Methods for deciding *what to do next* and learning*

Henry Hexmoor[ab]
Donald Nute[ac]

[a]Artificial Intelligence Programs, The University of Georgia, Athens, GA 30605
[b] Department of Computer Science, University at Buffalo, Buffalo, NY 14260
[c] Department of Philosophy, The University of Georgia

September 10, 1992

1 Introduction

Recent years have seen intense analysis and questioning of the importance of the role of classical AI-planning in deciding moment-to-moment actions. This has led to development of several new AI-planning paradigms such as reactive planning [Firby, 1987, Georgeff & Lansky, 1987]. Informally, a planning paradigm refers to principles of representing an agent's acts, beliefs and expectations about concepts-relating-acts-to-conditions-in-the-world, and methods for prescribing acts in response to conditions. Each paradigm (independent of a domain) develops a uniform way of interacting with the world and as such it attempts to account for complexities in the agent's environment.

The oldest AI-planning paradigm, known as classical planning, is closely related to reasoning and it is a highly cognitive behavior involving explicit goals. On the other hand, the reactive planning paradigms use little or no reasoning and goals are implicit. Reactive planning clearly has a less cognitive character than classical planning. We are interested in agents that both act based on reactions, producing whatever goals are implicit in those reactions, and generate plans to achieve explicit goals. We will refer to these ways of behaving as the agent's modes of behavior.

By *what to do next* we have in mind the very next physical action an agent situated in the world performs. Actions we consider are provoked either by direct sensing and a specific purpose (i.e., a goal) or by direct sensing and a general condition (i.e., an implicit goal). "Methods for deciding *what to do next*" in the title of our paper is meant to cover all planning paradigms that advance a physical action for execution, subject to a few simplifying assumptions. This includes actions prescribed by a plan produced by a classical planner, actions suggested by a reactive planner, and actions suggested by experimenting either to gain information about an agent's capabilities or to achieve a goal heuristically.

Since the scope of planning methods differ about when and in what ways the agent may use the output, we suggest the following assumptions on planning methods to limit their scope for deciding *what to do next*. These assumptions do not affect how the methods work. They only affect the inputs given to the method and the application of its output.

- The decision to execute an action takes into account present situations perceived. This assumption insures that the agent continually redecides what to do next as the situation changes.

*We appreciate comments made by Beth Preston on an earlier draft.

- We ignore the agent's prior goals and commitments beyond the scope of goals generated from the start of interaction with the environment. This is to deny long term memory of goals to the agent and to avoid problems associated with interaction among goals. Specifically, we ignore commitment to executing actions at non-present times.

- An agent's internal attitudes, such as needs and personal policies, are not considered. This excludes need-based reflexes and actions in response to needs. We make this assumption to avoid confusion between planning methods and agent architecture.

Adopting a planning paradigm assumes that the paradigm is universally applicable across domains. This assumption promotes a uniformity in the agent's mode of interaction with its environment that is unwarranted. Said differently, the agent adopts a fixed location for tasks in its cognitive/noncognitive spectrum of behavior generation. But for any of the existing planning paradigms we may pick, there are situations which are difficult or impossible to subsume. Another problem is that existing planning paradigms are not rich enough to account for behaviors other than deciding what to do next, eg., learning. These problems suggest that what is needed is some kind of integrated approach.

We assume that an agent generates behaviors at different levels of cognition. We observe that the internal states of an agent bias its mode of response generation, i.e., its level of cognition for tasks. For instance, an agent playing chess makes its moves reactively or plans against hypothesized counter-moves depending on its chosen level of cognition for the task.

We observe that the level of cognition for tasks is determined as part of the process of perception. The external stimuli in conjunction with the agent's internal states (eg., perceived-urgency, perceived-criticality, or perceived-resource-availability) determines a cognitive level for the task at hand.

We posit that a successful agent that functions in a dynamic environment needs to have a repertoire of behavior generation capabilities at different cognitive levels (corresponding to different paradigms). We call such a system a hybrid planning system. The system is hybrid since it incorporates different paradigms, potentially concurrently. We are particularly interested in the relationship between perception and behavior generation. In behavior-based AI [Brooks, 1990], this relationship is encapsulated in modules that pair-up perceptions with actions. In the traditional approach to intelligent agency, perception is a separate and independent module from behavior generating modules. These two approaches represent two opposite architectures for modeling intelligent agents.

We sketch an approach that suggests a cognitive continuum for sensory-based perception and behavior generation. At the less cognitive end, the approach resembles behavior-based AI in that sensors and actuators are closely connected; at the more cognitive end of the spectrum, it resembles the traditional approach in that there is a substantial distance between sensors and actuators. This approach suggests that naturally-occurring predictions/envisionments pursuant to sensory-based perception fill the space between perception and behavior generation. These play a key role in what method of behavior generation is dominant at any given moment. Furthermore, information gained from this extended perceptual processing is used in behavior generation. We call the processing that follows a more general and more familiar kind of perception *Extended Perception* (EP). EP itself operates on a cognitive continuum.

In this paper we will review general assumptions underlying planning paradigms, integrated approaches, and learning. We will describe agents that function in the world as hybrid systems that integrate perception and subsequent processing with behavior generation. After a general discussion, we will present a finite state machine model for representing reactive plans and a scheme for compiling classical plans into reactive plans.

We are proposing an approach to developing an agent that exhibits some of the behaviors of humans and other organisms, an approach to be tested and improved or discarded. While our approach is motivated by examples involving human agents, we are not making any claims about the psychological reality of this approach nor do we make any claims about this being a theory of human intelligence. It would be a bonus if it could be shown that humans work in some manner similar to the agents we will describe, but we will consider our approach successful if it can produce agents that exhibit the target behaviors whether or not this is the way humans produce these behaviors.

2 Planning Paradigms

In the following subsections we will review classical planning and recent planning paradigms as they can be used for "deciding *what to do next*". We broadly cover new planning paradigms under the general title of reactive systems to emphasize the common assumptions in these paradigms. We omit review of other paradigms such as memory-based (case-based) planning.

2.1 Classical Planning

Classical planning systems [Fikes & Nilsson, 1971, Sacerdotti, 1977, Tate, 1977, Wilkins, 1984] consider a plan to be a partial order over a set of operators (i.e., actions) with pre- and postconditions. Given a goal, a means-end mechanism uses matching of the goal with operator postconditions to instantiate operators if the preconditions are satisfied. Otherwise, it recursively uses preconditions to continue its search for applicable operators. Planners built according to this paradigm formulate a plan before commencing execution of the plan.

Classical planners share the following assumptions.

- STRIPS Assumption: The agent (and nothing else) is responsible for all changes in the environment.
- Plan-Then-Execute Assumption: A concrete plan is formulated prior to execution of the plan. No planning takes place while executing the plan.
- Action Fixation Assumption: An action has a set of preconditions and a set of postconditions. Postconditions may be partitioned into add and delete lists.

Classical planners often also make the following assumptions but these are not meant to be used in defining classical planners.

- Least Commitment: Decisions as to the order of planning for subgoals are delayed as long as possible.
- Linearity Assumption: "subgoals are independent and thus can be sequentially achieved in an arbitrary order" (Sussman [Sussman, 1973]).

Acts in classical planning are often represented as operators with pre- and postconditions. Preconditions are conditions that must be true before an act is applicable. Postconditions are effects of an action that hold after the act is executed. STRIPS divided the postconditions into add-lists and delete-lists. An add-list contains all that becomes true after executing an act. A delete-list contains all that becomes false after executing an act.

Knowledge about acts can be represented declaratively as propositions [Shapiro et al., 1988]. One way to do this is to associate a rule with each precondition of the act in statements of the form of [Q: A $\longrightarrow$ P] where Q is a quantifier on the object types involved in this act, A is the act name, and P is the precondition proposition. Similarly, postconditions are represented as [Q: A $\longrightarrow$ E] where E stands for an effect, i.e., a postcondition. Add and delete lists are not differentiated in [Shapiro et al., 1988] since a proposition can represent either an add condition or a delete condition. In [Shapiro et al., 1988], a proposition represents an add condition by a positive proposition about the condition while a negative proposition represents a delete condition.

A classical planner becomes a "method for deciding *what to do next*" by constraining it to generate plans a step at a time and to execute each step as it is generated. McDermott's NASL system [McDermott, 1978] is an example of this restriction on classical planners. Such a system reassesses the environment after executing a plan step. If the environment is consistent with its expectations of effects of its executed action, it generates the next step of the plan. Otherwise, it plans anew.

2.2 Reactive Systems

Reactive planning takes the situated action view of planning. That is, the agent decides on an action based on the current situation. All reactive systems share the following properties:

- Applicability of only one action is decided upon at one time.
- No predictions are made about what will be true after completing an action.

Since reactive planners make no predictions about the results of actions, the concepts of plan failure and replanning are avoided in reactive systems. A plan fails if either a plan step becomes inappropriate with respect to the conditions in the environment or a postcondition of a plan step after execution is not true in the environment. A planner that fixes a failed plan is said to be replanning. The reactive control strategy of James Firby's RAPs [Firby, 1987] helps it to avoid replanning. Firby states, " ... execution monitoring is an intrinsic part of the execution algorithm, and the need for separate replanning on failure disappears".

In the following subsections we will review a variety of paradigms that make different assumptions about the interactions of planners with their environment. We have included these paradigms under reactive systems since they share the above two assumptions. Otherwise, these paradigms are independent. Methods defined by these paradigms are all consistent with "methods for deciding *what to do next*". These paradigms reflect the current research in the area although other similar paradigms are conceivable.

2.3 Reacting as a Stateless Device

Reactive planning where the system reacts as a stateless device generates plans that put off the decisions to react until execution. These are planners that generate reactive plans. Such a planner is considered to be an automatic constructor of reaction plans. Here, the reactivity lies in the plan and not in the planning strategy. After Marcel Schoppers [Schoppers, 1987], let's call this type of reactive planner a "reactive reaction planner", abbreviated as RRPlanner. Schoppers' universal planning [Schoppers, 1987] has an RRPlanning component. RRPlanning does not interact with the environment; it is rather the plan interpreter that interacts with the environment. In this respect, building a reaction plan is like building a device. Reaction plans are an extreme approach to *acting with little or no thinking.* This might be suitable for general-purpose agents in some activities or for some insect-like agents. These planners share the following properties.

- Situatedness assumption: Applicability of an action is decided based on the current situation perceived and not on the analysis of the history of what has happened.
- Infallible reactions assumption: Interference among actions is computed and situated-rules are designed to steer away from undesirable states. This is also known as the *causal independence assumption.*

In Schoppers Universal Planner [Schoppers, 1987] an action is represented as a set of goal-reduction rules, i.e., action-i = $\bigvee$ Q $\times$ S $\longrightarrow$ A $\times$ M where Q is a set of quantifiers, S is a set of subgoals, A is an action, and M is a set of maintenance goals. Subgoals can be inperpreted as situations under which actions are applicable. The set of situations in goal-reduction rules for an action action-i covers all possible situations for applicability of action-i. Each goal-reduction rule is called an *action description* that maps a number of subgoals S to achievement goals A. Subgoals are situations under which the action will achieve a goal.

A universal plan for a goal is a decision tree that is derived from backchaining through action descriptions. If a condition is false at a node of the decision tree, then all subtrees below it must maintain that the condition is false. The universal planner generates and uses *confinement rules.* These rules are constraints on causal interference among actions.

Another example of this work is Drummond's Situated Action Rules. Drummond [Drummond, 1989] described a planning system that generates Situated Control Rules (SCRs) from *plan nets.* Plan nets are intended to be a language for specifying basic capabilities of an agent. A plan net is a bipartite graph of conditions and events. Edges connecting events to conditions are *enablement edges* and edges between conditions and events are *causal* edges. Through a process of *projection*, SCRs are generated from plan nets. SCRs are rules that embody causal knowledge about the domain. SCRs are used to guide action selection and to steer away from undesirable states.

Brooks' subsumption architecture [Brooks, 1985, Brooks, 1987, Brooks, 1990] clusters behaviors into layers. Low level behaviors like deciding the direction of motion and speed can be interrupted by behaviors determined at higher levels such as avoiding obstacles. Subsumption behaviors are written as finite state machines augmented with timing elements. A compiler is used to simulate the operation of finite state machines and parallelism. This architecture is implemented on a variety of mobile robots. More often used behaviors are placed at a lower level than less frequently used behaviors. This organization of behaviors gives the system faster response time and higher reactivity. Maes has experimented with a version of a behavior-based architecture which she calls ANA, [Maes, 1991]. This architecture consists of competence modules for action and a belief set in a network relating modules through links denoting successors, predecessors, and conflicts. Competence modules have activation levels. Activations are propagated and the competence module with the highest activation level is given control. We will review Maes' reinforcement-based learning later in this paper.

An underlying assumption in subsumption architecture is that all tasks are automatic, i.e., operating at the noncognitive level; operations of processing modules may seem to be like an electronic circuit, fast and without reason or explanation. Computational models of interaction like Brooks' subsumption are fine for the tasks to which they are applied. However, they do not provide a uniform model of interaction at various cognitive levels.

2.4 Reacting as an Automaton

Kaelbling [Kaelbling, 1986, Kaelbling, 1988] describes a reactive architecture for robot planning. This architecture defines a hierarchical organization of robot behaviors and a control scheme that determines appropriateness of available behaviors. Rosenschein's and Kaelbling's work [Kaelbling, 1988, Kaelbling & Rosenschein, 1990] describes tools (REX, GAPP, RULER) that, given descriptions of the world and tasks, construct reactive control mechanisms termed *situated automata*. Their architecture consists of perception and action components. The robot's sensory input and its feedback are inputs to the perception component. The action component computes actions that suit the perceptual situation. We should note that unlike Brooks' behavior modules, situated automata use internal states. They are mainly intended to produce circuits that operate in real-time, and they can prove properties about their operation. So their decisions are not Markovian.

2.5 Reacting as Improvisation

Agre and Chapman [Agre & Chapman, 1990] have championed acting as emergent interactions between an agent and its environment. Pengi and Sonja are two computer video games developed by Agre and Chapman that exemplify reacting as improvisation. In these games, the game-hero engages in sophisticated interactions with the enemy without any prior planning. However, strategies the hero exhibits appear to be planned.

Reacting as improvisation emphasized the following properties:

- Ready to act assumption: The planner is always ready to interact with the environment.
- Deictic Representations assumption: An agent interacting in its environment only needs to represent entities in terms of how they are related to the current task and how they are related to the agent.
- Lack of Control assumption: "...abandon the notion that activity has a particular locus of control at all. Activity arises through interaction, not through control." ([Agre & Chapman, 1990] p. 22)

The second assumption is worth elaborating. A situated agent, at any moment, attends to only a handful of entities and relationships in its immediate surroundings. In this type of setting, the agent often does not care or need to uniquely identify objects. It is sufficient to know the current relationship of the relevant objects to the agent, and what role the objects plays in the agent's activities. Agre and Chapman [Agre & Chapman, 1987] proposed indexical-functional (Deictic) representations to be the more natural way agents refer to objects in common everyday environments. They called entities and relationships of interest *entities* and *aspects* respectively. With respect to its current activities, the agent needs only to focus on representing those entities and relationships. Although the objects in the environment come and go, the representations of entities and relationships remains the same. For example, the-cup-that-I-am-holding is an indexical-functional notation that abstracts the essentials of what the agent needs to know in its interaction. This kind of designation is merely a mnemonic representation intended to suggest the entity and aspect under consideration, for the purpose of our exposition. It is not the actual representation that would be used by an agent. These representations serve to limit the scope of focus on entities. For example, if the agent wants to pick up a cup, it does not need to know *who owns the cup* or *how much coffee the cup can hold*. It may not even need to know that it is a cup, but only that it is an item to be cleared from the table. Only the relevant attributes of the item apply. We see the main use of this representation to be its ability to narrow the focus of attention about the entities in causal relationships with the agent.

3 Learning

Learning is an important feature of an agent that has to interact in the world. Since we are not interested in domain specific agents, we would like the agent to be able to cope with what it finds in the world. We are particularly interested in agents that learn to extend their abilities to interact with the world. For example, if all an agent knows is picking up square boxes by using its two fingers to squeeze on opposing faces of a square box and lifting it, and now it faces a round ball, it has to learn either to use more fingers or use its two fingers to squeeze on diametrically opposing points of the ball. An agent, novice in performing an act or novice in choosing an act, consciously reasons about its act and controls its execution. As the agent acquires more experience with an act, it is able to unconsciously pick the act and execute it. In psychological terms, this is called *automaticity*. In this paper, we are primarily concerned with learning knowledge of acts and knowledge of using acts and omit review of broader learning techniques.

Learning can stem from perception of an example as in explanation-based learning, perception of frequency of a recurring pattern of interaction, or perception of rewards for actions as in reinforcement-based learning. Learning can be *reflexive* in that the agent has no choice in when it learns as in Soar's chunking, or *deliberative*. Learning can be based on externally provided samples or instructions, or purely introspective. Learning may be incremental or one-shot.

Macrops [Fikes et al., 1972] were the earliest approach to learning in planning. Explanation Based Learning (EBL) is a descendant of Macrops style learning. Recently, Reinforcement Learning (RL) has had a revival in AI [Sutton, 1988] and has been applied to planning [Whitehead, 1992, Kaelbling, 1990, Lin, 1992].

3.1 What is learned

In the context of planning, an agent may learn one or more of the following: 1) preconditions (physical and epistemic) to when an act is applicable; 2) postconditions or effects of actions; 3) new acts; 4) local strategies for applying actions; 5) recurring sequences of actions, i.e., schemas; and 6) decisions for optimal/reliable selection of actions. In learning the first three things, we may assume that the agent does not have a *complete and correct* theory of its acts. In this case, the learning agent is gaining domain knowledge. Furthermore, an agent that learns new acts is learning new concepts. If the agent assumes it has a *complete and correct* theory of its acts, it may still learn pre- and postconditions by learning new relationships among its acts. In explanation-based learning, an agent may learn preconditions to its acts that are both derivable from its current theory of acts and supported by an example.

The latter three items in the above list most often concern learning domain knowledge about how to improve arbitration among applicable actions. RL combines learning pre- and postconditions with local strategies and decisions for optimal/reliable selection of actions (i.e., policies in RL terms).

3.2 Analytic: Explanation Based Learning (EBL)

Analytic learning is learning by reasoning. Analytic learners are typically cited in systems with rich domain knowledge. The bulk of work in analytic learning has converged on a technique known as explanation based learning [Mitchell et al., 1986, DeJong & Mooney, 1986]. EBL applications are found in [Segre & DeJong, 1985, Minton, 1988, Blythe & Mitchell, 1989, Gervasio & DeJong, 1989]. Explanation-based learning is learning from a single example. The key concepts in EBL are the four inputs and an output (adapted from [Mitchell et al., 1986]): goal concept– a concept definition describing the concept to be learned; training example– an example of the goal concept; domain theory– a set of rules and facts to be used in explaining how the training example is an example of the goal concept; operationality criterion– a predicate over concept definitions, specifying the form in which the learned concept definition must be expressed; and output– generalization of the training example that is a sufficient concept definition for the goal concept and that satisfies the operationality criterion.

We will illustrate EBL with an example (similar to the example in [Mitchell et al., 1986]). Goal concept: pairs of objects <X, Y> such that X can be safely stacked on top of Y. In predicate form we might represent this as **safeToStack(X, Y)**. X is safely stacked on top of Y if Y is not fragile or X is lighter than Y, **safeToStack(X, Y)** $\leftrightarrow$ **NOT fragile(Y)** or **lighter(X, Y)**. Training example: it is safe to stack book-1 on table-1, **safeToStack(book-1, table-1)**. Domain theory: book-1 is a book, **isBook(book-1)**; table-1 is a table, bf isTable(table-1); something weighs 4 pounds if it is a book, **weigh(X, 4)** $\leftarrow$ **isBook(X)**; something weighs 20 pounds if it is a table, **weigh(X, 20)** $\leftarrow$ **isTable(X)**; it is safe to stack X on Y if X is lighter than Y, **safeToStack(X, Y)** $\leftarrow$ **lighter(X, Y)**; it is safe to stack X on Y if Y is not fragile, **safeToStack(X, Y)** $\leftarrow$ **not fragile(Y)**; X is lighter than Y if X weighs X1 pounds and Y weighs Y1 pounds and X1 is less than Y1 pounds, **lighter(X, Y)** $\leftarrow$ **weigh(X, X1), weigh(Y, Y1), X** $<$ **Y**. Operationality criterion: The concept definition is expressed in terms of predicates **isBook** and **isTable**. EBL proves the training example as a query using deduction over the domain theory and deduction bottoms out in the atomic literals **isBook(book-1)** and **isTable(table-1)**. After replacing variables for constants, the result of EBL explanation boils down to **isBook(X)** and **isTable(Y)**.

A strong assumption in EBL is the assumption that the domain theory is complete and correct. This makes EBL unsuitable for applications where agents learn from their interaction with the environment. There are other analytic learning techniques that fall outside EBL. One example is work reported in [Christiansen et al., 1990]. This work describes a tray with blocks on it handled by a robot. A camera observes the state of the tray. In a loop, the robot forms plans for goals to have the blocks slide in a certain configuration by tilting. However, unlike EBL, the theory of tilting is assumed to be subject to errors. With each tilt, the robot's theory of the effects of tilting is updated. This is an example of learning action-postconditions.

Another example of analytic learning that is not based on EBL is [Shen, 1989]. Shen gives a system she calls LIVE for learning prediction rules of the form [percepts $\times$ action $\mapsto$ prediction]. The focus of rule learning is to learn effects of actions based on experimentation. Experiments try actions with respect to the agents current percepts and then observes which of its percepts persisted and any new percepts. This is an example of learning effects of actions. It also is an example of a symbolic solution to problems solved by RL (see next section).

3.3 Inductive: Reinforcement-Based Learning (RL)

Inductive learning is drawing inductive inferences from observed facts in the environment or by being told. A recent trend in inductive learning, especially as applied to planning, is reinforcement learning. Reinforcement learning is a trial-and-error process where, upon executing an action, applicability of actions to situations are updated based on the reinforcement signal received in response to the action. A style of reinforcement-based learning popularized by [Sutton, 1984] and followers is called Reinforcement Learning (RL). We make this distinction so we can talk about other systems that use reinforcement, but they are not strictly based on the the RL paradigm. [Maes & Brooks, 1990] is an example of reinforcement-based planning different from RL. Maes and Brooks describe a robot that learns to walk. They explored an application of learning to the subsumption architecture of behavior-based AI [Maes, 1990]. Their scheme differs from RL in that while RL only considers positive feedback, they considered negative feedback. In [Maes & Brooks, 1990] actions can be executed in parallel. Also, their algorithm works on reinforcing action-preconditions and not (state,action) pairs as in RL. This work is an example of learning action preconditions.

Another example of reinforcement-based learning that differs from RL is that of [Mani & Uhr, 1992]. Mani presents a system that deliberates about its actions based on a combination of perception and internal needs and then observes effects of actions, i.e., resulting states. It then strengthens the links between the starting state and the state that results when the action is performed. The reinforcement in this work measures the level to which an action satisfies a need, e.g., eating an apple satisfies hunger 80 percent of the time. Unlike the goal of learning (state, action) pairs in RL, Mani's goal is to learn action-postconditions, i.e., effects of actions.

In RL what is given is a set of *states* **S** (i.e., **S** is a set of labels for states) and a set of *actions* **A** (i.e., action names). Unknown is the *transition function* **T**. This is a function that tells us effects of actions, i.e., all transitions from states to states as a result of executing an action. Also unknown are the *rewards* **R** received by the agent from the environment as a result of executing actions. In RL terms, a *policy* is a mapping $\mathbf{S} \mapsto \mathbf{A}$ (i.e., a set of stimulus-response rules that for each state suggest an action to execute). For a given policy f, a *cumulative reward* **C** is used to measure the goodness of f. The *expected* **C** (**EC**) for a starting state x is defined to be the expected reward at that state plus a fraction of cumulative rewards at the next state y. The resulting state y is the transition policy f prescribes when in state x. Let $\mathbf{ER}_f(\mathrm{x})$ be expected reward in state x under policy f. Let $\gamma <$ denote the coefficient for computing the fraction of cumulative rewards. This is a factor for the relevance of previous rewards on the present situation. Then

$$\mathbf{EC}_f(\mathrm{x}) = \mathbf{ER}_f(\mathrm{x}) + \gamma \mathbf{EC}_f(\mathrm{y}), \text{ where } \mathbf{T}_f(\mathrm{x}) = \mathrm{y}.$$

The objective of RL is to find the optimal policy **f**.

An *action-value function* $\mathbf{Q}$(x,a) gives the expected **C** given that at state x, action a is chosen for execution. $\mathbf{Q}(\mathrm{x,a}) = \mathbf{ER}_f(\mathrm{x,a}) + \gamma\ \mathbf{EC}_f(\mathrm{y})$ where $\mathbf{T}_f(\mathrm{x,a}) = \mathrm{y}$. **Q** is also known as the evaluation function. When **Q** is used in an RL algorithm, the algorithm is called Q-learning [Watkins, 1989]. Q-learning is a delayed reinforcement learning technique.

A simple algorithm for RL is the policy iteration listed below ([Whitehead, 1992]).

```
f := an arbitrary initial policy
Repeat
   1. compute EC for f
   2. compute Q(x,a) for all x and all a
   3. update the policy for all x by choosing f(x) = a such that
      Qf(x,a) = max Qf(x,b) over all b in A
Until there are no changes in f at step 3
```

Since EC and Q depend on reinforcement signals received from the environment, they are not known *a priori*. In actual implementations of this algorithm, approximation techniques are used to speed up computation of EC and Q. McDermott [McDermott, 1991] gives a list of pairs of alternative assumptions for a feedback (i.e., reinforcement) on actions: deterministic or stochastic, stationary or non-stationary, binary or graded, immediate or delayed, memory-less or state-dependent.

General disadvantages to RL are twofold. First, reinforcements from the environment may mislead RL into learning coincidental effects and not true effects of actions. Second, computations of EC and Q are slow. This makes it unsuitable for most real-time applications. The closest thing to an application of RL in a physical setup appears to be [Lin, 1991] and [Mahadaven & Connel, 1991].

Work described in [Mahadaven & Connel, 1991] is an application of Q-learning in which a physical robot learns to find and push a cardboard carton around. Actions are grouped into subsumption behaviors *find box*, *push box*, and *unwedge box* with a given priority ordering among them. RL is applied to each behavior separately.

4 Integrated Approaches

Recent literature is full of suggestions for integrating different planning paradigms mediated by learning. Mitchell and his group are developing agents that integrate planning and learning [Blythe & Mitchell, 1989, Mitchell, 1990, Mitchell et al., 1991]. Mitchell's agents are initially classical planners. Using explanation-based learning (EBL), caching results, and a third learning strategy, agents form new stimulus-response rules to use in the future. Using learned rules, agents become faster. Another system that starts as a classical system and learns reactive rules using reasoning is that of [Gervasio & DeJong, 1989, Gervasio, 1990]. Gervasio suggests a form of EBL which she calls contingent EBL (CEBL). CEBL is shown to work on learning reactive rules over continuously changing situations. The key point of CEBL introduces *conjectured variables*. These are variables that do not have exact values until execution.

In contrast to the above, [Chien et al., 1992] is a system that starts out reactive and falls back on classical planning when reactivity proves fatal. Lyons and Hendriks describe a formal model in which the reactive module is controlled by the planner module. The planner develops a plan for a given goal and instructs the reactor to elaborate on substeps of the plan by considering pertinent reaction rules. This system is applied to assembling three parts on a tray. The tray and parts on it arrive on a conveyor belt. The planner constructs a high level plan to fetch the tray and then to assemble the parts. At the reactive module, each of these high level steps is decomposed into appropriate reactions.

5 A Hybrid System of Methods for *What to do Next* Tempered with Learning

Ideally, we wish to give an algorithm that models how an agent integrates various planning methods. However, such an algorithm will undoubtably be criticized for its simple-minded integration or else it will be questioned whether such an algorithm psychologically exists. Furthermore, we are interested in integration issues in general such as integration of reactive, planned, and experimental behavior. As such the integration depends heavily on the agent architecture. For example, a robot that is sent to gather rock samples has a goal and should act in a way to achieve its goal. It should not wander off exploring interesting things. It should however reactively interact with the environment in accomplishing its task. The upshot of this is that an important part of how an agent acts and how it decides what to do next is due to its architecture (i.e., its disposition to integration of behaviors).

A common and simple-minded architecture for an agent is most predominantly to decide on the next action based on *situation rules*. These are cached decisions of interaction with the environment that the agent has learned from a variety of sources such as direct experience and introspection. When such an agent receives a nontrivial goal, it reasons about its capabilities and produces a plan (i.e., a sequence of acts) and considers the first step of the plan for doing next. The action decided by a plan supersedes all other decisions to act. This is one way of prioritizing modes of behavior.

Two key elements for integration, invariant to agent architecture, are automaticity and perception. In addition to assumptions contained in the introduction to this paper, we make the following assumptions in our approach.

- Agents are assumed to possess a capability to treat acts less cognitively as they become routine, i.e., a capacity for increasing automaticity.

- Agents are assumed to have an internal state that retains recent perceptions along with acts performed and their outcomes.

Automaticity is the ability of the agent to treat activities less cognitively as they become routine. This is also a key factor for an agent's choice of method of interaction with its environment. Furthermore, we say automaticity is cognitively impenetrable, to use Pylyshyn's term. This means that this capability to acquire automatic behaviors is innate for the agent. Automaticity is a term used in human psychology. In our broader interest of presenting an integrated approach to planning methods, our reference to automaticity is somewhat limited. When we say a routine is automatic, we mean the routine is available for the agent (i.e., it is learned) and avoid issues about lack of conscious monitoring of execution of routines and so on typically associated with the term as used in human psychology.

Webster's Third New International Dictionary gives the following definition of 'perceive': "1a: to become conscious of, 1b: to recognize or identify, esp as a basis for or as verified by action, 2: to become aware of through the senses". At the lowest level, perception consists merely in locating an object in ones perceptual field, to see or feel or hear that *something or other* is *there*. At a higher level, we not only discriminate an entity from the perceptual background and locate it in perceptual space, but we also assign it to some category. We call both of these varieties of sensory-based perception, corresponding to Webster's second sense of 'perceive', *General Perception* (GP). It is possible to classify something as a hammer or a carburetor or a microchip without having any notion of the causal properties or intended functions of hammers, carburetors, or microchips. Even if we are aware of the causal properties of microchips, it is possible to notice that something is a microchip without thinking about those properties. We call the processes augmenting sensory-based perception, those processes which call to mind the causal properties or functions of things, *Extended Perception* (EP). This kind of perception roughly corresponds to Webster's sense of 'perceive' labeled '1b'. GP is responsible for noticing features of entities and changes in features of entities in the agent's environment in an ongoing basis and on demand. GP does not use the agent's causal theory about the environment. EP involves prediction/envisionment of events or recognition of causal relationships between the entities in the agent's environment and the agent. Notice that EP can take place at least in part independently of the classification function of GP, but location of an entity in perceptual space is a necessary precondition for EP. For example, one can perceive a hammer as *that*-thing-which-can-be-used-to-drive-*this*-nail even if one lacks the concept of a hammer. One could also perceive a brick as *that*-thing-which-can-be-used-to-drive-*this*-nail although this is not a normal function of bricks. EP receives as input the output of GP, whereas GP receives its inputs directly from the sensors.

GP and to a greater extent EP are the faculty by which an agent assesses the environment to determine the current *situation*.

5.1 Situations and Perceptions

An agent perceives itself to be in various *situations*. Put another way, situations are what agents perceive that serve as parameters in methods for deciding *what to do next*. Situations are products of perception. However, perception generates more than situations. Perception generates percepts used in learning and in forming theories about the environment.

While parallels can be drawn between what we mean by situations and the situations defined by Barwise and Perry in [Barwise & Perry, 1983] and other works, there is at least one essential difference. Crucial to our notion is the idea of *situatedness*. By 'situation', we mean something that an agent perceives from the perspective of being *in* it, whether or not we would say that the agent has an awareness of self. An agent might *see* the situation to be one where there is *something-or-other* directly in front of it or might *feel* the situation to be one where there is *something-or-other* beneath its hand (both instances of GP). More information might be perceived: that the thing in front of the agent is a brick (still an instance of GP) or that the thing in front of the agent is something-that-can-be-used-to-drive-*this*-nail (again, a case of EP).

As another example, in a grocery store some parts of the situation that the agent perceives are that there are checkout lines, how long the lines are, and how fast the lines move. Still another example is an agent that has interacted with doors only in daylight. When it confronts a new but typical door in the dark that it has to unlock, it perceives it to be a door, assesses it to be stationary, and as it tries to find the door lock, it perceives it to be more difficult to open the door in the dark than it is in daylight.

We first consider types (functionalities) of EP used in situations. One form of EP is subjectifying objective representations of objects. We call this perceive-subjective-entities/aspects. Agre and Chapman [Agre & Chapman, 1987] introduced indexical-functional representations as the agent's subjective view of entities and relationships between entities and agents. For instance, when an agent sees an apple in its field of view, the apple is an objective entity, i.e., it exists independently of the agent. But when the agent interacts with the apple, it may become the-apple-I-am-eating if the agent is eating the apple. The process of forming this representation, or focusing of attention is extended perception. EP of this type produces typical situations.

Perceive-potential-actions is a form of EP by which the agent develops tendencies to do certain actions. When an agent finds itself in a situation, it assesses the situation with respect to its goals and any actions available to it. For an agent that finds itself in a machine shop with the aim of building a widget, certain actions like sawing the raw parts suggest themselves over other actions like polishing the part. Here we assume that the agent has never built a widget before but nevertheless it can foresee that perhaps sawing the parts is a good starting point. This type of perception is related to what James Gibson called *affordances* [Gibson, 1979]. Gibson defines affordances of the entities in the environment as how they allow the animal to interact with them. For example, a table-top affords objects like books to be laid upon it. Perceive-affordances is discussed by Gibson and we include it as a form of EP. Perceiving the affordances with the most potential is preliminary to perceive-potential-actions.

Perceive-a-goal is an EP that might be pursuant to perceive-affordances. Perceive-the-need-to-plan is often pursuant to this EP. However, the latter does not lead to a situation. Perceive-success/failure-of-planned-action-performed is another EP. Observing results of executing actions are interpreted in light of the action expectations. If they are consistent, the action is successful;

otherwise, failed.

When an agent has successfully achieved a goal, perceive-achievement-violations is an EP which alarms the agent about any threat of the goal coming 'undone' or of an event having undone the goal. This is often accompanied by perceive-the-need-to-react after which the agent uses situation rules to choose an action quickly. The latter is a form of extended perception where the agent bypasses goal formation and directly proceeds to act. For instance, when a moving agent confronts the wall, there is no need to form a goal to avoid the wall. An immediate collision avoidance reaction will often suffice. Perceive-the-need-to-react appears to rely on a quick analysis of urgency of the situation and availability of reactions. Since it is common in subcognitive actions, i.e., reactive actions, it too appear to be at a low-cognitive level. By this we mean that this form of EP is like behavior-based modules of subsumption architecture [Brooks, 1985] where each behavior is composed of a perception and action sub-modules. In behavior-based modules, each module independent of others becomes active when its corresponding perception submodules are triggered. Note that this EP may in fact be distributed among many modules and if any behavior becomes active, this EP is active.

We now turn to EP that does not produce situations. Several forms of EP are perceptions that span over time. An agent interacting with its environment may produce interesting, unexpected behaviors. These are what Agre calls emergent behaviors. An agent may perceive the long-range effects of its actions, or emergent behaviors. The agent can then cache and/or generalize these observations for future use. In this way, the agent learns routines or recurrent patterns of interaction with its environment that directly stem from its perception. We call this form of extended perception *perceive-routines*. This type of EP appears to be highly cognitive. Although this type of EP often seems to take place sub-consciously and suddenly, the agent may notice the result consciously. EP of this type is found in experimentation (trial and error) tasks. The difference between EP in experimentation tasks and other tasks is that often the agent is more conscious about its EP and is aiming to find the interesting behaviors as a result of combining its primitive actions.

Perceive-reactions is another type of EP that involves perception over a time interval. It follows perceive-success/failure and either perceive-the-need-to-plan or perceive-the-need-to-experiment. When an action prescribed by a plan or an experiment is successfully executed, the agent may perceive that under the situations given by the action's precondition, the context of execution, and with respect to a goal, the action is appropriate and form a rule to this effect. Perceive-reactions is similar to EBL of reaction rules [Blythe & Mitchell, 1989, Mitchell, 1990].

Perceive-reinforcement is another type of EP that spans a time interval. This kind of EP assigns credit/penalty to reaction rules, plans, or experiments associated with a goal. This type of EP supports reinforcement-based learning. However, unlike the RL paradigm it is not based on (situation,action) pairs but is goal/strategy oriented. We believe that the RL method is too fine-grained, too time consuming, and can be misleading.

Perceive-the-need-to-experiment is a type of EP where the agent falls into an experimentation mode. It is a common experience to have keys that don't fit the lock right; to encounter an overhead transparency screen that won't stay in the extended position when pulled down; or not to know which way to turn the blind/curtain rods. In these examples, the agent automatically falls into an experimentation mode where she tries variations of her acts. These examples suggest a subcognitive level for this EP. However, there are other examples where the agent may choose to operate cognitively. These tend to be experimentations to gain knowledge. For example when a person receives a new toy, she deliberately experiments with the toy.

It has been argued that routine activity in everyday life is improvisational. However, there are occasions when the agent may routinely decide on optimizing its actions by carefully planning them before executing them. Instances include when the agent wants to conserve its resources, to minimize repeating an action such as in a chess game, or to avoid a large-mud-puddle on the sidewalk. Planning, reacting, and experimenting are competing modes of behavior. Perceive-the-need-to-plan is an EP that like planning (here we mean deductive reasoning) is highly-cognitive.

We think there are many other types of EP arising out of prior goals and internal needs which must be built into an agent, but we will not explore them in this paper.

5.2 Integrating Modes of Behavior

We cited planning, reacting, and experimenting as three modes of behavior. Perceive-the-need-to-react and perceive-the-need-to-plan are mutually exclusive and at different levels of cognition. Having decided on which to choose, a reactive or a planning strategy becomes applicable. It is the task of perception to perceive adequate clues from the environment to decide between this pair. Perception is responsible for perceiving factors like the dynamics of the environment and the criticality of bad decisions about acting. For example, playing chess with a novice player versus a chess-master can be perceived to require different strategies for producing actions.

Similarly, perception has to provide information to arbitrate between perceive-the-need-to-plan and perceive-the-need-to-experiment. Of course, we are not suggesting that the agent has an explicit arbitration scheme. Once adequate information is provided, only one of these two strategies become dominant. Perceive-the-need-to-experiment and perceive-the-need-to-react can coexist. For example, playing a game of Pengo (Agre and Chapman's video-game [Agre & Chapman, 1987]) with a slow bee may trigger perceive-the-need-to-experiment. Once triggered, experimentation may suggest a strategy of successively kicking blocks to find the one that kills the bee. If the bee becomes a bit faster, perceive-the-need-to-react is also triggered to flee from the bee while still experimenting. If the bee becomes much faster, perceive-the-need-to-react completely dominates. The strategies for action generating is embodied in interaction between methods for deciding *what to do next* and a knowledge base of actions. The methods we have cited are reaction, planning, and experimentation. Let us call the component containing these methods *Methods for Actions* (MA).

Information may flow from one component to another with the dominant flow of information being from GP to EP to MA. Each component operates independently and no one controls the other. GP and EP may be directed by MA or vice versa. Internal workings of an agent that operates based on these general components can be observed at each instant by instantaneous examination of the contents of GP, EP, MA and an Internal State (IS). Let's examine a few examples.

Example 1: keyhole in the dark– GP: standing in front of the door holding the key, static environment. EP: failure to find the keyhole, perceive-the-need-to-experiment. MA: experiment by successive refinement in the proximity of keyhole. IS: goal of key in keyhole, perceive-a-goal.

Example 2: mud-puddle on sidewalk– GP: standing in front of the mud-puddle, static environment. EP: perceive-a-goal, perceive-the-need-to-plan. MA: plan a path around the mud-puddle results in the first act of veering right. IS: null.

Example 3: chess move– GP: a chess-board configuration. EP: perceive-a-goal, perceive-the-need-to-react. MA: react by a standard retreat motion. IS: Agent is unable to form a plan due to a check and a threat of check-mate.

It might occur to the reader that we intend to recast traditional problems of integrating methods for deciding *what to do next*, in our terms of extended perceiving. To some extent this is correct. Starting off with EP gives us a vantage point for unifying most recent reactive planning paradigms. Using EP, we can view an agent's interaction with its environment to take place in a continuum of levels of cognition/consciousness. EP itself spans this continuum. At the less cognitive end of the spectrum, EP leads to actions directly without the interveining formation of goals and conflicts among actions and such. At the more cognitive end of the spectrum, EP leads to reasoning about actions and from this a form of classical planning naturally emerges. A feature of extended perception and of perception in general is that it is ongoing. It continually updates the agent's internal state to reflect the impact of changes in the environment.

5.3 Perceptuo-Motor Automata

We define a perceptuo-Motor automata (PMA) as a finite state machine in which each state is associated with an act and arcs are associated with perceptions. In each PMA, a distinguished state is used to correspond to the no-op act. Each state has associated with it some action or process which may be either 'one-shot' or continuing: pushing-button, walking, chewing, etc. Each state also contains an auxiliary part we call its Internal State (IS). The IS is used in arbitrating among competing arcs. IS at each state with competing arcs contains a method for deciding how much EP to employ in arbitrating among competing arcs. This might be a time limit or a condition. Arcs in a PMA are situations that the agent perceives in the environment. These perceptions range from GP to EP. When there are competing arcs, EP is used to arbitrate among them. Since perception is a continuous function over time, the arcs are designed to be anytime algorithms. In contrast to one-shot algorithms, these algorithms improve over time. When a PMA arc emanating from a state becomes active, it behaves like an asynchronous interrupt to the act in execution in the state. This causes the PMA to stop executing the act in the state and to start executing the act at the next state at the end of the arc connecting the two states. This means that in our model the agent is never idle, and it is always executing an act. We will illustrate the operation of PMA with two examples:

B Not Last ([Drummond, 1989]): Before the agent is a table on which the agent is to assemble three blocks in a row: block A on the left, at location 1; block B in the middle, at location 2; and block C on the right, at location 3. The blocks are initially available for placement, and each block can be placed on the table once it's available. The exact means for moving the blocks does not matter: when a block is available it may be placed. The only constraint on assembly is that *block B cannot be placed last*: once A and C are down, there is not enough room to place B. B must be swept in from the left or from the right, and cannot be placed if A and C are in the way.

We build a PMA for this problem with four acts: placeA, LplaceB, RplaceB, placeC, and our start state no-op. On the arcs we have perceptions free1, free2, free3, availableA, availableB, and availableC. State transitions here are triples of the form [current state, perceptions, next state]:

- [no-op, (free1, availableA), placeA]
- [no-op, (free2, free3, availableB), RplaceB]
- [no-op, (free1, free2, availableB), LplaceB]
- [no-op, (free3, availableC), placeC]
- [placeA, (free2, free3, availableB), RplaceB]
- [placeC, (free1, free2,availableB), LplaceB]
- [RplaceB, (free3, availableC), placeC]

- [LplaceB, (free1, availableA), placeA]
- [LplaceB, (free3, availableC), placeC]
- [RplaceB, (free1, availableA), placeA]

In this example, IS in each state other than no-op is the conditions perceived in arcs leading to the state and the previous state. IS at placeA is free1, availableA, and one of (no-op, RplaceB, LplaceB). IS at RplaceB is free2, free3, availableB, and one of (no-op, placeA). IS at LplaceB is free1, free2, availableB, and one of (no-op, placeC). IS at placeC is free3, availableC, and one of (no-op, LplaceB, RplaceB). IS can be interpreted as the situation under which an act is applicable. This can be used to rewrite the PMA in terms of *situation rules* where the antecedent is a situation (i.e., conditions on arcs and the previous state) and the consequent is an act. The resulting rules are similar to Drummond's situation control rules [Drummond, 1989].

Waiting queues: There are two queues in parallel heading for a ticket booth. Let's assume three acts are available: walking straight in the current queue (W), switching to the right queue (R), and switching to the left queue (L). With each act we will associate a state. Let's have a starting state corresponding to the no-op act. IS in state W contains which state (R or L) was visited last. Arcs are the following perceptions: visited L last, visited R last, current queue is okay, left queue is better, right queue is better. The state transitions are triples of the form [current state, perceptions, next state]:

- [no-op,null,L]
- [L,right queue is better,R]
- [R,left queue is better,L]
- [L,current queue is ok,W]
- [R,current queue is ok,W]
- [W,(visited L last,right queue is better),R]
- [W,(visited R last,left queue is better),L]

An agent starting at the no-op will join the left queue. If the queue seems to move with normal speed, the agent will walk down the line. Otherwise, the agent will switch queues. But what if there was a barrier between the queues so once a queue is chosen, no switching is allowed? In this case, R or L can be picked only once. With a barrier, we will add to the IS in no-op the condition to pick the significantly shorter line or, if the two lines are about the same length, to pick the line that moves faster. Now the agent starting at the no-op state will spend more time in choosing a queue. This example shows the larger role of EP.

5.4 Goal Perceptuo-Motor Automata

Goal Perceptuo-Motor Automata (GPMA) is a PMA constructed from a plan produced by a classical planner. IS in GPMA contains the problem goal (i.e., the goal for which the plan is constructed.)

Blocks world: We are given a table on which to assemble a tower of three blocks A, B, and C in the configuration of on(A,B), on(B,C), on(C,table). Initially, the blocks are in the on(C,A), on(A,table), on(B,table) configuration. This problem is solved by the plan clear(A) followed by puton(B,C) followed by puton(A,B) using classical planning. Here puton(X,Y) and clear(X) are the only two acts available. In classical planning terms, preconditions for puton(X,Y) are isClear(X) and isClear(Y) and the precondition for clear(X) is on(-,X) (i.e., something is on X).

A GPMA is built for this problem by making each of the acts in the plan into a state. The necessary states are puton(A,B), puton(B,C), clear(A) and the starting state no-op. Arcs between states are given by the act preconditions. The incoming arcs to a state are the preconditions of the act at the state:

- [no-op, on(-,A), clear(A)]
- [clear(A), (isClear(B), isClear(C)), puton(B,C)]
- [puton(B,C), (isClear(A, isClear(B)), puton(A,B)]

IS contains the previous state and the incoming arc to the state as well as the problem goal. Note that the resulting problem can be rewritten using situation rules of the form [situation $\times$ goal $\mapsto$ act].

The GPMA constructed here is incomplete in the sense that not all conditions are accounted for. What if we execute clear(C) and perceive on(C,B)? No arcs apply. The problem goal has not changed; so we plan again using the current situation and add additional states to GPMA (i.e., states corresponding to additional acts in the new plan) and add the missing links to the GPMA. In the case of on(B,C) we add the state clear(C), and transitions [no-op, on(-,C), clear(C)], and [clear(C), (isClear(B), isClear(C)), puton(B,C)] to our GPMA. Since plans for a goal can be compiled into situation rules, a plan is decomposed into local decisions for action. All subsequent plans for the same goal also decompose into local decision rules for action. Therefore, all subsequent plans for a goal only augment the original plan and are consistent with it.

But what if when we execute clear(A), we perceive on(B,C). This is the result of puton(B,C) (i.e., postcondition) but we never got to execute it. Furthermore, once again no arcs in clear(A) apply. If we planned for this new situation and added the arc [clear(A), (isClear(A), isClear(B)), puton(A,B)], that would be misleading. At this point, if we started over with the same initial conditions and the problem goal as before and reached clear(A), there would be two arcs, one to puton(A,B) and another to puton(B,C), that will apply. The solution to this problem is to add additional arcs corresponding to skipping plan steps after constructing a GPMA for a plan . We call this *transitive completion* of the GPMA and it proceeds as follows. Between any three successive steps of the plan, corresponding to three successive states S1, S2, S3, add additional arcs between states S1 and S3 such that the transition is the union of the preconditions of the act in S3 and postconditions of the act in S2. In our example plan, we would add transitions [no-op, isClear(A), puton(B,C)], [no-op, (isClear(A), on(B,C), isClear(A), isClear(C), puton(A,B)], [clear(A), (isClear(A, isClear(B), on(B,C)), puton(A,B)] to our GPMA.

6 Concluding Remarks

Our a brief review of recent planning trends and related learning techniques convinces us that what is needed is an integration of planning and learning methods suitable for deciding *what to do next*. We suggest that the gap between perceiving and acting is sometimes completely and always at least partially filled by extended perception - prediction/envisionment processes that follow general perception. We have explored the role of perception in acting in general and in producing situations used in acting. The role of perception and automaticity in the integration of modes of behavior based on reaction in planning is crucial here. Our approach relies on the predominant role of the interaction between the agent and its environment in suggesting what the agent does from moment to moment. We introduced a finite state machine representation for modeling continuous reconsideration of acts (i.e., reactive behavior) using perception. This provides one way that plans can be cached into structures for reactivity. Our initial investigation provides a conceptual framework for further research in the integration of classical and reactive planning methods into systems that can sometimes determine and other times learn *what to do next.*

References

[Agre & Chapman, 1990] Agre, P. & Chapman, D. (1990). What are plans for? In P. Maes (Ed.), *Designing Autonomous Agents* chapter 2, (pp. 17–34). MIT Press.

[Agre & Chapman, 1987] Agre, P. E. & Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87, Seattle, Wa.* (pp. 268–272).

[Barwise & Perry, 1983] Barwise, J. & Perry, J. (1983). *Situations and Attitudes.* MIT Press.

[Blythe & Mitchell, 1989] Blythe, J. & Mitchell, T. (1989). On becoming reactive. In *Machine Learning Workshop* (pp. 255–257).

[Brooks, 1985] Brooks, R. (1985). *A Robust Layered Control System for a Mobile Robot.* Technical Report 864, MIT AI Labs, MIT.

[Brooks, 1987] Brooks, R. (1987). *Planning is Just a Way of Avoiding Figuring Out What to do Next.* Technical Report 303, MIT AI Labs.

[Brooks, 1990] Brooks, R. A. (1990). Elephants dont play chess. In P. Maes (Ed.), *Designing Autonomous Agents* (pp. 3–15). MIT Press.

[Chien et al., 1992] Chien, S., Gervasio, M., & DeJong, G. (1992). Learning to integrate reactivity and deliberation in uncertain planning and scheduling problems. In *AAAI Spring Symposium.*

[Christiansen et al., 1990] Christiansen, A., Mason, M., & Mitchell, T. (1990). Learning reliable manipulation strategies without initial physical models. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 1224–1230).

[DeJong & Mooney, 1986] DeJong, G. & Mooney, R. (1986). Explanation-based learning: An alternative view. In *Machine Learning 1* (pp. 144–176).

[Drummond, 1989] Drummond, M. (1989). Situated control rules. In *Proceedings Of the first conference on principles of knowledge representation* (pp. 103–113).: Morgan Kaufman.

[Fikes et al., 1972] Fikes, R., Hart, P., & Nilsson, N. (1972). Learning and executing generalized robot plans. In *Artificial Intelligence 3.*

[Fikes & Nilsson, 1971] Fikes, R. E. & Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5, 189–208.

[Firby, 1987] Firby, R. J. (1987). An investigation into reactive planning in complex domains. In *Proceedings of AAAI-87.*

[Georgeff & Lansky, 1987] Georgeff, M. & Lansky, A. (1987). Reactive Reasoning and Planning. In *Proceedings of AAAI-87* (pp. 677–682).

[Gervasio, 1990] Gervasio, M. (1990). Learning general completable reactive plans. In *Proceedings of AAAI-90* (pp. 1016–1021).

[Gervasio & DeJong, 1989] Gervasio, M. & DeJong, G. (1989). Explanation-based learning of reactive operators. In *Machine Learning Workshop* (pp. 252– 254).

[Gibson, 1979] Gibson, J. (1979). *The Ecological Approach to Visual Perception.* Lawrence Erlbaum.

[Kaelbling, 1986] Kaelbling, L. (1986). An architecture for intelligent reactive systems. reasoning about actions and plans. In M. G. nd A. Lansky (Ed.), *Reasoning about Actions and Plans.* Morgan Kauffman.

[Kaelbling, 1988] Kaelbling, L. (1988). Goals as parallel program specifications. In *Proceedings of AAAI-88*: Morgan Kaufman.

[Kaelbling, 1990] Kaelbling, L. (1990). *Learning in Embedded Systems.* PhD thesis, Stanford University. Teleos TR 90-04.

[Kaelbling & Rosenschein, 1990] Kaelbling, L. & Rosenschein, S. (1990). Action and planning in embedded agents. In P. Maes (Ed.), *Designing Autonomous Agents* (pp. 35–48). MIT Press.

[Lin, 1991] Lin, L.-J. (1991). Programming robots using reinforcement learning and teaching. In *Proceedings of AAAI-91* (pp. 781–786).

[Lin, 1992] Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning, and teaching. In *Machine Learning 8* (pp. 293–321).

[Maes, 1990] Maes, P. (1990). Situated agents can have goals. In P. Maes (Ed.), *Designing Autonomous Agents* chapter 4, (pp. 49–70). MIT Press.

[Maes, 1991] Maes, P. (1991). Action selection. In *Proceedings of Cognitive Science Society Conference.*

[Maes & Brooks, 1990] Maes, P. & Brooks, R. (1990). Learning to coordinate behaviors. In *AAAI-90* (pp. 796–802).

[Mahadaven & Connel, 1991] Mahadaven, S. & Connel, J. (1991). Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Maching Learning Workshop* (pp. 328–332).

[Mani & Uhr, 1992] Mani, G. & Uhr, L. (1992). Acquiring rules for need-based actions aided by perception and language. In *Proceedings of the 14th Annual Cognitive Science Conference.*

[McDermott, 1978] McDermott, D. (1978). Planning and acting. In *Cognitive Science Vol. 2.*

[McDermott, 1991] McDermott, D. (1991). *Robot Planning.* Technical Report CS-861, Yale University.

[Minton, 1988] Minton, S. (1988). *Effective Search Control Strategy: An Explanation-Based Approach.* Technical Report CMU-CS-88-133, Carnegie Mellon University.

[Mitchell et al., 1991] Mitchell, T., Allen, J., Chalasani, P., Cheng, J., Etzioni, O., Ringuette, M., & Schlimmer, J. (1991). Theo: A framework for self-improving systems. In K. VanLehn (Ed.), *Architectures for Intelligence* (pp. 323–356). Lawrence Erlbaum.

[Mitchell, 1990] Mitchell, T. M. (1990). Becoming increasingly reactive. In *Proceedings of AAAI-90* (pp. 1051–1058).: AAAI.

[Mitchell et al., 1986] Mitchell, T. M., Keller, R., & Hedar-Cabelli, S. (1986). Explanation based generalization: A unifying view. In *Machine Learning 1*, volume January 86 (pp. 47–80).

[Sacerdotti, 1977] Sacerdotti, E. D. (1977). *A Structure for Plans and Behavior*. New York, NY: Elsevier North Holland.

[Schoppers, 1987] Schoppers, M. J. (1987). Universal plans for unpredictable environments. In *Proceedings 10th IJCAI* (pp. 1039–1046).

[Segre & DeJong, 1985] Segre, A. & DeJong, G. (1985). Explanation-based manipulator learning: Acquisition of planning ability through observation. In *IEEE Robotics and Automation* (pp. 555–560).

[Shapiro et al., 1988] Shapiro, S. C., Kumar, D., & Ali, S. S. (1988). A propositional network approach to plans and plan recognition. In *Proceedings of the AAAI-88 Workshop on Plan Recognition*: Morgan Kaufmann.

[Shen, 1989] Shen, W.-M. (1989). *Learning from the Environment Based on Actions and Percepts*. PhD thesis, Carnegie Mellon University.

[Sussman, 1973] Sussman, J. (1973). *A Computational Model of Skill Acquisition*. Technical Report TR-297, AI Labs, MIT, Cambridge, MA.

[Sutton, 1984] Sutton, R. (1984). *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts.

[Sutton, 1988] Sutton, R. (1988). Learning to predict by the methods of temporal differences. In *Machine Learning 3* (pp. 3–44).

[Tate, 1977] Tate, A. (1977). Generating project networks. In *Proceedings 5th IJCAI* (pp. 888–93).

[Watkins, 1989] Watkins, C. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.

[Whitehead, 1992] Whitehead, S. (1992). *Reinforcement Learning for the Adaptive Control of Perception and Action*. PhD thesis, University of Rochester. Technical Report 406.

[Wilkins, 1984] Wilkins, D. E. (1984). Domain-independent planning: Representation and plan generation. *Artificial Intelligence*, 22.