A Cognitive Model of Decision Making: Chunking and the Radar Detection Task

Constantine P. Papageorgiou[1]
Kathleen Carley

December 13, 1993.
CMU-CS-93-228

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1] BBN Systems and Technologies
70 Fawcett Street
Cambridge, MA 02138

Abstract

We present Radar-Soar, a model of a cognitive agent learning to classify aircraft as friendly, neutral, or hostile based on a discrete set of characteristics. This agent is built using the Soar cognitive architecture which has as one of its main features a powerful form of explanation-based learning called chunking. We show that the performance this agent exhibits is extremely high and surpasses that of a comparable agent that has previously been built.

Keywords: artificial intelligence, machine learning, organizational behavior, Soar, pattern matching, radar detection

Contents

1 Introduction **3**

2 Soar **5**
2.1 High Level View 5
2.2 Modular Decomposition 6
2.3 Chunking 6

3 The Radar Detection Task **8**

4 Training and Testing **10**

5 Model **11**
5.1 Models of Belief Change 11
5.1.1 Reinforcement Model 11
5.1.2 Information Processing Model 11
5.1.3 Constructural Model 12
5.2 Agent Model 12
5.2.1 General Features 12
5.2.2 Decision Making Process 13
5.3 Soar Implementation 14
5.3.1 High Level Spaces 14
5.3.2 Low Level Spaces 14

6 Experiments and Analysis **17**
6.1 Experiments 17
6.1.1 Baseline 17
6.1.2 Training with the LTS 18
6.1.3 Training with the STS 18
6.1.4 Training with the SSTS 21
6.2 Analysis 21
6.2.1 Performance 21
6.2.2 Chunks 24

7 Conclusion **25**

List of Figures

5.1 Radar-Soar Problem Spaces 15

6.1 Percentage of Correct Classifications With No Chunking during Training 18
6.2 Percentage of Correct Classifications (LTS) during Training 19
6.3 Percentage of Correct Classifications (LTS) during Testing 19
6.4 Percentage of Correct Classifications (STS) during Training 20
6.5 Percentage of Correct Classifications (STS) during Testing 20
6.6 Percentage of Correct Classifications (SSTS) during Training 21
6.7 Percentage of Correct Classifications (SSTS) during Testing 22
6.8 Percentage of Correct Classifications during Testing 22
6.9 Agent Performance Table 23
6.10 SSTS Data 23

Chapter 1

Introduction

Organizational performance is an issue that pervades many aspects of our society. The value of this subject becomes evident upon realizing that most societal interactions can be regarded as interactions between different organizations. Taking this one step further, the performance of these organizations is of central importance; if bad or incorrect decisions are made most of the time, the organization will fail, whereas making a majority of good or correct decisions leads to the success of the organization. It also becomes obvious that there are many internal and external factors or stressors that can cause the performance of an organization to deteriorate. Thus, organizational performance is, in general, not constant. Training may mitigate the effect of such stressors.

Training procedures are of special interest to organizations where performance is more critical than in other organizations. The targeted organizations are those which are involved in certain dangerous or critically important tasks; for example, it is easy to see that more is at stake when designing a training procedure for an agent who is guiding airplanes from a control tower is more than one who is separating balls by color at a toy manufacturing plant. As such, there has been much research into identifying different training procedures and the effects of training in improving performance. A significant increase in recent research in this area has been a direct result of an instance of the identify-friend-or-foe task where the incorrect identification of an aircraft resulted in the direct loss of more than 200 civilian lives – the July 3, 1988 tragic shooting of an Iranian passenger plane over the Persian Gulf by the U.S.S. Vincennes.

A training procedure is a process used to give an agent a basis from which to make decisions. Two different training processes that have been examined within the organizational context are experiential training, where the agents are trained such that their current decisions are a function of certain aspects of their past that are labeled "experience", and operational training, where the agents use a standard operating procedure (SOP) to evaluate the situation and make a decision [Car91, LC91].

Much research has been done on organizational performance when the agents in the organization have been experientially trained such that their current decisions are made using a historical dominance rule - a simple mathematical procedure in which the agent chooses as its decision that choice that in its experience was most often correct, assuming perfect recall. Cognitive agents can certainly learn through experience, but unlike the foregoing procedure, they can learn in a more piecewise fashion, e.g. through chunking. A consequence of this is that cognitive agents should be capable of making decisions using other procedures, e.g. generalization.

This research focuses on the use of a cognitive model of an agent within an organizational framework. Specifically, we examine whether the learning and decision making capability of a cognitive agent exhibits better behavior than the more mathematical agent. This research will be carries out by applying Soar to a task that has been frequently studied in the organizational literature, the radar task.

Soar, a cognitive architecture for general learning, provides an ideal medium through which to examine these issues. Soar's knowledge is encoded as rules in a production system; these rules are used in a goal-directed decision making process to solve problems. The specific aspect of this process that is relevant here is the use of "chunking". As Soar searches its problem space moving towards a goal, it often reaches a state where the immediately available knowledge is not sufficient for further progress. At this point, it sets up a subgoal which it attempts to solve in the same manner as the supergoal. Special new productions called chunks are built after Soar is done working on a subgoal. These chunks use the path of information that led to the current subgoal results as the conditions of these productions and the results of the subgoal as the new productions' actions. One of the most important aspects of this process is that the chunks created are general so as to apply in comparable situations [LRN84].

The task that we examine is the radar detection task: given a single flying object in a physical air space being scanned by some agent, it is the goal of this agent to determine if the object is friendly, neutral, or hostile. The ultimate decision is some function of the values of certain characteristics of the flying object; some examples of pertinent characteristics are speed, direction, altitude, and identification.

The insight gained from the research on this specific task can easily be applied to many other areas where there is some form of decision making occurring; it is possible to abstract away from this research the general task of learning to make societal decisions based on the characteristics of a given situation. The performance of this agent on the radar detection task will be evaluated. Specifically, the value of chunking as a mechanism for the development of cognitive models of learning is assessed as a result of this research. The results of this research are compared to previous related research done on the radar detection task [LC91] in order that we obtain a more complete view of chunking's actual capabilities. Thus, this research extends our understanding of Soar as a cognitive agent.

Chapter 2

Soar

Soar is a cognitive architecture [LNR87] that has been successfully applied in such diverse domains as: cognitive modeling [Aky90, DS90, JRS91], development [SK91], expert systems [HPS89], machine learning [LRN84], natural language processing [LLN91], and robotics [LYHT91].

2.1 High Level View

Soar's goal directed activity is based around searches in problem spaces. Problem spaces are sets of states that contain the information representing the system's knowledge. A problem space is searched by using operators to transform the current state to successive new states.

When Soar initially starts solving a task, it enters some initial state in the highest level problem space. As operators are applied in that problem space, the state is changed. It is often the case that to achieve a goal in the current problem space, it is first necessary to solve another goal. In such situations, a new problem space is set up in which Soar pursues this more important goal.

Soar's problem solving mechanism often encounters situations where there is not enough direct knowledge available to proceed in solving the current goal. These situations are known as impasses. A tie impasse occurs when there is more than one operator that can be applied to the current state and not enough search knowledge to differentiate or preferentially treat any of the possible choices. An impasse also occurs when two operators can be applied to the current state and the system's knowledge dictates both that choosing operator 1 is better than choosing operator 2 and choosing operator 2 is better than choosing operator 1; this is known as a conflict impasse. Whenever an impasse does not allow Soar's decision making mechanism to proceed, Soar's default knowledge dictates that it set up a subgoal to resolve the impasse. This is known as automatic subgoaling because the information that leads to the detection and resolution of impasses is contained within the Soar architecture. It is important to note that the only times that Soar subgoals is when there is an impasse blocking further action. These subgoals are satisfied whenever the discrepancy or lack of knowledge that caused the impasse is overcome using the same goal-directed search as in conventional problem spaces.

Chunking is an explanation-based learning mechanism [RL86] used by Soar to increase system knowledge. When Soar satisfies a subgoal, the information that was used to reach the decision as well as the information in the state that led to the impasse are embodied in a new piece of knowledge, called a chunk. A full discussion of chunking and its implications is reserved for its own section.

2.2 Modular Decomposition

Soar is composed of four modules: Recognition Memory, IO, Decide, and Chunking. While this research deals mainly with chunking, an introduction to all the modules is necessary for the reader to understand the process by which Soar makes decisions.

- Recognition Memory is a cognitive memory model based in a production system. All permanent knowledge is represented as productions with left hand sides and right hand sides. The left hand sides of productions are conditions that must be satisfied so that the actions on the right hand side can be instantiated. Therefore, productions can be regarded as complex if-then rules.
- IO allows Soar to interact with the world, by receiving perceptions and issuing motor actions through simulations or user-inputed instructions.
- Decide [Lai84] controls Soar by maintaining a goal stack organized around a search in the multiple problem space/state space model discussed above. Soar operates in a temporal framework composed of decision cycles. The first part of the decision cycle is spent in matching productions to current knowledge. The second part of each decision cycle is where Soar selects which operator to fire, which problem space or state to choose, or which goal or subgoal to attend to next.
- Chunking creates new productions by observing and remembering the performance of Recognition Memory and Decide. These new rules are generalized to apply in situations which are comparable to, but differ from the exact situation in which they were learned. This phenomenon is discussed later in more detail.

The process by which Soar's decision making process progresses is as follows. Through the IO module, Soar perceives input from what we can conveniently label the "outside world", entailing any sort of knowledge that is current and ephemeral. This knowledge is placed in Working Memory. Working Memory holds all the knowledge that the system uses to attend to current goals. The elements of Working Memory are matched against the left hand sides of the rules in Recognition Memory; those Working Memory Elements that do match some production(s) form the Match Set. Working Memory is modified by either changing or adding information; it is not possible to explicitly remove information from Working Memory. As Soar matches and fires productions, the traces generated by doing so are placed in Trace Memory. The Chunking module analyzes these traces by backtracing through the productions contained therein. New productions are created based on Trace Memory and are placed into Recognition Memory.

2.3 Chunking

Chunking essentially extracts that part of the state of the system that was used to resolve the impasse in the subgoal from the traces in Trace Memory and creates a production which encapsulates this information. By backtracing through the traces, Soar identifies the parts of the state before the impasse arose that led to a solution in the subgoal and puts this information into the left hand side of a chunk. The right hand side of the chunk contains the new knowledge elicited from the subgoal – those Working Memory Elements that were added or changed as a result of the subgoaling procedure.

A significant feature of the process by which chunks are created is that the information that is extracted from the traces is generalized to a certain degree. This allows chunks to apply in situations which are similar, though not equivalent, to the situations in which they were created. A negative aspect of this generalization process that often arises is the phenomenon of overgeneralization. Overgeneralization of a chunk occurs when chunking cannot completely determine the conditions that led to the resolution of an impasse. For instance, suppose that Soar builds the following chunk:

```
if
    it is hot outside and
    the grass is brown
then
    it is not raining
```

Seeing that the condition that `there are no clouds in the sky` was somehow not captured in the trace that led to the notion that it is not raining, it becomes evident that the proper chunk should have had the form:

```
if
    it is hot outside and
    the grass is brown and
    there are no clouds in the sky
then
    it is not raining
```

The actual chunk that was built is therefore grossly overgeneral; Soar will decide that it is not raining whenever it has the knowledge that it is hot outside and the grass is brown. This simple example effectively illustrates the concept of overgeneral chunks.

The implications of chunking are summarized:

- When Soar is in a situation which it has been in before, it remembers the correct solution. This supports a permanent memory model of human cognition.
- Given that Soar operates in a problem space search model, with chunking much of this search process can be eliminated. This leads to more efficient search which is evidenced empirically through faster answers. This supports the phenomenon observed in human cognition that, with practice, response time decreases in the field of interest.

Chapter 3

The Radar Detection Task

The radar detection task, also known as the identify-friend-or-foe (IFF) task, is a highly studied problem in the literature. The problem is for one or more agents to classify an arbitrary number of aircraft in some airspace as friendly, neutral, or hostile. This decision is based on some combination of characteristics of the aircraft and their interrelation. It can easily be seen that for a novice this task is exceedingly complex and difficult to accomplish successfully.

The task that we address in this paper is a stylized radar detection task. The problem space is some physical air space which a single agent is scanning for aircraft and the task of the agent is to classify each plane as friendly, neutral, or hostile. Aircraft permeate this airspace one at a time in succession; at any point in time, there is at most one aircraft in the airspace that the agent is responsible for. Each aircraft has nine characteristics that the agent can observe:

- speed
- direction
- range
- altitude
- angle
- corridor status
- identification
- size
- radar emission type

Each of these characteristics can have a value of low (1), medium (2), or high (3); singly, each of these values is to be interpreted as being indicative of a friendly, neutral, or hostile aircraft, respectively. With three possible values for each of the nine characteristics, we see that there are 19,683 (3^9) different unique combinations satisfying the given constraints. The true classification of aircraft is determined by an unbiased, decomposable model of the task at hand. The task is decomposable in the sense that each characteristic has the same weight in determining the true classification of the aircraft. We chose that the task be unbiased so that there is an equal number of each type of aircraft (friendly, neutral, hostile) that the agent is exposed to. With these constraints, the formulae for determining the true classification of aircraft are as follows:

sum = speed + direction + range + altitude + angle + corridor + status + identification + size + radar emission type

If 9 <= sum <= 16 then the true classification is friendly
If 17 <= sum <= 19 then the true classification is neutral
If 20 <= sum <= 27 then the true classification is hostile

Assumptions are made in this stylized radar task that differentiate it from the general task in several aspects. First, we assume that the agent correctly observes all of the characteristics of a given plane. This is an important assumption since decisions made when there is missing information may affect the data; we assume that the agent is always presented with complete information for simplicity. Second, the classification of the plane is based on exactly the set of characteristics that are listed above, nothing more and nothing less.

While it does not share the excessively rich nature of the general radar detection task, studying the stylized version is important for many reasons.

- As stated above, this task has been studied to a great degree due to its ramifications on civilian and military air-traffic control. As such, there is much interest in developing models of learning for this task.
- While there are not an arbitrary number of possible aircraft, the 19,683 combinations in the stylized version still allow for a good deal of variation in the aircraft that the agent observes.
- In assuming that a single agent is observing and classifying the aircraft, we can develop a model of learning for this case and eventually extend it for the more interesting case of multiple agents.
- Even though the task has been labeled the stylized radar detection task, this nomenclature is misleading. Essentially, this task is a glorified ternary decision task which could be adapted to any situation in which there are three distinct choices for some agent.
- Since the true classification of each plane can be determined, the agent can receive correct, constructive feedback and, in such a manner, learn how to classify aircraft.

Chapter 4

Training and Testing

The radar detection task can be broken into two parts: training and testing. The agent spends the first part of the radar detection task in a training procedure where it learns to classify aircraft. In the second part, the agent uses the knowledge obtained during training to identify more aircraft. The agent's performance is evaluated during both sections but is considered especially important in the testing section.

There are several common training scenarios that are relevant to the radar-detection task – experiential training and operational training are two of the most widely used. In the experiential training procedure, the agent bases its decision on its historical experience with the task. For example, if an agent sees an object with the set of characteristics X and remembers that it had previously seen an object with the same characteristics, it will assign the classification of this earlier categorized object to the current object. When an agent is operationally trained, it makes decisions using a standard operating procedure (SOP). An SOP is an enumeration of a set of conditions that the agent might be exposed to, coupled with the corresponding actions or decisions for each condition. As stated in the introduction, the training procedure that we are investigating is an experiential procedure where the agent develops a cognitive model of the task it is attending to and uses this model to make further decisions. The model of learning is discussed in full in the next chapter.

In the training procedure, the agent is shown, in succession, single aircraft with different sets of characteristics. As the agent sees the aircraft, it is asked to classify them. After making the recommendation, it is given feedback as to the exact classification of the aircraft. After a predetermined number of trials, the agent is considered trained.

In testing, the succession of events is the same as in training with the exception that now the agent is given no feedback as to the correctness of the classifications and must rely solely on the knowledge obtained during the training portion of the task.

Chapter 5

Model

5.1 Models of Belief Change

The model of learning and belief change that Radar-Soar uses to create new knowledge and apply existing knowledge to classify aircraft adheres to the basic principles of the Soar architecture and, at the same time, reflects the structure of empirical results in the literature. An introduction to several popular models of belief change will serve as a background against which the actual model used by Radar-Soar can be discussed.

Without delving into confounding philosophical argument, for the purposes of this paper it is sufficient to define a belief as an idea that is drawn from facts or other beliefs the individual ascribes to. Beliefs are contrasted with facts; facts are necessarily beliefs, while beliefs are not necessarily facts. In other words, a fact is a truthful belief. For obvious reasons, belief change is an integral aspect of any training process; without it, there would be no modification of knowledge leading to better performance.

The several models of belief change that are pertinent to the discussion are:

- reinforcement model
- information processing model
- constructural model

5.1.1 Reinforcement Model

The reinforcement model argues that beliefs, while becoming heavily reinforced when complementary information is provided, can be changed by a proportional amount of contradictory information. Extreme beliefs are more resistant to change than conservative beliefs. An interesting aspect of all three models that will be addressed in the later discussion in this section is the direction and degree of belief change given a sequence of observations. Reinforcement theory says that, while the "sign" of the final belief is independent of the order in which the observations occur, the magnitude or confidence of the beliefs are affected by the ordering.

5.1.2 Information Processing Model

Information processing theory postulates that incoming information is compared with current beliefs and leads to an incremental change in the belief in the direction of the new information. The impact on the final belief can be gauged using a metric where +1 is an extreme positive belief, 0 is a neutral belief, and -1 is an extreme negative belief. Assume that the agent currently has belief X and received information Y. If belief

X is at level +0.5 and observation Y affords a level of +1 – the new information is more positive than the currently held belief – the agent's belief will be shifted to a more positive level. On the other hand, a belief at level +1, when augmented with information at level +0.5, yields a shift to a less positive belief [HDC84]. More recent information is given greater attention or weight than earlier information. In the case that an agent's earlier impressions are heavily reinforced by other observations, a large number of new information is required to change this set belief.

5.1.3 Constructural Model

The constructural model of belief change merges key aspects of both the reinforcement and information processing models, creating a theory whose predictions are supported by a wide range of empirical data [Car91]. Like the information processing model, the constructural model predicts that beliefs are affected in the direction of the discrepancy between the belief and the current information. Furthermore, it argues that extreme beliefs are affected to a greater degree than conservative beliefs when contradictory information is presented. Newer beliefs and information are given the most weight in any belief change process.

5.2 Agent Model

The Soar model of the agent learning to classify aircraft is relatively simple, yet has certain characteristics that warrant mentioning based on what we currently know about human cognition and the theories that were just presented. Recognition Memory (permanent memory) is the main storage for the agent's knowledge of the classification of aircraft.

5.2.1 General Features

Perfect Permanent Memory One of the major assumptions behind the Soar agent is that its memory is perfect. Information that is recalled from memory is always exactly what was initially stored in memory. When this model is viewed in the context of a lifetime of development and refinement, one observes that information the agent has, or beliefs that it ascribes to, never degrade over time. This permanence at first might seem like an implausible assumption. If we consider that humans have consistently been shown to develop this type of error-free decision making or action in tasks like classification of objects and depth perception when developed over the course of years, this feature of the model could be justified.

Temporal Memory Every belief is temporally related to other beliefs. Soar does not support temporal relations between objects but some functionality was added to the system in the form of LISP code that communicates with the Soar model, thereby emulating a basic temporal memory.

Strictly Positive Beliefs Each belief that the agent has is a positive belief; the agent can have no negative beliefs. The ramifications of this are that an agent can believe that a plane "is friendly" but cannot believe that the same plane "is not neutral or hostile". This design choice was instantiated mainly for simplicity. In the Soar architecture information in permanent memory cannot be removed or changed, only added to. In some sense, negative beliefs are encapsulated in the beliefs that are not temporally current and the beliefs that the agent does not have.

Lack of Belief Confidence Levels In the Radar-Soar model, there are no provisions for the amount of confidence or information associated with a belief. This is certainly a feature which should be changed in the future; the idea that every possible belief has the same amount of information backing it is not realistic. For

the purposes of this model, though, some of the effects of this shortcoming are hidden by using the temporal relation of knowledge as a confidence level.

5.2.2 Decision Making Process

The following paragraphs describe those parts of the model that are directly related to the agent's classification process.

Temporal Belief Weighting To classify aircraft, the agent uses more recent beliefs in place of older beliefs. If the agent sees information X leading to belief A and then sees the same information X leading to belief B at a later time, the agent's further decisions regarding X will be based on belief B. This part of the model conforms to constructural theory predictions – all of the agent's decisions are based on the most temporally current beliefs.

Pattern Matching The Soar agent learns to classify aircraft by pattern matching against the values of characteristics of the objects it sees in the airspace. For instance, let us assume that the agent observes an aircraft with the following set of characteristics and values:

- speed = low
- direction = high
- range = high
- altitude = medium
- angle = high
- corridor = low
- identification = medium
- size = high
- emission = high

If the agent finds out that this aircraft was hostile, rather than learning that an aircraft which has speed = low, direction = high, ..., and emission = high is classified as hostile, he will learn that an agent which has 2 low values, 2 medium values, and 5 high values is classified as hostile.

Guessing Under Uncertainty If the agent cannot find any information in permanent memory regarding the current aircraft, the current system model is such that the agent guesses its classification. In this way, the agent is forced to generate an answer and has no response of the type "I don't know" or "I don't have enough information". In the second generation of Radar-Soar, whenever the agent's memory does not have the exact information that matches the current aircraft, it should be able to pick the closest match to the knowledge it has via some heuristic function.

5.3 Soar Implementation

This section describes the actual implementation of the Radar Task in detail. The discussion is accomplished through an analysis of the problem spaces that emerged in the Soar system. For the purposes of this paper, a trial signifies the following loop:

1. observe aircraft
2. classify aircraft

5.3.1 High Level Spaces

The problem spaces that are listed here are those which, when seen as a whole, are the minimal subset of all the problem spaces that form a coherent picture of the actions that occur during a training and testing session with Radar-Soar.

radar-driver The radar-driver problem space serves to sequence the the set of trials for the agent during training and testing. It initializes the agent's temporal memory so that it has no information or knowledge about classifying aircraft.

radar The radar space is the Soar representation for one trial. Initially, the agent observes the aircraft and, from the nine characteristics that it sees, it automatically generates a series of correspondences between the values low, medium, and high, and the number of characteristics with the different values. These correspondences are the pattern of characteristics that the agent sees and will be what the agent uses to classify the aircraft. The agent then classifies the aircraft in the classify-aircraft problem space and updates its memory with the correct classification of the aircraft.

classify-aircraft In this space, the Soar agent classifies the current aircraft. The agent draws information from the temporal memory and the appropriate chunks that are contained in permanent memory, if any, in the find-answer problem space to arrive at a classification for the aircraft. If no such information exists, the current model dictates that the agent simply guess the aircraft's classification.

find-answer This space is that which attempts to find an answer as to the classification of the aircraft by retrieving the classification from permanent memory if it is there.

5.3.2 Low Level Spaces

These problem spaces add the necessary support to the previous three problem spaces to facilitate chunking. As such, though they are integral to the system's functionality and success, they do not contribute to the overall picture of what the agent is doing at a high level. Their significance, though, necessitates a discussion of their characteristics.

assimilate-situation The assimilate-situation problem space draws out the individual aspects of a situation in the assimilate-aircraft space and creates a chunk out of them. The chunk is given a symbol which serves as a unique identifier to aid in recall. This space, while not absolutely necessary for the current model of the radar task where there is strictly one aircraft at a time in the air space that the agent is scanning, was provided to help enable possible expansion of the system to handle multiple aircraft in the air space concurrently.

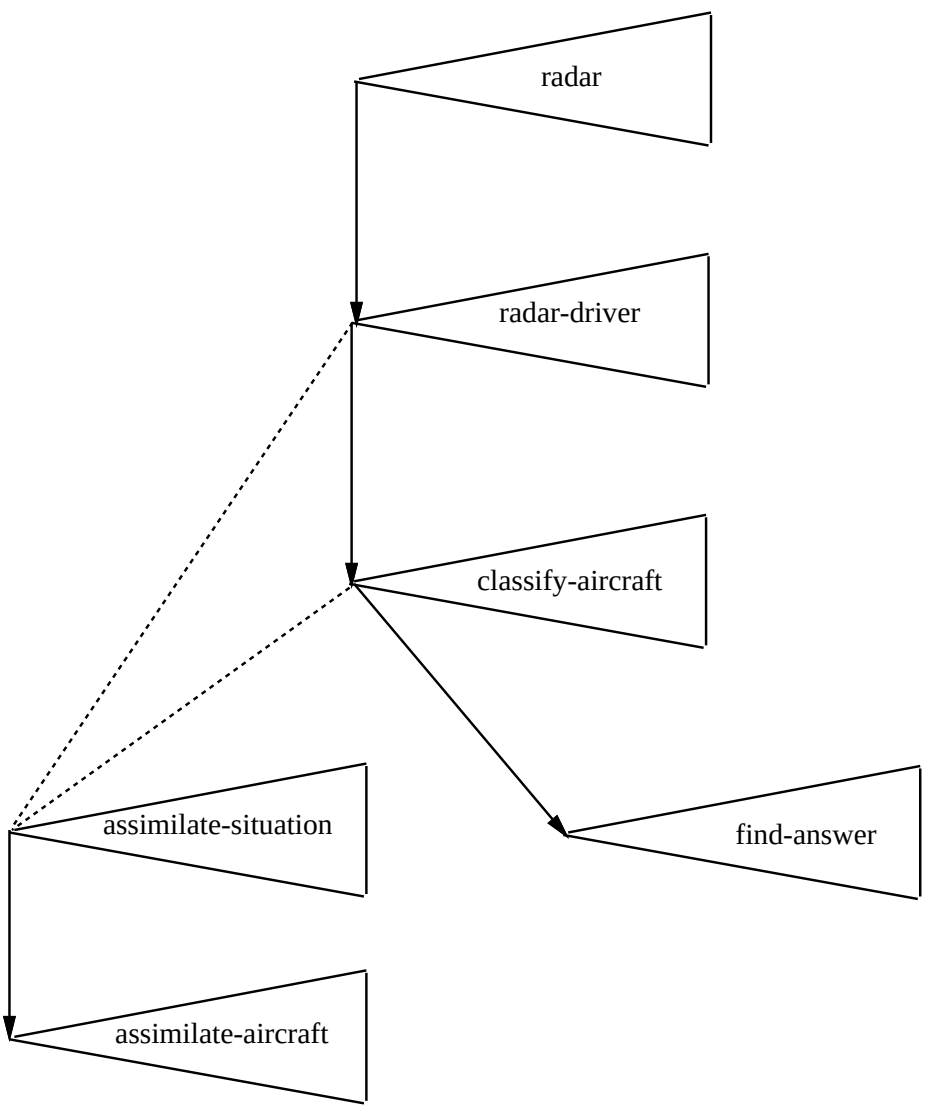

Figure 5.1: Radar-Soar Problem Spaces

assimilate-aircraft The assimilate-aircraft space draws out the individual characteristics of an aircraft in the air space and allows the system to create chunks where the individual characteristics are recognized once the aircraft has been recognized.

Chapter 6

Experiments and Analysis

6.1 Experiments

To analyze the Soar agent that we created, we measured the agent's performance by running it through several different training and testing sessions. The agent's performance on the task is defined to be the percentage of correct classifications that the agent decides on. To determine the effects of the size and composition of the aircraft training sets on the agents performance during testing, we created three different training sets:

- Large Training Set (LTS) — This training set consists of 50 friendly, 50 neutral, and 50 hostile aircraft making a total of 150 aircraft. Each of the different aircraft is randomly chosen from the set of 19,683 different possibilities.
- Small Training Set (STS) — This training set consists of four friendly, four neutral, and four hostile aircraft making a total of twelve aircraft. Each of the different aircraft is randomly chosen from the set of 19,683 different possibilities.
- Small Special Training Set (SSTS) — This training set, like the STS, consists of four friendly, four neutral, and four hostile aircraft. Each of these subsets of four aircraft, though, were chosen to be representative elements of their classification. For instance, the subset of friendly aircraft includes the aircraft whose characteristics are all low as well as three other friendly aircraft: one whose characteristics are low except for a medium speed and medium size, one whose characteristics are all low except for a medium size and a high altitude, and so on.

Like the STS, the Test Set (TS) is composed of 50 friendly, 50 neutral, and 50 hostile aircraft all of which are different and randomly chosen. The general task setup is that the agent is first run through the training set and then the testing set.

6.1.1 Baseline

To acquire baseline data representing an untrained agent who guesses the classification of every aircraft, the agent's performance was measured at the end of the training part of the trial where the LTS is used and chunking was turned off. Given that in each of the training sets there are an equivalent number of friendly, neutral, and hostile aircraft, we would expect the agent to correctly guess each classification approximately one third of the time. This is in fact supported by the data we collected. After the 150 aircraft the agent has

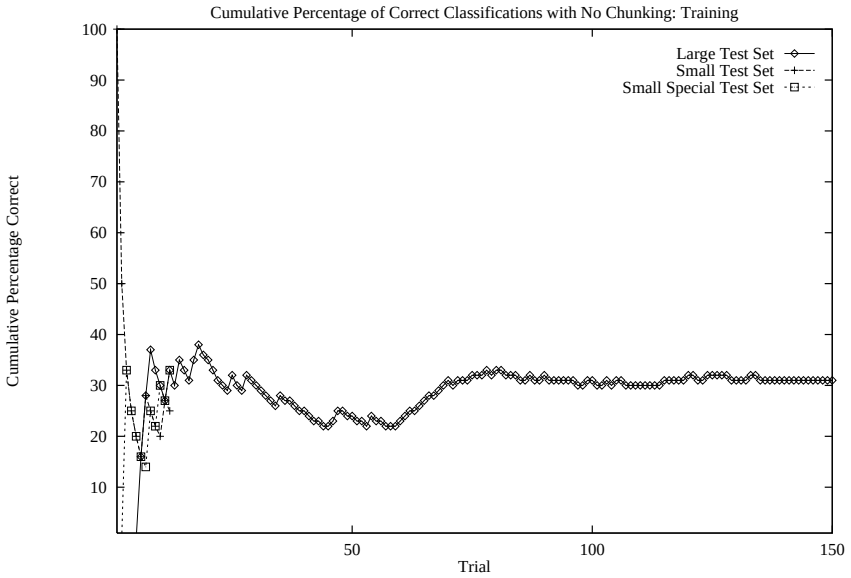

Figure 6.1: Percentage of Correct Classifications With No Chunking during Training

correctly guessed approximately 31% of the aircrafts' classifications. The cumulative percentage of correct guesses are shown in Figure 6.1.

6.1.2 Training with the LTS

For this section, chunking was turned on, the agent was trained with the LTS, and then attempted to classify the 150 new planes in the TS without chunking. Figure 6.2 shows the percentage of aircraft that were classified correctly during training.

From the graphs, we can see that at the end of training, the Soar agent has correctly classified approximately 84% of the aircraft, significantly above the rate for guessing. Figure 6.3 shows the agent's performance during testing using the knowledge that was acquired during training.

In this setup, the agent classifies almost every aircraft (99.33%) correctly. With the large number and variety of aircraft in the LTS, the agent learns how to classify aircraft nearly perfectly.

6.1.3 Training with the STS

In this part, chunking was turned on, the agent was trained with the STS, and then attempted to classify the planes in the TS without using chunking. The results at the end of the training session are shown in Figure 6.4.

The graphs show that the agent has classified approximately 50% of the aircraft correctly at the end of the training session. Figure 6.5 show the number and percentage of correctly classified aircraft after training.

While the performance of the agent after learning with the STS is not as high as after learning with the LTS, classification still occurs at a 62% correctness rate. It is important to note the difference in training set size when regarding the LTS-trained agent as "better" than the STS-trained agent. The LTS-trained agent had previously seen 150 aircraft while the STS-trained agent saw only 12 aircraft and still managed to successfully classify over 60% of the aircraft that followed.

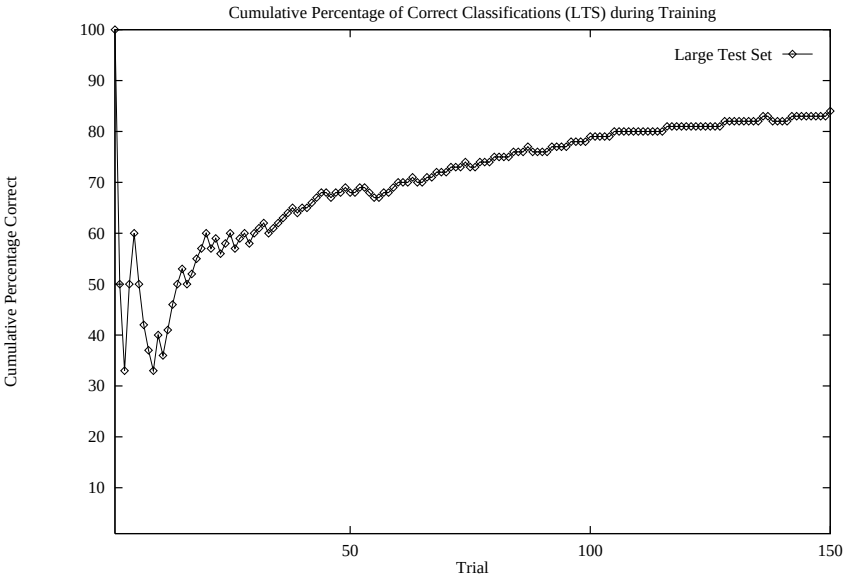

Figure 6.2: Percentage of Correct Classifications (LTS) during Training

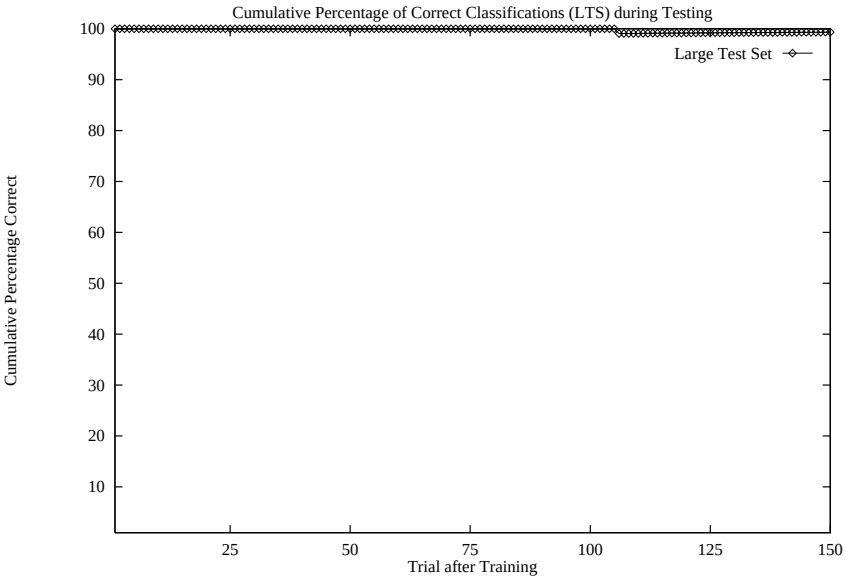

Figure 6.3: Percentage of Correct Classifications (LTS) during Testing

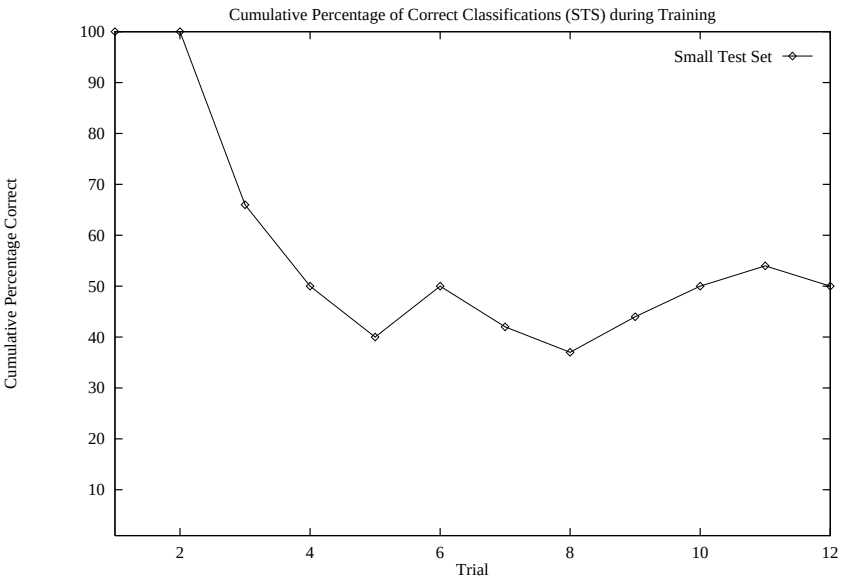

Figure 6.4: Percentage of Correct Classifications (STS) during Training

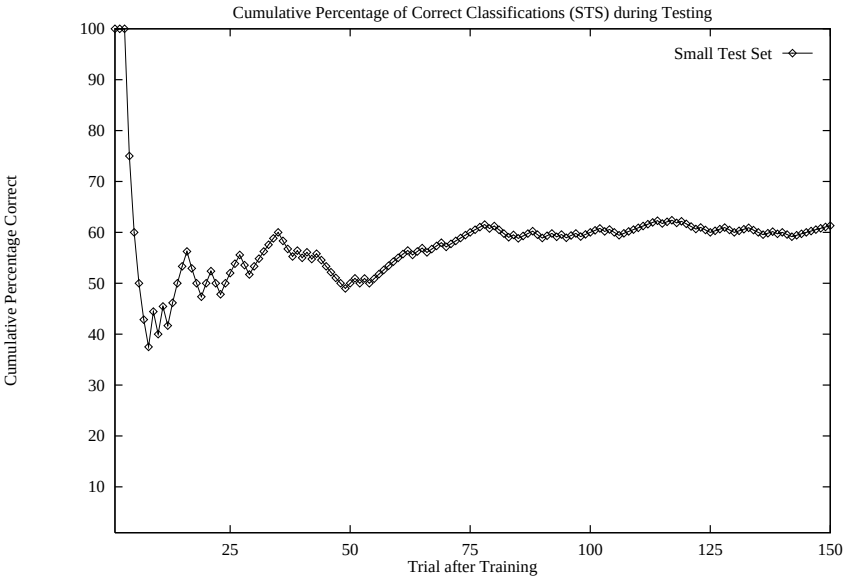

Figure 6.5: Percentage of Correct Classifications (STS) during Testing

6.1.4 Training with the SSTS

The final experimental setup was with the agent being trained on the SSTS with chunking and then classifying the 150 aircraft in TS without chunking. Figure 6.6 show the cumulative results of learning during training.

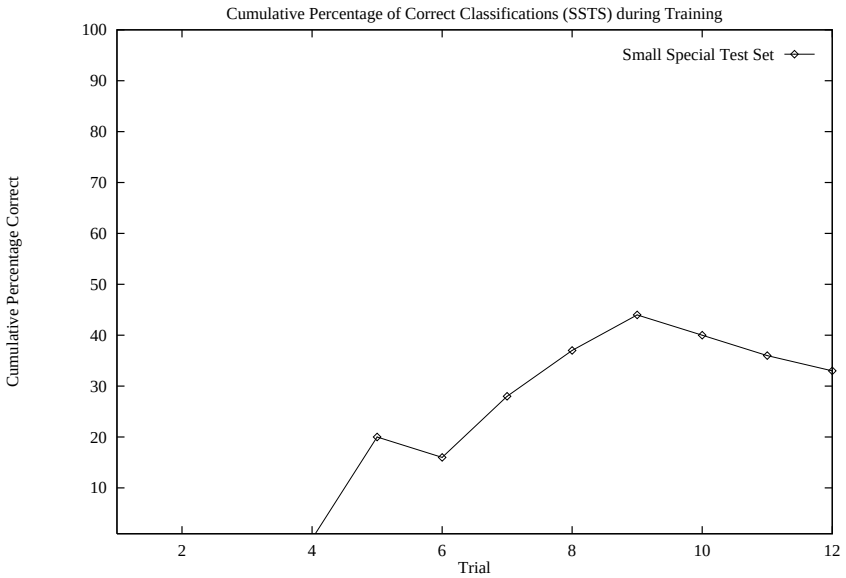

Figure 6.6: Percentage of Correct Classifications (SSTS) during Training

During training with the SSTS training set, the agent correctly classifies 25% of the aircraft that are observed. The post-SSTS training results are in Figure 6.7.

We can see that the agent correctly determines 33% of the aircrafts' classifications after training with the SSTS, approximately the same as we would expect with random guessing.

6.2 Analysis

6.2.1 Performance

In this section, we analyze the data that is presented in Section 6.1 and the chunks that are built during the trials. The results of these experiments are summarized in Figure 6.8 and the table that follows.

One of the first things that one notices about the data presented in Figure 6.9 is that the performance of the SSTS-trained agent is exceedingly low. As a matter of fact, the rate of 33.33% matches the performance of an untrained agent guessing the aircraft classifications. Having been trained on a set of data that is composed of just the most typical aircraft from every class, we would expect that this method of training be the most powerful and yield the highest post-training performance. The situation is completely the opposite as evidenced by the above table. To further discuss this phenomenon, the actual SSTS data is presented below (1 is friendly, 2 is neutral, 3 is hostile) in Figure 6.10.

The reason that the performance is so low has to do with the relative uniformity of the aircraft in the SSTS and can be uncovered through a combinatorial argument. In short, the agent that is trained with the SSTS data is exposed to a more limited set of patterns in relation to the total set of 19,683 possible aircraft.

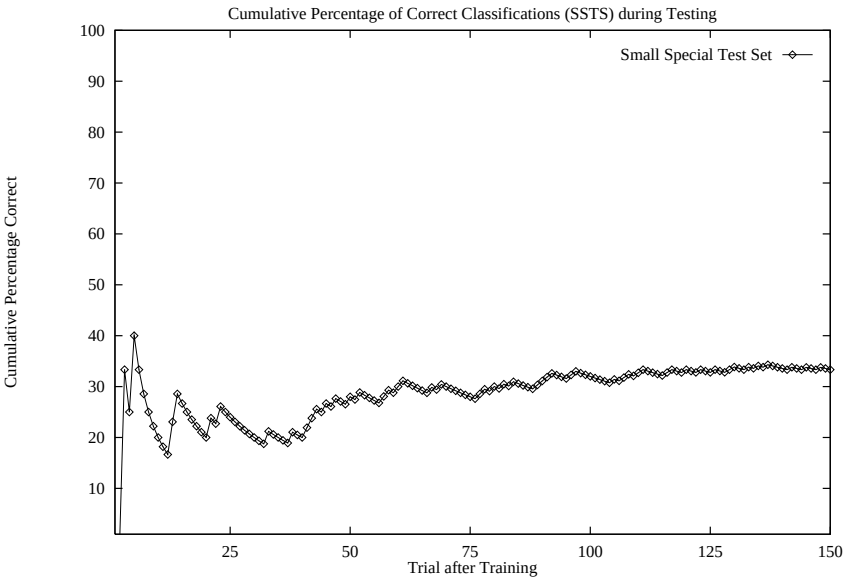

Figure 6.7: Percentage of Correct Classifications (SSTS) during Testing

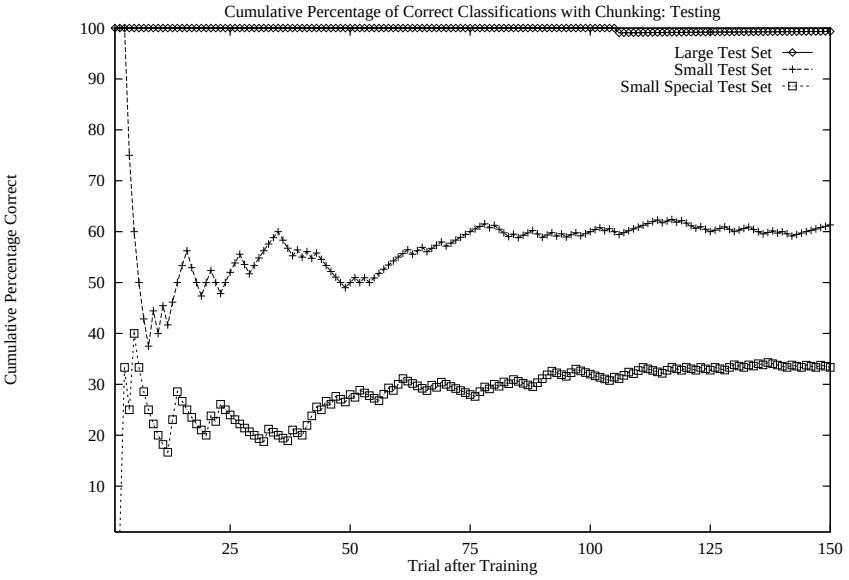

Figure 6.8: Percentage of Correct Classifications during Testing

Training Set	*Performance During Training*	*Performance After Training*
Baseline	NA	31.00
LTS (150)	83.00	99.33
STS (12)	50.00	61.33
SSTS (12)	25.00	33.33

Figure 6.9: Agent Performance Table

friendly	133111111
	111311121
	111111111
	211111121
neutral	212222122
	222222222
	222312222
	222232232
hostile	113333333
	332133333
	323333233
	333333333

Figure 6.10: SSTS Data

We can look at the pattern afforded by one of the test cases in the SSTS, 332133333 for instance. Out of all 19,683 different combinations, there are 72 aircraft which have this pattern of one low characteristic, one medium characteristic, and seven high characteristics. On the other hand, with a less extreme case where the aircraft's characteristics are 322212311, the pattern is three low characteristics, four medium characteristics, and two high characteristics. There are approximately 1,000 different aircraft with this pattern. Since the training and testing sets are composed of different planes, the probability of the agent seeing another aircraft which has the first pattern is extremely low when compared to the chance of the agent seeing the second pattern. This is where the low performance value is originating from. While low performance with training on the most typical planes does not follow our intuition, both the manner in which the TS was set up and the model of the agent which does not attempt to classify aircraft according to the closest pattern it has in memory yield this degraded performance.

Another interesting aspect of the data in Figure 6.9 is the high performance on the agent trained with the LTS; it misclassifies only one of the 150 aircraft in TS. After being trained to classify aircraft on a training set of size 150, it is evident that the Soar agent has learned this process to an impressive degree. The next agent, that which is trained with the STS, correctly determines aircraft classification at a rate of 61.33% after training. This is significantly less spectacular than the LTS-trained agent's performance when taken simply as a percentage but deserves closer look. The STS-trained agent correctly classifies 85 of the 150 aircraft: approximately seven correct classifications per aircraft trained on. On the other hand, the LTS-trained agent correctly classifies 149 of the 150 aircraft: approximately one correct classification per aircraft trained on. This means that the STS-trained agent's training was in effect more valuable than the training of the LTS-trained agent; in a sense, we could say that the LTS overtrained the agent since after a point it seems that it gained only redundant information. We can postulate that there is a medium sized training set ($12 < n < 150$) which generates optimal performance without overtraining the agents. This will be addressed in the conclusion.

We can also compare this system's performance to that of a previous system built for the radar task

[LC91]. This experientially trained agent (organization) kept a record of every previous aircraft that was seen and the number of times each different decision was correct. With this model, overall performance was 59.93% after being trained on all 19,683 aircraft. In our model, after being trained on only 12 aircraft, the performance of the agent is slightly better: 61.33%. The Soar model, therefore, learns efficiently.

6.2.2 Chunks

The chunks that are generated in each of the three training cases prove to be of little interest as their form is identical. This is due to the manner in which correct knowledge is added to Recognition Memory – it is constant over all training sets. It is expected that as the model is enhanced, the nature of the chunks will change and will become more rich in nature. For completeness' sake, though, one set of chunks learned during a training session is presented below.

```
(sp p180
    (goal <g1> state <s1> operator <o1>)
    (state <s1> dummy-att* true)
    (operator <o1> name assimilate-aircraft object <n1>)
    (object <n1> -name low 3 medium 4 high 2 timetag 1)
-->
    (object <n1> name g178 &, g178 +))

(sp p193
    (goal <g1> state <s1> operator <o1>)
    (state <s1> dummy-att* true)
    (operator <o1> name assimilate-aircraft object <n1>)
    (object <n1> low 3 timetag 1 high 2 medium 4 name g178)
-->
    (object <n1> timetag 1 -))

(sp p194
    (goal <g1> state <s1> operator <o1>)
    (state <s1> dummy-att* true)
    (operator <o1> name assimilate-aircraft object <n1>)
    (object <n1> low 3 timetag 1 high 2 medium 4 name g178)
-->
    (object <n1> high 2 -))

(sp p195
    (goal <g1> state <s1> operator <o1>)
    (state <s1> dummy-att* true)
    (operator <o1> name assimilate-aircraft object <n1>)
    (object <n1> low 3 timetag 1 high 2 medium 4 name g178)
-->
    (object <n1> medium 4 -))
```

Chapter 7

Conclusion

From the above analysis, a complete picture of the Soar agent emerges. The agent is one which learns everything that it sees and can generalize what it has learned to other congruent situations through pattern matching. The current model of the agent is not able to match patterns that it has never seen, though; this is perhaps the single most important shortcoming of the system. Creating an agent that classifies the aircraft it sees against the closest match it has in memory, rather than guessing the classification when it does not know it, should be the highest priority in furthering this research.

Another significant fact that has emerged is that there must exist a training set of optimal size and composition which yields high performance. In other words, the agent can be both over and undertrained as was shown through the comparison of the LTS and STS test results. Identifying this training set would be of great significance.

A possible second step which could be investigated would be the development of multiple versions of the single agent, each which would decide if a certain flying object was friendly, neutral, or hostile based on just one of its characteristics. There would also be some sort of higher order decision making process, such as the voting or managerial processes used in human organizations. This would greatly enhance the system's functionality and would allow more comparisons with organizational hierarchies that exist in society.

In short, while this research leaves several questions unanswered and opens up new subjects which the system should take into account, the creation of Radar-Soar represents a valuable first step. It has been shown that the Soar agent's performance on the radar task is significantly better that a previously built agent's performance. We can attribute this performance to the chunking mechanism by which knowledge is acquired. The next steps that this research should take are as follows, in order of importance:

1. Enhance the model so that the agent classifies aircraft based on the closest match in memory. With an agent trained in this manner, there will be no explicit guessing involved. Learning rate and performance is expected to increase with such an augmented model.

2. Identify a subset of the 19,683 aircraft that yields the most effective learning for its size.

3. Create a different model where the decision making process is hierarchical in fashion; high-level managers make final decisions based on the decisions of their specialized analysts.

Bibliography

[Aky90] A. Akyurek. Means-Ends Planning, Operator Subgoaling, and Operator Valuation: An Example Soar Program. Technical Report RUG-FA-90-3, University of Groningen, August 1990.

[Car91] K. Carley. On the Persistence of Beliefs. working paper, September l991.

[DS90] M. DeJongh and Jr. Smith, J. W. Blood Typing: Functional Modeling Put to the Tests. Laboratory for Artificial Intelligence Research, Ohio State University, November, 1990, Unpublished., 1990.

[HDC84] J.E. Hunter, J.E. Danes, and S.H. Cohen. *Mathematical Models of Attitude Change: Change in Single Attitudes and Cognitive Structure*. Academic Press, New York, New York, 1984.

[HPS89] W. Hsu, M. Prietula, and D.M. Steier. Merl-Soar: Scheduling within a general architecture for intelligence. In *Proceedings of the Third International Conference on Expert Systems and the Leading Edge in Production and Operations Management*, pages 467–481, May 1989.

[JRS91] B.E. John, R.W. Remington, and D.M. Steier. An Analysis of Space Shuttle Countdown Activities: Preliminaries to a Computational Model of the NASA Test Director. Technical Report CMU-CS-91-138, School of Computer Science, Carnegie Mellon University, May 1991.

[Lai84] J.E. Laird. *Universal Subgoaling*. PhD thesis, Department of Computer Science, Carnegie Mellon University, May 1984.

[LC91] Zhiang Lin and K. Carley. Maydays and Murphies. working paper, November l991.

[LLN91] J. F. Lehman, R. L. Lewis, and A. Newell. Integrating knowledge sources in language comprehension. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 461–466, 1991.

[LNR87] J.E. Laird, A. Newell, and P.S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence @*, 33(1):1–64, 1987.

[LRN84] J.E. Laird, P.S. Rosenbloom, and A. Newell. Towards chunking as a general learning mechanism. In *Proceedings of the National Conference on Artificial Intelligence*, pages 188–192, August 1984.

[LYHT91] J.E. Laird, E.S. Yager, M. Hucka, and C.M. Tuck. Robo-Soar: An integration of external interaction, planning, and learning using Soar. In Van de Velde, editor, *Machine Learning for Autonomous Agents*. MIT Press: Bradford Books, Cambridge, Massachusetts, 1991.

[RL86] P. S. Rosenbloom and J. E. Laird. Mapping explanation-based generalization onto Soar. In *Proceedings of the National Conference on Artificial Intelligence*, pages 561–567, August 1986.

[SK91] A. Simon, T.and Newell and D. Klahr. Q-Soar: A computational account of children's learning about number conservation. In *Working Models of Human Perception*. Morgan Kaufman, Los Altos, California, 1991.