8

Planning and Plan Recognition from a Computational Point of View *

CHARLES F. SCHMIDT AND STACY C. MARSELLA

Introduction

Beliefs, goals and intentions are particularly interesting entities to the psychologist. They are interesting because an organism that has the capacity to use representations which capture the semantics associated with their use can reason about states of affairs that are not true of its current world, and, perhaps, not even true of any possible world which the organism will enter. By definition, planning to achieve some goal involves the use and representation of states of affairs that are not true of the current world. And, plan recognition involves attributing to some other acting agent beliefs about past, present, and possible future states of affairs as well as an intent to bring about some future state of affairs. Further, the ability to plan and recognize the possible plans of other agents does not require an ability to use language. Thus, the absence of language does not imply that the organism does not have the ability to plan or recognize the plans of other agents.

The possession of a symbol system is required in order to achieve the full semantics that are typically associated with the capacity to represent beliefs and goals. In order to explicate the sense in which this statement is true, we will focus on the possible role of such representations in the various models of planning and plan recognition that have been developed within AI. It is necessary to consider such models because the question of whether an organism possesses and uses such representations cannot be decided on strictly empirical grounds. As will become clearer as we proceed, it turns out that for any particular situation within which some pattern of behaviour is observed we can always write down a computational model that can generate that behaviour without resorting to any symbolic representation of beliefs and the like. Consequently, we will complement the empirical argu-

* In A. Whiten (Ed.) *Natural Theories of Mind.* Oxford: Basil Blackwell, 1991. Pp. 109-126.

ments with a more top–down argument. This argument will involve showing what must be presumed if particular observed patterns of behaviour are explained without resorting to the use of representations of beliefs and goals. Thus, the pattern of the argument will be similar to that of *reductio ad absurdum* rather than the presentation of a specific computational model that utilizes beliefs and goals to control its behaviour and to recognize the plans realized in the behaviour of another organism.

From a computational point of view, questions of the specific content of a representation are subsidiary to questions of how it is used. In particular, the concern is with whether the use of a particular representation is required to realize the computation of some function or allows a more efficient realization of the computation of some function. What function corresponds to our intuitive ideas about planning or about plan recognition? This can not be answered directly. But, we can develop various computational models of planning or plan recognition. Such models provide a basis for understanding the computational role of beliefs as well as the difficulties and advantages that accrue from reasoning with such representations.

Computationally Characterizing Planning

Artificial intelligence (AI) has focused much of its research efforts on problems which humans can usually solve successfully but for which it is known or believed that any algorithm that guarantees a correct solution to the full set of problems will for some subset of these problems be so inefficient that the algorithm's use is not practical. We will say that problems with this characteristic are intractable.

An AI approach to studying planning and plan recognition begins by specifying a functional definition, the desired input/output behaviour, of planning or plan recognition. The next step is to attempt to define, implement and study an algorithm that meets these specifications by viewing the problem as one of symbolic search. That is, the algorithm is designed to transform syntactically structured symbolic expressions until some expression is recognized that constitutes a solution to the given problem. Thus, the AI method begins with a computational system that represents and uses a symbolic system (cf. Newell and Simon, 1976). This computational approach is useful since creating some well-defined models of planning and plan recognition can at least provide a basis for describing what might be meant when we say that a person can plan or can recognize the plans of other agents.

Planning is generally defined within AI as a function that takes as input a problem and provides as output a solution to the given problem. More

specifically, a problem is defined as a pair of partial situation descriptions where the first member of the pair describes the starting or initial situation and the second member describes the goal situation. A solution is typically defined as a fully or partially ordered sequence of actions such that the final action in this sequence results in a situation that implies that the partial description of the goal situation is true. A more general definition of the idea of a solution, and the one that we employ, is that a solution is a structuring of the information required to support the execution of a sequence of actions that result in a situation implied by the goal description. A partially ordered sequence of actions is a special case of this definition of a solution. This more general definition is preferable for studying planning when the planner is actually going to execute the actions.

Note that it is assumed that a method for realizing the planning function represents initial and goal situations symbolically. More informally, situations represent the planning system's beliefs about the world and its own goals. If its world is populated by other intelligent agents, then it may also have beliefs about the beliefs and goals of these other agents if these are required by the planning process.

Planning algorithms usually employ some variation of a search method known as *problem reduction*. In this method a problem is decomposed into sub-problems. A sub-problem is syntactically of the same form as a problem, that is, it consists of a situation pair describing its starting situation and goal situation. These sub-problems may themselves be further decomposed until only primitive sub-problems remain. In planning, a sub-problem is primitive if the agent can carry out an action in the starting situation of that sub-problem which yields the goal situation associated with that sub-problem. The recursive decomposition of a problem into sub-problems yields an ANDed hierarchy or tree of sub-problems. The sub-problems are ANDed in the sense that the problem dominating the ANDed node is solved only if each sub-problem dominated by that node is solved. Thus, a problem is solved if the AND tree terminates in primitive sub-problems. This search through a space of possible problem decompositions is realized using two types of rules. One type is referred to as non-terminal rules and the other as terminal rules. Non-terminal rules serve to rewrite a problem as a set of ANDed sub-problems. Terminal rules recognize primitive sub-problems.

In this method, planning is determined by (1) the problem, i.e., the starting situation and the goal situation; (2) the planner's knowledge of possible decompositions, the non-terminal rules; and (3) the actions that an actor is capable of carrying out, the terminal rules. The planner's knowledge is extensible if new non-terminal rules can be acquired. Note that the *derivation* of a course of action, i.e, the tree of sub-problems, is obtained in a top-down and goal-driven fashion. This derivation constitutes the

solution to the planning function. Planning via this method is very similar to the style of reasoning involved in providing a proof for some conjecture (cf. Amarel, 1967). The conjecture in planning is that a particular goal can be achieved from a particular starting situation. The non-terminal rules can be thought of as introducing further conjectures about the intermediate situations that, if reached, will yield achievement of the goal. The terminal rules allow a conjecture to be 'proved'. Retention of the derivation tree allows a plan to be critiqued and revised in a potentially efficient fashion. Retention of the derivation can also serve to support monitoring the execution of the plan. The assumptions on which the plan was based can be checked and, if incorrect, attempts to revise the plan can be initiated or the plan abandoned. Thus, plans that fail are still in some sense rational, they were simply based on an incorrect model or set of assumptions.

Computational Difficulties in Planning

There are two major difficulties that are encountered by this method of planning. Recall that the assumptions introduced in planning are assumptions about some partial ordering of situations that can be achieved. These situations represent assumptions or beliefs about a possible history that can be realized in the world. Now, the world is usually quite a complicated place and our model of the world is at best only partially correct. Further, we often don't have certain knowledge about the world on which to base our model. So one major difficulty is simply that of modelling the world and the effects that our actions might have on the world. And, to make matters worse, it appears that for most worlds that we are interested in, a true model would itself be computationally intractable even if we possessed it. Consequently, planning is limited by the ability to reason correctly and efficiently about the effects of actions on the world. This limitation becomes even more serious when we must reason about the effects that other causal agents may have on the world in which we are planning to achieve some goal. No computational model solves this problem of tractably modelling a complex world. What problem reduction does provide is a way in which to plan and act despite its ultimate ignorance. That is, a plan is based on the assumptions about the world that the planner possesses. To the degree these assumptions constitute a good approximation to the way the world works, the planner will on average be successful. In social situations there is the additional possibility the actors can attempt to constrain their actions to accord with their assumptions about each other in order to sucessfully achieve coordination and cooperation.

In addition to the problem of modelling the effect of actions on the world,

there is also the problem of dependency among sub-problems. It is usually the case that goals and sub-goals involve a conjunction of conditions and no single action can bring about this conjunctive goal or subgoals. Sub-problem dependency holds whenever the choice of *how* or *when* a sub-problem is achieved affects the ability to achieve other sub-problems. Sub-problem dependency is the norm rather than the exception. For example, if the goal is to have dinner and to see a play in New York City then seeing-a-play and having-dinner are two quite dependent sub-problems. One had best work out the seeing-a-play sub-problem first since the location and time of the performance will constrain how the having-dinner sub-problem is solved. None the less, the actions that achieve having-dinner may be executed prior to those involved in seeing-a-play.

There are three basic ways in which planning systems have attempted to cope with sub-problem dependency. One is to solve the sub-problems as if they are independent and await the 'verdict' of the terminal rules which in essence determine whether the action(s) associated with each primitive sub-problem can be carried out in the starting situation of the sub-problem. If not, the procedure backtracks and tries another. If this procedure is followed exhaustively, the search is through a space of possible decompositions and their permutations (e.g., Fikes, Hart and Nilsson, 1971). This can be a very large space to search (Chapman 1987). Another procedure defines a metalevel of knowledge which can access the current tree of derivations and recognize such dependencies and then potentially adjust the current derivation appropriately (e.g., Sacerdoti, 1977; Amarel, 1983; Stefik, 1981; Schmidt, 1985). A third way is simply to know in advance that such dependencies exist and allow this knowledge to guide the derivation of a solution (Bresina, Marsella and Schmidt, 1987). These variations on ways of dealing with sub-problem dependencies yield differing planning models that have been developed within AI.

In summary, the method of problem reduction provides a computational model of planning that: (1) ensures that each action selected is logically related to the goal; and, (2) depends on the agent's beliefs about, and model of, the world in which the plan is to be executed. Further, the employment of this method yields a tree of derivation which, if retained in some fashion, can be used by the agent to critique and modify the plan as well as intelligently monitor its execution. These features of the computational model of planning accord quite well with our informal concept of a plan.

From Planning to Plan Recognition

Given a definition of planning, recognizing (others') plans can be defined as a kind of inverse of the planning function; that is, as a function which takes as input a temporally ordered description of a sequence of actions and associated situation changes and provides as output the assumptions about the actor's beliefs, goals and plan or plans that imply the observations. Because plans can fail or be abandoned, assumptions about the actor's beliefs often must be explicitly provided. An action sequence describes what happened whereas the plan attributed to the actor describes what the actor intended to happen.

If actions are understood by attempting to recover the plan or plans that the actor was executing, then this model of planning accounts for a sense in which the observer attributes rationality to the actor. The derivation of a plan creates a logically coherent structuring of the beliefs, intents, and goal of the actor. However, a cursory glance at a statement of the basic axioms that govern the inference of plans from actions suggests quite forcefully that this might be a computationally formidable problem (cf. Schmidt, 1976). These axioms are presented in Box 8.1. The statement termed the Axiom of Personal Causation asserts that for all actions, A, if some person, P, causes the action then there is some plan, M, of which A is a part. In addition, P believes that P can do M and P has some reason for choosing to do M. Thus, each action is assumed to be constrained by the situation,

Box 8.1 Some 'mindreading axioms

Axiom of Personal Causation:

(Act A) (P cause A) → ((Some Plan M)
(A part of M)
(P believe (P can M))
(P choose M))

Axiom of Shared Assumption of Rationality:

(Person Q) (Q believes
((Person P) (Act A) (P cause A) →
((Some Plan M)
(A part of M)
(P believe (P can M))
(P choose M))))

capacities and beliefs of the actor together with the actor's reasons for choosing to carry out the action. The reasons for the actions are typically either the actor's belief that this action is necessary for the achievement of the overall goal or the action is a final action of the plan which achieves the overall goal directly. Note that the axiom simply states that each action is part of some plan. If one action is observed followed by another, the axiom states that they are each part of some plan. It does not state that each contiguous action is part of the *same* plan. Although the construction of a plan follows an orderly and logical process, this process is not necessarily reflected in a simple way in the action sequence. Contiguous actions may be part of the same plan or differing plans which happen to be temporally interleaved during execution.

Plan recognition is further complicated by the next axiom which states that this axiom is shared over all persons. This, of course, gives rise to the familiar recursive nature of beliefs (see chapter 1) and also provides a basis for cooperation, competition and deception. This creates, from a logical point of view, an extremely complex world in which to plan and recognize plans. Note, there is nothing that guarantees that any of the actors possess correct beliefs about the physical world or about the beliefs and goals of each other. Nor can there be any guarantee that the actors share the same beliefs, correct or incorrect. In fact, in general there is no way in which to guarantee that asynchronous distributed processes can always maintain a consistent 'set of beliefs' that constitute their common knowledge (cf. Halpern and Moses, 1984; Fischer and Immerman, 1986).

The observer's uncertainty about the beliefs and motivations of the actor, the fact that the actor may have based the plan on incorrect beliefs, the fact that plans can be interleaved, and the fact that a plan can fail or be abandoned all conspire to make plan recognition a challenging task. Further, the sequence of actions that are observed is simply a temporally ordered sequence of events. It may have a temporal rhythm or temporal breaks associated with it, but it has no logical beginning and end. A plan, on the other hand, has a definite beginning, the beliefs about the starting situation that served as a basis for the plan. It also has a definite end. When the world satisfies the goal situation of the plan, it has been completed. If the plan fails or is abandoned, it ends as well but in a less satisfactory sense. The planning process was driven by the planner's goal, but goal achievement is the last observation that the observer has available and goal achievement is an inference that results from the recognition process itself, not something that is perceptually marked.

A Plan-Recognition Strategy: Hypothesize and Revise

For these reasons, a plan-recognition process that exhaustively maintains the space of plans consistent with a set of observations will generally be intractable. A more tractable computational model generates a predictive hypothesis about the actor's goal or goals and utilizes these goal hypotheses to focus the observations and their interpretations. We have referred to this type of recognition strategy as a *hypothesize and revise strategy* (Schmidt, Sridharan and Goodson, 1978). It proceeds by generating the higher levels of the plan derivation as early as possible. This hypothesis is then revised in light of the unfolding observations. There is no guarantee that such a heuristic strategy will always succeed in identifying a set of plans that cover the observations. And, there is certainly no guarantee that the hypothesis is correct. None the less, it can yield a tractable strategy which can serve the observer well especially if the goals of the actor and the actions of the actor's plan obey the norms and conventions that are used to guide the observer's process of hypothesize and revise. Further, the logical coherence of a plan serves as a quite reliable basis for *disconfirmation* of the observer's hypothesis. Thus, even if we are not sure as to exactly what someone is up to, without an explicit attempt by the actor to deceive us, we often know that we dσn't know. This is more than an interesting consequence. It is crucial since it can itself serve as a basis for action on the part of the observer. Again, this ability to recognize failure to understand the beliefs and intents of another is crucial for the eventual achievement of cooperation and coordination of action among several actor/observers. If the logical coherence were not assumed, then even this strong basis for disconfirmation would be unavailable.

An Alternative to Symbolic Representation of Beliefs

Now up to this point, we have simply presented a brief sketch of some of the ways that planning and plan recognition have been formulated from a computational point of view. This computational approach assumes beliefs, beliefs about beliefs, intents and goals as part of its method for planning and plan recognition. But, must we make such seemingly strong assumptions about the cognitive ability of a device in order to achieve performance that we might informally characterize as intentional and plan-like? For many, the idea that young children, much less primates, have such capacities is at least debatable if not downright absurd. We leave to others in this volume

who are much more expert in these matters to debate the specific merits of the case.

However, we can briefly sketch the alternative, again from a computational point of view. That is, let us systematically consider some computational models that can be phrased in a formalism in which symbols, and therefore beliefs and goals, cannot be represented. One reason for this tack is that the arguments *against* attributing beliefs and goals to some biological organism almost never present a constructive alternative account. There is, as we hope to briefly illustrate, good reason for this. Alternative accounts, accounts which do not resort to a computational model which can represent beliefs and goals themselves suffer from some rather severe problems.

It is often argued that a psychological theory of an organism's behaviour that does *not* assume that the organism can represent beliefs and goals is *simpler*, and thus preferable, to one that does make this assumption. Consequently, the burden of proof is often placed on the theorist who claims that such representations are part of the cognitive mechanism. But what would a computational theory that does *not* assume that the organism has the ability to represent and use beliefs look like? In this final section, we turn to an examination of the kind of theory of 'planning' or 'plan recognition' that can be stated if we restrict ourselves to a computational framework in which symbols, and thus beliefs, are unavailable. Planning and plan recognition are placed in quotes because these terms are defined in terms of the computational model that carries out these processes. A computational model in which beliefs and goals cannot be represented and used explicitly will obviously result in a different characterization of the process of planning and plan recognition.

In what sense is a theory that eschews the use of symbols and thus beliefs simpler? To say that one thing is simpler than another presumes that we have some uniquely appropriate metric to use to determine the simplicity of theories or explanations. If we take the view that our theories of behaviour are computational in character, then there is a well-defined metric used to order formal models of computation. Associated with each formal model of computation is a set of functions which can be computed using that model. If the set of functions that can be computed using one model are strictly included in the set that can be computed using another model, then we say that the more inclusive model is more powerful than the first. This yields a hierarchy of formal models of computation. At the bottom of this hierarchy, the least powerful, are finite state machines (FSM) or equivalently finite state grammars (FSG). At the top of this hierarchy are Turing machines or equivalently, unrestricted grammars. In this hierarchy, it is only the finite state machine model (FSM) that provides a formal model of computation that does not employ symbols. Thus, one well-defined construal

of *simpler* is to say that a computational model or theory of behaviour is simpler than another if it is computationally less powerful than another. On this interpretation, any theory that is cast as a finite state machine would be simpler than another theory that is cast in a more powerful computational formalism. Then, a decision procedure might be to only state our theory within a formal system more powerful than a FSM when the 'data' demand it.

There are difficulties with this seemingly straightforward approach. The difficulties can be more easily appreciated if the basic structure of a finite state machine is defined (cf. Hopcroft and Ullman, 1979). A deterministic finite state machine (DFSM) is a 5-tuple $(Q, \Sigma, \partial, q_\Phi, F)$, where Q is a finite set of states, Σ is a finite input alphabet, q_0 in Q is the initial state, F subset of Q is the set of final states, ∂ is the transition function mapping $Q \times \Sigma$ to Q, that is, $\partial(q,a)$ specifies a state from Q for each state q and input a. The input to a DFSM is a string, s, of elements from Σ. The set of all such possible input strings is symbolized as L, subset of Σ^*. This set may be infinite. The output of a DFSM is typically defined as 1 if the device accepts the string and 0 otherwise.

Figure 8.1 provides a transition diagram of a DFSM that accepts L, where L is the set of strings of the form $a^n b^m$ where $n > 0$ and $m > 0$. In this example, $\Sigma = \{a,b\}$; the set $Q = \{q_0, q_1, q_2\}$ are represented as nodes; q_0 is the initial state and q_2 the final state. The transition function ∂ is represented as the set of directed arcs between elements of Q annotated with an element from Σ.

It will often be more natural for our purposes to associate an output with

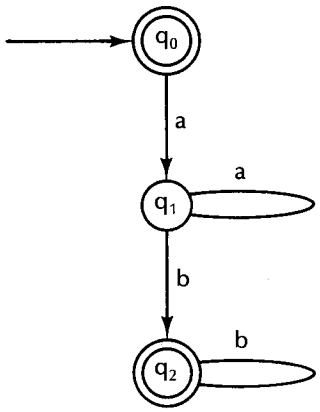

Figure 8.1 *State transition diagram of a finite state machine that accepts* $a^n b^m$.

each transition of the machine. In this case, the DFSM is defined by the 6-tuple $(Q, \Sigma, \triangle, \partial, \lambda, q_0)$. Here Q, Σ, ∂, and q_0 are defined as before. The set $\triangle$ is the output alphabet and λ is a mapping from $Q \times \Sigma$ to $\triangle$. DFSM with an associated output function are equivalent to the standard DFSM.

Figure 8.2 provides a transition diagram that exemplifies this type of DFSM. In this example $\Sigma = \{N, D, Q, C, P\}$ and $\triangle = \{a, r, c, p\}$. In the diagram, states are represented as nodes and the arcs represent state transitions. Each arc has a label of the form 'x/y' where 'x' is the input symbol associated with the transition and 'y' is the output. This machine can be thought of as accepting (a) nickels (N), dimes (D), or quarters (Q) as long as they contribute to an exact sum of 50 cents and rejecting (r) them otherwise. In the state 50 it accepts C and emits the action of providing a coke (c) or accepts P and emits the action of providing a pepsi (p). For future reference we will refer to this DFSM as the structured soda seller (SSS).

Finally, it will be useful to define a nondeterministic finite state machine (NFSM). This is again defined as a 5-tuple $(Q, \Sigma, \partial, q_0, F)$, where Q, Σ, q_0, F have the same meaning as before, but ∂ is a mapping from $Q \times \Sigma$ to 2^Q rather than to Q. The set 2^Q is the power set of Q; that is, the set of all subsets of Q. An equivalent DFSM can be constructed for any NFSM.

It is important to keep the algebraic structure in mind to avoid reading more into this formal model than is really there. The state-transition view of this model of computation makes clear that the control of the computation is realized by the states and their connections. At any point in the computation, only the current state and current input event can affect the state transition. There is no memory for the path of state transitions that led to the current state and no memory for the previously processed input events. Similarly, future input events and future possible state transitions are unavailable to control the choice of a state transition.

It would seem to be extremely difficult to account for observations of an organism's actions or outputs in response to inputs from the environment using such an austere computational model. Surprisingly, the opposite is the case. As long as the set of <I,O> pairs is finite, we can always write down a FSM that can generate these <I,O> pairs. This can be done by adding, as needed, new elements to Σ, new states to Q, additional actions to $\triangle$, an extension of the transition function, ∂, and extension to the output mapping, λ, depending on which may be required to include any additional <I,O> pairs that may be observed. Even the most driven experimentalists among us will never observe more than a finite set of <I,O> pairs. Thus, we have a kind of paradox. What has often been taken by psychologists as the simplest and most parsimonious model, can in fact be very hard to

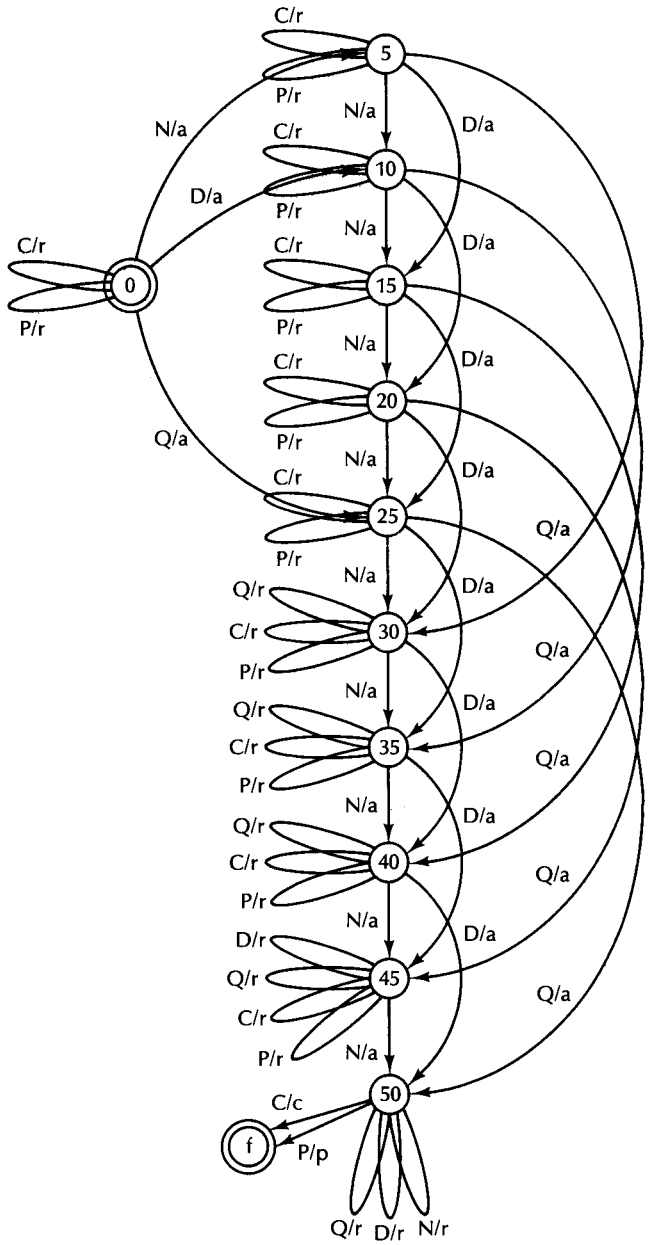

Figure 8.2 *State transition diagram of the Structured Soda Seller (SSS).*

falsify empirically. At the limit a finite state machine can have no more structure than an arbitrary finite list of rules. A theory that is simply a list is certainly simplistic. Whether it also corresponds to our intuitive notion of a simple theory is another matter.

Figure 8.3 illustrates a NFSM that is nothing more than a list of a set of <I,O> pairs. For purposes of this example we again consider a machine that sells soda, but in this case the rules for selling the soda are represented in this list form. Only some of the states required to define this machine are represented in figure 8.3, since there are a great many. We will refer to this version of our soda seller as the list soda seller (LSS). Note, that in contrast to the SSS, the structure of LSS does not at all reflect the combinatorial structure of L. There are two consequences of this difference in the structure of the machine from the computational point of view. First, L, the input language or event strings accepted by SSS is infinite and L' the input language accepted by LSS is a finitely bounded subset of L. Secondly,

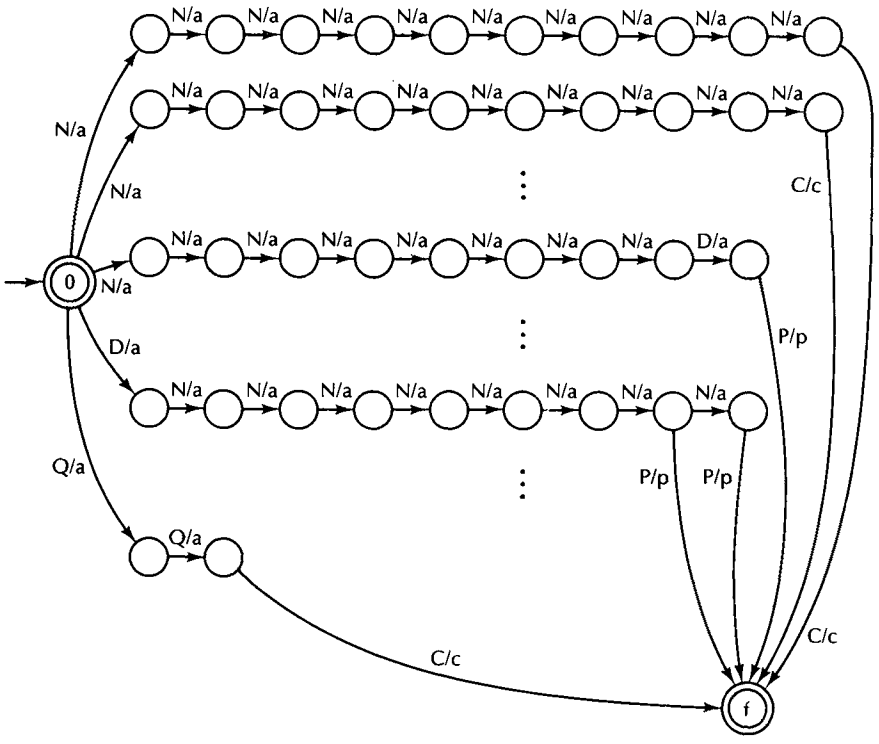

Figure 8.3 *Very, very partial state transition diagram of the List Soda Seller (LSS).*

the LSS requires many more states than SSS. However, neither SSS nor LSS appreciate their respective intelligence or ignorance regarding the combinatorial structure of L. We can see the structure or lack thereof in the respective graphical representations of these machines. But the processes realized in each of these machines 'see' only the current state, current input event, and next state.

Now the discovery or conjecture that an input language, L, possesses some combinatorial structure is of little relevance unless there is concern with the issue of creating or modifying an FSM. If a FSM comes prepackaged with the hardware or neuronal wetware, then aside from physical considerations of its size or density of connections, one is just as good as the other. But, modifying or building an FSM suggests the need for use of a computational model more powerful than a FSM. Thus, we are left with the dilemma that either our organism or device must already possess a suitably general set of FSMs to handle all of the exigencies of its everyday life or it must possess a computational model more powerful than a FSM.

To appreciate and illustrate the inflexibility of a given FSM let us return once again to our soda sellers. Now, obviously, what is crucial to this soda selling game is not the particular sequence in which the money is received and the type of soda specified. What is crucial is that the purchase price be provided and the item desired be specified. A permuted sequence of inputs such as <quarter, coke, dime, nickel, dime> is just as good as any other sequence that satisfies these requirements. Thus, using such permutations, there is a very abstract, at least for this mundane example, way in which to form an equivalence class over the particular input language. But the only way in which to capture this equivalence class in an FSM is to increase drastically the number of states. For example, one can extend the SSS to accept the input of soda selection followed by a sequence of coins summing to 50 cents by creating two additional copies of the existing machine and attaching these to the start state. One of these copies covers the selection of a pepsi and the other the selection of a coke. These copies allow the dependency between soda selected and soda dispensed to be retained over the ensuing coin sequence. To accept the permutation set created by specifying the selected beverage somewhere within the sequence of coins will again require an enormous proliferation of states.

This equivalence class is quite simple to capture within the more powerful planning formalisms discussed earlier. Planning can capture the equivalence class at a level of description quite removed from the input actions by referring to abstract situations or sub-problems; having-received-the-price and having-received-the-request. The dependency among events can be stated correctly and generally at this level of abstraction. Further, the enumeration of the permutation set of events is no longer required. What

is required is only the ability to generate a correct sequence from this set in order to achieve this goal. Thus, for a problem reduction planner, generating a correct sequence would entail recursively decomposing these abstract sub-problems (having-received-the-price and having-received-the-request) into primitive sub-problems that represent the various actions, such as depositing a quarter. On the plan-recognition side, what is required is the ability to recognize if a sequence does achieve the goal. In part, the flexibility and economy is achieved because planning and plan recognition are captured as a generative process, the problem reduction search, rather than as a fixed and fully enumerated structure.

At this point, it should be obvious that these same problems beset any attempt to characterize plan recognition as a FSM. Indeed, it is difficult to even distinguish 'planning' and 'plan recognition' within a FSM model. The difference between generation and recognition can be captured by a change in the input and output alphabets of the machines. There is, however, something interesting to be said if we consider the possibility that two interacting machines/agents may have differing structure. By interacting machines we refer to a case where some of the inputs to one are the outputs of the other and conversely. Consider first the case where both machines are FSMs. To illustrate this, let us create a soda purchasing machine whose outputs are precisely the elements of the soda selling machine's input language. In this case, the soda purchaser may be a generator of output sequences where either the selection is first specified and then the money provided or vice versa. Assume that the soda seller is our now well-understood soda seller, SSS. In this case, the soda buyer will be successful whenever the money is provided prior to the selection and unsuccessful otherwise.

To take an even more extreme case, consider the degenerate soda seller (DSS) shown in figure 8.4. This machine has a single state and provides

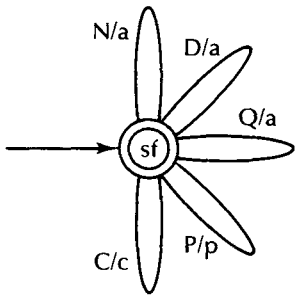

Figure 8.4 *State transition diagram of the Degenerate Soda Seller (DSS).*

the specified soda as soon as it is input. It will also accept as much money as the interacting machine cares to provide it. Note, that if the soda purchaser always provides 50 cents followed by a selection of a soda, the DSS will seem to accept exactly the same input strings as the SSS. Alternatively, the soda purchaser may actually be a planning system of the type reviewed earlier. In this case, this planning soda purchaser may provide any permutation of coins summing to 50 cents and a soda selection and DSS will consistently provide the appropriate soda. In fact, the DSS might appear to the planning soda purchaser to be more powerful than the SSS; and, DSS will now receive all of his business. The only anomaly that might suggest to the planning soda purchaser that DSS is not itself a powerful plan recognizer rather than a FSM is that DSS provides the soda as soon as requested regardless of whether the proper amount of coinage has been output by the soda purchaser. But, perhaps our planning soda purchaser simply assumes that DSS is a very trusting soul/machine!

Now we have rather thoroughly and painstakingly illustrated the limitations of finite state machines as a model of computation. However, despite their limitations they also have certain advantages. First, as the DSS has illustrated, a very simple and computationally tractable model can perform quite appropriately in a suitably benign world. Secondly, we have illustrated that as long as the input events are of finite length, it is always possible to specify a FSM that 'behaves appropriately'. Again, this can yield an extremely tractable computational solution to the generation of action. The FSM model may well be an appropriate computational formalism in which to capture the generation of actions by some organisms or by some organisms at various stages of development or by some organisms within certain types of problem domains.

As some in AI have turned to problems of real-time control that arise in robotics there has been increased interest in the ability to map from plans to a corresponding FSM. Note, the plan is not created via a FSM, but once created the plan and an associated logic of belief can serve as a basis for constructing a FSM whose behaviour is correctly correlated with the plan (Rosenschein, 1985). In general it is not possible to predict if and when a planning algorithm of the type specified earlier will arrive at a solution. If a real-time response is required, this requirement can sometimes be met by mapping a symbolically generated plan onto a suitable FSM. Consequently, it is certainly possible that we, along with other species, have the ability to, in some fashion, 'compile out' the representation of beliefs, goals, and actions by creating a structure of control akin to the simplicity of a finite state machine model.

Concluding Remarks

Where does this leave us as psychologists interested in determining the kind of computational model required to explain some observed behaviour? It certainly reaffirms what most of us knew all along; namely, the explanation of behaviour is not a simple task. However, our hope is that we can draw some more interesting conclusions from this examination of the computational approach that will inform our empirical attempts to study behaviour. These conclusions were implicit in many examples that were presented.

The first conclusion is that attempts to characterize the computational model required to explain an organism's behaviour must begin with a careful analysis of the combinatoric structure of the problems presented to, or in the case of naturalistic studies, responded to by the organism. Precisely what input events can appropriately be grouped into equivalence classes? What are the dependencies among these equivalence classes and how do these dependencies constrain the permissible action sequences?

If the combinatorial structure or structures allowed by a problem are understood, then this provides several ways in which to attempt to probe the generative power of the organism's computational model. The first is to ascertain whether or not the full combinatorics of the input language, the equivalence classes allowed, are evidenced by the organism's observed behaviour. The second is to violate systematically the combinatoric structure of the input language to determine if such violations are evidenced in the organism's behavior. Thirdly, if by hypothesis a generative model of planning or plan recognition is appropriate, then it may be possible to devise problems which differ in their difficulty of solution. And finally, if a planning or plan recognition model is appropriate, then the organism's knowledge may be at a level of abstraction that allows the transfer of appropriate portions of this knowledge to an entirely different problem.

We have probably meandered down this computational path at a rapid enough rate to catch our breath and provide a brief summation of the gist of what we have tried to communicate. First, as you undoubtedly expected, we have not come to any final answers to the various questions that have been posed. Rather we have tried to direct your attention and intuitions here to a certain quandary. It appears that any device, whether biological or artificial, that is required to engage in flexible and asynchronous modes of social interaction, especially if that social interaction includes a cooperative component, needs to be able to compute functions, create plans and recognize the plans of others, that are from a computational point of view generally intractable. None the less, we do interact with each other quite successfully.

The answer to this success is probably to be found in several places. First, just as in the case of natural language, we will probably be led to conclude that biological devices are biased by their nature in ways that make effective social interaction tractable. Second, and this is not independent of the first point, devices which can interact successfully are probably able to find and learn computationally tractable approximations of the function that seems to be logically required to interact with guaranteed success. The task of understanding our understanding of ourselves and others is certainly a challenging one, and, probably one that will all keep us all occupied as long as we wish.