Applied Intelligence 11, 109–127 (1999)

A Hybrid Architecture for Situated Learning of Reactive Sequential Decision Making

RON SUN
NEC Research Institute and University of Alabama, 4 Independence Way, Princeton, NJ 08540
rsun@cs.ua.edu

TODD PETERSON AND EDWARD MERRILL
The University of Alabama, Tuscaloosa, AL 35487

Abstract. In developing autonomous agents, one usually emphasizes only (situated) procedural knowledge, ignoring more explicit declarative knowledge. On the other hand, in developing symbolic reasoning models, one usually emphasizes only declarative knowledge, ignoring procedural knowledge. In contrast, we have developed a learning model CLARION, which is a hybrid connectionist model consisting of both localist and distributed representations, based on the two-level approach proposed in [40]. CLARION learns and utilizes both procedural and declarative knowledge, tapping into the synergy of the two types of processes, and enables an agent to learn in situated contexts and generalize resulting knowledge to different scenarios. It unifies connectionist, reinforcement, and symbolic learning in a synergistic way, to perform on-line, bottom-up learning. This summary paper presents one version of the architecture and some results of the experiments.

Keywords: hybrid models, sequential decision making, neural networks, reinforcement learning, cognitive modeling

1. Introduction

We will present a hybrid connectionist model that unifies connectionist, symbolic, and reinforcement learning into an integrated and coherent architecture. The model is named CLARION, which stands for *Connectionist Learning with Adaptive Rule Induction ON-line*. In this summary paper, we will emphasize the motivations for developing this architecture, summarize some experimental results, and discuss the advantages of the model.

We will focus on reactive sequential decision tasks, which involve selecting and performing a sequence of actions to accomplish an objective on the basis of moment-to-moment perceptual information (hence the term "reactive"). In general, there is an agent (or a set of agents) that can select (from a finite set of actions) an action to perform at each time step (in either a discrete or a continuous sense). The selection decision is (mainly) based on the current state of the world. The world is presented to the agent, through sensory input, as a state vector which contains various pieces of information (with some correlated and some uncorrelated). The world changes either autonomously or as a result of some action by an agent. Thus, over time, the world is presented to an agent as a sequence of states.[1] At certain points in a sequence, the agent may receive *payoffs* or *reinforcements* for their actions performed at or prior to the current state. Thus, the agent may need to perform *credit assignment* in learning, to attribute the payoffs/reinforcements to actions at various points in time (the temporal credit assignment problem; [1]), in accordance to various aspects of a state (the structural credit assignment problem). There is in general no teacher input. The agent starts with little or no a priori knowledge.

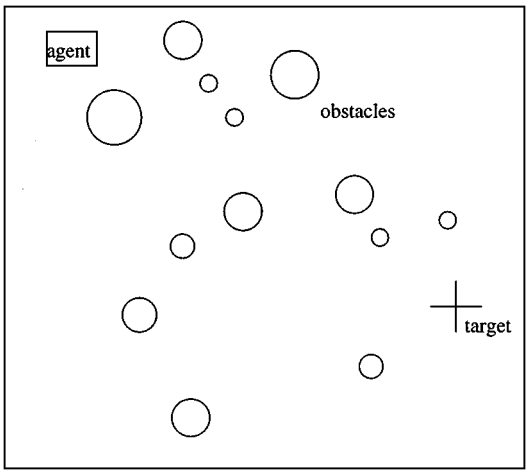


Figure 1. Navigating through a minefield.

One example involves learning to navigate through mines (see Fig. 1; [2]). There is one agent and a target to be reached. Between the agent and the target there are many mines that can destroy an agent on contact. The agent has only local sensory information—short-range rudimentary vision that allows it to see a small section of space in front of it. The agent has to reach the target in a limited amount of time. Another example is learning to navigate a maze (see Fig. 2; [3]). An agent has to start in one cell of the maze and navigate to reach another, predesignated destination cell. It has limited sensory input consisting of local information about adjacent cells. The agent has no knowledge of its actual current location and previous locations. It can move forward, turn left or turn right, based on its current input.

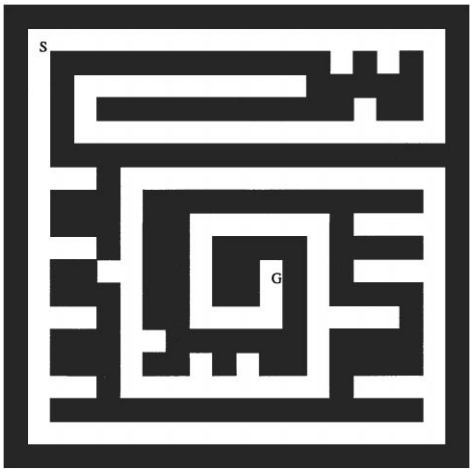


Figure 2. Running a maze.

While performing this kind of task, the agent is under time pressure: Often a decision has to be made in a fraction of a second; it cannot do much of "information processing", and falls outside of Allen Newell's "rational band" (i.e., cognitive processes that take minutes or hours to complete, which is what AI traditionally deals with; [4]). As in humans, the agent may also be severely limited in other resources, such as memory, so that memorizing and analyzing all the previous episodes in detail is impossible (although some form of episodic memory may exist).[2] The perceptual ability of the agent may also be limited (as in humans), so that only local information is available. In addition, learning in such tasks is an experiential, trial-and-error process; the agent develops skills, concepts, and rules *tentatively* on an on-going basis, because it cannot wait until the end of an episode (a sequence) before making a decision and starting to learn.[3] In general, as pointed out by Nosofsky et al. [5] and others, human learning is mostly gradual, on-going, and concurrent (on-line), which is especially true in this type of task. Goals may not be explicit and a priori to an agent either. They may be implied in reinforcements/payoffs received and pursued by an agent as a side-effect of trying to maximize payoffs (for example, an agent can maximize the average or total payoffs/reinforcements received, or maximize a recency-weighted total.)

The characteristics of a world need not be stationary from the viewpoint of the learning agent—the world may be "drifting". It can be nonstationary in several ways: (1) the world can change over time; thus, the revision of knowledge learned by an agent may be necessary (if changes are substantial enough). (2) Even when the world per se is stationary, it may still seem uncertain and evolving to an agent learning to cope with the world, because different regions of the world may exhibit different characteristics and thus require the revision of concepts over time [6]. In general, there is no preselected set of (positive and negative) instances (and a closed world assumption) that provide a fixed view of the world. (3) Once a knowledge structure is revised, the agent has to view whatever it experienced before in new ways (because knowledge provides a "filter" through which the agent sees the world), and thus the experience may seem different and the world nonstationary. (4) In experience-driven learning based on reinforcement, there is a lack of a clear and steady criterion for learning. Payoffs may be received sporadically,

and it is up to the agent to decide what to make of them. The agent has to assign credits/blames on the basis of what is already known, which is constantly changing. So the learning criterion is a moving target, and the learning process becomes nonstationary.

To acquire low-level specific skills in these tasks, there are some existing methods available. Chief among them is the temporal difference method [7], a type of reinforcement learning that learn through exploiting the difference in evaluating actions in successive steps and thus handling sequences in an incremental fashion. This approach has been applied to learning in mazes, navigation tasks, and robot control [8–12]. Another approach, genetic algorithm [13], can also be used to tackle this kind of task [2, 14, 15]. They handle the temporal credit assignment problem and "drifting" worlds. But they do not distinguish between procedural and declarative knowledge.

In terms of learning declarative knowledge or rules for such tasks, however, the afore-identified characteristics of the task render most existing algorithms inapplicable, because they require either preconstructed exemplar sets [16–18], incrementally given consistent instances [19–22], or complex manipulations of learned structures when inconsistency is discovered (which is typically more complex than the limited time a reactive agent may have; [23]).[4] "Drifting" as analyzed above is clearly more than noise and inconsistency as considered by some learning algorithms (e.g., [24]), because it involves changes over time and it leads to radical changes in learned knowledge. It often requires extra dedicated mechanisms [6]. Above all, most of the rule learning algorithms do not handle the learning of sequences which necessarily involves temporal credit assignment.

In the remainder of this paper, we will highlight a hybrid model, CLARION, for this type of task (in Section 2). Then we will discuss briefly several sets of experiments using the model, which demonstrate the advantages of the model (Section 3). CLARION is then compared with other models (Section 4). Some concluding remarks end the paper (Section 5).

2. Hybrid Models

Below we will first discuss (in Sections 2.1–2.2) the motivations for the model (parts of which have been published before; see e.g., [25, 26]). Then, in Sections 2.3–2.7, we will discuss some details of the model, which mainly involve a rule learning algorithm that is different from the previously published algorithms for the model (cf. [25, 26]).

2.1. Procedural and Declarative Knowledge

How can an agent develop a set of skills that are highly specific (geared towards particular situations) and thus highly efficient but, at the same time, acquire sufficiently general knowledge that can be readily applied to a variety of different situations and be communicated to others? Although humans seem to possess such abilities and be able to achieve an appropriate balance between the two sides, existing systems fall short. What appears to be missing is the duality and coexistence of both procedural and declarative knowledge (or both specific skills and generic knowledge). There has been a great deal of work demonstrating the difference between procedural knowledge and declarative knowledge: e.g., [27–32]. It is believed that a balance of the two is essential to the development of complex cognitive agents. This is based on two lines of argument. First, there are ample psychological data that support the distinction between procedural and declarative knowledge and the need for both. Anderson [33] initially proposed the distinction between declarative and procedural knowledge based on such data; Fitts and Posner [34], Keil [30], and Sun [32] subsequently made similar points based also on psychological data. Second, there are many philosophical arguments for making this distinction and achieving an appropriate balance of the two. Dreyfus and Dreyfus [35] proposed the distinction of analytical and intuitive thinking; Smolensky [36] proposed the distinction between conceptual (publicly accessible) and subconceptual processing; in addition, the distinction between conscious and subconscious processes, although controversial, is well known [37, 38]. The inadequacy of systems that ignore these points in dealing with the full range and complexity of intelligent behaviors lends additional support for these arguments.[5]

Declarative knowledge has some advantages which make it indispensable to a learning agent despite the fact that procedural skills are more efficient or easier to learn.

- It helps to guide the exploration of new situations, and reduces the time (i.e., the number of trials) necessary to develop specific skills in new situations. In other words, it helps the transfer of learned skill (as shown psychologically by Willingham et al. [39]).

- It can help to speed up learning. If properly used, declarative knowledge that is extracted on-line during skill learning can help to facilitate the learning process itself.
- Declarative knowledge can also help in communicating learned knowledge and skills to other agents.

Because declarative knowledge is usually easily accessible, it constitutes the conceptual knowledge base in an agent. On the other hand, procedural knowledge, or specific skills, is usually not conceptually accessible (i.e., impenetratable): there is generally a lack of conceptual-level thinking in performing procedural skills; details of such skills are in general inaccessible to self introspection (consciousness), as acknowledged by most leading researchers, including Anderson [28, 33] and Rosenbloom et al. [4]; thus it constitutes the subconceptual knowledge base in an agent. This is consistent with the framework of Sun [32, 40].

2.2. Two Levels

A two-level hybrid model seems to provide the needed framework for representing both types of knowledge. There have been various two-level architectures proposed, e.g., [32, 41–45].

Let us discuss the two levels in detail based on the idea of Sun [32, 40]. Existing evidence shows that the difference between the two levels lies mainly in their representations [46]. First, how do we capture procedural knowledge? In terms of representation, we prefer a subsymbolic distributed representation, such as that provided by a backpropagation network. This is because of the implicit nature of procedural skills (inaccessibility). A distributed representation naturally captures this property of procedural skills [32, 40], with representational units that are capable of accomplishing tasks but are in general uninterpretable (subsymbolic). (A symbolic representation may be used, but then we would have to artificially assume that these representations are not accessible, while some other similar representations are accessible—the distinction is arbitrary and not intrinsic to the media of representations; cf. [4, 28].) In terms of learning, we would like to use reinforcement learning (i.e., the temporal difference method, which can handle sequential decision situations; [7, 11]. This is because in a reactive sequential decision-making situation (such as navigation), there is seldomly any uniquely correct action. In general, for each situation, there are various possible responses that are roughly equally good; thus, supervised learning procedures do not seem applicable. However, for each situation, there are indeed good actions and bad actions; we measure the goodness of actions through a payoff/reinforcement signal from the world; thus, we can adopt the reinforcement learning paradigm. In such learning, an adjustment can be made to some parameters to increase the chance of selecting the actions that will lead to positive reinforcement and to reduce the chance of selecting the actions that will lead to negative reinforcement.

Second, how do we capture declarative knowledge? In terms of representation, we prefer a symbolic or localist representation, in which each unit has a clear conceptual meaning or interpretation, because declarative knowledge is highly accessible and inferences are performed explicitly at the conceptual level [32, 36, 40]. Because of the reactive nature of the tasks, the symbolic representations that we need are relatively simple. We thus focus on propositional rules. There are a number of ways symbolic representations can come into being. As pointed out before, those learning algorithms (either incremental or batch) that assume a stationary world are not suitable for use here. Some other algorithms are also not suitable because of nonconcurrent updating, which does not bode well with related cognitive phenomena. Developing a suitable algorithm means going beyond these learning algorithms. We can make use of the other level—the network that is trained with reinforcement learning and capable of specific procedural skills, and there is thus no need for a completely separate learning mechanism for the top level. We can extract information from the bottom-level and thereby form rules (i.e., bottom-up learning).[6] In addition, we should dynamically acquire rules and modify rules in subsequent encounters when necessary. Human learning is often on-line and gradual, which is especially true in rule learning as specifically discussed by Dominowski [47], Medin et al. [48], Nosofsky et al. [5] and others.

2.3. An Outline of the Model

These desiderata lead to CLARION, which is similar to the model developed in [40] but is specifically designed for reactive sequential decision making. It consists of two levels: The bottom level contains procedural knowledge [33] and the top level contains declarative knowledge in the form of propositional rules. An overall pseudo-code algorithm that describes the operation

of CLARION is as follows:

1. Observe the current state x.
2. Compute in the bottom level the Q-values of x associated with each of all the possible actions a_i's: $Q(x, a_1), Q(x, a_2), \ldots, Q(x, a_n)$.
3. Find out all the possible actions $(b_1, b_2, \ldots, b_m)$ at the top level, based on the input x and the rules in place.
4. Compare the values of a_i's with those of b_j's, and choose an appropriate action b.
5. Perform the action b, and observe the next state y and (possibly) the reinforcement r.
6. Update the bottom level in accordance with Q-LEARNING.
7. Update the rule network with RULE-EXTRACTION-REVISION.
8. Go back to Step 1.

Below we will describe the details of one version of the architecture: the deterministic rule learning version, which is the simplest version but illustrates the basic idea of the architecture (see [25, 26] for details of the statistical rule learning versions).

2.4. The Bottom Level

A Q-value is an evaluation of the "quality" of an action in a given state: $Q(x, a)$ indicates how desirable action a is in state x (which consists of sensory input). We choose an action based on Q-values. To acquire the Q-values, we use the standard Q-*learning* algorithm (a temporal difference reinforcement learning algorithm as determined earlier). For details of Q-learning, see [49] as well as more recent developments, [11, 12, 50–52]. The Q-learning algorithm is implemented here using Backpropagation neural networks [10, 53, 54].[7] (See Appendix A.1 for a brief review of Q-learning and its implementations in the bottom level of the model.)

In a nutshell, such a learning process performs (in an approximate manner) both structural credit assignment (with backpropagation), so that the agent knows which element in a state should be assigned credit/blame, as well as temporal credit assignment (with Q-learning), so that the agent knows which action leads to success or failure (in terms of reinforcement received). The combination of Q-learning and backpropagation enable the development of procedural knowledge solely based on the agent exploring the world on-line (on a continuous, on-going basis). It requires no external teacher or a priori knowledge and it allows "drifting". (For more explanations of the working of the bottom level, see our previous publications, such as Sun and Peterson [25, 26].)

2.5. The Top Level

Declarative knowledge is captured in a simple propositional rule form, in accordance with the previous considerations. Although we can use directly a symbolic rule representation, to facilitate correspondence with the bottom level and to encourage uniformity and integration, we chose to use a localist connectionist model instead. Basically, we connect the nodes representing conditions of a rule to the node representing the conclusion. That is, we translate the structure of a set of rules into the structure of a network. See e.g. [55–58] for details (see Appendix A.2 for a quick review).

As discussed earlier, we devised a different rule learning algorithm because of the differing characteristics of our tasks: concurrent (on-line) learning, reactivity, lack of teacher input and a priori knowledge, "drifting" environments, and requisite revisions. The basic idea for our algorithm is as follows: if some action decided by the bottom level is successful (here, being successful could mean a number of different things; the details are specified later), then the agent extracts a rule that corresponds to the action selected by the bottom level and adds the rule to the rule network. Then, in subsequent interactions with the world, the agent verifies the extracted rule by considering the outcome of applying the rule (when it is applied): if the outcome is not successful, then the rule should be made more specific and exclusive of the current case; if the outcome is successful, the agent may try to generalize the rule to make it more universal. Specialization and generalization is done based on actually encountered situations, and thus search is minimal. Below we will present some details, which are necessary to convey the essense of our model.

2.5.1. Rule Extraction. We perform rule extraction at each step, which is associated with the following information: (x, y, r, a), where x is the state before action a is performed, y is the new state entered after an action a is performed, and r is the reinforcement received after action a. Rules are of the following form: *conditions* $\rightarrow a$, where a is an action and *conditions* are made up of the conjunction of individual conditions each of which refers to the value of an element in the (sensory) input state x. Three different criteria can be used for

Methods:

three phases	criteria	testing
phase 1	direct reinforcement	$r > r_t$
phase 2	temporal difference in Q-values	$r + \gamma e(y) - Q(x, a) > s_2$
phase 3	maximum Q-values	$Q(x, a) > \max_i Q(x, i) - s_3$

Figure 3. Three phases in rule extraction. r_t, s_2, and s_3 are threshold parameters.

rule learning at each step: (1) direct reinforcement (r) received at a step, (2) temporal difference (as used in updating Q-values), and (3) maximum Q-values in a state. The first criterion is an indication of whether or not an action taken in a given state is *directly* beneficial, but it fails to take into account sequences of actions. The second criterion indicates if further improvement in a Q-value is possible. The third criterion concerns whether the Q-value of an action is close enough to the maximum Q-value in that state, indicating the optimality of the action. We adopt a three-phase approach, with each phase lasting for a certain number of episodes.[8] The three criteria are applied in their respective phases (in Fig. 3, each criterion is listed with its appropriate phase and the corresponding test to be used). At each step, we apply the current-phase criterion to determine whether we should construct a rule. If so, a rule is wired up in the rule network.

2.5.2. Rule Revision. After rules are extracted, at each step, the algorithm reexamines the rules matching the current step to decide if each of them should be kept, revised, or discarded. A number of operations are used for this purpose (in each of the 3 phases):

- *Expansion*: when a rule is successfully applied (according to the criterion in the current phase), the value range of a condition is expanded by one interval (randomly selected).[9] Generally speaking, we can expand j different conditions of a rule (selected randomly); we can do that for k times resulting in k rules. When a condition is expanded to the full range of its value, then in effect the condition is dropped from the rule. (This is similar in some way to [59], different from temporal abstraction such as [12, 52].)
- *Shrinking*: when a rule leads to unsuccessful results (as judged by the criterion in the current phase), we reduce the value ranges of some or all conditions (cf. [59, 60]). In general, we can select u conditions to shrink; the selection can be random, or based on recency. We can perform a shrinking operation on the same rule for v times and thus create v shrunk rules.
- *Deletion*: remove a rule from the rule network when a counter example to the original setting (x, y, r, a) from which the rule was extracted is encountered (according to the current-phase criterion).
- *Merge*: when the conditions of two rules are close enough, the two rules may be combined so that a more general rule can be produced. We require that corresponding conditions of the two rules must overlap (for up to two conditions) or be identical in their value ranges. We combine the range of each corresponding pair of conditions of the two original rules, so a new rule covers the union of the two original rules and more.

Such rule learning is not guaranteed, in a formal sense, to find a generic rule whenever it exists. The process is tentative and heuristic. "Backtracking", in the sense of reextracting a rule and testing other generalizations, may happen.

2.6. Combination

In the overall algorithm, Step 4 is for making the final decision on which action to take by incorporating outcomes from both levels. (To justify this cognitively, as shown by Willingham et al. [39], declarative knowledge can influence procedural performance.) It allows different operational modes: e.g., relying only on the top level, relying only on the bottom level, or combining the outcomes from both levels weighing them differently. These operational modes roughly correspond to the folk psychological notions of the intuitive (reactive) mode, the deliberative mode, and the various mixture of the two with a different percentage of each [35].

Several methods of combining outcomes from the two levels were tried. For example, in the *stochastic method*, we combine the corresponding values for an action from the two levels by a weighted sum; that is,

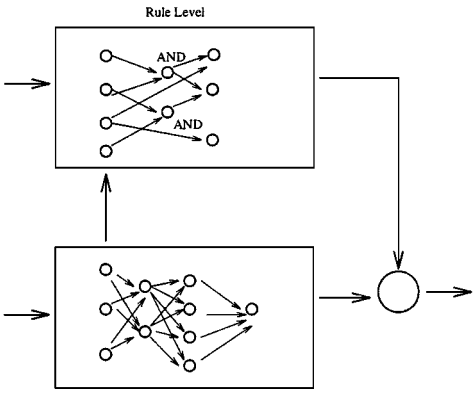


Figure 4. The detailed CLARION architecture.

if the top level indicates that action a has an activation value v (which should be 0 or 1 as rules are binary) and the bottom level indicates that a has an activation value q (the Q-value), then the final outcome is $w_1 * v + w_2 * q$ [61]. Stochastic decision making with Boltzmann distribution (based on the weighted sums) is then performed to select an action out of all the possible actions. (See [39] for psychological justifications of this combination method.)

2.7. The Whole Model

Putting the two levels together, implemented in a connectionist fashion, we now have the whole model (see Fig. 4).

We can contrast the characteristics of the two levels. The top level is discrete, all-or-nothing, rigorously verified (through experience), and without random exploration, and it learns through trial-and-error in a one-shot fashion. The bottom level is continuous, graded, statistical in nature (i.e., not rigorously verified), and with random exploration, and it learns in a gradual and cumulative fashion. Thus they complement each other. Note also that the generalization of rules complements the generalization in the bottom level: While the bottom level generalization is continuous/graded, rule generalization at the top level is discrete/crisp, thus capturing different kinds of regularities. Because they possess different characteristics, each level tends to learn differentially; thus a combination of the two, through stochastic "averaging", is likely to result in improved performance [61, 62].

The necessity of having a two-level architecture can be summed up as follows:

- Without the bottom level, the agent will not be able to represent procedural skills sufficiently. Such skills may involve graded, uncertain knowledge and autonomous stochastic exploration (with numeric calculation and probabilistic firing). Thus they may not be captured by simpler mechanisms (such as those in [33]).
- Without learning in the bottom level, the agent will not be able to learn from experience, and therefore will not be able to dynamically acquire either procedural skills in the bottom level (cf. [63]) or rules in the top level (as in the current model). The bottom level also captures the gradual learning of skills, different from one-shot rule learning.
- Without the top level, the agent will not be able to (1) represent generic, easily accessible, and crisp knowledge and (2) explicitly access and communicate that knowledge to other agents (i.e., the explanation capability, which is absent in e.g., [8, 63]; but see [64]). When novel situations are encountered and/or when precision, crispness, consistency, and certainty are needed, declarative knowledge is preferred. Explicit access and explanation is also important in facilitating cooperation among agents.
- Without rule learning, the agent will not be able to acquire quickly and dynamically declarative knowledge for the top level, and therefore have to resort to externally given declarative knowledge.
- Moreover, without rules (and rule learning), the performance of the bottom level alone will be significantly worse, as analyzed earlier and has been shown in experiments.

3. A Sketch of Some Experiments

We will sketch two sets of experiments, which suggest, respectively, (1) the performance advantage of the two-level model (in addition to the previously mentioned advantages) and (2) its cognitive validity. (We believe that it is important to highlight some empirical results for a summary paper like this. These two experiments have not been published in journals before.)

3.1. Experiments with Navigation

We tested CLARION on the simulated navigation task as shown in Fig. 1. Developed by Naval Research Lab, the task is complex and realistic ([65]). The agent has to navigate an underwater vessel through

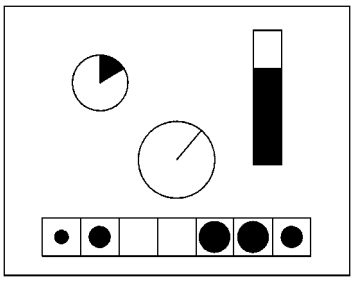


Figure 5. The navigation input. The display at the upper left corner is the fuel gauge; the vertical one at the upper right corner is the range gauge; the round one in the middle is the bearing gauge; the seven sonar gauges are at the bottom.

a minefield to reach a target location. The agent receives information only from a number of instruments. As shown in Fig. 5, the sonar gauge shows how close the mines are in seven equal areas that range from 45 degrees to the left of the agent to 45 degrees to the right. The fuel gauge shows how much time is left before fuel runs out. The bearing gauge shows the direction of the target from the present direction of the agent. The range gauge shows how far the target is from the current location. Based only on such information, the agent decides on (1) how to turn and (2) how fast to move. The agent, within an allotted time period, can either (a) reach the target (a success), (b) hit a mine (a failure), or (c) run out of fuel (a failure again). The agent is under severe time pressure, so it has to be reactive in decision making.

A random environment is generated for each episode, so that there is no repetition of experience for the agent throughout the experiment. Learning is thus difficult because the agent has to take into account varying environments. The reinforcements for an agent are produced based on how successful the agent is at the end of an episode [25, 26].

3.1.1. Learning Speeds. In this experiment, the minefield contains a given number of mines, which in this case is 10 or 30. The mines are randomly placed between the starting point of the agent and the target. The target location is also randomly selected each time. The time allotted to the agent for each episode is 200 steps.

Figure 6 shows the difference between CLARION and the bottom level alone (trained with pure Q-learning) in terms of learning effectiveness, which is measured by the number of successful episodes out of a total of 1000 episodes during training (averaged over 10 runs). In the figure, *Stoc.z* refers to the stochastic combination of the two levels with rules being weighted at $w_1 = z\%$ (and $w_2 = 1-w_1$). The symbol *gen* indicates that generalization/revision is performed; otherwise, generalization/revision operations (such as *expansion* and *shrinking*) are omitted. For CLARION, we show only the performance of *Stoc.20* and *Stoc.20.gen*, because different methods for combining the two levels do not make much difference. However, we varied the temperature (randomness) parameter (in stochastic decision making), and examined learning in minefields consisting of either 10 or 30 mines. The superiority of CLARION over the bottom level alone (with Q-learning) is statistically significant in all the cases (with t-tests, $p < 0.05$), except for *Stoc.20* in the 10-mine case with temperature $= 0.01$.

3.1.2. Transfer. Transfer is defined here loosely as the performance improvements of an agent when tested in a new setting after it was pre-trained in a different setting, relative to an agent who had no similar pretraining. To assess transfer, after training various models

		Temperature= 0.01	Temperature= 0.1
For 30 mines	Q-learning	95.0	6.2
	Stoc.20	397.8	33.6
	Stoc.20.gen	388.3	35.5
		Temperature= 0.01	Temperature= 0.1
For 10 mines	Q-learning	525.5	32.8
	Stoc.20	689.1	279.9
	Stoc.20.gen	644.1	308.5

Figure 6. Learning: the numbers of successful episodes.

	Temperature= 0.01	Temperature= 0.1
Q-learning	37.8%	0.6%
Stoc.20	31.2%	46.3%
Stoc.20.gen	29.5%	48.0%

Figure 7. Transfer: the percentages of successful episodes in a new setting.

on 10-mine minefields for 1000 episodes each (using a randomly generated minefield for each episode), we applied these models to new minefields that consisted of 30 mines. Figure 7 shows the differences in transfer, where transfer is measured by the percentage of successful episodes in the new setting by the trained models (each trained model was applied to 30-mine minefields for a total of 20 episodes; the data was averaged over 10 runs). As indicated by the table, CLARION outperforms the bottom level alone (trained with only Q-learning) in transfer. The difference between the best of the bottom level (37.8%) and the best of CLARION (48.0%) is statistically significant.[10]

3.2. Matching Human Data

Let us look into the match between the performance of the model and the performance of the human subjects whom we tested. Although these results have never before been published in journals, some details regarding the human experiments can be found in [66, 67].

Three training conditions were used in human experiments:

- The standard training condition. Subjects received five blocks of 20 episodes on each of five consecutive days (100 episodes per day). In each episode the minefield contained 60 mines.
- The verbalization training condition. This condition was identical to the standard training condition except that subjects were asked to step through slow replays of selected episodes and to verbalize what they were thinking during the episode.
- The 30-to-60 transfer condition. This condition was also identical to the standard training condition except that subjects performed the task with 30 mines on the first two days of training and switched to 60 mines starting the third day.

The results of the experiments were as follows.

- The standard training condition. We averaged the data over 10 human subjects. We also averaged 10 model runs in correspondence with human experiments (each model run was initialized with different random number sequences and thus produced different results). These data are presented in Fig. 8. Both sets of data were best fit by power functions (for failure rate). The degree of similarity is evident. A Pearson product moment correlation coefficient was calculated. The analysis yielded a high positive correlation ($r = 0.82$), indicating a high degree of similarity between human subjects and model runs (although there is some disimilarity in the initial segments of the curves).
- The verbalization training condition. Obviously, we could not require verbalization from the model. However, we posited that much of the effect of verbalization on learning was associated with rehearsing previous steps and episodes (although there may be additional factors involved). Thus for the model, we used episode memory playback [10], in

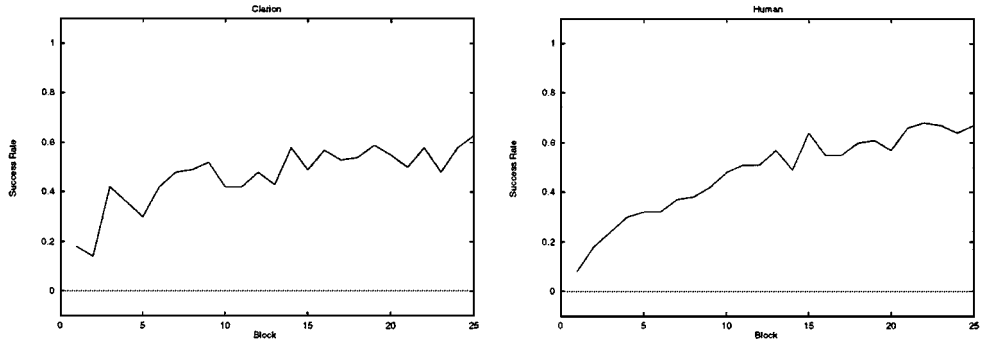


Figure 8. The learning curves in terms of success rates in the standard condition. The right side is the human data and the left side is the model data.

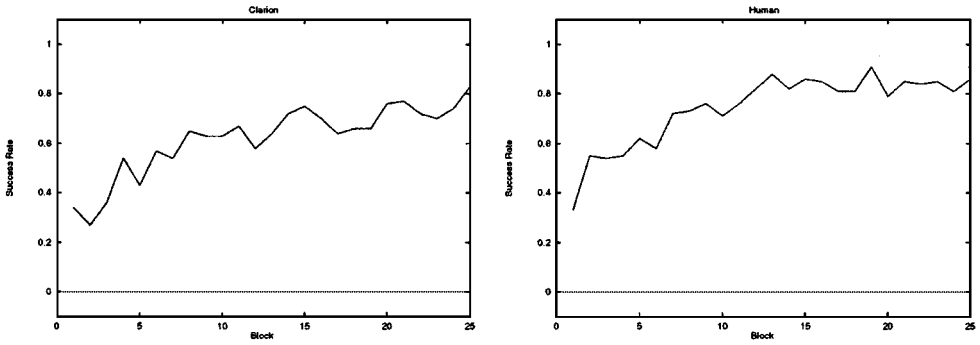


Figure 9. The learning curves in terms of success rates in the verbalization condition.

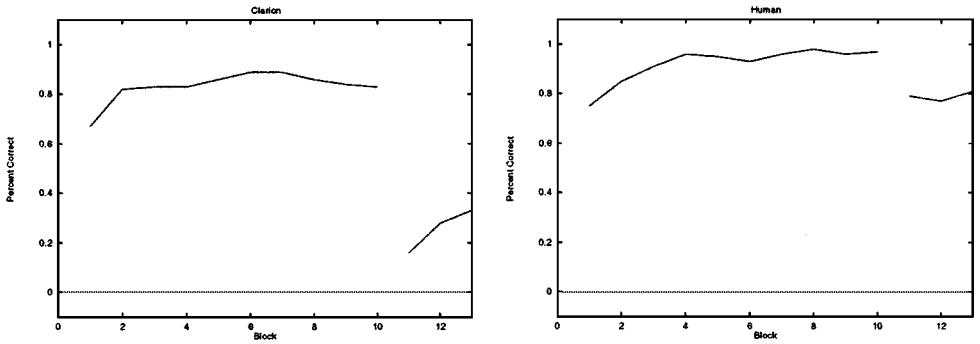


Figure 10. The 30-to-60 transfer data in terms of success rates.

an attempt to capture this effect.[11] In this case, the data from 5 human subjects was compared to that of 5 model runs (see Fig. 9). Again, both sets of data were highly similar and both were best fit by power functions. We also calculated a Pearson product moment correlation coefficient, which yielded a high positive correlation ($r = 0.84$).

We also compared the changes in performance due to verbalization for the human subjects and the model runs. This was done by averaging failure rates across blocks separately for each human subject and for each model run and subjecting that data to a 2×2 ANOVA. The result indicated the both groups exhibited a significant increase in performance due to verbalization ($p < 0.01$), and that the changes due to verbalization for the two groups were not significantly different.

- The 30-to-60 transfer condition. Subjects were first trained on 30-mine minefields, and then transferred to 60-mine minefields. The model was tested under the same condition. Both human and model data were averaged over 10 subjects. Comparing the human and model data (see Fig. 10), we noticed that both learned well at 30 mines, although human data was slightly better. When transferred to 60 mines, both exhibited a significant drop in performance, although the model exhibited a deeper drop.[12]

The afore-sketched comparison (see [66, 67]) demonstrates, to a certain extent, the cognitive validity of the model, in the sense of being a reasonable interpretation of the data (although, of course, it does not uniquely proves the model).

3.3. Other Experiments

There are many other experiments with the architecture that have been published in other journal articles and so will not be repeated here. For example, Sun and Peterson [25, 26], contain details of experiments using alternative versions of the CLARION architecture

(the statistical rule learning versions). Sun and Peterson [25] contain details of computational experiments in another domain, a maze domain. These results also demonstrate the performance advantages of the architecture.

3.4. Discussions of Experiments

3.4.1. Verifying Predictions. Several earlier predictions (made in Section 2.1) have received empirical support. We have shown (see also [25, 26]) that (1) rule learning helps to speed up learning (as shown earlier); (2) rule learning sometimes helps to improve trained performance (as shown in [25]); (3) rule learning helps to facilitate transfer (as shown earlier), and sometimes learned rules transfer better than the whole system (see [25]). (Note that these results serve only as existence proofs, not a proof of the necessity of the model advantages.)

3.4.2. Concept Formation and Symbolic Reasoning. While learning rules, CLARION forms concepts. Although there are available the distributed (feature) representations in the bottom level for specifying rule conditions, a separate node is instead set up in the top level to represent the conditions of a rule, that connects to the feature representations. So along with the induced rules, localist representations are formed. This kind of conceptual representation is basically a *prototype model* [68, 69]. Localist nodes serve as identification and coding of features, in a bottom-up direction. Localist nodes also serve to trigger relevant features in a top-down direction once a concept is brought into attention. They can also facilitate inheritance reasoning based on features [32, 40].

Concepts formed in this way are context-dependent and goal (task)-oriented because they are always formed with regard to the tasks at hand while exploiting environmental regularities. A concept is formed as part of a rule, which is learned to accomplish a task in a certain environment. The task context and the experience can help an agent to determine which features in the environment need to be emphasized, which kind of object should be grouped together, and thus what constitute a separate category of objects (that is, a concept). Therefore, the acquired concepts are functional.

We can combine both learning and (complex) reasoning in CLARION (see [70]). The rules need not to be in the "state → action" form. Rather, rules can be in the "state → action result" form (which constitute *schemas* as they are often called; [71]); this form can be easily incorporated into all the aforementioned experiments. This simple addition allows some powerful operations from traditional AI, such as backward chaining reasoning, means-ends analysis, and counterfactual reasoning, to be performed. These operations can be used either to speed up learning or to help with post-processing, such as providing explanations and facilitating communication among agents.

4. General Discussions

4.1. Situated Cognition

The above dealing with reactive sequential decision tasks is consistent with the situated cognition view, in the sense that coping with the world means acting in an environmentally driven fashion and dealing with moment-to-moment contingencies. The "reactive" approach embodied in this work reflects such a view through a focus on reacting to the current state of the world. Also in line with the situated cognition view, learning in this work is tied closely to specific situations as experienced, with learning reflecting and exploiting environmental contingencies and regularities.

But there are some obvious differences too. The situated cognition view claims that there should not be any elaborate model of the world or elaborate representation (including goals). However, instead of being antithetical to the representationalist view and avoiding abstract/generic rules and models, we take a more inclusive approach: we show that declarative knowledge *can* be constructed on the basis of situated learning by situated cognitive agents, thus unifying the two contradictory views through a bottom-up process.

4.2. Rule Learning in AI

Though it learns rules, CLARION tackles tasks that differ from what is usually dealt with by traditional rule learning algorithms. Most of the supervised concept/rule learning algorithms (such as AQ and ID3; [16, 72]) require consistent data and pre-classification, which is not available to CLARION. As "batch" algorithms, they require the agent to obtain all data before learning starts, which means higher space complexity, slow start in learning, and no "drifting" (without extra mechanisms). They cannot be applied directly to sequential decision tasks because they do not perform

temporal credit assignment. There is also the incremental variety of supervised learning (e.g., the version space algorithm of [19] and also [22]). Although incremental, they require labeled, complete, and consistent descriptions of instances (no "drifting"), which is not available to CLARION. They do not handle sequences either.

Rule learning algorithms might be able to avoid the problem of sequences by evaluating the values of states and/or actions statically (in isolation). Such static evaluation must assume a lot of a priori knowledge about the task on the part of the agent and thus changes the nature of the learning problem. In contrast, we assume minimum a priori knowledge in the agent.

Unsupervised rule/concept learning algorithms, such as [73–75], are also unsuitable for our type of task, in that (1) in our task there is feedback available (reinforcements or payoffs) although there is no direct supervision; barring static evaluation of states, such feedback must be taken into consideration in order to achieve successes; (2) a complete description of instances on which a system can base its decisions are usually not available; (3) temporal credit assignment is necessary.

Some recent work, such as [70, 76], tried to incorporate reasoning into learning agents, which is also what CLARION strives for. Learning and reasoning are inseparable in cognition. For example, [77] demonstrated how reasoning ("theory") interacts closely with concept learning. [78] showed that even categorization does not rely on just similarity but also reasoning from rules. However, these above systems did not utilize a principled dichotomy of conceptual/subconceptual processing.

4.3. Connectionist Rule Extraction

Fu [56] proposed a search-based algorithm to extract conjunctive rules from perceptron networks. To find rules, the learner first searches for all the combinations of positive conditions that can lead to a conclusion; then, in the second phase, with a previously found combination of positive conditions, the learner searches for negative conditions that should be added to guarantee the conclusion. In the case of three-layered networks, the learner can extract two separate sets of rules, one for each layer, and then integrate them by substitution. Towell and Shavlik [58] used rules of an alternative form, the *N-of-M* form: *If N of the M conditions, $a_1, a_2, \ldots, a_M$, is true, then the conclusion b is true*. (It is believed that some rules can be better expressed in such a form, which more closely resembles the weighted-sum computation in connectionist networks, in order to avoid the combinatorial explosion and to discern structures.) A four-step procedure is used to extract such rules, by first grouping similarly weighted links and eliminating insignificant groups, and then forming rules with the remaining groups. However, these rule extraction algorithms are meant to be applied at the end of the training of a network. Once extracted, the rules are fixed; there is no modification on the fly, unless the rules are reextracted after further training of the network. On the other hand, in CLARION, an agent can extract and modify rules dynamically. Connectionist reinforcement learning and rule learning can work together simultaneously; thus we utilize the synergy of the two algorithms to improve learning. Dynamically extracting and modifying rules is computationally less expensive. It also helps the agent to adapt to changing environments, by allowing the addition and the removal of rules at any time. For more recent work on rule extraction, see the collection [79].

4.4. Hybrid Models

Let us now turn to the form of the model. Hybrid models (connectionist and symbolic, reactive and deliberate, rules and non-rules, representation and non-representation, etc.) have become popular. These models include [32, 40–44, 80–82], among others. However, some have objected to such an approach, claiming that such research will "most likely be frustrated by its lack of a consistent conceptual framework". In our view, hybridness can be consistent or even principled. From a cognitive perspective, the two-level architecture is principled, as it is based on the theories concerning the dichotomy of the conceptual and the subconceptual. In the experiments reported here, we see that there can also be a performance advantage in combining declarative knowledge/rule learning and procedural skills/reinforcement learning.

Similarly, adopting multiple learning methods in hybrid models has been dubbed the "kitchen sink approach". We insist that the present model is not a mere "kitchen sink". This is because it *unifies* two different learning methods in an integrated architecture, so that their synergy is explored. Moreover, it can be viewed

as an extension of Q-learning, because rule learning here is distinct from existing algorithms and specifically designed to help with Q-learning.

Some other hybrid connectionist models try to implement all types of knowledge (symbolic and non-symbolic) in one particular kind of connectionist network or another; for example, [55, 83–85] try to implement knowledge in localist networks, and [43, 80, 86, 87] try to implement knowledge in distributed networks. CLARION, among others such as [40, 42, 45, 82], takes a different tack and attempts to develop a principled dichotomy of the conceptual vs. the subconceptual in architectures. Among those models that incorporate such a dichotomy, some tend to simply juxtapose the two sides of the dichotomy. Instead, CLARION attempts to explore their synergy.

Some existing hybrid models do not or cannot perform learning, such as [40, 44, 83, 88, 89], although their representations are more complex and sophisticated. Others perform learning in a batch fashion, such as [45, 80], although their learning task may be quite difficult. Instead, CLARION performs on-line (concurrent) learning [42, 90]. CLARION is thus more cognitively plausible and more ecologically realistic in this regard (see e.g. [5, 49]). CLARION is also capable of integrated learning (that is, developing connectionist and symbolic representation along side of each other), which is unlike any of the existing models. In addition, most of the existing learning models explore mainly top-down learning (including advice taking), in which externally given declarative knowledge is turned into procedural knowledge through practice [28, 29, 33, 42, 90, 91]. Instead, CLARION explores bottom-up learning, to demonstrate how conceptual/symbolic knowledge can emerge in interacting with the world through the mediation of subconceptual procedural knowledge.

4.5. Cognitive Modeling

The major difference between CLARION and other cognitive architectures is the fact that CLARION utilizes a combination of two processes and two representations, and exploits synergy of them (see [45]). Both representations are acquired through autonomous exploration by the learner, instead of being externally given or requiring a large amount of a priori knowledge to begin with. This is in sharp contrast to well known cognitive architectures such as ACT [27–29, 33] and SOAR [4].

In [92] an architecture was developed that attempted to implement the Piagetian constructivist view of child development. The learning mechanism is based on statistics collected during interaction with the world. New schemas (i.e., rules) are created and their conditions identified and tuned through statistical means based on relevance. It can also build abstractions out of primitive actions. However, the model does not make the dichotomatic distinction of procedural vs. declarative knowledge and thus does not account for the distinction of implicit vs. explicit learning [46, 93].

The process difference in the two levels of CLARION resembles those in some other psychological models of learning: for example, (1) the difference between the strategies of look-ahead vs. table lookup as proposed by [93] for explaining the difference between explicit and implicit learning; (2) the difference between algorithms vs. instance retrieval as proposed by [94] for accounting for the difference between initial trials and later skillful performance in the course of skill learning; (3) the difference between mental models/theories vs. experiences as proposed by [95] in relation to the difference between explicit verbalization and implicit skill. The former type in each dichotomy is slower (more serial) and more deliberate, while the latter is relatively effortless and automatic. It appears that such a mechanistic difference may be applicable in modeling a variety of cognitive processes. It has been applied in some of our previous cognitive models (see [32, 40, 44]).

4.6. Relations to Consciousness

As having been discussed in Sun [38], the crucial link between this model of procedural/declarative knowledge and the conscious/unconscious distinction in humans was in the psychological work on implicit learning (e.g., [39, 46, 95]). Such work showed the dissociation between conscious and unconscious learning. That is, human knowledge, and its acquisition process, could be partially or completely unconscious. The connection from such data to our model laid in the ability of the model to account for some of the most important characteristics of human implicit/explicit learning, as described in [38]. The CLARION model explained consciousness by accounting for phenomena in psychological literature in terms of the two levels in the model and their associated mechanisms. Moreover, the model readily accommodated important features of existing models of consciousness (see [38]).

4.7. Extensions

In order to further explore the performance of the model, experiments are needed in a variety of different domains, especially more complex domains. We shall consider the following tasks: (1) tasks in which location information is available [8], which makes it more difficult to extract generic rules, in order to extend the transfer ability of the model; (2) tasks in which even more sensory information is available so that the learning space is even larger; (3) tasks in which the environment is more noisy or more probabilistic, and the probability distribution changes over time.

We shall also further investigate concept formation by an agent embedded in a task environment, to develop concepts automatically that are pertinent and beneficial to the task at hand, through exploring the task environment. There are a variety of other learning techniques that can be used in CLARION. We need to try out, and to explore the interplay of, these different learning techniques.

Another extension is incorporating top-down learning, in addition to bottom-up learning, by accepting external advice and assimilating it internally (see [28, 29, 33, 96]). In addition, we shall also look into implementing a variety of subconceptual, intuitive processes in the bottom level, such as constraint satisfaction and similarity-based (analogical) reasoning and categorization [40].

5. Conclusions

In this paper, we focused on an example of a hybrid architecture. Its learning was reactive, experience-driven, and on-line, developing both subsymbolic and symbolic representations. CLARION was able to learn autonomously, and to develop both procedural skills and declarative knowledge, in a bottom-up direction, through exploring the world.

Utilizing a principled dichotomy, CLARION was able, at least in some circumstances, to learn faster, perform better, and transfer more effectively than models that neglect such a dichotomy. Experiments in various different tasks demonstrated in the three aspects the potential for such advantages (some experiments in the navigation domain was presented). They suggested that the combination of the two types of knowledge could yield synergistic results.

Moreover, CLARION was able to match human skill learning data to a certain extent. This matching demonstrated the cognitive validity of the architecture (to the extent that it accounted for human data to a certain degree of accuracy).

A. Appendix

A.1. Q-learning at the Bottom Level

In the Q-learning algorithm, $Q(x, a)$ estimates the maximum discounted cumulative reinforcement that the agent will receive from the current state x on: $\max(\sum_{i=0}^{\infty} \gamma^i r_i)$, where γ is a discount factor that favors reinforcement received sooner relative to that received later, and r_i is the reinforcement received at step i (which may be 0). The updating of $Q(x, a)$ is based on minimizing

$$r + \gamma e(y) - Q(x, a)$$

where γ is a discount factor, y is the resulting state, and $e(y) = \max_a Q(y, a)$. Thus, the updating is based on the *temporal difference* in evaluating the current state and the action chosen.[13] Through successive updates of the Q function, the agent can learn to take into account future steps in longer and longer sequences notably without explicit planning [49].

To implement Q-learning, we chose to use a four-layered network (see Fig. 11), in which the first three layers form a backpropagation network for computing Q-values and the fourth layer (with only one node) performs stochastic decision making. The network is internally subsymbolic as decided earlier. The output of the third layer (i.e., the output layer of the backpropagation network) indicates the Q-value of each action (represented by an individual node), and the node in the fourth layer determines probabilistically the action to be performed based on a Boltzmann distribution [49]:

$$p(a \mid x) = \frac{e^{1/\alpha Q(x,a)}}{\sum_i e^{1/\alpha Q(x,a_i)}}$$

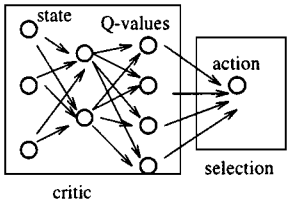


Figure 11. The Q-learning method.

Here α controls the degree of randomness (temperature) of the decision-making process.

The training of the backpropagation network is based on minimizing the following:

$$err_i = \begin{cases} r + \gamma e(y) - Q(x, a) & \text{if } a_i = a \\ 0 & \text{otherwise} \end{cases}$$

where i is the index for an output node representing the action a_i. The backpropagation procedure is then applied as usual to adjust weights.

The lookup table implementation of Q-learning is out of question here because of the (likely) continuous input space and when discretized, the resulting huge state space (e.g., in the minefield navigation task there are more than 10^{12} states). Some kind of function approximator has to be used that can generalize from some states to others [8–10, 53]. But as a result the convergence of learning is not guaranteed.

A.2. Rule Encoding

There are a number of previous proposals of rule encoding in connectionist networks (e.g. [55–58]) that we can draw upon. For each rule, a set of links can be established, each of which connects a concept in the condition of a rule to the conclusion of the rule. So the number of incoming links to the conclusion of a rule is equal to the number of conditions of the rule. If the concept in the condition is in a positive form, the link carries a positive weight w; otherwise, it carries a negative weight $-w$. Sigmoidal functions are used for node activation (as an obvious choice; other functions are also possible):

$$\frac{1}{1 + e^{\sum_i i_i w_i - \tau}}$$

The threshold τ of a node is set to be $n * w - \theta$, where n is the number of incoming links (the number of conditions leading to the conclusion represented by this node), and θ is a parameter, selected along with w to make sure that the node has activation above 0.9 when all of its conditions are satisfied, and has activation below 0.1 when some of its conditions are not met. (Activations above 0.9 are considered 1, and activations below 0.1 are considered 0; so rules are crisp/binary.) In addition, if there is more than one rule that leads to the same conclusion, an intermediate node is created for each such rule: all the concepts in the condition of one rule are linked to the same intermediate node, and then all the intermediate nodes are linked to the node representing the conclusion. For more complex rule forms including predicate rules and variable binding, see [55].

Acknowledgments

This work is supported in part by Office of Naval Research grant N00014-95-1-0440 to University of Alabama. We wish to thank Helen Gigley and Susan Chipman for the support, Dave Waltz, Jack Gelfand, Jeff Shrager, and Devika Subramanian for comments or discussions. Diana Gordon, Jim Ballas and Alan Schultz provided the navigation simulator. Thanks to Franz Kurfess for his suggestion of structuring this as a summary paper. Thanks also to the four anonymous reviewers.

Notes

1. Sequentiality is clearly essential in human intelligent behavior as work in areas ranging from instrumental conditioning to cognitive skill acquisition has demonstrated [1, 97, 98].
2. An agent might have some *internal* states, such as limited memory. Usually these internal states can be set and reset by appropriate (internal) actions of the agent [99]. From a formal view point, these internal states and their related actions are indistinguishable from external ones and can be viewed simply as part of the world. In case multiple agents are present in the world, to an individual agent, the existence of other agents can also be viewed as a part of the external world. The actions of other agents and the resulting change of the world can be simply viewed as autonomous changes occurred in the external world over time, and can thus be handled in the same way as described before.
3. It may not even be clear what constitutes a sequence and how long it is; the resource limitation may prevent it from remembering sequences.
4. In an "incremental" algorithm based on explicit statistics, changes can be gradually accommodated [100].
5. Note that we are not aiming to capture all of the above dichotomies. Denoting more or less the same thing, these dichotomies serve as justifications for our main distinction between declarative knowledge and procedural skills.
6. This two-level framework can be parsimonious, as learning procedural skills serves a dual purpose.
7. The lookup table implementation of Q-learning is out of question here because of the (likely) continuous input space and when discretized, the resulting huge state space.
8. Phase transition can be automatically determined based on the current performance level of the model.
9. The interval of a condition is determined by the granularity of the condition. For a binary condition, one expansion will reach its full range.

10. The performance of an agent without pre-training (which takes a lot more steps) is of no relevance and thus not shown here.
11. Episode memory playback involves training the model with previously performed episodes between blocks of actual trial episodes in exactly the same manner as in human experiments.
12. Specifically, we compared performance of the last block before the change in mine density and the first block after the change. Success rates were 98 and 79% for the human subjects and 83 and 26% for the model runs respectively. The drops were both statistically significant.
13. In the above formula, $Q(x, a)$ estimates, before action a is performed, the (discounted) cumulative reinforcement to be received if action a is performed, and $r + \gamma e(y)$ estimates the (discounted) cumulative reinforcement that the agent will receive, after action a is performed; so their difference (the temporal difference in evaluating an action) enables the learning of Q-values that approximate the (discounted) cumulative reinforcement.

References

1. Sutton and Barto (1981).
2. A. Schultz, "Using a genetic algorithm to learn strategies for collisionavoidance and local navigation," in *Proc. of 7th International Symp. on Unmanned Untethered Submersible Technology*, University of New Hampshire, Durham, pp. 213–225, 1991.
3. R. Sun and T. Peterson, "A hybrid learning model of reactive sequential decision making," in *The Working Notes of The IJCAI Workshop on Connectionist-Symbolic Integration*, edited by R. Sun and F. Alexandre, 1995.
4. P. Rosenbloom, J. Laird, and A. Newell, *The SOAR Papers: Research on Integrated Intelligence*, MIT Press: Cambridge, MA, 1993.
5. R. Nosofsky, T. Palmeri, and S. McKinley, "Rule-plus-exception model of classification learning," *Psychological Review*, vol. 101, no. 1, pp. 53–79, 1994.
6. G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden context," *Machine Learning*, vol. 23, no. 1, 1996.
7. R. Sutton, "Learning to predict by the methods of temporal difference," *Machine Learning*, vol. 3, pp. 9–44, 1988.
8. R. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proc. of Seventh International Conference on Machine Learning*, Morgan Kaufmann: San Mateo, CA, 1990.
9. S. Mahadevan and J. Connell, "Automatic programming of behavior-based robot with reinforcement learning," vol. 55, pp. 311–365, 1992.
10. L. Lin, "Self-improving reactive agents based on reinforcement learning, planning, and teaching," *Machine Learning*, vol. 8, pp. 293–321, 1992.
11. L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
12. T. Dietterich, Hierarchical reinforcement learning with MAXQ value function decomposition, 1997, ftp://www.cs.orst.edu.
13. J. Holland, N. Nisbitt, P. Thagard, and J. Holyoak, *Induction: A Theory of Learning and Development*, MIT Press: Cambridge, MA, 1986.
14. J. Grefenstette, "The evolution of strategies for multiagent environments," *Adaptive Behavior*, vol. 1, no. 1, pp. 65–90, 1992.
15. L. Meeden, "An incremental approach to developing intelligent neural network controllers for robots," *Adaptive Behavior*, 1995.
16. R. Michalski, "A theory and methodology of inductive learning," *Artificial Intelligence*, vol. 20, pp. 111–161, 1983.
17. R. Quinlan, "Inductive learning of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.
18. R. Quinlan, "Learning logical definition from relations," *Machine Learning*, vol. 5, pp. 239–266, 1990.
19. T. Mitchell, "Generalization as search," *Artificial Intelligence*, vol. 18, pp. 203–226, 1982.
20. M. Lebowitz, "Experiments with incremental concept formation: UNIMEM," *Machine Learning*, vol. 2, pp. 103–138, 1987.
21. D. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, vol. 2, pp. 139–172, 1987.
22. P. Utgoff, "Incremental induction of decision trees," *Machine Learning*, vol. 4, pp. 161–186, 1989.
23. H. Hirsh, "Generalizing version spaces," *Machine Learning*, vol. 17, pp. 5–46, 1994.
24. P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, pp. 261–284, 1989.
25. R. Sun and T. Peterson, "Some experiments with a hybrid model for learning sequential decision making," *Information Sciences*, vol. 14, pp. 83–107, 1998.
26. R. Sun and T. Peterson, "Autonomous learning of sequential tasks: Experiments and analyses," *IEEE Transaction on Neural Networks*, vol. 9, no. 6, pp. 1217–1234, 1998.
27. J. Anderson, "Acquisition of cognitive skill," *Psychological Review*, vol. 89, pp. 369–406, 1982.
28. J. Anderson, *Rules of the Mind*, Lawrence Erlbaum Associates: Hillsdale, NJ, 1993.
29. J. Anderson and C. Lebiere, *The Atomic Components of Thought*, Lawrence Erlbaum Associates: Mahwah, NJ, 1998.
30. F. Keil, *Concepts, Kinds, and Cognitive Development*, MIT Press: Cambridge, MA, 1989.
31. A. Damasio et al., "Neural regionalization of knowledge access," *Cold Spring Harbor Symp. on Quantitative Biology*, CSHL Press, vol. LV, 1990.
32. R. Sun, *Integrating Rules and Connectionism for Robust Commonsense Reasoning*, John Wiley and Sons: New York, NY, 1994.
33. J. Anderson, *The Architecture of Cognition*, Harvard University Press: Cambridge, MA, 1983.
34. P. Fitts and M. Posner, *Human Performance*, Brooks/Cole, Monterey, CA, 1967.
35. H. Dreyfus and S. Dreyfus, *Mind Over Machine*, The Free Press: New York, NY, 1987.
36. P. Smolensky, "On the proper treatment of connectionism," *Behavioral and Brain Sciences*, vol. 11, no. 1, pp. 1–74, 1988.
37. W. James, *The Principles of Psychology*, Dover: New York, 1890.
38. R. Sun, "Learning, action, and consciousness: A hybrid approach towards modeling consciousness," *Neural Networks*, special issue on consciousness, vol. 10, no. 7, pp. 1317–1331, 1997.
39. D. Willingham, M. Nissen, and P. Bullemer, "On the development of procedural knowledge," *Journal of Experimental*

Psychology: Learning, Memory, and Cognition, vol. 15, pp. 1047–1060, 1989.
40. R. Sun, "Robust reasoning: Integrating rule-based and similarity-based reasoning," *Artificial Intelligence*, vol. 75, no. 2, pp. 241–296, 1995.
41. J. Hendler, "Marker passing and microfeature," in *Proc. 10th IJCAI*, Morgan Kaufmann: San Mateo, CA, 1987, pp. 151–154.
42. J. Gelfand, D. Handelman, and S. Lane, "Integrating knowledge-based systems and neural networks for robotic skill acquisition," in *Proc. IJCAI*, Morgan kaufmann: San Mateo, CA, 1989, pp. 193–198.
43. W. Schneider and W. Oliver, "An instructable connectionist/control architecture," in *Architectures for Intelligence*, edited by K. VanLehn, Erlbaum: Hillsdale, NJ, 1991.
44. R. Sun, "A Connectionist model for commonsense reasoning incorporating rules and similarities," *Knowledge Acquisition*, vol. 4, pp. 293–321, 1992.
45. M. Erickson and J. Kruschke, *Rules and Examplars in Category Learning*, manuscript, 1997.
46. A. Reber, "Implicit learning and tacit knowledge," *Journal of Experimental Psychology: General*, vol. 118, no. 3, pp. 219–235, 1989.
47. R. Dominowski, *How do People Discover Concepts*?, manuscript, 1975.
48. D. Medin, W. Wattenmaker, and R. Michalski, "Constraints and preferences in inductive learning: An experimental study of human and machine performance," *Cognitive Science*, vol. 11, pp. 299–339, 1987.
49. C. Watkins, "Learning with Delayed Rewards," Ph.D. Thesis, Cambridge University, Cambridge, UK, 1989.
50. D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific: Belmont, MA, 1996.
51. R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," *Advances in Neural Information Processing Systems*, MIT Press: Cambridge, MA, 1997.
52. D. Precup, R. Sutton, and S. Singh, "Multi-time models for temporary abstract planning," *Advances in Neural Information Processing Systems 10*, MIT Press: Cambridge, MA, 1998.
53. T. Tesauro, "Practical issues in temporal difference learning," *Machine Learning*, vol. 8, pp. 257–277, 1992.
54. J. Boyan and A. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Neural Information Processing Systems*, edited by J. Tesauro and D. Touretzky, and T. Leen, MIT Press: Cambridge, MA, pp. 369–376, 1995.
55. R. Sun, "On variable binding in connectionist networks," *Connection Science*, vol. 4, no. 2, pp. 93–124, 1992.
56. L.M. Fu, "Rule learning by searching on adapted nets," in *Proc. of AAAI'91*, 1991, pp. 590–595.
57. R.C. Lacher, "Expert networks: Paradigmatic conflict, technological rapprochment," *Minds and Machines*, vol 3, pp. 53–71, 1993.
58. G. Towell and J. Shavlik, "Extracting refined rules from knowledge-based neural networks," *Machine Learning*, vol. 13, no. 1, pp. 71–101, 1993.
59. R. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The multipurpose incremental learning system AQ15," in *Proc. of AAAI-86*, Morgan Kaufmann: San Mateo, CA, 1986, pp. 1041–1045.
60. A. McCallum, "Learning to use selective attention and short-term memory in sequential tasks," in *Proc. Conference on Simulation of Adaptive Behavior*, MIT Press: Cambridge, MA, 1996, pp. 315–324.
61. L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
62. Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proc. of ICML'97*, Morgan Kaufmann, San Francisco, CA, 1996, pp. 148–156.
63. R. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, pp. 139–160, 1991.
64. P. Tadepalli and T. Dietterich, "Hierarchical explanation-based reinforcement learning," in *Proc. International Conference on Machine Learning*, Morgan Kaufmann: San Francisco, CA, 1997, pp. 358–366.
65. D. Gordon, A. Schultz, J. Grefenstette, J. Ballas, and M. Perez, *User's Guide to the Navigation and Collision Avoidance Task*, Naval Research Lab, Washington, DC, 1994.
66. R. Sun, E. Merrill, and T. Peterson, "A bottom-up model of skill learning," in *Proc. of 20th Cognitive Science Society Conference*, Lawrence Erlbaum Associates: Mahwah, NJ, pp. 1037–1042, 1998.
67. R. Sun, E. Merrill, and T. Peterson, "Skill learning using a bottom-up hybrid model," in *Proc. of The Second European Conference on Cognitive Modeling*, Nottingham University Press: Nottingham, UK, pp. 23–29, April 1998.
68. E. Smith and D. Medin, *Categories and Concepts*, Cambridge, MA: Harvard University Press, 1981.
69. E. Rosch, "Principles of categorization," in *Cognition and Categorization*, edited by E. Rosch and B. Lloyd, Erlbaum: Hillsdale, NJ, 1978.
70. C. Giraud-Carrier and T. Martinez, "An integrated framework for learning and reasoning," *Journal of Artificial Intelligence Research*, vol. 3, pp. 147–185, 1995.
71. D. Waltz, "How to build a robot," in *Proc. of Conf. on Simulation of Adaptive Behaviors*, edited by S. Wilson, MIT Press: Cambridge, MA, 1991.
72. R. Sun, E. Merrill, and T. Peterson, "A bottom-up model of skill learning," in *Proc. of 20th Cognitive Science Society Conference*, Lawrence Erlbaum Associates: Mahwah, NJ, pp. 1037–1042, 1998.
73. M. Lebowitz, "Experiments with incremental concept formation: UNIMEM," *Machine Learning*, vol. 2, pp. 103–138, 1987.
74. D. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, vol. 2, pp. 139–172, 1987.
75. Stepp and Michalski (1983).
76. W. Shen, "Discovery as autonomous learning from the environment," *Machine Learning*, vol. 12, pp. 143–165, 1993.
77. E. Wisniewski and D. Medin, "On the interaction of data and theory in concept learning," *Cognitive Science*, vol. 18, pp. 221–281, 1994.
78. L. Rips, "Similarity, typicality, and categorization," in *Similarity and Analogical Reasoning*, edited by S. Vosniadou and A. Ortony, Cambridge University Press: New York, NY, 1989.
79. R. Andrews and J. Diederich (Eds.), *Proceedings of the NIPS'96 Workshop on Rule Extraction From Trained Artificial Neural Networks*, NIPS Foundation, 1996.
80. R. Miikkulainen and M. Dyer, "Natural language processing

with modular PDP networks and distributed lexicons," *Cognitive Science*, vol. 15, no. 3, pp. 343–399, 1991.
81. D. Gordon and D. Subramanian, "A cognitive model of learning to navigate," in *Proc. of 18th Cognitive Science Conference*, Lawrence Erlbaum: Mahwah, NJ, 1997, pp. 271–276.
82. T. Johnson, J. Zhang, and H. Wang, "A hybrid learning model of abductive reasoning," in *Connectionist-Symbolic Integration*, edited by R. Sun and F. Alexandre, Lawrence Erlbaum Associates: Mahwah, NJ, 1998.
83. V. Ajjanagadde and L. Shastri, "From simple association to systematic reasoning," Tech. Report MS-CIS-90-05, University of Pennsylvania, Philadelphia, PA, 1990.
84. J. Barnden, "The right of free association: Relative-position encoding for connectionist data structures," in *Proc. 10th Conference of Cognitive Science Society*, Lawrence Erlbaum Associates: Hillsdale, NJ, pp. 503–509, 1988.
85. J. Kruschke, "ALCOVE: an examples-based connectionist model of category learning," *Psychological Review*, vol. 99, pp. 22–44, 1992.
86. D. Touretzky and G. Hinton, "Symbols among neurons," in *Proc. 9th IJCAI*, Morgan Kaufmann, 1987, pp. 238–243.
87. C.L. Giles and M. Gori, "Adaptive processing of sequences and data structures," Lecture Notes in Artificial Intelligence, Springer Verlag, 1998.
88. J. Barnden, "The right of free association: Relative-position encoding for connectionist data structures," in *Proc. 10th Conference of Cognitive Science Society*, Lawrence Erlbaum Associates: Hillsdale, NJ, pp. 503–509, 1988.
89. E. Gat, "Integrating planning and reacting in a heterogeneous architecture," in *Proc. AAAI*, Morgan Kaufmann: San Mateo, CA, 1992, pp. 809–815.
90. W. Schneider and W. Oliver, "An instructable connectionist/control architecture," in *Architectures for Intelligence*, edited by K. VanLehn, Erlbaum: Hillsdale, NJ, 1991.
91. R. Maclin and J. Shavlik, "Incorporating advice into agents that learn from reinforcements," in *Proc. of AAAI-94*, Morgan Kaufmann: San Meteo, CA, 1994.
92. G. Drescher, *Made-up Minds*, MIT Press: Cambridge, MA, 1991.
93. D. Broadbent, P. Fitsgerald, and M. Broadbent, "Implicit and explicit knowledge in the control of complex systems," *British Journal of Psychology*, vol. 77, pp. 33–50, 1986.
94. G. Logan, "Toward an instance theory of automatization," *Psychological Review*, vol. 95, no. 4, pp. 492–527, 1988.
95. W. Stanley, R. Mathews, R. Buss, and S. Kotler-Cope, "Insight without awareness: On the interaction of verbalization, instruction and practice in a simulated process control task," *Quarterly Journal of Experimental Psychology*, vol. 41A, no. 3, pp. 553–577, 1989.
96. D. Gordon and D. Subramanian, "A multistrategy learning scheme for agent knowledge acquisition," *Informatica*, vol. 17, pp. 331–346, 1993.
97. E. Thorndike, *Animal Intelligence*, Hafner: Darien, Connecticutt, 1911.
98. K. VanLehn, "Cognitive skill acquisition," in *Annual Review of Psychology*, edited by J. Spence, J. Darly, and D. Foss, Annual Reviews, Palo Alto, CA, vol. 47, pp. 513–539, 1996.
99. S. Whitehead and D. Ballard, "Learning to perceive and act by trial and error," *Artificial Intelligence*, vol. 7, pp. 45–83, 1991.
100. J. Schlimmer, "Incremental adjustment of representations for learning," in *Proc.of 4th Workshop on Machine Learning*, Morgan Kaufmann: San Mateo, CA, pp. 502–507, 1987.
101. P. Agre and D. Chapman, "What are plans for?," in *Designing Autonomous Agents*, edited by P. Maes, Elsevier: New York, 1990.
102. A. Barto, R. Sutton, and C. Watkins, "Learning and sequential decision-making," in *Learning and Computational Neuroscience*, edited by M. Gabriel and J. Moors, MIT Press: Cambridge, MA.
103. J. LeDoux, "Brain mechanisms of emotion and emotional learning," *Current Opinion in Neurobiology*, vol. 2, no. 2, pp. 191–197, 1992
104. R. Shiffrin and W. Schneider, "Controlled and automatic human information processing II," *Psychological Review*, vol. 84, pp. 127–190, 1977.
105. R. Stepp and R. Michalski, "Conceptual clustering," in *Machine Learning, II*, edited by R. Michalski et al., Morgan Kaufmann: Los Altos, CA, 1986.
106. R. Sun and L. Bookman (Eds.), *Computational Architectures Integrating Neural and Symbolic Processes*, Kluwer Academic Publishers: Norwell, MA, 1994.
107. D. Touretzky and G. Hinton, "Symbols among neurons," in *Proc. 9th IJCAI*, Morgan Kaufmann, 1987, pp. 238–243.
108. K. VanLehn, "Rule acquisiton events in the discovery of problem-solving strategies," *Cognitive Science*, vol. 15, pp. 1–47, 1991.
109. S. Whitehead and L. Lin, "Reinforcement learning of non-Markov decision processes," *Artificial Intelligence*, vol. 73, nos. 1–2, pp. 271–306, 1995.

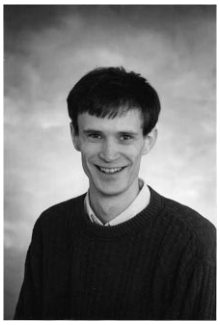
Dr. Ron Sun is an associate professor of computer science and psychology at the University of Alabama and a visiting scientist at NEC Research Institute. He received his Ph.D. in 1991 from Brandeis University in computer science.

Dr. Ron Sun's research interest centers around the studies of intelligence and cognition, especially in the areas of commonsense reasoning, human and machine learning, and hybrid connectionist models. He is the author of over 80 papers, and has written, edited or contributed to 10 books, including authoring the book *Integrating Rules and Connectionism for Robust Commonsense Reasoning*, published by John Wiley and Sons, and co-editing *Computational Architectures Integrating Neural and Symbolic Processes*, published by Kluwer, and *Connectionist-Symbolic Integration*, published by Lawrence Erlbaum Associates. For his paper on integrating

rule-based reasoning and connectionist models, he received the 1991 David Marr Award from Cognitive Science Society.

He organized and chaired the AAAI Workshop on Integrating Neural and Symbolic Processes, 1992, and the IJCAI Workshop on Connectionist-Symbolic Integration, 1995, as well as co-chairing the AAAI Workshop on Cognitive Modeling, 1996. He has also been on the program committees of the National Conference on Artificial Intelligence (AAAI-93, AAAI-97, AAAI-99), International Joint Conference on Neural Networks (IJCNN-99), International Two-Stream Conference on Expert Systems and Neural Networks, International Symposium for Integrating Knowledge and Neural Heuristics, and other conferences, and has been an invited/plenary speaker for some of them.

Dr. Sun serves on the editorial boards of Connection Science, Applied Intelligence, and Neural Computing Surveys. He was a guest editor of the special issue of the journal Connection Science on architectures for integrating neural and symbolic processes and the special issue of IEEE Transaction on Neural Networks on hybrid intelligent models. He is a member of AAAI and Cognitive Science Society, and a senior member of IEEE.



Todd Peterson graduated from Brigham Young University with a bachelor's degree in Computer Science in 1993. He received his masters' degree in Computer Science from the University of Alabama in 1996. He is currently finishing his Ph.D. in Computer Science at The University of Alabama, specializing in function approximation in reinforcement learning. He will be an assistant professor at Brigham Young University starting in 1999. His research interests are in the areas of Machine Learning and Neural Networks.

Edward Merrill has a Ph.D. in Developmental Psychology with a specialty emphasis in Cognitive Psychology, Learning, and Mental Retardation Research. His research has focused on semantic information processing (broadly defined), cognitive aspects of mental retardation/development disabilities, and learning as it relates to various individual difference parameters. He received his Ph.D. from Peabody College of Vanderbilt University in 1982.