,i}) = (\mathcal{R}_j, \mathcal{F}_j, \mu_j)$ for all runs $r \in \mathcal{R}_j$ and all agents i. If there is such a partition, the system is *standard*. In a standard system, the probability spaces $\mathcal{PR}_i(r,m)$ all satisfy UNIF; moreover, if $m = 1$ (so that $\mathcal{R}_1 = \mathcal{R}$), the probability spaces satisfy SDP (Exercise 9.4). If $m = 1$, the system is said to be a *standard SDP system.* A standard SDP system can be viewed as satisfying CP as well; there is a common prior on the runs from which all the probability assignments are derived by conditioning (via the intermediate step of projecting the probability on runs to a probability on points).

Example 9.4.1 Consider again the system consisting of the runs $\{r^1, \ldots, r^4\}$, as described in Example 9.1.1. Split these runs into two subsets $\mathcal{R}_1 = \{r^1, r^2\}$ (the runs where Alice chose 0) and $\mathcal{R}_2 = \{r^3, r^4\}$ (the runs where Alice chose 1). Since Alice uses a fair coin, let μ_i be the probability measure on $\mathcal{R}_i$ that gives each run in $\mathcal{R}_i$ probability 1/2, for $i = 1, 2$. In the standard system corresponding to these prior probabilities, $\mu_{r^1,2,\mathrm{Bob}}$ is the probability measure on $\{(r^1, 2), (r^2, 2)\}$ that assigns each point probability 1/2. This is because, for example, $\mu_{r^1,2,\mathrm{Bob}}(r^2, 2) = \mu_1(r^2)$ $(= 1/2)$, since $\{(r^2, 2)\} = \mathcal{K}_{\mathrm{Bob}}(r^1, 2)(\{r^2\})$, $\mathcal{R}_1(\mathcal{K}_{\mathrm{Bob}}(r^1, 2)) = \{r^1, r^2\} = \mathcal{R}_1$, and $\mu_1(\{r^2\}|\mathcal{R}_1) = \mu_1(r^2)$. Similar arguments show that $\mu_{r^1,2,\mathrm{Alice}}(r^1, 2) = 1$.

The standard system corresponding to splitting the runs of $\mathcal{R}$ into two subsets $\mathcal{R}_1$ and $\mathcal{R}_2$ corresponds to the structure M^2 from Section 8.1. If instead we do not split $\mathcal{R}$ at all, but take the only measurable subsets to be $\emptyset$, $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}$, the resulting standard system corresponds to M^1. Finally, M^3 corresponds to the system obtained by splitting $\mathcal{R}$ into four subsets, $\mathcal{R}'_1, \ldots, \mathcal{R}'_4$, where $\mathcal{R}'_i = \{r_i\}$, and putting the trivial probability measure on each one. ∎

What is the connection between $\mathcal{PR}_i(r, m)$ and $\mathcal{PR}_i(r, m + 1)$ in a system satisfying PRIOR? Since each of these spaces is obtained by conditioning the prior probability on agent i's current information, it seems that $\mathcal{PR}_i(r, m + 1)$ should be, roughly speaking, the result of conditioning $\mathcal{PR}_i(r, m)$ on $\mathcal{K}_i(r, m + 1)$. In general, this is not true.

Example 9.4.2 Consider the system described in Example 9.2.1, where Alice tosses two coins. Assume that although Alice knows the outcome of the first coin toss after she has tossed it, she forgets it after she tosses the second coin. Thus, Alice's state has the form (i, o), where $i \in \{0, 1, 2\}$ is the time and $o \in \{\langle\rangle, \mathit{heads}, \mathit{tails}\}$ describes the outcome of the previous coin toss (at time 0, $o = \langle\rangle$ since there was no coin toss at the previous step). Suppose that in fact Alice tosses two heads (so that run r^1 occurs). In this case, $\mu_{r^1,1,Alice}(r^1, 1) = 1/2$. This seems reasonable; at time 1, after observing heads, Alice assigns probability 1/2 to the point $(r^1, 1)$, where the coin will land heads at the next step. If we were conditioning, after observing heads at the next step, Alice should assign the point $(r^1, 2)$ probability 1. However, $\mu_{r^1,2,Alice}(r^1, 2) = 2/3$ since Alice forgets the outcome of the first coin toss at time 2. ∎

As Example 9.4.2 suggests, a necessary condition for conditioning to be applicable is that agents do not forget, in a sense I now make precise. This observation is closely related to the observation made back in Chapter 4 that conditioning is appropriate only if the agents have perfect recall.

How can we model perfect recall in the systems framework? In this framework, an agent's knowledge is determined by his local state. Intuitively, an agent has perfect recall if his local state is always "growing", by adding the new information he acquires over time. This is was essentially the case in the way we modeled the local states in Example 9.1.1. Local states are not required to grow in this sense, quite intentionally. It is quite possible that information encoded in $r_i(m)$—i's local state at time m in run r—no longer appears in $r_i(m+1)$. Intuitively, this means that agent i has lost or "forgotten" this information. There is a good reason not to make this requirement. There are often scenarios of interest where we want to model the fact that certain information is discarded. In practice, for example, an agent may simply not have enough memory capacity to remember everything he has learned.

Nevertheless, there are many instances where it is natural to model agents as if they do not forget. This means, intuitively, that an agent's local state encodes everything that has happened (from that agent's point of view) thus far in the run. That is, an agent with perfect recall should, essentially, be able to reconstruct his complete local history from his current local state. This observation motivates the following definition.

Let *agent i's local-state sequence at the point* (r, m) be the sequence of local states she has gone through in run r up to time m, without consecutive repetitions. Thus, if from time 0 through time 4 in run r agent i has gone through the sequence $\langle s_i, s_i, s_i', s_i, s_i\rangle$ of local states, where $s_i \neq s_i'$, then her local-state sequence at $(r, 4)$ is $\langle s_i, s_i', s_i\rangle$. Agent i's local-state sequence at a point (r, m) essentially describes what has happened in the run up to time m, from i's point of view.

An agent has perfect recall if her current local state encodes her whole local-state sequence. More formally, *agent i has perfect recall in system $\mathcal{R}$* if at all points (r, m) and (r', m') in $\mathcal{R}$, if $(r, m) \sim_i (r', m')$, then agent i has the same local-state sequence at both (r, m) and (r', m'). Thus, agent i has perfect recall if she "remembers" her local-state sequence at all times. In a system with perfect recall, $r_i(m)$ encodes i's local-state sequence in that, at all points where i's local state is $r_i(m)$, she has the same local-state sequence. A system where agent i has perfect recall is shown in Figure 9.4, where the vertical lines denote runs (with time 0 at the top) and all points that i cannot distinguish are enclosed in the same region.

One might expect that in a system $\mathcal{I}$ where agents have perfect recall, if an agent knows a fact φ at a point (r, m), then she will know φ at all points

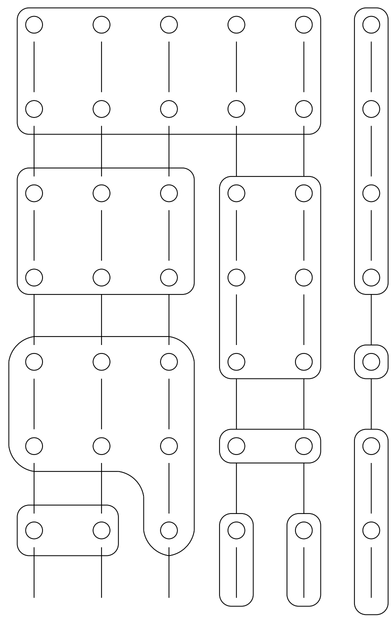

Figure 9.4: A system where agent i has perfect recall

in the future, that is, one might expect that $\mathcal{I} \models K_i\varphi \Rightarrow \Box K_i\varphi$. This is not quite true. One problem arises with statements talking about the situation "now". For example, if φ is the statement "it is currently time 0", then at time 0, an agent i may know φ (say, if she has access to a clock) but agent i will certainly not always know that it is time 0! Another problem comes from knowledge about ignorance. Suppose that φ is the formula $\neg K_1 p$. Then it is not hard to construct a system where agents have perfect recall such that agent 1 initially does not know p, but she later learns p. Thus, $\neg K_1 p$ and hence $K_1 \neg K_1 p$ both hold at time 0, but $K_1 \neg K_1 p$ does not hold at all times in the future, since, by assumption, eventually $K_1 p$ holds (see Exercise 9.5). So, knowledge about ignorance does not necessarily persist in the presence of perfect recall. Nevertheless, the intuition that in systems where agents have perfect recall, once an agent knows φ, then she never forgets φ, is essentially correct. Namely, it is true for all *stable* formulas, where a stable formula is one that, once true, remains true (Exercise 9.6).

How reasonable is the assumption of perfect recall? That, of course, depends on the application. It is easy to see that perfect recall requires every agent to have a number of local states at least as large as the number of distinct local-state sequences she can have in the system. In systems where agents change state rather infrequently, this may not be too unreasonable. On the other hand, in systems where there are frequent state changes or in long-lived systems, perfect recall may require a rather large (possibly unbounded) number of states. This typically makes perfect recall an unreasonable assumption over long periods of time, although it is often a convenient idealization, and may be quite reasonable over short time periods.

In any case, perfect recall gives the desired property:

Proposition 9.4.3 *If $\mathcal{I}$ is a probabilistic interpreted system satisfying PRIOR where agents have perfect recall, then for all points (r,m), agents i, if $\mathcal{S}$ is a measurable subset of runs, then*

$$\mu_{r,m+1,i}(\mathcal{K}_i(r,m+1)(\mathcal{S})) = \mu_{r,m,i}(\mathcal{K}_i(r,m)(\mathcal{S})|\mathcal{K}_i(r,m)(\mathcal{R}(\mathcal{K}_i(r,m+1)))).$$

Proof See Exercise 9.7. ∎

Note that $\mathcal{K}_i(r,m')(\mathcal{S})$ is the set of points in $\mathcal{S}$ that i considers possible at (r,m'). Proposition 9.4.3 says that the probability of $\mathcal{K}_i(r,m+1)(\mathcal{S})$ is determined by computing the conditional probability of $\mathcal{K}_i(r,m)(\mathcal{S})$ given the time-m projection of information acquired by i in step m of run r. The projection of the information acquired in step m is computed by taking the set of runs going through the points in $\mathcal{K}_i(r,m+1)$ and seeing which points in $\mathcal{K}_i(r,m)$ lie on these runs. The assumption of perfect recall is crucial in this proposition; it does not hold in general without it (Exercise 9.8).

Proposition 9.4.3 is perhaps easier to understand if we make one more assumption that is standard in the literature, namely, that agents know the time. This assumption already arose in the discussion of Example 9.1.1, when I assumed that Bob knew at time 1 that it was time 1, and thus knew that Alice had chosen a number. This assumption is also implicit in all the other examples I have considered so far. It is also made (almost always) by game theorists when analyzing games and by linguists when analyzing a conversation. It is not quite as commonplace in computer science—*asynchronous systems*, where agents do not necessarily have any idea of how much time has passed between successive steps, are often considered. Nevertheless, even in the computer science literature, protocols that proceed in rounds, where no agent starts round $m+1$ before all agents finish round m, are often considered.

The assumption that agents know the time is easily captured in the systems framework. $\mathcal{R}$ is a *synchronous system* (i.e., one where agents know what time it is) if for all agents i and points (r,m) and (r',m') in $\mathcal{R}$, if $(r,m) \sim_i (r',m')$, then $m = m'$. Thus, at time m in a synchronous system, agent i knows that it is time m, because it is time m at all the points he considers possible.

It is easy to see that the system of Example 9.1.1 is synchronous, precisely because Bob's local state at time is *tick*. If Bob's state at both time 0 and time 1 would have been $\langle\,\rangle$, then the resulting system would not have been synchronous.

In a synchronous system where agents have perfect recall, the set of runs that an agent considers possible is nonincreasing at each time step. Intuitively, the set partitions into subsets, as the agent acquires more information. The situation is illustrated in Figure **??**.

Proposition 9.4.3 can be stated more simply in the special case of synchronous systems. Given a set U of points, let $U^- = \{(r,m) : (r,m+1) \in U\}$; that is, U^- essentially consists of all the points preceding points in U. It is easy to check that in a synchronous system with perfect recall,

$$\begin{array}{l} \mathcal{K}_i(r,m)(\mathcal{R}(\mathcal{K}_i(r,m+1))) = \mathcal{K}_i(r,m)^- \text{ and} \\ \mathcal{K}_i(r,m)(\mathcal{S}) \cap \mathcal{K}_i(r,m+1)^- = \mathcal{K}_i(r,m+1)(\mathcal{S})^- \end{array}$$

(Exercise 9.9). Thus, the following special case of Proposition 9.4.3 is immediate.

Proposition 9.4.4 *If $\mathcal{I}$ is a synchronous probabilistic interpreted system satisfying PRIOR where agents have perfect recall, then for all points (r,m), agents i, if $\mathcal{S}$ is a measurable subset of runs, then*

$$\mu_{r,m+1,i}(\mathcal{K}_i(r,m+1)(\mathcal{S})) = \mu_{r,m,i}(\mathcal{K}_i(r,m+1)(\mathcal{S})^- | \mathcal{K}_i(r,m+1)^-).$$

9.5 Protocols

Systems provide a useful way of representing situations. But where does the system come from? If we start in some initial global state, changes occur as a result of *actions*. These actions, in turn, are often performed as the result of agents using a *protocol* or *strategy*.

Actions change the global state. Typical actions include tossing heads, going left at an intersection, sending a message. A protocol for agent i is a description of what actions i may take as a function of her local state. For simplicity, I assume here that all actions are deterministic, although protocols may be nondeterministic. Thus, for example, Alice's protocol at some step may be to toss a coin. I view "coin tossing" as consisting of two deterministic actions—tossing heads and tossing tails.

This can be formalized as follows. Fix a set L_i of local states for agent i (intuitively, these are the local states that arise in some system) and a set ACT_i of possible actions that agent i can take. A *protocol P_i for agent i* is a function that associates with every local state in L_i a nonempty subset of actions in ACT_i. Intuitively, $P_i(\ell)$ is the set of actions that agent i may perform in local state ℓ. The fact that P_i is a function of the local state of agent i, and not of the global state, is meant to capture the intuition that an agent's actions can be a function only of the agent's information.

If P_i is *deterministic*, then P_i prescribes a unique action for i at each local state; that is $|P_i(\ell)| = 1$ for each local state in $\ell \in L_i$. For protocols that are not deterministic, rather than just describing what actions agent i may take at a local state, we may also want to associate a measure of likelihood, such as probability, possibility, or plausibility, with each action. A *generalized protocol* associates with each local state some measure of uncertainty over a subset of actions in ACT_i. A *probabilistic protocol for* i is a special case of a generalized protocol, where we associate with each local state a probability measure over a subset of actions in ACT_i.

Example 9.5.1 In the situation in Example 8.1.1, where Alice chooses a number then tosses a fair coin, we can take ACT_A to consist of the actions *choose-0*, *choose-1*, *toss-heads*, and *toss-tails*. Note that since I have insisted that actions be deterministic, choosing a number is not an action; it is viewed as consisting of two actions, choosing 0 and choosing 1. Similarly, tossing a coin is not an action by itself, but is viewed as consisting of tossing heads and tossing tails. Alice can be viewed as using a generalized protocol P_A. Formally,

$$P_A(\langle\,\rangle) = \{\textit{choose-0, choose-1}\} \text{ and}$$
$$P_A(\langle 0\rangle) = P_A(\langle 1\rangle) = (\{\textit{toss-heads, toss-tails}\}, \mu_A),$$

where μ_A gives each of *toss-heads* and *toss-tails* probability 1/2. This protocol captures the intuition that choosing the number is a nonprobabilistic choice, while tossing the coin is probabilistic—heads and tails each occur with probability 1/2. ∎

In all the examples discussed so far, only one agent (Alice) has performed actions. But, in general, agents do not run their protocols in isolation. A *joint protocol* P is just a tuple $(P_1, \ldots, P_n)$ consisting of protocols for each of the agents. A joint protocol P associates with each global state to a set of *joint actions*, that is, a subset of $ACT = ACT_1 \times \cdots \times ACT_n$. Suppose that the protocol for each agent i associates with each local state a measure of uncertainty over the actions performed by agent i at that state. This induces a measure of uncertainty over the joint actions performed at each global state, obtained by treating each of the local protocols as independent. Thus, for example, if Alice and Bob each toss a coin simultaneously, then taking the coin tosses to be independent leads to an obvious measure on $\{\textit{toss-heads, toss-tails}\} \times \{\textit{toss-heads, toss-tails}\}$.

(There is a minor technical point worth observing here. Although I am taking the measures to be independent, this does not mean that there cannot be correlated actions. Rather, it says that if there are, there must be something in the local state that allows this correlation. For example, if, rather than tossing coins independently, Alice and Bob observe Charlie tossing a coin and then base their actions on the outcome of Charlie's coin toss, then the outcome of Charlie's coin toss must be reflected in the local states of Alice and Bob.)

To capture the effect of joint actions, associate with every joint action a function from global states to global states. For example, the joint action consisting of Alice choosing 1 and Bob doing nothing maps the initial global state $(\langle\,\rangle, \langle\,\rangle, \langle\,\rangle)$ to the global state $(\langle 1\rangle, \langle 1\rangle, \langle \mathit{tick}\rangle)$. A joint protocol and a set of initial global states generate a system in a straightforward way. Intuitively, the system consists of all the runs that are obtained by running the joint protocol from one of the initial global states. More formally, say that run r is *consistent with protocol* P if it could have been generated by P, that is, for all m, $r(m{+}1)$ is the result of applying a joint action $\mathbf{a}$ that could have been performed according to protocol P to $r(m)$. (More precisely, there exists a joint action $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$ such that $\mathbf{a}_i \in P_i(r_i(m))$ and $r(m{+}1) = \mathbf{a}(r(m))$.) Given a set *Init* of global states, the system $\mathcal{R}(P, \mathit{Init})$ consists of all the runs consistent with $\mathcal{R}$ that start at some initial global state in *Init*. The system $\mathcal{R}$ *represents* P if $\mathcal{R} = \mathcal{R}(P, \mathit{Init})$ for some set *Init* of global states.

If P is a probabilistic protocol and $\mathcal{R}$ represents P, then there should be a probability on the runs of $\mathcal{R}$. The basic idea for constructing this probability is straightforward: the probability of a run r is essentially the product of the probabilities of the joint actions that were performed in r. While the basic idea is easy to understand (and, indeed, represents precisely the way we handled all the earlier probabilistic examples), it is easy to get lost in all the notation required to make it completely formal. In the rest of this section, I try to steer a middle course between providing a completely formal definition and giving the intuition.

Suppose that the set ACT of joint actions is finite and $\mathcal{R}$ is a system representing a protocol P in which the environment state encodes (among other things) the sequence of joint actions that has been performed thus far. That is, if $r(m+1)$ was obtained by applying the joint action $\mathbf{a}$ to $r(m)$, then $\mathbf{a}$ is encoded in the environment state. I do not formalize this here, since this assumption is only used once (in the next paragraph). It is perhaps best understood by considering Example 9.1.1, where the environment state encodes which number Alice chose and the outcome of the coin toss. Under this assumption, given a probabilistic joint protocol P and an initial global state s, let $\mathcal{R}_s$ consist of all runs consistent with

P that start with state s. Think of these runs as the branches of a tree, as in Figure 9.2. Given a sequence $\vec{\mathbf{a}}$ of k joint actions of the protocol, let $\mathcal{R}_s^{\vec{\mathbf{a}}}$ consist of all runs where the first k actions are precisely those in $\vec{\mathbf{a}}$. Since the environment state encodes the sequence of actions performed, if $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$ are two sequences of joint actions of the same length, then $\mathcal{R}_s^{\vec{\mathbf{a}}}$ and $\mathcal{R}_s^{\vec{\mathbf{b}}}$ are disjoint sets. Moreover, if $\vec{\mathbf{a}}$ is a sequence of length k and $\mathcal{A}$ consists of all sequences of length $k+1$ that extend $\vec{\mathbf{a}}$, then $\mathcal{R}_s^{\vec{\mathbf{a}}} = \cup_{\vec{\mathbf{a}}' \in \mathcal{A}} \mathcal{R}_S^{\vec{\mathbf{a}}'}$. For example, in Figure 9.2, if s is the unique initial state $(\langle\,\rangle, \langle\,\rangle, \langle\,\rangle)$, then $\mathcal{R}_s^{heads}$ consists of the leftmost two runs and $\mathcal{R}_s^{tails}$ consists of the rightmost two runs. Moreover, $\mathcal{R}_s^{heads} = \mathcal{R}_s^{heads,heads} \cup \mathcal{R}_s^{heads,tails}$.

Define a probability μ_s on sets of the form $\mathcal{R}_s^{\vec{\mathbf{a}}}$ by taking $\mu_s(\mathcal{R}_s^{\vec{\mathbf{a}}})$ to be the product of the probabilities of performing each of the joint actions in $\mathbf{a}$. If only finitely many actions are performed in a run r, then the probability of r is indeed the product of the probabilities of the actions that are performed in r. This is exactly how the probabilities of runs in Figure 9.2 were computed. For example, the probability of $\mathcal{R}_s^{heads}$ is $2/3$ and the probability of $\mathcal{R}_s^{heads,tails}$ is $1/3$ $(2/3 \times 1/2)$. Let $\mathcal{F}_s$ consist of all finite unions of sets of the form $\mathcal{R}_s^{\vec{\mathbf{a}}}$. It is easy to check that $\mathcal{F}_s$ is an algebra; moreover, there is a unique way to extend μ_s to a probability on all the sets in $\mathcal{F}_s$. This construction depends in a crucial way on the assumption that the environment state encodes the sequence of joint actions that have been performed so far (Exercise 9.10).

This procedure gives a probability on all the runs starting with a given initial state. If μ_{init} is a probability measure on the set of all initial states, then a single probability measure $\mu_{\mathcal{R}}$ can be defined on all runs by defining $\mu_{\mathcal{R}}(\mathcal{R}_s^{\vec{\mathbf{a}}}) = \mu_{init}(s) \times \mu_s(\mathcal{R}_s^{\vec{\mathbf{a}}})$. Similarly, if the set of initial global states can be partitioned into subsets $I_1, \ldots, I_k$ and μ_i is a probability measure on I_i, $i = 1, \ldots, k$, then the set of runs can be partitioned into k subsets $\mathcal{R}_1, \ldots, \mathcal{R}_k$, where $\mathcal{R}_i$ consists of those runs with initial state in I_i, and this technique can be used to put a probability on the runs in $\mathcal{R}_i$, for $i = 1, \ldots, k$.

This idea extends to situations with protocols of the type discussed in Example 9.3.3, where there is an initial nonprobabilistic step, and probabilistic steps from then on. We can partition the runs according to what happened at the initial nonprobabilistic step, and then put a natural probability on each of these subsets of runs. That is precisely what was done in both Examples 9.1.1 and 9.3.3. In Figure 9.1, we can put a probability on $\{r^1, r^2\}$, the two runs where Alice chooses 0, and on $\{r^3, r^4\}$, the two runs where Alice chooses 1.

Finally, for protocols where probabilistic steps alternate with nonprobabilistic steps, we can reduce to the case where the only nonprobabilistic step happens at the beginning by using strategies along the lines sketched in Example 9.3.3.

9.6 Using Protocols to Specify Situations

The use of protocols helps to clarify what is going on in many examples. In this section, I illustrate this point with three examples, two of which were introduced in Chapter 1—the second-ace puzzle and the Monty Hall puzzle. The first (admittedly somewhat contrived) example formalizes some of the discussion in Chapter 4 regarding when conditioning is applicable.

9.6.1 A Listener-Teller Protocol

Suppose that the world is characterized by n binary random variables, $X_1, \ldots, X_n$. Thus, we can describe the world by an n-tuple of 0s and 1s, $(x_1, \ldots, x_n)$, where x_i is the value of X_i. There are two agents, a *Teller*, who knows what the true values of the random variables are, and a *Listener*, who initially has no idea what they are. There are n primitive propositions, $p_1, \ldots, p_n$, where p_i is true in a world iff $X_i = 1$ at that world. At each step, the Teller gives the listener very limited information: she describes (truthfully) one world that is not the true world. To make this precise, define an *atom* to be a formula of the form $q_1 \wedge \ldots \wedge q_n$, where q_i is either p_i or $\neg p_i$. An atom describes a possible world. For example, if $n = 3$, the atom $p_1 \wedge \neg p_2 \wedge p_3$ describes the world $(1, 0, 1)$. We assume that at each step, the Teller says $\neg\alpha$, where α is an atom (which is not the atom describing the actual world).

We now want to model this situation as a system. Thus, we need to decide what the local states are and what protocol the Teller is following. Since the Teller is presumed to know the actual situation, her state should include the atom describing the actual world. What else should it include? That depends on what the Teller remembers. If she remembers remembers everything, then her local state would have to encode the sequence of formulas she has told the Listener. If she does not remember everything, it might include only some of the formulas she has told the Listener—perhaps even none of them—or only partial information about these formulas, such as the fact that the formulas all included the conjunct $\neg p_1$. The local state could, in principle, also include other features, such as the values of certain variables. The form of the local state affects what protocol the Teller could be using. For example, the Teller can't use a protocol that says "Tell the Listener something new at each step" unless she remembers what she has told the Listener before. And if the local state of the Teller includes the values of certain variables, these values could influence what the Teller says.

For definiteness, I assume that the Teller remembers everything she has said, and the local state includes nothing else. Thus, the Teller's local state has the form $(\alpha, \langle \neg\beta_1, \ldots, \neg\beta_m \rangle)$, where, intuitively, α describes the real world and each each β_i is an atom different from α; the sequence of formulas represent the facts that the Teller has thus far told the Listener. For simplicity, I take the environment's state to be identical to the Teller's state, so that the environment is also keeping track of what the actual world is like and what the Teller said.

Finally, we need to deal with the Listener. What does he remember? I consider just three possibilities here:

1. the Listener remembers everything he was told by the Teller,
2. the Listener remembers only the last thing the Teller said,
3. the Listener remembers only the last thing the Teller said, but is also aware of the time.

If the Teller's local state is $(\alpha, \langle \neg\beta_1, \ldots, \neg\beta_m \rangle)$, then in the first case, the Listener's local state would be $\langle \neg\beta_1, \ldots, \neg\beta_m \rangle)$; in the second case, it would be just $\neg\beta_m$ (and $\langle\,\rangle$ if $m = 0$); in the third case, it would be $(m, \neg\beta_m)$ (the m encodes the time). Since the environment state and the Teller's state are the same in all global states, I denote a global state as $(\cdot, ((\alpha, \langle \neg\beta_1, \ldots, \neg\beta_m \rangle), \ldots)$ rather than $((\alpha, \langle \neg\beta_1, \ldots, \neg\beta_m \rangle), (\alpha, \langle \neg\beta_1, \ldots, \neg\beta_m \rangle), \ldots)$, where exactly what goes into the ellipsis depends on the form of the Listener's state.

Just specifying the form of the global states is not enough; there are also constraints on the allowable sequences of global states. In the first case, a run r has the following form:

- $r(0) = (\cdot, (\alpha, \langle\,\rangle), \langle\,\rangle)$,
- if $r(m) = (\cdot, (\alpha, \langle \neg\beta_1, \ldots, \neg\beta_m \rangle, \langle \neg\beta_1, \ldots, \neg\beta_m \rangle)$, then $r(m+1) = (\cdot, (\alpha, \langle \neg\beta_1, \ldots, \neg\beta_m, \neg\beta_{m+1} \rangle, \langle \neg\beta_1, \ldots, \neg\beta_m, \neg\beta_{m+1} \rangle$ for some atom β_{m+1}.

The first component of the Teller's (and environment's) state—α—remains unchanged throughout the run; this implicitly encodes the assumption that the external world is unchanging. The second component—the sequence $\langle \neg\beta_1, \ldots, \neg\beta_m \rangle$—grows by one at each step; this implicitly encodes the assumption that the Teller tells the Listener something at every step. Neither of these are necessary assumptions; I have just made them here for definiteness. There are analogous constraints on runs if the Listener remembers only the last thing the Teller says.

I have now specified the form of the runs, but this still does not tell us what the system is. That depends on the Teller's protocol. Consider the following two protocols:

- The first is probabilistic. At each step $m < 2^n$, the Teller chooses an atom γ uniformly at random among the $2^n - m$ atoms different from that describing the actual world and the $(m-1)$ it has previously told the Listener, and tells the Listener $\neg\gamma$. After step 2^n, the Teller says nothing.

- The second protocol is deterministic. Order the 2^n worlds from $(0, \ldots, 0)$ to $(1, \ldots, 1)$ in the obvious way. Let β_k be the description of the kth world in this ordering. At step $m < 2^n$, the Teller tells the Listener $\neg\beta_m$ if the actual world is not among the first m worlds; otherwise, the Teller tells the Listener $\neg\beta_{m+1}$. Again, after step 2^n, the Teller says nothing.

If the Listener considers all the initial global states to be equally likely, then it is easy to construct probabilistic interpreted systems representing each of these two protocols, for each of the three assumptions made about the Listener's local state. (Note that these constructions implicitly assume that the Teller's protocol is common knowledge: the Listener knows what the protocol is, the Teller knows that the Listener knows, the Listener knows that the Teller knows that the Listener knows, and so on. The construction does not allow for uncertainty about the Teller's protocol, although the framework can certainly model such uncertainty.) Call the resulting systems $\mathcal{I}_{ij}$, $i \in \{1, 2\}$, $j \in \{1, 2, 3\}$ (where i determines which of the two protocols is used and j determines which of the three ways the Listener's state is being modeled). It is easy to see that $\mathcal{I}_{11}$ and $\mathcal{I}_{21}$ are synchronous, and both the Teller and Listener have perfect recall. The Listener does not have perfect recall in any of the other systems, but both $\mathcal{I}_{13}$ and $\mathcal{I}_{23}$ are synchronous (Exercise 9.11).

In the system $\mathcal{I}_{11}$, where the Teller is using the probabilistic protocol and the Listener has perfect recall, the Listener can be viewed as using conditioning when we project to the level of worlds. That is, if the Listener is told $\neg\beta$ at the mth step, then the set of worlds he considers possible consists of all the worlds he considered possible at time $m-1$, other than that whose atom is β. Moreover, the Listener considers each of these worlds equally likely (Exercise 9.12). Thus, for example, if r is a run in $\mathcal{I}_{11}$ and $m < 2^n$, then $(\mathcal{I}_{11}, r, m) \models \ell_L(\gamma) = 1/(2^n - m)$ if γ is not one of the m atoms the Listener has heard about so far, and $(\mathcal{I}_{11}, r, m) \models \ell_L(\gamma) = 0$ if the Listener has heard that γ is not the atom describing the actual world.

In both $\mathcal{I}_{12}$ and $\mathcal{I}_{13}$, conditioning on worlds is not appropriate. Because the Listener forgets everything he has been told (except for the very last thing), the Listener considers possible all worlds other than the one he was just told was impossible. Thus, if γ is not the atom he heard about at the mth step, then $(\mathcal{I}_{13}, r, m) \models \ell_L(\gamma) = 1/2^n - 1$. Note, nevertheless, that we still use conditioning at the level of runs to construct the Listener's probability space in $\mathcal{I}_{13}$. Equivalently, we are conditioning the Listener's initial probability (all the 2^n worlds are equally likely) on what he knows at time m (which is just the last thing the Teller told him).

In $\mathcal{I}_{21}$, even though the Listener has perfect recall, conditioning on worlds is also not appropriate. That is, if the Listener is told $\neg\beta$ at the mth step of worlds, then the set of worlds he considers possible is not necessarily the set of worlds he considered possible at time m other than that characterized by β. In particular, if β is the description of the $(m+1)$st world in the ordering rather than the mth world, then the Listener will know that the actual world is the mth world in the ordering. Since he has perfect recall, he will also know it from then on.

By way of contrast, in $\mathcal{I}_{22}$, because the Listener has forgotten everything he has heard, he is in the same position as in $\mathcal{I}_{12}$ and and $\mathcal{I}_{13}$. After hearing $\neg\beta$, he considers possible all $2^n - 1$ worlds not other than the one whose atom is β, and considers them all equally likely. On the other hand, in $\mathcal{I}_{23}$, because he knows the time, if the Listener hears $\neg\beta_{m+1}$ at time m in run r (rather than β_m), then he knows, at the point (r, m), that the actual world is characterized by β. However, he forgets this at the next step. (This presumes, of course, that the Listener has sufficient computational power to tell that β_{m+1} characterizes the $(m+1)$st world in the ordering, not the mth. I have implicitly assumed that computational power is not an issue throughout this example and, indeed, in all the other examples in the book. If the agent cannot compute what worlds are compatible and incompatible with his information, then this whole approach is clearly inappropriate.)

While this example is quite contrived, it does show that, in general, if an agent starts with a set of possible worlds and learns something about the actual state of the world at each step, then in order for conditioning to be appropriate, not only must no forgetting be assumed, but also that agents learn nothing from what is said beyond the fact that it is true. This point is clarified somewhat in Exercise 9.13, and may help to explain why conditioning is inappropriate in Example 4.1.2 (see Exercise 9.14).

9.6.2 The Second-Ace Puzzle

Thinking in terms of protocols also helps in understanding what is going on in the second-ace puzzle from Chapter 1. Recall that in this story, Alice is dealt two cards from a deck of four cards, consisting of the ace and deuce of spades and the ace and deuce of hearts. Alice tells Bob she has an ace and then tells him that she has the ace of spades. We want to compute the probability, according to Bob, that Alice has both aces. The calculation done in Chapter 1 gives the answer 1/3. The trouble is that we would also get 1/3 if Alice has said that she had the ace of hearts. It may seem unreasonable that the probability that Alice has two aces drops from 1/5 (Bob's probability that Alice has two aces after she says she has one ace) to 1/3, no matter what ace Alice says she has, once Bob already knows that Alice has an ace.

To analyze this puzzle in the systems framework, we first need to specify the global states and exactly what the protocol is. We can take Alice's local state to consist of the two cards she was dealt together with the sequence of things she has said; Bob's local state consists of the sequence things he has heard. (The environment state plays no role here; we can take it to be the same as Alice's state.)

What about the protocol? One protocol that is consistent with the story is that, initially, Alice is dealt two cards. At step 1, Alice tells Bob whether or not she has an ace. Formally, this means that in a local state where she has an ace, she performs the action of saying "I have an ace"; in a local state where she does not have an ace, she says "I do not have an ace". Then, at step 2, Alice tells Bob she has the ace of spades if she has it and otherwise says she hasn't got it. This protocol is deterministic; there are six possible pairs of cards that Alice could have been dealt; each one determines a unique run. Since we assume a fair deal, each of these runs has probability 1/6. I leave it to the reader to specify this system formally (Exercise 9.15(a)).

In this system, the analysis of Chapter 1 is perfectly correct. When Alice tells Bob that she has an ace in step 1, then at all time-1 points, Bob can eliminate the run where Alice was not dealt an ace, and his conditional probability that Alice has two aces is indeed 1/5, as suggested in Chapter 1. At time 2, Bob can eliminate two more runs (the runs where Alice does not have the ace of spades), and he assesses the probability that Alice has both aces as 1/3 (Exercise 9.15(b)). Notice, however, the concern as to what happens if Alice had told Bob that she has the ace of hearts does not arise. This cannot happen, according to the protocol. All that Alice can say is whether or not she has the ace of spades.

Now consider a different protocol (although still one consistent with the story). Again, at step 1, Alice tells Bob whether or not she has an ace. However, now, at step 2, Alice tells Bob which ace she has if she has an ace (otherwise we can assume she says nothing). This still does not completely specify the protocol. What does Alice tell Bob at step 2 if she has both aces? One possible response is for her to say "I have the ace of hearts" and "I have the ace of spades" with equal probability. This protocol is almost deterministic. The only probabilistic choice occurs if Alice has both aces. With this protocol, there are seven runs: each of the six possible pairs of cards that Alice could have been dealt determines a unique run, with the exception of the case where Alice is dealt two aces, for which there are two possible runs (depending on which ace Alice tells Bob she has). Each run has probability 1/6 except for the two runs where Alice was dealt two aces, which each have probability 1/12. Again, I leave it to the reader to model the resulting system formally (Exercise 9.16(a)); it is sketched in Figure **??**.

It is still the case that at all time-1 points in this system, Bob's conditional probability that Alice has two aces is 1/5. What is the situation at time 2, after Alice says she has the ace of spades? In this case Bob considers three points possible, those in the two runs where Alice has the ace of spades and a deuce, and the point in the run where Alice has both aces and tells Bob she has the ace of spades. Notice, however, that after conditioning, the probability of the point on the run where Alice has both aces is 1/5, while the probability of each of the other two points is 2/5! This is because the probability of the run where Alice holds both aces and tells Bob she has the ace of spades is 1/12, half the probability of the runs where Alice holds only one ace. Thus, Bob's probability that Alice holds both aces at time 3 is 1/5, not 1/3, if this is the protocol. The fact that Alice says she has the ace of spades does not change Bob's assessment of the probability that she has two aces. Similarly, if Alice says that she has the ace of hearts at step 3, the probability that she has two aces remains at 1/5.

Now suppose that Alice's protocol is modified so that, if she has both aces, the probability that she says she has the ace of spades is α. Again, there are seven runs. Each of the runs where Alice does not have two aces has probability 1/6. Of the two runs where Alice has both aces, the one where Alice says she has the ace of spades at step 2 has probability $\alpha/6$; the one where Alice says she the ace of hearts has probability $(1-\alpha)/6$. In this case, a similar analysis shows that the Bob's probability that Alice holds both aces at time 2 is $\alpha/(\alpha+2)$ (Exercise 9.16(b)). In the special case that we started with, $\alpha = 1/2$, so $\alpha/(\alpha+2)$ reduces to 1/5. If $\alpha = 0$, then Alice never says "I have the ace of spades" if she has both aces. In this case, when Bob hears Alice say "I have the ace of spades" at step 2, his probability that Alice has both aces is 0, as we would expect. If $\alpha = 1$, which corresponds to Alice saying "I have the ace of spades" either if she has only the ace of spades or if she has both aces, Bob's probability that Alice has both aces is 1/3.

What if Alice does not choose which ace to say probabilistically, but uses some deterministic protocol which Bob does not know? In this case, all Bob can say is that the probability that Alice holds both aces is either 0 or 1/3, depending on which protocol Alice is following.

9.6.3 The Monty Hall Puzzle

Last but not least, consider the Monty Hall puzzle. The standard argument says that you ought to switch: you lose by switching if the goat is behind the door you've picked; otherwise you gain. Since the goat is behind the door you've picked 1/3 of the time and behind one of the other two doors 2/3 of the time, the probability of gaining by switching is 2/3. Is this argument reasonable? It depends. I'll just sketch the analysis here, since it's so similar to that of the second-ace problem.

What protocol describes the situation? Assume that, initially, Monty places a car behind one door and a goat behind the other two. For simplicity, let's assume that the car is equally likely to be placed behind any door. At step 1, you choose a door. At step 2, Monty opens a door (one with a goat behind it other than the one you chose). Finally, at step 3, you must decide if you'll take what's behind your door or what's behind the other unopened door. Again, to completely specify the protocol, we have to say what Monty does if the door you choose has a car behind it (since then he can open either of the other two doors). Suppose that we take the probability of him opening door j if you choose door i and it has a car behind it to be α_{ij} (where α_{ii} is 0—Monty never opens the door you've chosen). Computations similar to those above show that, if you initially take door i and Monty then opens door j, the probability of you gaining by switching is $1/(\alpha_{ij}+1)$ (Exercise 9.17). If $\alpha_{ij} = 1/2$, then we get 2/3, just as in the standard analysis. If $\alpha_{ij} = 0$, then you are certain that the car can't be behind the door you opened once Monty opens door j. Not surprisingly, in this case, you certainly should switch; you are certain to win. On the other hand, if $\alpha_{ij} = 1$, you are just as likely to win by switching as not. Since, with any choice of α_{ij}, you are at least as likely to win by switching as by not switching, it seems that you ought to switch.

Is that all there is to it? Actually, it's not quite that simple. This analysis was carried out under the assumption that, in step 2, Monty *must* open another door. Is this really Monty's protocol? In the real game show, Monty did not always open another door. Thus, to do a more careful analysis of this puzzle, we must consider what Monty's protocol is for opening a door. For example, if we modify Monty's protocol so that he opens another door only if the door that you chose has a car behind it (in order to tempt you away from the "good" door), then you should certainly not switch! Indeed, once Monty opens the door, you become certain the the door you chose has the car behind it.

The analysis of the three-prisoners puzzle from Example 4.3.1 is, not surprisingly, quite similar to that of the second-ace puzzle and the Monty Hall puzzle. I leave this to the reader (Exercise 9.18).

9.7 Markovian Systems

Although assuming a prior probability over runs helps explain where the probability measure at each point is coming from, runs are still infinite objects and a system may have infinitely many of them. Indeed, even in systems where there are only two global states, there may be uncountably many runs. (Consider a system where a coin is tossed infinitely often and the global state describes the outcome of the last coin toss.) Where is the probability on runs coming from? Is there a compact way of describing and representing it?

In many cases (especially if the system is generated by a probabilistic protocol) it is possible to assign a probability to the transition from one state to another. Consider the two-coin example described in Figure 9.2. Because the first coin has a probability 2/3 of landing heads, the transition from the initial state to the state where it lands heads has probability 2/3; this is denoted by labeling the left edge coming from the root by 2/3 in Figure 9.2. Similarly, the right edge is labeled by 1/3 to denote that the probability of making the transition to the state where the coin lands heads is 1/3. All the other edges are labeled by 1/2, to denote that the probability of transitions resulting from tossing the second (fair) coin are all 1/2. Because the coin tosses are assumed independent and there is a single initial state, the probability of a run in this system can be calculated by multiplying the probabilities labeling its edges. Similar calculations were used to calculate the probabilities of the runs in all the probabilistic systems I have discussed so far. Indeed, for probabilistic protocols, this is how I defined the probability of runs.

These systems are all instances of *Markovian systems*. In Markovian systems, appropriate independence assumptions are made that allow the prior probability on runs to be generated from a probability on state transitions.

To make this precise, let $\mathcal{R}$ be a system whose global states come from a set Σ. For $m = 0, 1, 2, \ldots$, let G_m be a random variable on $\mathcal{R}$ such that $G_m(r) = r(m)$—that is, G_m maps r to the global state in r at time m. (This is a case of a random variable whose range is not the real numbers.) A *time-m* event is a Boolean combination of events of the form $G_i = g$ for $i \leq m$. Of particular interest are time-m events of the form $(G_0 =$

$g_0) \cap \cdots \cap (G_m = g_m)$, which is abbreviated as $[g_0, \ldots, g_m]$; this is the set of all runs in $\mathcal{R}$ with initial prefix $g_0, \ldots, g_m$. Such an event is called an *m-prefix*. It is easy to show that every time-m event is a union of m-prefixes; moreover the set $\mathcal{F}_{pref}$ which consists of the time-m events for all m is an algebra (Exercise 9.19).

Definition 9.7.1 A probability measure μ on the algebra $\mathcal{F}_{pref}$ is *Markovian* if

- $\mu(G_{m+1} = g_{m+1} \mid U \cap G_m = g_m) = \mu(G_{m+1} = g_{m+1} \mid G_m = g_m)$, where U is any time-m event in $\mathcal{R}$,
- $\mu(G_{m+1} = s' \mid G_m = s) = \mu(G_{m'+1} = s' | G_{m'} = s)$. ∎

The first requirement states that the probability of $G_{m+1} = g_{m+1}$ is independent of preceding states given the value of G_m; that is, the probability of going from state $G_m = g_m$ to $G_{m+1} = g_{m+1}$ is independent of *how* the system reached $G_m = g_n$. The second requirement essentially says that, for each pair (g, g') there is a well-defined *transition probability*—the probability of making the transition from g to g'—that does not depend on the time of the transition.

It is not hard to check that these requirements are satisfied by the systems we have considered in our examples (or subsystems of them). This is not so surprising. These requirements are often fulfilled by the most natural way of modeling a system. Indeed, by putting enough information into the environment state, it is almost always possible to model a system in such a way as to make both assumptions applicable. (This is not to say that it is necessarily the most useful way to model the system.)

The main interest in Markovian probability measures on systems is not that they admit well-defined transition probabilities, but that, starting with the transition probabilities and a prior on initial states, a unique Markovian probability measure can be defined. Define a *transition probability function* τ to be a mapping from pairs of global state g, g' to $[0, 1]$ such that $\sum_{g' \in \Sigma} \tau(g, g') = 1$ for each $g \in \Sigma$. The requirement that the transition probabilities from a fixed g must sum to 1 just says that the sum of the probabilities over all possible transitions from g must be 1. (When modeling a finite protocol, I assume that the system just stays in a final state once it reaches it, so that if g is a final state of the protocol, then $\tau(g, g) = 1$. Thus, there is a well-defined transition probability function for probabilistic protocols.)

Proposition 9.7.2 *Given a transition probability function τ and a prior μ_0 on 0-prefixes, there is a unique Markovian probability measure μ on $\mathcal{F}_{pref}$ such that $\mu(G_{n+1} = g' | G_n = g) = \tau(g, g')$ and $\mu([g_0]) = \mu_0([g_0])$.*

Proof Since $\mathcal{F}_{pref}$ consists of time-m events (for all m) and, by Exercise 9.19(a), every time-m event can be written as the disjoint union of m-prefixes, it suffices to show that μ is uniquely defined on m-prefixes. This is done by induction on m. If $m = 0$, then clearly $\mu([g_0]) = \mu_0([g_0])$. For the inductive step, assume that $m > 0$ and that μ has been defined on all $(m-1)$-prefixes. Then

$$\begin{array}{rl} & \mu([g_0, \ldots, g_m]) \\ = & \mu(G_m = g_m | [g_0, \ldots, g_{m-1}]) \times \mu([g_0, \ldots, g_{m-1}])) \\ = & \tau(g_{m-1}, g_m) \times \mu([g_0, \ldots, g_{m-1}]. \end{array}$$

Thus, μ is uniquely defined on m-prefixes. Moreover, an easy induction argument now shows that

$$\mu([g_0, \ldots, g_m]) = \mu([g_0]) \times \tau(g_0, g_1) \times \cdots \times \tau(g_{m_1}, g_m). \quad \blacksquare$$

It is easy to check that in the standard SDP systems discussed so far, the probability measure on runs is Markovian, while the standard systems that are not SDP systems can be decomposed into a disjoint union of systems, each of which has a Markovian probability on runs. For example, consider the second representation discussed in Example 9.3.3, where the set of runs is partitioned according to Charlie's initial strategy. For each of Charlie's strategies s, there is a natural Markovian probability measure on the set of runs $\mathcal{R}_s$ where Charlie uses s. Moreover, this is precisely the way the probability on runs for that example was generated.

Not only are many naturally-arising systems *Markovian* (in that they can be partitioned into sets of runs which have a Markovian probability measure on them), but typically in the subsets of runs with a Markovian probability measure, there is a unique initial state (this in fact is the case in all the examples in this chapter); that makes the probability measure μ_0 trivial—it gives probability one to the 0-prefix $[g]$ where g is the unique initial state of the runs in $\mathcal{R}$ and probability 0 to all other 0-prefixes. This observation just reinforces the usefulness of the Markovian assumption.

Exercises

9.1 Show that $(\mathcal{I}, r, m) \models \Diamond\varphi$ iff $(\mathcal{I}, r, m') \models \varphi$ for some $m' \geq m$.

9.2 Show that in the system $\mathcal{I}$ from Example 9.2.1,

$$(\mathcal{I}, r^1, 0) \models \neg c_0 \wedge \neg c_1 \wedge \neg h \wedge \neg t \wedge \bigcirc(c_0 \wedge \neg h \wedge \neg t) \wedge \bigcirc\bigcirc(c_0 \wedge h).$$

9.3 This exercise refers to the second system constructed in Example 9.3.3, where we consider all possible strategies for Charlie.

(a) Show that there are 2^7 strategies for Charlie. (Hint: recall that a strategy is a function from the history up to a given point to a decision of who goes next. Show that there are only 7 relevant histories.)

(b) Describe the probability on the set of runs corresponding to a fixed strategy for Charlie.

(c) Show that each of the 64 runs in the original representation, where we do not include Charlie's strategy in his local state, corresponds to 16 runs in the new representation.

9.4 Show that a standard interpreted probability system must satisfy CONS and UNIF. Show that if, in addition, there is one probability measure on all of $\mathcal{R}$, the resulting probability system satisfies SDP.

9.5 Construct an interpreted system $\mathcal{I}$ where agents have perfect recall such that agent 1 initially does not know p, but she later learns p. Show that $(\mathcal{I}, r, 0) \models \neg K_1 p \wedge \Diamond K_i p$ for all runs r in $\mathcal{I}$.

9.6 A formula φ is said to be *stable (with respect to the interpreted system $\mathcal{I}$)* if once φ is true it remains true; i.e., if $\mathcal{I} \models \varphi \Rightarrow \Box\varphi$. Assume that φ_1 and φ_2 are stable.

(a) Show that $\varphi_1 \wedge \varphi_2$ and $\varphi_1 \vee \varphi_2$ are stable.

(b) If, in addition, $\mathcal{I}$ is a system where agents have perfect recall, show that $K_i\varphi_1$ is stable. Thus, in a system where agents have perfect recall, if an agent knows a stable formula at some point, then he knows it from then on.

9.7 Prove Proposition 9.4.3

9.8 Show that the analogue of Proposition 9.4.3 does not hold in general in systems where agents do not have perfect recall.

9.9 Show that in a synchronous system with perfect recall,

$$\mathcal{K}_i(r,m)(\mathcal{R}(\mathcal{K}_i(r,m+1))) = \mathcal{K}_i(r,m+1)^- \text{ and}$$
$$\mathcal{K}_i(r,m)(\mathcal{S}) \cap \mathcal{K}_i(r,m+1)^- = \mathcal{K}_i(r,m+1)(\mathcal{S})^-.$$

9.10 Show that in the general construction of a probability measure on runs in Section 9.5, the set $\mathcal{F}_s$ is an algebra. Show that there is a unique way to extend the the definition of μ_s from sets of the form $\mathcal{R}_s^{\vec{\mathbf{a}}}$ to all the sets in $\mathcal{F}_s$ so that it is a probability measure on $\mathcal{F}_s$. Carefully explain why the assumption that the environment state encodes the sequence of joint actions that have been performed so far is needed in this construction.

9.11 For the systems constructed in Section 9.6.1, show that

(a) $\mathcal{I}_{11}$, $\mathcal{I}_{21}$, $\mathcal{I}_{13}$, and $\mathcal{I}_{23}$ are synchronous, while $\mathcal{I}_{21}$ and $\mathcal{I}_{22}$ are not,

(b) the Teller has perfect recall in all six systems,

(c) the Listener has perfect recall in $\mathcal{I}_{11}$ and $\mathcal{I}_{21}$, but not in the other four systems.

9.12 Show that in the system $\mathcal{I}_{11}$, if r is a run in which the Listener is told $\neg\beta$ at the mth step, then the set of worlds he considers possible consists of all the worlds he considered possible at time $m-1$, other than that whose atom is β. Moreover, the Listener considers each of these worlds equally likely.

* **9.13** This problem considers the conditions on the Teller's protocol under which probabilistic conditioning is appropriate. Suppose that the Listener has perfect recall and the Teller is following some probabilistic protocol where he chooses something new to say at each step, but the choice is not necessarily made uniformly at random, and may in general depend on what he has said before. Suppose that the Teller has said $\langle\neg\beta_1,\ldots,\neg\beta_m\rangle$ up to time m in run r, and then says $\neg\gamma$ at time m. Further suppose that, at time m, the Listener considers the set U of worlds possible and has a probability measure $\mu_{r,m}$ on these worlds. Under what conditions is it the case that his probability measure $\mu_{r,m+1}$ on worlds at time $m+1$ is $\mu_{r,m}(\cdot|U(\neg\gamma))$, where $U(\neg\gamma)$ consist of all those worlds whose atom is not γ? According to Exercise 9.12, this will be true if the Teller's protocol is just to choose some world uniformly at random. More generally, show that it is true if and only if the Teller's protocol P is such that the probability of saying γ according to $P((\alpha,\langle\neg\beta_1,\ldots,\neg\beta_m\rangle))$ is the same for all worlds $\alpha\neq\gamma$ in U. However, note that the probability of saying $\neg\gamma$ does not have to be the same as that of saying $\neg\gamma'$, for $\gamma'\neq\gamma$. In this sense, hearing γ gives the Listener no information beyond the fact that $\neg\gamma$ is true, since the Listener is just as likely to hear $\neg\gamma$ for all worlds compatible with this information.

9.14 Describe a simple system that captures Example 4.1.2, where Alice is about to look for a book in a room where the light may or may not be on.

Explain in terms of the discussion of Exercise 9.13 under what conditions conditioning is appropriate.

9.15 Consider Alice's first protocol for the second-ace puzzle, where at the second step, she tells Bob whether or not she has the ace of spades.

(a) Specify the resulting system formally.

(b) Show that in the runs of this system where Alice has the ace of spades, at time 2, Bob knows that that the probability that Alice has both aces is 1/3.

9.16 Consider Alice's second protocol for the second-ace puzzle, where at the second step, she tells Bob which ace she has. Suppose that if she has both aces, she says that she has the ace of spades with probability α.

(a) Specify the resulting system formally.

(b) Show that in the runs of this system where Alice has the ace of spades, at time 2, Bob knows that that the probability that Alice has both aces is $\alpha/(\alpha+2)$.

9.17 Consider the Monty Hall puzzle, under the assumption that Monty must open a door at step 2. Suppose that the probability of him opening door j if you choose door i and it has a car behind it is α_{ij}.

(a) Specify the resulting system formally (under the assumption that you know the probabilities α_{ij}).

(b) Show that in this system, the probability of you gaining by switching is $1/(\alpha_{ij}+1)$.

(Note that if you do not know the probabilities α_{ij}, they become part of the uncertainty that you face. This can be captured using, for example, a probability structure as in Chapter 6, where different probability measures are associated with the possible world.)

9.18 Analyze the three-prisoners puzzle from Example 4.3.1 under the assumption that the probability that the jailer says b given that a lives is α. That is, describe the jailer's protocol carefully, construct the set of runs in the system, and compute the probability that a lives given that the jailer actually says b.

9.19 (a) Show that every time-m event in a system is the disjoint union of m-prefixes.

(b) Show that if $m < m'$, then an m'-prefix is a union of m-prefixes.

(c) Show that if $m < m'$, then the union of a time-m event and a time-m' event is a time-m' event.

(d) Show that $\mathcal{F}_{pref}$ is an algebra; that is, it is closed under union and complementation.

Notes

The general framework presented here for ascribing knowledge in multi-agent systems originated with Halpern and Moses [1990] and Rosenschein [1985]. Slight variants of this framework were also introduced by Fischer and Immerman [1986], Halpern and Fagin [1989], Parikh and Ramanujam [1985], and Rosenschein and Kaelbling [1986]. The presentation here is based on that of [Fagin, Halpern, Moses, and Vardi 1995, Chapter 4], which in turn is based on [Halpern and Fagin 1989]. The reader is encouraged to consult [Fagin, Halpern, Moses, and Vardi 1995] for further references and a much more detailed discussion of this approach, examples of its use, and a discussion of alternative approaches to representing multi-agent systems.

Temporal logic (for reasoning about time) was introduced by Prior [1957]. It has been used extensively in the computer science literature for proving program correctness. An excellent introduction to temporal logic can be found in [Manna and Pnueli 1992]. The combination of temporal logic and epistemic logic was first studied in [Halpern and Vardi 1989]; axioms for reasoning about knowledge and time can be found in [Halpern, van der Meyden, and Vardi 1997].

As was mentioned in the text, there are no known efficient algorithms for primality testing. The probabilistic primality-testing algorithms were developed by Rabin [1980] and Solovay and Strassen [1977]. The one sketched in Example 9.3.2 is due to Rabin. The RSA algorithm was developed by Rivest, Shamir, and Adleman [1978]; their article also gives a brief introduction to public-key cryptography.

The idea of factoring out all the nonprobabilistic steps in a system and viewing them as being under the control of some adversary is standard in the distributed computing and theoretical computer science literature (see, for example, [Rabin 1982; Vardi 1985]; it was first formalized in the context of reasoning about knowledge and probability by Fischer and Zuck [1988]. The general approach to reasoning about probability in interpreted systems used here—that is, the idea of starting with a probability on runs and then going to a probability space at each point, by conditioning—was first discussed in [Halpern and Tuttle 1993].

The formal definition of synchronous systems and of systems where agents have perfect recall is taken from [Halpern and Vardi 1989]. For a discussion of perfect recall in the context of game theory, see [Fudenberg and Tirole 1991]. The definition of perfect recall given here is actually not quite the same as that used in game theory; see [Halpern 1997b] for a discussion of the differences.

The importance of the role of the protocol in analyzing the second-ace puzzle was already stressed by Shafer [1985]. Morgan, Chaganty, Dahiya, and Doviak [1991] seem to have been the first to observe in print that the standard analysis of the Monty Hall puzzle (for example, that given by vos Savant [1990a, 1990b, 1991]) depends crucially on the assumption that Monty Hall must open a door at step 2. The analysis of the second-ace puzzle and the Monty Hall puzzle presented here is essentially taken from [Halpern 1998b].

Markov chains (which is essentially what Markovian systems are) are of great practical and theoretical interest, and have been studied in depth; a good introduction to the field is the book by Kemeny and Snell [**?**]. An extension of Markov chains allows an agent to take an action at each step. The transition probabilities then depend on the action $\mathbf{a}$ as well as the states involved, and thus have the form $\tau(g, \mathbf{a}, g')$. Intuitively, $\tau(g, \mathbf{a}, g')$ is the probability of making the transition from g to g' if action $\mathbf{a}$ is taken. With each action is associated a *reward* or utility. The problem is then to find an optimal *policy* or strategy, that is, an optimal choice of action at each state (under various definitions of optimality). This model is called a *Markov decision process* (MDP); see [**?**] for an introduction to MDPs.