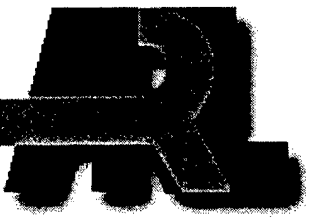
ARMY RESEARCH LABORATORY



Developing an ACT-R Model of Mental Manipulation

Troy D. Kelley
Frank J. Lee
Patrick W. Wiley

ARL-TR-2179 MAY 2000

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5425

ARL-TR-2179 May 2000

Developing an ACT-R Model of Mental Manipulation

Troy D. Kelley
Patrick W. Wiley
Human Research & Engineering Directorate, ARL

Frank J. Lee
Carnegie Mellon University

Abstract

In order to better understand the use of cognitive architectures within the Human Research and Engineering Directorate of the U.S. Army Research Laboratory, and with the intent of supporting the Land Warrior program, members of the Manned Systems Design Methods Team of the Integration Methods Branch developed an adaptive control of thought - rational (ACT-R) (Anderson & Lebiere, 1998) model of mental manipulation. The goal of the model was to reproduce errors made during mental manipulation, which were similar in type and number to those made by actual soldiers who took a paper-and-pencil test of mental manipulation as part of a cognitive assessment battery. The final model was able to replicate errors made by the soldiers in the mental manipulation test; however, further work and data collection are needed to make the model predictive of the types of errors soldiers could make if they were exposed to specific types of mental manipulation problems. Lessons learned from this initial application of ACT-R are also discussed.

TABLE OF CONTENTS

FIGURES

INTENTIONALLY LEFT BLANK

1. Introduction

Modeling all aspects of human behavior, including cognition, has become an important research thrust within the Department of Defense (DoD). This is evidenced by a recent review of the state of the art in DoD's modeling of human performance (Pew & Mavor, 1998). The National Research Council's panel on Modeling Human Behavior and Command Decision Making concluded that "the modeling of cognition and action by individuals and groups is quite possibly the most difficult task humans have yet undertaken. Developments in this area are still in their infancy. Yet, important progress has been and will continue to be made." This technical report represents further progress toward the challenging goal of modeling human cognition.

The data used to develop the cognitive model reported here were collected by members of the Human Research and Engineering Directorate of the U.S. Army Research Laboratory (ARL) as part of a study by Glumm et al. (1998). In that study, 12 male infantry soldiers were required to perform a series of navigational tasks while wearing a helmet-mounted display (HMD) or while using the current navigational equipment (i.e., a map and compass). Following a training period, soldiers were tested before and after the two navigational conditions via the cognitive performance for stress and endurance (CPASE) test. The CPASE battery included tests of verbal memory, logical reasoning, addition, and mental manipulation. Results showed that there was an effect of mental manipulation across the pre- and post-tests in that subjects were improving in the test after both the HMD and the non-HMD conditions. Further analysis indicated that the improvement was because soldiers were actually attempting more problems during the post-test mental manipulation sections. However, their overall error rates remained relatively constant. It was this apparent ability to increase the processing speed of spatial information that originally drew our interest to the data. However, as the model developed, we became more interested in replicating the type and kind of errors produced. The details of how the data from the study were used in the model development are described in the next section.

Adaptive control of thought - rational (ACT-R), a cognitive modeling architecture developed by Anderson and Lebiere (1998), was used for this study. It is freely available for government and academic research from Carnegie Mellon University. It is a symbolic production system architecture, capable of low-level representations of memory structures. ACT-R is implemented in the common list processing (LISP) programming language as a collection of LISP functions and subroutines that can be accessed by the cognitive modeler. For this project, we used Macintosh common LISP (MCL) and ACT-R 4.0 running on a G3 Apple Macintosh computer running system 8.5.1. More detail about the ACT-R architecture is included in later sections.

2. Developing the Data for the Model

2.1 Test Questions

The mental manipulation test used for the HMD study was based on a spatial rotation test developed by Shepard (1978). Note, we have chosen to refer to the test as a test of mental manipulation of spatial images instead of a test of spatial rotation. We make this distinction because the data analysis and our verbal protocol (see Appendix A) indicated that subjects were not primarily using a spatial rotation strategy to solve the test problems but were using a variety of problem-solving strategies.

On the paper-and-pencil multiple choice test, participants were asked to compare an image with three alternatives (see Figure 1). Each test variation included 18 mental manipulation questions, and participants were given 2 minutes to complete as many questions as possible on the test. The test questions were balanced on three variables: rotation (the correct answer was rotated 90°, 180°, or 270° clockwise relative to the referent), the number of darkened blocks that comprised each polygon (7, 9, 11), and connectivity (the referent polygon was comprised of one or two polygons). A number of different test variations were used for the overall study. Our analysis included eight of the test variations of 18 questions, which gave us 144 test questions for the item analysis.

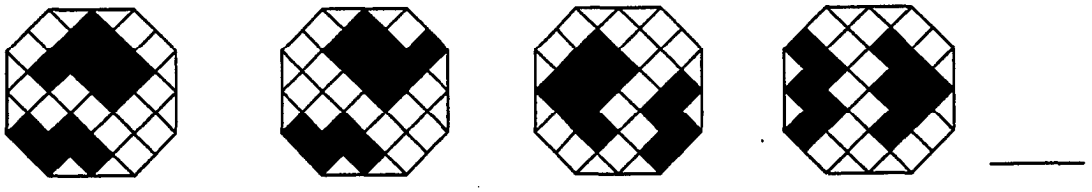


Figure 1. Sample Multiple Choice Test Question of Disconnected Polygons Used for the Study.

2.2 Data Analysis

In order to simulate the types and number of errors soldiers committed while taking the mental manipulation test, the questions were analyzed to investigate what image components made some questions more difficult than others. The results of this error analysis would help determine how to build the cognitive model within ACT-R.

Our initial analysis was of the major variables that comprised each mental rotation question: rotation, connectivity, and the number of blocks in a polygon. The mean error rate for 90°, 180°, and 270° of rotation was $\underline{M}$ = .022, $\underline{M}$ = .026, and $\underline{M}$ = .022, ($\underline{N}$ = 144), respectively. The mean error rate for 7, 9, and 11 blocks in the polygon

was M = .018, M = .026, and M = .027, respectively; and the mean error rate for the connected and disconnected polygons was M = .021 and M = .027, respectively.

We were also interested in examining if subjects who had made a mistake in test questions were more likely to select mirrored distracters over other types of distracters. A chi-square analysis of mirrored versus non-mirrored selection for incorrect answers yielded significant effects X2 (1, N = 445) = 4.55, $p = .033$.

Finally, the compactness score of each referent image was analyzed (the square root of the area over the perimeter (see Podgorny & Shepard, 1982). As expected, the compactness score was inversely related to the number of errors made in questions that contained the referent. Error data were negatively correlated with the compactness scores of the referent r(198) = -.25, $p < .001$.

Our basic finding was that if a mirror image was included as one of the two distracter images, the test question became more difficult, especially if the number of blocks in the polygons increased from 7 to 9 bits (see Figure 2). This then was the basic result that we selected to model.

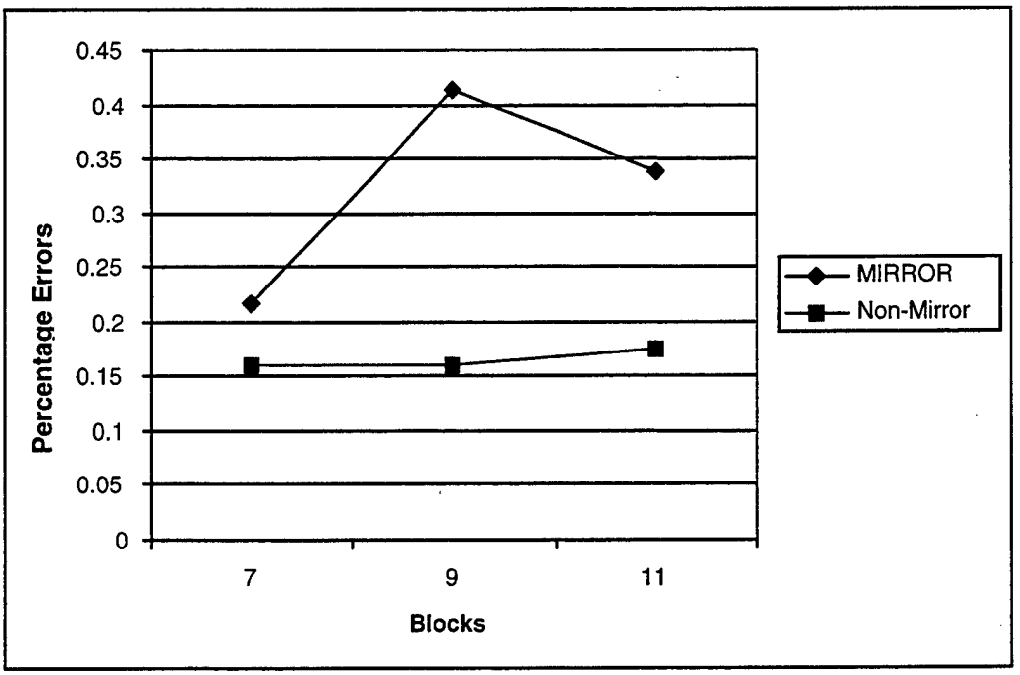


Figure 2. Subject Mirror and Non-Mirror Distracters by Number of Blocks in the Polygon.

2.3 Verbal Protocol

The collection of verbal protocol data is becoming more common in cognitive psychology and it is highly recommended for cognitive modeling of complex tasks since it yields a wealth of information that relates directly to the cognitive processes

of the subject. Given that the general pattern of error rates did not support the fact that subjects were using a mental rotation strategy, verbal protocol data were collected from one subject in order to determine what strategies subjects were using to solve the mental manipulation test questions. The subject was told to talk aloud while taking the test, and the subject was recorded while taking the test. The subject's recording was then transcribed for further analysis. The test variation selected was chosen because it showed the most change in score from the pre-test to the post-test HMD condition for one soldier. The complete protocol is included in Appendix A.

3. ACT-R Architecture

This section gives a brief overview of ACT-R and how it constrained the implementation of this model. However, this overview is not intended to be an extensive review of ACT-R. For more information about the ACT-R architecture, please refer to Anderson and Lebiere (1998).

The ACT-R cognitive modeling environment is comprised of two distinct sections: a declarative memory section and a procedural memory section. The declarative memory section can be thought of as the long-term memory (LTM) storage area of the mind where rules and facts are stored. The procedural memory area is comprised mainly of operations that act upon declarative memory in order to work toward a specific goal, and these operations more or less follow an "if-then" format. Theoretically, the information contained in procedural memory comprises cognitive skills. The basic claim of ACT-R theory is that cognitive skill is composed of "production rules" where "production rules are if-then or condition-action pairs" (Anderson, 1993, page 4).

Declarative memory structures are split into two components: type and slots. The type of structure indicates a relationship among the slots. For example, a typical declarative memory structure might be "type animal slot1 dog slot2 cat." For our model, the declarative memory structures contained information about the images being manipulated in the problem.

ACT-R maintains a goal stack that contains the most recent goal at the top of the stack. The modeler manipulates the goal stack to force the model into doing different actions. For example, we might set the first goal of a model to "solve a problem." Next, the modeler would develop code that would determine "if the problem has not been solved, then set a new goal to look at the first line of the problem." Now, the modeler would have a new goal, "look at the first line of the problem," which is located on top of the previous goal, "solve the problem". The modeler uses the goal stack in this way to direct the actions of the model.

As previously mentioned, procedural memory structures are composed in an "if-then" framework called productions. In general, the "if" part of the production (also called the "left-hand side" of the production) will attempt to match to the current goal. If a production matches the goal on the left-hand side, it will execute the "then" structure, also known as the "right-hand side" of the production.

Production firing is constrained by architectural implementations that control the speed and rate of firing so that each is consistent with current psychological theory. For example, productions that access declarative memory (considered a memory retrieval) are executed at rates that are consistent with current experimental literature about memory access. These rates are controlled by the architecture and, in turn, provide the architecture constraints that make the completed model psychologically plausible.

4. Model Implementation

4.1 Defining Problem Space

Deciding how to implement the data we collected about mental manipulation tasks into an ACT-R model was one of the most difficult parts of this project. The difficulty arose because the constraints imposed by the cognitive architecture will allow certain implementations but not others. Therefore, selecting a representation of the problem space that was consistent with the architectural constraints proved to be a difficult challenge.

One difficulty with developing an ACT-R model is determining what information is represented in declarative memory and how this information becomes part of declarative memory. This has been a theoretical argument with respect to ACT-R for some time (Anderson, 1976). Obviously, a human's perceptual system processes incoming information which then may eventually become encoded into a declarative memory store. To address this shortfall, a perceptual component for ACT-R is currently being developed (Byrne & Anderson, 1998). It was not used for the development of this model since is it primarily designed for interaction with a computer interface and is still quite new in its development. Therefore, assumptions about the declarative memory store had to be made in order to develop the model.

The perceptual implementation that was used for this model was to separate the "external world" from the "internal world" of cognition by using separate LISP function calls to simulate the external world. The LISP commands modified memory elements after a specified production fired. Therefore, the LISP commands represented the "external world," which modified declarative memory structures, depending on which productions previously had fired.

4.2 Modeling Errors

Why do errors occur in mental manipulation tasks? This was the basic problem for the ACT-R model. Our data indicated that people have significant amounts of trouble with mirrored objects and this problem is compounded if the images are also complex (i.e., as measured by the number of polygon blocks of the polygons, connectivity, and compactness).

ACT-R currently has two ways of implementing errors: commission and omission. Errors of commission occur when an incorrect memory (declarative memory element) is retrieved instead of a correct memory. The example given in the ACT-R tutorial is a child attempting to count numbers and confusing the numbers 4 and 5, so that 5 is retrieved in place of 4. This is similar to the psychological construct of interference, in which memory elements become confused with each other because of their similarities. Errors of omission are memory failures when the correct memory is never retrieved. This is similar to Norman's (1988) concept of loss-of-activation errors when one simply forgets what to do. Errors of omission are controlled by activation levels in ACT-R. Each memory element has a certain level of activation. If that memory falls below a certain retrieval threshold because of decay, the memory will not be retrieved.

The data from our analysis indicated that more complex images resulted in more errors. Further, when an error was made, a mirror image distracter was more likely to be chosen than a non-mirrored distracter. It was decided to handle errors from mirrored images as errors of commission, which can be implemented via ACT-R's partial matching functions. Errors with more complex images were assumed to be caused by errors of omission, since the more complex the image, the more one has to remember during the rotation process.

4.2.1 Strategies

The verbal protocol data indicated that subjects were not always using rotation as a means for solving the problems; instead, they frequently used rotation as a last resort for solving the problem. Subjects were using a variety of strategies including shape comparisons, distance from the edge (images were inside a grid that allowed comparisons to the edge of the grid), and alignment comparisons. Sometimes, a problem-solving strategy was applied, and if no solution was found, the same strategy might have been re-applied or a different strategy was applied. Also, the selection of strategy was stimulus specific, so depending on the stimulus, a certain strategy may or may not have been selected.

Strategy selection proved to be one of the more difficult constructs to model within ACT-R. The final solution was to model strategies, based on the number of manipulations required to match the referent to the correct answer. The higher the number of manipulations, the more difficult the strategy. Strategies were essentially broken into single units, with each unit or manipulation being the one unit. The only difference in strategies was the number of units or manipulations for each strategy.

This made programming the model much simpler, and it also captured the essential elements of the different strategies, that is, that some strategies are more difficult than others.

4.3 Defining the Model

Initially, multiple small models were built with the hope of combining each small model into a larger single model. These separate small models included modeling perception separately in LISP, modeling errors of commission via partial matching, modeling errors of omission via decay functions, and modeling strategy selection. However, the process of building smaller models with the hope of combining them proved to be unsuccessful. It was profitable in that it provided small demonstrations of how the architecture worked, but in the long run, when the small models were combined, the interactions between the once separated ACT-R routines produced unforeseen outcomes. The ACT-R architecture is so dynamic and interconnected that adding modules of code can change the whole outcome of the model.

Another problem that proved difficult to address after the smaller models were combined into a larger model was how to represent each image of each question. Recall that each question contained a referent image that the person was comparing with three possible answering images (see Figure 1). A possible representation was one declarative memory store (chunk) for each image, thus yielding four chunks per question and 72 chunks for the entire test. However, from the verbal protocol and existing literature (Biederman, 1987), it was clear that subjects had a tendency to break images into pieces and manipulate certain portions of the image, depending on the strategy. This fact led to another possible representation in ACT-R—multiple chunks for each image representation, with some of these chunks being susceptible to decay, depending on the strategy being employed. Further complicating representation of the items in each question was the ACT-R theory of spreading activation (also known as associative priming) in which exposure to objects primes a subject to other similar objects. In our model, this meant that there was a significant amount of spreading activation between image chunks and across image chunks of different questions. In other words, the model was confusing previous questions with the question it was attempting to solve. This has some psychological validity, but our human subjects were not nearly as susceptible to this problem as the model was. Our smaller models that operated on one question did fine, but once multiple questions were added, the model had too much spreading activation across problems.

4.4 The Final Model

The final model was eventually quite small. (See Appendix B for the more complex commented model and Appendix C for the final smaller model.) Many ACT-R models become quite small (see Lebiere, 1997), so having a rather small model is not unprecedented.

An important distinction needs to be made concerning the final model. The final model can be viewed as a model of the process of image transformation and not a model of the representation of images. In other words, the external LISP calls were unable to represent the outside world. Instead, the process was defined by how the model varied the number of manipulations in order to represent the different strategies. However, representation of the outside world could be more fully developed within the formula such as the one shown in Figure 3. It was developed as a parallel effort but was not fully implemented into the final model. Once the representation of each image is completely detailed within the model, then it could become more predictive.

$$ImageSimilarity = R + B(Sw + Ew)$$

in which

R = Rotation (90, 180, 270)
B = Bits (7, 9, 11)
S = Shape (percent overlap)
E = Edge (1+|Q-A|)
Q = question
A = answer
W = weights of regression equation

Figure 3. Image Similarity Formula (by Troy Kelley and Patrick Wiley).

The logic of the final model was very simple. We manipulated the partial matching command to represent errors of commission, but errors of omission were not explicitly represented. The partial matching command used to represent errors of commission depended on both the complexity of the image and whether the question included a mirror image as a distracter. Partial matching in ACT-R is controlled by the "*set similarities*" command. Essentially, the modeler sets the similarities of two chunks in memory; the greater the similarities, the more likely the two chunks are to be confused with each other. The final result was one *set similarities* command that incorporated whether the distracter was a mirror image and whether the image was more or less complex. This gave variables that depended on the type and complexity of each object.

The model was able to reproduce the general trends of scores obtained by real soldiers (see Figure 4). The model captures the changes in error rates from the mirrored distracters as well as the more complex images. The correlation of the model with the obtained error data at the different number of bits in the polygon (7, 9, 11) was r = .96 for the mirror polygons and r = .98 for the non-mirror polygons.

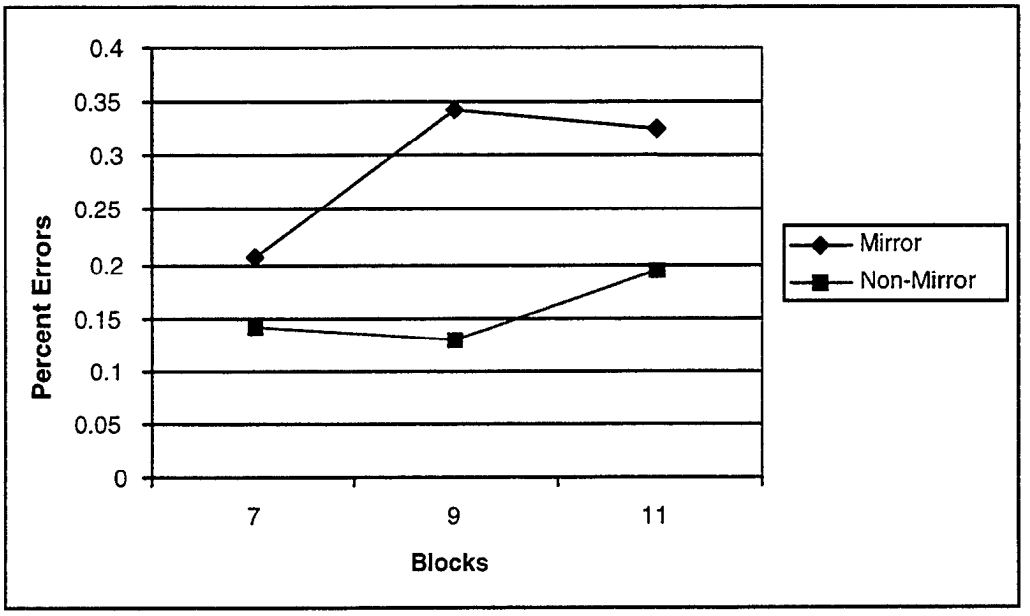


Figure 4. Final Model Output of Percentage of Errors for Mirror and Different Images.

5. Recommendations

The final model was able to map the error responses of subjects who had previously taken the mental manipulation test. This is what many ACT-R models attempt to do: match the data taken from real subjects. We accomplished this with our ACT-R model.

During the development of this model, we learned many things about the strengths and weaknesses of ACT-R and how to proceed in the development of a cognitive model. Some of our suggestions are as follow.

1. ACT-R is best designed for relatively simple memory tasks, and this seems to be its strength. More complex tasks that may involve different components of the human perceptual system may exceed ACT-R's current capability. The perceptual component of ACT-R is being developed and is called ACT-R perceptual/motor (P/M) (Anderson & Lebiere, 1998). It is capable of interacting with a LISP-generated computer interface through production-driven modules of vision, motor, and auditory behaviors. As with the cognitive components of ACT-R, the perceptual modules are constrained by physiological and psychological data and thus interact with the simulated interface in a human-like manner. This is clearly an important advancement in the ACT-R architecture and will make ACT-R more capable of addressing human-computer interaction design issues in the future.

2. Do not attempt to model small units of a more complex model and then expect the smaller units of code to work when they are all assembled. Modeling the smaller units is appropriate for learning the architecture, but ACT-R is a very dynamic and intertwined environment, and assembling smaller bits of code can have unforeseen consequences for the final model.

3. If at all possible, collect verbal protocol data. These data will provide insights to the cognitive actions of subjects, which are not immediately apparent in raw data.

Finally, the real test of a computational cognitive model is to be predictive. One further step needs to be taken in order to make this model of mental manipulation predictive. For the current model, the assignment of the weights and values for the *set similarities* command was chosen somewhat arbitrarily in order to find the best match to the existing data. However, no formula was developed to assign the values for the *set similarities* command. A formula could be incorporated into the *set similarities* command, which would allow predictions of error rates for different types of images.

A tentative formula was proposed; however, more data need to be collected in order to increase the sample size to allow for the development of appropriate multiple regression equations. This formula (see Figure 3) would be a calculation used to set the similarities of each image. The formula would take into consideration the rotation of the image and the number of blocks in the image, which are expressed in the formula "between-question" variables. The "within-question" constants of image shape and its relationship to the edge of the outside polygon are also included.

The DoD research community must find ways to bridge the gap between the psychological theory of the laboratory and behavioral applicability of the battlefield. With the ever-increasing complexity of the modern battlefield, computational cognitive modeling is clearly an important tool for the DoD community. While the field of cognitive modeling may appear to be still in its infancy, much work has already been done to ensure that the foundations of cognitive architectures, such as ACT-R, are well grounded in psychological theory and laboratory experimentation. This grounding in psychological theory has allowed cognitive architectures to thrive in certain situations, primarily the laboratory, allowing cognitive psychologists to gain new insights into the mechanisms of cognitive processes. However, more work must be done to allow these architectures to thrive in modern, real-world situations. Future development of the perceptual-motor components of ACT-R P/M will certainly allow it to have a greater impact on dynamic, cognitive tasks that are becoming more ubiquitous on the modern battlefield. However, current implementations of ACT-R have much to offer and the DoD researcher must select appropriate questions for ACT-R to answer.

References

Anderson, J.R., & Lebiere, C. (1998). The atomic components of thought. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J.R. (1976). Language, memory, and thought. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J.R. (1993). Rules of the mind. Lawrence Erlbaum Associates.

Biederman, I. (1987). Recognition by components: A theory of human image understanding. Psychological Review, 97(2), 115-147.

Byrne, M.D., & Anderson, J.R. (1998). Perception and Action. In J. R. Anderson & C. Lebiere (Eds.) Atomic components of thought, (pp. 167-200). Mahwah, NJ: Lawrecne Erlbaum.

Glumm, M.M., Marshak, W.P., Branscome, T.A., McWesler, M., Patton, D.J., & Mullins, L.L. (1998). A comparison of soldier performance using current land navigation equipment with information integrated on a helmet-mounted display (ARL-TR-1604). Aberdeen Proving Ground, MD: U.S. Army Research Laboratory.

Lebiere, C. (1997). ACT-R tutorial. http://128.2.248.57 /inter/ACT-R-tutorial.html.

Norman, D.A. (1988). The design of everyday things. New York: Doubleday.

Pew, W.R., & Mavor, A.S. (Eds.) (1998). Modeling human and organizational behavior. Washington DC: National Academy Press.

Podgorny, P., & Shepard, R.N. (1982). Distribution of visual attention over space. Journal of Experimental Psychology: Human Perception and Performance, 9, 380-393.

Shepard, R.N. (1978). The mental image. American Psychologist, 33, 125-137.

INTENTIONALLY LEFT BLANK

APPENDIX A

VERBAL PROTOCOL

INTENTIONALLY LEFT BLANK

VERBAL PROTOCOL

Test V5a
(does examples first)
When I look at no. 1, I see that there's 2 shapes, a square and a T.
So I'm basically looking for the same type of orientation.
As I look across here at a, b, and c, I see that the square is 1 box off from the end all my other ones are 1 box off from the end.
I look at the T, and I see that it's butted up against the side of the whole pattern here.
So I guess, basically, I just gotta rotate this in my mind.
As I turn it around in my mind I see that b is the appropriate answer, because of the tip of the T, when I rotate that around, is to the left of the square.
So my answer would be B.

For no.
2, it's just one basic shape -- it's an L.
So I wanna see where that is with respect to the block.
And it's...
both sides...
the long part of it...
There's a box...
it's one box away from the side, and then on the other end of one part it's one box away from the other side.
So, I wanna find that out in my assessment of the other 3.
It looks like that's the case for all 3 of 'em, so my next thought is, 'Where is the end of the short length at?' It's up against the last triangle before the side
And I see that's also the case for all 3 of these
So my next thing to do is to look at which direction the long..
how the long part and the short part are connected
Is it pointing..
if it were standing up, would it be a...? Looking at it now it makes an L shape, if I were to look at it straight
So I wanna see if I can rotate this, which ones would still maintain that L shape
And as I look around here, I see that both..
the A does maintain the L shape, but B does not
And neither does C
Both of those have more of a J shape, so my answer would be A

For #3, again we have 2 shapes
And my first thought is to see where they are with respect to each other, and also to see where they are with respect to the pattern
First of all I see that they are 1 block apart from each other
I also see that the Z-shaped pattern is connected to the side of the pattern or the grid
And right away, looking at A, I can eliminate A because that's 2 blocks away
And I can eliminate C because they're also 2 blocks away from each other
So..
and just double- checking B which is gonna be my answer, I see that there's..
it's a..
the Z part is connected to the side
The other part is 1 block away from the side, which is the same for the example
And if I would rotate that in my mind, I would see that's what I would get..
that same pattern.
So my answer would be B.

Again in #4, we have a L shape.

And it's the same as #2, as far as it's orientation.
I see, just a quick look, I see that if I were to rotate any of these patterns, I see that they're also all being an L shape vs.
a J shape.
So I can't eliminate any that way, but looking at where this L shape is with respect to the grid...
It's 1 block away...
the long part is 1 block away from both sides.
So when I look at A, I can eliminate A because it's up against the side of the grid on one side, and 2 blocks away on the other side.
And I can also eliminate B for the same reason, because it's 2 blocks away on one side and up against the side of the grid on the other.
So my answer would be C.

(start of test)
If I look at the next one here.
It's a multiple pattern, here.
And so I'm pretty much gonna look to see if I can eliminate anything that doesn't look like this.
Right away I see that A doesn't have the same type of pattern, because I see that looking at the left side of this, that there's a hump that's more in the center of the whole piece.
And A has the hump that's more to the, in this case, the right of the piece...
so I'm looking at this orientation, okay.
And then my next thing to notice is that there is a...
the longer side of this has one side that has 2 blocks wide and the other side is 1 block wide.
And the hump is closer to the 1 block side.
So if I look for the next one down here, I see that B has that same type of shape, where a hump is closer to the short...
to the 1 block side.
And so then I'm gonna consider B, but then I'm gonna go look at C.
And I see that C is right in the middle.
It is closer to the 2 block side than the 1 block side.
So, based on that, I'm gonna choose B.

Looking at the next example.
It's 2 pieces.
I see that one is just 2 blocks.
And it's connected to the one, well I guess you could call it a corner of the pattern.
So right away I know that if anything's not in that corner then it's not correct for the matchup.
It's also...
I'm looking at the other piece, which is almost a rectangle with 3 blocks wide and 2 blocks high.
But there's an additional block on the left end.
So I'm looking across here to see...
I'm also seeing that this short...
the one piece which is 2 blocks is parallel to the shorter side of this rectangle, because the other side has an additional block.
And it's 1 block away.
So with my respect to each other, I know that they're 1 block away.
I can look across here real quickly and see that C is not the answer because the two are 2 blocks away.
So I can eliminate that.
I can also look at B and see that the 2 are not parallel in the same respect.
B is parallel on another part/side of the rectangle, not the side that I see in the example, here.
So I'm just gonna go ahead and choose A, which seems to be the only obvious answer.
And if I would look at that, I would see that, just rotating that in my head, that that does match up.

Going on to the next example...

It's one shape, kind of an odd shape, that looks like a square part for a solid part, and it has one block that sticks out to the right and one that sticks up to the top, if I would kind of look at it from the side, here.
So I'm automatically looking...
I also notice that the part that sticks to the top is 2 blocks away from the other edge and that the other stuff that sticks up from the square is 1 triangle away from the edge.
And looking across, here, I can eliminate B, because that long part that sticks out from the square is not 2 blocks away from the edge -- it is right at the edge.
So B is not the correct answer.
If I look at A, real quickly, it could be possibly the right answer.
C could possibly be the right answer.
Now I have to look at the orientation where the...
just looking at where the long part of the stub...
the long stub off of the square and the short stub off of the square are with respect to each other.
And so I see that the short stub is to the right of the long stub.
That, itself, if you look at it, makes an L shape.
So if I rotate these other 2 in my head I see that A does not make that L shape.
It actually makes a J shape.
So that one is not the right answer.
If I would look at C, I could see that rotates to where it makes the same orientation.
There's an L-shape with respect to the square.
So my answer would be C.

This next one is 2 shapes.
makes a U.
see another shape that makes a small rectangle of 2 long and 1 high.
And they're 1 block apart from each other.
The shorter piece is parallel to one of the sides of the U.
If I look across, I see that right away that A is not the answer.
Neither is B, because the short piece is not parallel to the sides of the U.
It is actually parallel to the tops of the U.
And then I can just see in my mind when I rotate the example that C does come out to be the correct answer.
Looking at the next example.
It either looks like a backwards F...
or if I were to rotate that in my mind it looks like the number, 4.
So, right away I eliminate A because rotate in my mind it doesn't look like a 4.
When I look at the last one it has the same shape as the first one.
And as I look closer...
even though in my mind I had thought that looked like a 4...
after I rotate that around, it turns out that that's a backwards 4...
which I had to kind of teach myself, looking back at one of the examples or looking at one of the solutions...
and seeing that my first assessment was not 4 but a backwards 4.
And as I look at the letter B, example...
with the example that I'm looking on to match, I see that they rotate correctly based on...
first, just process of elimination, and second, because of where it is with respect to the grid and its actual shape.
So B is my answer.

The last one on this page is 2 shapes.
Again it's a square with one stub sticking out of it.
And then also another piece that is 1 block high and 2 blocks wide.
With respect to each other...
Well, first of all, with respect to the grid they're both against the sides of the grid on a corner...
and one is in the top left corner and one is in the bottom left corner.

The short piece is parallel to the opposite side of the other one which would be square with the stub on top of it.
And between the stub and the other piece is 2 blocks away.
So, when I look at A, I see that, first of all, they're...
the short piece is not on the opposite side of the stub.
It's actually on the same side and it's only 1 block away.
So I can eliminate A.
If I go across to B...
that's a possible solution.
But I look at C, which is quite obviously not the same thing, because the short piece, the smaller piece, is not parallel to the side that has the stub on it.
And it also is...
well, that's pretty much it.
It's 2 blocks away, but it's 2 blocks away in a different way and it's not parallel.
So going back to B...
if I rotate that 90 degrees, counterclockwise, I can see in my head that that is the actual answer.
So B is my answer.

Looking at the next example, the first one on this page...
I see a shape that to the right of it makes a W, but then the left part has 2 extra blocks that go back downward in a wavy way.
So, right way, I'm just gonna see if any of these don't look like the exact same pattern.
And from just what I see in this first one, I don't see as many peaks and valleys for B or C.
Regardless of where they are - orientation.
They just don't look to be the same shape.
And I look at A, and I see that it does have the same...
if I would rotate that in my mind, 90 degrees, to...
clockwise...
then I get the same type of peaks and valleys, and the same depth for the one side.
So A would be my answer.

The next one is...
the second one on this page...
is kind of an odd shape.
The first thing I would consider here is whether I can recognize the shape.
I see that the...
it almost looks like a, kind of a gangster machine gun, here.
I want to see if I can find anything that looks similar to that, without going into great detail.
What would look like the handle part, which sticks out to the right has a stub that would be of...
now facing downward.
That is on the top part.
And then there's a stub, which would be right now, facing upward that's on the bottom part of that rectangle that connects to the (what I would consider) the handle.
So, right away, I'm looking for something that has this exact same shape.
I look across here, and I see that C does not coincide because the handle part doesn't...
looks like it kinda sticks up through the thicker part.
There's a part of a stub but it comes out of the top which is not the case in the example.
So C is eliminated.
The next 2 is to compare A and B.
And I see that I've got a...
Again I look at my stubs that are sticking out on both ends of the thicker part, not the part that I consider the handle.
And again I have a stub that's sticking out of the top, that's facing down now, and one that's sticking out on the bottom, on the other side.
So I'm going to look at A and I see that there's stubs sticking out on what I would consider the bottom part closest to the handle.

So there's a stub on the top of this picture, on A, on bottom...
and then the stub on the other end is sticking out.
So they're both on the bottom with respect to what I would consider the handle.
So that one doesn't look right, but just to confirm this would be...
Let's see, B is my answer.
B does have one that's sticking out.
If you're looking at, with respect to the handle...
I would say that...
have a stub sticking out...
on the top part of that, upward...
and one that's sticking out on the bottom part, downward.
If I just mentally rotate that in my mind I see that that does match up.
So B would be my answer.

The next one is kind of an odd shape.
Again it's 2 pieces.
First I notice that they are 1 block apart from each other.
It's somewhat...
the big piece is somewhat symmetric in that...
if I would say that the center part is a rectangle that's a shape of 3 by 2...
and then there's a block on each side, right & left of that.
And then there's one block that's up and one block that's down, but they are not on the same side.
The one on the top is to the right; the one on the bottom is to the left.
So, I notice that the other little block that's separate is one block away from one of these ones, these other blocks that're on the side.
When I look at my shapes I first of all, see that C has the same shape, looks like it has the shape as the big block part, big section, but the small block would not be in the same place with respect to the big block is, actually...
off by almost 90 degrees.
So I can eliminate C just because it doesn't have the same orientation with respect to each other.
The next thing I have to do is to look at A and B.
I see that B does not have the same shape for the big block section as it does in the example, because looking at this again from just the rectangle part beens (means?) the core part, and then seeing extra blocks put around that I see that the 2 that were on the side of the wide part of this rectangle, the 3 block side are not in the same location across from each other.
So, even though that's only 1 block apart from the smaller block, they're not in the same location.
So I eliminate B from that.
And just to confirm A...
I see that, first of all, that the 2 blocks on the side are at the right place on the longer part of the rectangle and the 2 blocks on the end of this rectangle are across from each other in opposite locations.
And they're...
the ones inside -- one of those blocks is one block away from the smaller block.
So that's okay.
If I would rotate this in my mind, counterclockwise, 90 degrees, I see that they match up.
So A would be my choice.

The next one is again 2 shapes.
One is a kind of a backwards Z.
It's on the bottom part of the grid.
And then the other one is a hat shape.
I see that the closest part of the Z to the hat is one block on one side and 2 blocks on the other side.
The first thing I need to do is see if just these blocks are in the same...
I'm going to either try to see if they're orientated with each other or see if they're orientated on the grid at the right place.
This one looks a little bit trickier.
It looks like in the first 2 examples, the hat has been rotated 90 degrees clockwise and...

on the last one, it's been rotated 180 degrees clockwise, or 180 counterclockwise -- whichever way you want to look at it.
The next thing to do is see where...
which part of the Z is...
which side of the hat is the Z closest to.
If I rotate that part in my mind...
if I rotate the whole thing in my mind...
90 degrees, to the clockwise...
I see that A would end up in the same location for both of these...
just based on, just spinning this round in my head.
If I look at B, the orientation of the 2 is not the same with respect to each other.
As a matter of fact, the Z shape is not the same type of shape.
It's reverse of the one that I see in the example.
So I'm gonna eliminate that one.
If I rotate the Z in my head, for the last part...
I also see that's not the right answer, because the Z is a reverse of what it is in the example.
So my answer, based on that, would be A.

The next example is an odd shape.
Right away, just glancing across, here, I see that there's a couple different shapes that just don't seem to have the same...
shape in particular, let alone what their orientation is.
A does not look anything like the example.
B doesn't seem to have the same depth as far as peaks and valleys.
So I'm just going to quickly eliminate A and B And then I can just kinda confirm in my head, rotating C 180 degrees to see that, yes it does, fall in the same...
does have the same shape.
Its one side is right-connected to the side of the grid and then the other one is a triangle away from the edge of the grid on the other side.
So C is my answer.

The last one on this page is another odd shape.
I would look at it as...
the center...
the base part being a rectangle being 3 long and 2 deep (2 wide, whatever way)...
and then there's 3 other squares connected around those edges.
I see a similar ones for all the other ones in that the center part is 3 long and 2 deep.
But it's now just a matter of where do those 3 squares sit around that rectangle.
It's not quite obvious what is the best way to do this.
One thing I do see is that the example...
nowhere is there an area where part of this piece is 2 blocks away from the edge.
There's either stubs sticking out or the rectangle side is only 1 or less blocks away.
If I look at C, I see that that's eliminated, because there's a part of this piece that's actually 2 blocks...
the whole side of it is 2 blocks away from the edge of the grid.
So I eliminate that.
So I'm down now to A and B.
I see that A has a one side that has nothing on top of it.
It's just one long side that is 4 long.
That actually happens to be the top left side of this shape.
When I look at my shape over here, I see nowhere where there is a side that has no blocks on it that is 4 long.
So I eliminate that one.
And if I just rotate B in my head (or the example 2B), I see -- it's a little bit more difficult -- but can I see that by turning it 90 degrees counterclockwise that I get the same shape and orientation.
So my answer would be b.

The next page...
the first one on here is another odd shape.
And again I revert to going to the base part where I see a rectangle with little blocks connected to it.
I see a base of the blocks of 3 long and 2 wide.
And again, there are 3 blocks that are connected, 3 squares that are connected to this base rectangle.
In my mind, just quickly glancing, it looks like A is a possibility, if I would rotate this 90 degrees...
so this may be a quick one...
I just happen to notice...
would be the best case.
But I'll go ahead and see if I can eliminate B and C.
If I look at B, I can kinda see, right off the bat, that that's not the same shape...
even if I rotate that around in my mind, it just doesn't seem to have the same shape, because the rectangle has, in the example, has 3 blocks that are directly off of it.
If I look at B, I see there's 1 box that's directly off of the rectangle, but then there's another block that directly off of the block that was directly off of the rectangle.
So that one, in my mind, is eliminated.
The last one that I see here is C.
And I look at the orientation of...
what in my mind looks like the top...
Going back to the example I see that the top left kinda makes a backwards L shape (if I look at the edges of that).
So I'm gonna kinda use that as my thought as I rotate this around to see if that would make an L shape.
As I do that, I see now that C is still a possibility.
So the next thing I need to do is see is where is this orientated in the grid.
I see that that L shape is, in the example, is right up against the side, or one of the corners, of the grid.
If you're looking at this as a backward L shape, it's right up against the bottom of the L.
And the top corner is connected to the top of the grid.
So I have one part that is connected to the side, and one part that's just the corners connected.
If I rotate that around in my head, I see that that L shape is not like that, because the top part of the L shape you (?) C is not touching anything.
It's actually 2 blocks aaway from any side, but even though the bottom part of that L shape is connected to the side.
So I've eliminated C by that.
My answer for this is what I had initially thought, which would be A.

Going on the next one...
It looks like we have a couple simple shapes, here.
We have a square and we have a...
L shape or a V, whatever way you want to look at it, here.
The best thing to now determine is...
where they are with respect to each other, and where they are with respect to the grid.
The square on one side is 1 block away from the top left corner.
It is 2 blocks away on the other 2 sides.
And then the final side that is closest to this L shape is 1 block away.
The L shape also has 2 blocks...
one of the side that is 2 blocks long is facing the box.
So I want to make sure that I have something that is parallel with it.
That both shapes face each other, with 2 blocks of side as far as length.
Just to explain it a little bit better, I can look at A and see that regardless of how these are orientated, I can already eliminate it because with respect to each other, the L shape does not have a side parallel to the side...
of the square that is facing it that is 2 blocks...
...looking at A on this second example on page 3...
which is B5ap3...
the second one with the square and the L...
I've eliminated A because it does not have the same side facing the square...

the L shape does not have the same side facing the square.
If I look quickly at B and C...
That side, the long side of the L is facing the square for both parts.
So now I have to look at the orientation, where they are with respect to each other; where they are with respect to the grid.
Quickly I see that the L shape is up against the bottom right corner of the grid, one of the legs of it, one of the sides...
well, not the sides, but it's the top part of one of the sides is up against that.
Right away, I see that in B, this L shape has nothing touching the sides.
So I can eliminate that.
So I've eliminated A and B, so C would be my answer.
And just to confirm that, I can rotate that in my mind 180 degrees, and...
yes it does fall in the same place.
The third one is kindof a funky design.
It's kinda hard to see right off the bat.
I have to look to see how many...
First, I want to see where it is with respect to the grid.
And I see that there is only one part that is touching the end of the grid.
So I see a block, one block that's touching the right top corner.
Looking at these possible solutions, I see that I can eliminate B, because B has two sides that are touching...
2 parts of the shape that are touching the sides of the grid.
So I'm down to A and C.
And now I'm just looking at shape, in general.
If I rotate this in my mind, I see that the shape that is closest to the example is C.
I can look at that based on just picture it looks like.
Doesn't look like anything that I use as an example I see in A that the top right side makes a long L, laying down on its side.
When I rotate the example, I see nothing that looks like that, that has a long L.
For no other reason, without going into a great study, here, I just would choose C based on the way that it's rotated and how it looks in my mind to be the same thing.
So C is my answer.

The 4th one on this page is, again, 2 shapes.
We have, basically, a big L and a small L.
The bottom corner of the big L touches the side of the grid.
So I know, right away, that only the ones that have the big L, bottom corner touching the side of a grid at any orientation, is a possibility.
So, right away, I eliminate A, because when you look at that L...
that bottom corner, which is not rotated, it's where the short side long side come together at the back, is not touching the side of the grid.
So that's not a possibility.
I look at B and I see that that is a possibility.
Quickly, I look at C, and I see that C is not a possibility.
Because that bottom, that part of the L shape is not touching the side of the grid.
So going back to B, I see, just quickly in my mind, that if I would rotate those around 90 degrees, clockwise, that that would be my solution.
So my choice is B.

The next one on this page, second to the last, is the 2 pieces.
It's a long bar, 4 wide / 1 deep.
It's corners touch the top and left side of the grid.
The other one is kinda an odd shape.
It's basically the base of a square with 3 blocks sticking up from it.
With respect to each other, the part of this base, the square part, has a side that is parallel to this other object.

And that side is 2 long and it...
with respect to the bar it is central.
The other bar is 4 long.
One part from each other, with respect to each other...
And as far as orientation with each other, this odd object is centered with the bar from (It's hard to describe which orientation, or which direction...) But, the...
Again the bar has 4 boxes.
And the parallel part of the other shape is at the 2nd and 3rd box location.
So right away, I look at A and I see that the parts that are parallel, the sides that are parallel, are not (for the odd shape) is not 2 boxes wide / 2 boxes long.
It's only the side that is parallel with the long bar is 1 box.
So I'm gonna eliminate that one.
The next 2, I do see that there is a possibility.
The side of this odd shape that is parallel to the bar is 2 boxes long.
And it is central with the bar with respect to the example.
I see that the same thing is true for C.
So now I'm down to how this...
I also see that this bar is always connected to the sides.
One corner is connected to one side; one corner is connected to the other side.
If I was looking at only the bar, in both cases...
they would be true examples: one's rotateed 90 degrees clockwise; the other one is rotated 180 degrees clockwise.
So the question to ask is, 'which one of these odd shapes is the same as the example shape if it was rotated?' The first thing in my mind is, is that I see that the part of this odd shape that has this block that is touching the bottom right corner is on the bottom right side of this shape with respect to itself.
So I'm gonna try and rotate these other 2 shapes, the odd shape in B and the odd shape in C, and see which one of these would rotate back to the example, and have the...
that square be on the same side.
So if I look at B and I would rotate B 90 degrees counterclockwise, (I see that the square is not on the right side, but is on the left side touching that corner of the grid.
So it is not the same, it is symmetrically not the same shape.
If I look at C and I rotate that 180 degrees...
Or if I turn my paper...
I can see the same shape.
And have that be the same as the example.
So C is my answer.

The final one on this page is 2 shapes.
It's a...
In my mind that kinda looks like a long, thin Z.
There's also another shape that is kind of a small L.
That small L, the outer corner...
the 2, the lines meet for the small L...
is touched to the side, its one corner is touching the side, the right side.
So right away, I'm gonna go across here and see which ones are...
where that small L is, that back corner is touching the side.
I see in A that it is doing that.
When I look at B, I see that that back corner is not touching the side.
It's actually facing into the middle, so I can eliminate that one.
And I look at C.
It is touching the side.
But it's not centralized like the first one is.
The first one is touching just the...
It's actually at...
if you would...
It's not touching a corner part of the grid.

It's just touching the center of the right side.
I see in this last example, that little L is not touching the center of the top side, if it was rotated.
So I eliminate C.
Just to confirm that I see that I can rotate this in my mind 90 degrees and see that all these shapes match up as they should with that orientation.
So A is my answer.

APPENDIX B

COMPLEX MODEL CODE

INTENTIONALLY LEFT BLANK

COMPLEX MODEL CODE

```
;;; Code for mental rotation model by Troy Kelley, 1999 for ACT-R 4.0
;;; Code was eventually abandoned in favor of a more simple version


;;; The logic of the model is as follows: The person looks at the referent, which is
the first
;;; image which needs to be matched to some other image.  Then the person looks at the
other
;;; images (called non-referents) and selects one of the non-referents as a first
chioce.  This
;;; was something I could see done in the verbal protocols.  Once a "first choice" has
been selected
;;; as a "strategy", the person will attempt "manipulations" in order to make a match.
At this point
;;; I don't care what the manipulations are, like compare to edge, or rotate, I just
care how
;;; hard the manipulations are, or how many.  The harder the manipulation, or the more
of them
;;; the more likely there will be
;;; decay of the referent during the manipulation.  Right now, however, if there is
enough
;;; decay and person has forgotten the referent, the person will just go back and
study the
;;; referent more until activation levels are higher.  So in other words, forgetting
the
;;; referent doesn't produce an error.  If the referent is remembered clearly enough
after
;;; so many manipulations, the person will attempt a match.  This is where I use
partial matching
;;; to produce an error.  This is where I also need some type of mathematical
representation
;;; of the similarity between two images, and the representation needs to reflect how
people
;;; represent the similarity of items in there minds.

;;; As of right now I am representing each image (polygon) in three parts, center,
middle, and outside.  I don't
;;; have to continue with this idea, but right now this is how I have it.  I was
conceptualizing that people
;;; would have different levels of similarity for the different parts of the two
images.  So, for example,
;;; the middle of two images might be very similar, but the outside of the images
(polygons) were not
;;; at all similar, so I would have an .80 for the center, and a .40 for the outside.


;;; Most of the first part of the code are LISP functions which practice the referents
or non-referents.  Also, I am
;;; attempting to represent the outside world in LISP functions, and I use mod-chunk
calls to change
;;; chunks depending on which problem they are working on.  This reduces the amount of
interference I get
;;; across chunks.

;;; This call practices the referent x number of times

(defun practicereferent (x)
    (cond ((= x 1)
      (dotimes (i 1)
      (rehearse-chunk (referent1))))))

;;; I also have to practice my strategy or it will be forgotten

(defun practiceStrategy (n)
     (dotimes (x n)
     (rehearse-chunk (strategy1))
```

```
     (rehearse-chunk (strategy1))
     (rehearse-chunk (strategy1))
     (rehearse-chunk (strategy1))
     (rehearse-chunk (strategy1))
     (rehearse-chunk (strategy1))
     (rehearse-chunk (strategy1))
     (rehearse-chunk (strategy1))
     (rehearse-chunk (strategy1))))

(defun nonreferentbase (problem)
     (cond ((= problem 1)
         (set-base-levels (nonreferent1 6 0))))
     (cond ((= problem 2)
         (set-base-levels (nonreferent1 12 0))))
    (cond ((= problem 3)
         (set-base-levels (nonreferent1 18 0))))
    (cond ((= problem 4)
         (set-base-levels (nonreferent1 24 0))))
    (cond ((= problem 5)
         (set-base-levels (nonreferent1 30 0))))
    (cond ((= problem 6)
         (set-base-levels (nonreferent1 36 0))))
    (cond ((= problem 7)
         (set-base-levels (nonreferent1 42 0))))
    (cond ((= problem 8)
         (set-base-levels (nonreferent1 48 0))))
    (cond ((= problem 9)
         (set-base-levels (nonreferent1 54 0))))
    (cond ((= problem 10)
         (set-base-levels (nonreferent1 60 0))))
    (cond ((= problem 11)
         (set-base-levels (nonreferent1 66 0))))
    (cond ((= problem 12)
         (set-base-levels (nonreferent1 72 0))))
    (cond ((= problem 13)
         (set-base-levels (nonreferent1 84 0))))
    (cond ((= problem 14)
         (set-base-levels (nonreferent1 96 0))))
    (cond ((= problem 15)
         (set-base-levels (nonreferent1 108 0))))
    (cond ((= problem 16)
         (set-base-levels (nonreferent1 114 0))))
    (cond ((= problem 17)
         (set-base-levels (nonreferent1 122 0))))
    (cond ((= problem 18)
         (set-base-levels (nonreferent1 128 0)))))

;; This practices the strategy and makes modifications to the strategy depending on
the
;; second pick.

(defun practicenewnonreferent (n secondpicknum)
     (cond ((= secondpicknum 1)
         (mod-chunk strategy1 selection nonreferent3a)
         (dotimes (x n)
         (rehearse-chunk (strategy1))
         (rehearse-chunk (strategy1)))))
      (cond ((= secondpicknum 2)
         (mod-chunk strategy1 selection nonreferent3a)
         (dotimes (x n)
         (rehearse-chunk (strategy1))
         (rehearse-chunk (strategy1)))))
      (cond ((= secondpicknum 3)
         (mod-chunk strategy1 selection nonreferent1b)
         (dotimes (x n)
         (rehearse-chunk (strategy1))
         (rehearse-chunk (strategy1)))))
      (cond ((= secondpicknum 4)
         (mod-chunk strategy1 selection nonreferent1b)
         (dotimes (x n)
         (rehearse-chunk (strategy1))
```

```
    (rehearse-chunk (strategy1)))))
  (cond ((= secondpicknum 5)
    (mod-chunk strategy1 selection nonreferent1b)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 6)
    (mod-chunk strategy1 selection nonreferent3c)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
  (cond ((= secondpicknum 7)
    (mod-chunk strategy1 selection nonreferent3c)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 8)
    (mod-chunk strategy1 selection nonreferent3c)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 9)
    (mod-chunk strategy1 selection nonreferent3d)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 10)
    (mod-chunk strategy1 selection nonreferent3d)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
  (cond ((= secondpicknum 11)
    (mod-chunk strategy1 selection nonreferent3d)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 12)
    (mod-chunk strategy1 selection nonreferent3e)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 13)
    (mod-chunk strategy1 selection nonreferent3e)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 14)
    (mod-chunk strategy1 selection nonreferent3e)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 15)
    (mod-chunk strategy1 selection nonreferent3f)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 16)
    (mod-chunk strategy1 selection nonreferent3f)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 17)
    (mod-chunk strategy1 selection nonreferent3f)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 18)
    (mod-chunk strategy1 selection nonreferent3g)
    (dotimes (x n)
    (rehearse-chunk (strategy1))
    (rehearse-chunk (strategy1)))))
```

```
(cond ((= secondpicknum 19)
   (mod-chunk strategy1 selection nonreferent3g)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 20)
   (mod-chunk strategy1 selection nonreferent3g)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 21)
   (mod-chunk strategy1 selection nonreferent3h)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 22)
   (mod-chunk strategy1 selection nonreferent3h)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 23)
   (mod-chunk strategy1 selection nonreferent3h)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 24)
   (mod-chunk strategy1 selection nonreferent3i)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 25)
   (mod-chunk strategy1 selection nonreferent3i)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 26)
   (mod-chunk strategy1 selection nonreferent3i)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 27)
   (mod-chunk strategy1 selection nonreferent3j)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 28)
   (mod-chunk strategy1 selection nonreferent3j)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 29)
   (mod-chunk strategy1 selection nonreferent3j)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 30)
   (mod-chunk strategy1 selection nonreferent3k)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 31)
   (mod-chunk strategy1 selection nonreferent3k)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 32)
   (mod-chunk strategy1 selection nonreferent3k)
   (dotimes (x n)
   (rehearse-chunk (strategy1))
   (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 33)
```

```
  (mod-chunk strategy1 selection nonreferent31)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 34)
  (mod-chunk strategy1 selection nonreferent31)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 35)
  (mod-chunk strategy1 selection nonreferent31)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 36)
  (mod-chunk strategy1 selection nonreferent3m)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 37)
  (mod-chunk strategy1 selection nonreferent3m)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 38)
  (mod-chunk strategy1 selection nonreferent3m)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 39)
  (mod-chunk strategy1 selection nonreferent3n)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 40)
  (mod-chunk strategy1 selection nonreferent3n)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 41)
  (mod-chunk strategy1 selection nonreferent3n)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
(cond ((= secondpicknum 42)
  (mod-chunk strategy1 selection nonreferent3o)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 43)
  (mod-chunk strategy1 selection nonreferent3o)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 44)
  (mod-chunk strategy1 selection nonreferent3o)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 45)
  (mod-chunk strategy1 selection nonreferent3p)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 46)
  (mod-chunk strategy1 selection nonreferent3p)
  (dotimes (x n)
  (rehearse-chunk (strategy1))
  (rehearse-chunk (strategy1)))))
 (cond ((= secondpicknum 47)
  (mod-chunk strategy1 selection nonreferent3p)
```

```
          (dotimes (x n)
          (rehearse-chunk (strategy1))
          (rehearse-chunk (strategy1)))))
         (cond ((= secondpicknum 48)
          (mod-chunk strategy1 selection nonreferent3q)
          (dotimes (x n)
          (rehearse-chunk (strategy1))
          (rehearse-chunk (strategy1)))))
         (cond ((= secondpicknum 49)
          (mod-chunk strategy1 selection nonreferent3q)
          (dotimes (x n)
          (rehearse-chunk (strategy1))
          (rehearse-chunk (strategy1)))))
         (cond ((= secondpicknum 50)
          (mod-chunk strategy1 selection nonreferent3q)
          (dotimes (x n)
          (rehearse-chunk (strategy1))
          (rehearse-chunk (strategy1)))))
         (cond ((= secondpicknum 51)
          (mod-chunk strategy1 selection nonreferent3r)
          (dotimes (x n)
          (rehearse-chunk (strategy1))
          (rehearse-chunk (strategy1)))))
         (cond ((= secondpicknum 52)
          (mod-chunk strategy1 selection nonreferent3r)
          (dotimes (x n)
          (rehearse-chunk (strategy1))
          (rehearse-chunk (strategy1)))))
        (cond ((= secondpicknum 53)
          (mod-chunk strategy1 selection nonreferent3r)
          (dotimes (x n)
          (rehearse-chunk (strategy1))
          (rehearse-chunk (strategy1))))))

;; one referent condition for each question, with the first one being zero

(defun referentparameters (x)
  (cond ((= x 0)
          (mod-chunk referent1 center a middle b outside c next overall1)
          (mod-chunk strategy1 operations 3 original 3 firstpick nonreferent1a
firstpicknum 0
              secondpick nonreferent2a secondpicknum 1 selection 0 counter 1)
          (mod-chunk overall1 firstpick nonreferent1a)
          (mod-chunk attention1 problem 1)))
  (cond ((= x 1)
          (mod-chunk referent1 center a middle b outside c next overall1)
          (mod-chunk strategy1 operations 9 original 9 firstpick nonreferent3b
firstpicknum 5
            secondpick nonreferent2b secondpicknum 4 selection 0 counter 1)
          (mod-chunk overall1 firstpick nonreferent1b)
          (mod-chunk attention1 problem 2)))
  (cond ((= x 2)
          (mod-chunk referent1 center a middle b outside c next overall1)
          (mod-chunk strategy1 operations 4 original 4 firstpick nonreferent2c
firstpicknum 7
            secondpick nonreferent1c secondpicknum 6 selection nonreferent3c counter
1)
          (mod-chunk overall1 firstpick nonreferent1c)
          (mod-chunk attention1 problem 3)))
  (cond ((= x 3)
          (mod-chunk referent1 center a middle b outside c next overall1)
          (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent1d
firstpicknum 9
            secondpick nonreferent2d secondpicknum 10 selection 0 counter 1)
          (mod-chunk overall1 firstpick nonreferent1d)
          (mod-chunk attention1 problem 4)))
  (cond ((= x 4)
          (mod-chunk referent1 center a middle b outside c next overall1)
          (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent1e
firstpicknum 12
            secondpick nonreferent2e secondpicknum 13 selection 0 counter 1)
```

```
        (mod-chunk overall1 firstpick nonreferent1e)
        (mod-chunk attention1 problem 5)))
  (cond ((= x 5)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent1f
firstpicknum 15
           secondpick nonreferent2f secondpicknum 16 selection nonreferent3f counter
1)
        (mod-chunk overall1 firstpick nonreferent1f)
        (mod-chunk attention1 problem 6)))
  (cond ((= x 6)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent2g
firstpicknum 19
           secondpick nonreferent3g secondpicknum 20 selection 0 counter 1)
        (mod-chunk overall1 firstpick nonreferent1g)
        (mod-chunk attention1 problem 7)))
  (cond ((= x 7)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent3h
firstpicknum 23
           secondpick nonreferent1h secondpicknum 21 selection 0 counter 1)
        (mod-chunk overall1 firstpick nonreferent1h)
        (mod-chunk attention1 problem 8)))
  (cond ((= x 8)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent3i
firstpicknum 26
           secondpick nonreferent2i secondpicknum 25 selection nonreferent3i counter
1)
        (mod-chunk overall1 firstpick nonreferent1i)
        (mod-chunk attention1 problem 9)))
  (cond ((= x 9)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent1j
firstpicknum 27
           secondpick nonreferent2j secondpicknum 28 selection 0 counter 1)
        (mod-chunk overall1 firstpick nonreferent1j)
        (mod-chunk attention1 problem 10)))
  (cond ((= x 10)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 31 firstpick nonreferent1k
firstpicknum 30
           secondpick nonreferent2k secondpicknum 31 selection 0 counter 1)
        (mod-chunk overall1 firstpick nonreferent1k)
        (mod-chunk attention1 problem 11)))
  (cond ((= x 11)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent3l
firstpicknum 35
           secondpick nonreferent1l secondpicknum 33 selection nonreferent3l counter
1)
        (mod-chunk overall1 firstpick nonreferent1l)
        (mod-chunk attention1 problem 12)))
  (cond ((= x 12)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent1m
firstpicknum 36
           secondpick nonreferent2m secondpicknum 37 selection 0 counter 1)
        (mod-chunk overall1 firstpick nonreferent1m)
        (mod-chunk attention1 problem 13)))
  (cond ((= x 13)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent1n
firstpicknum 39
           secondpick nonreferent2n secondpicknum 40 selection 0 counter 1)
        (mod-chunk overall1 firstpick nonreferent1n)
        (mod-chunk attention1 problem 14)))
  (cond ((= x 14)
        (mod-chunk referent1 center a middle b outside c next overall1)
```

```
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent2o
firstpicknum 43
           secondpick nonreferent3o secondpicknum 44 selection nonreferent3o counter
1)
        (mod-chunk overall1 firstpick nonreferent1o)
        (mod-chunk attention1 problem 15)))
  (cond ((= x 15)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent1p
firstpicknum 45
           secondpick nonreferent2p secondpicknum 46 selection 0 counter 1)
        (mod-chunk overall1 firstpick nonreferent1p)
        (mod-chunk attention1 problem 16)))
  (cond ((= x 16)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent1q
firstpicknum 48
           secondpick nonreferent3q secondpicknum 50 selection 0 counter 1)
        (mod-chunk overall1 firstpick nonreferent1q)
        (mod-chunk attention1 problem 17)))
  (cond ((= x 17)
        (mod-chunk referent1 center a middle b outside c next overall1)
        (mod-chunk strategy1 operations 2 original 2 firstpick nonreferent1r
firstpicknum 51
           secondpick nonreferent2r secondpicknum 52 selection nonreferent3r counter
1)
        (mod-chunk overall1 firstpick nonreferent1r)
        (mod-chunk attention1 problem 18))))

;;; again modifying the nonreferents.  We have a chunk, nonreferent1, for each
nonreferent.  Overall is a type
;;; of strategy that helps determine the first pick.  He we need 1 condition for each
nonreferent.  There are
;;; 18 questions each with 3 non-referents, so eventually I will have 53 conditions
here (starting from zero).

(defun nonreferentparameters (x)
    (cond ((= x 0)
       (mod-chunk nonreferent1 name nonreferent1a number 1 center a middle b outside c
type different next nonreferent2a)
       (mod-chunk overall1 firstpick nonreferent1a)
       (mod-chunk strategy1 operations 3)))
    (cond ((= x 1)
       (mod-chunk nonreferent1 name nonreferent2a number 2 center j middle k outside l
type answer next nonreferent3a)
       (mod-chunk overall1 firstpick nonreferent2a)
       (mod-chunk strategy1 operations 9)))
     (cond ((= x 2)
       (mod-chunk nonreferent1 name nonreferent3a number 3 center a middle b outside c
type different next nonreferent1b)
       (mod-chunk overall1 firstpick nonreferent3a)
       (mod-chunk strategy1 operations 4)))
     (cond ((= x 3)
        (mod-chunk nonreferent1 name nonreferent1b number 4 center j middle k outside
l type answer next nonreferent2b)
        (mod-chunk overall1 firstpick nonreferent3b)
        (mod-chunk strategy1 operations 6)))
      (cond ((= x 4)
        (mod-chunk nonreferent1 name nonreferent2b number 5 center a middle b outside
c type different next nonreferent3b)
        (mod-chunk overall1 firstpick nonreferent2b)
        (mod-chunk strategy1 operations 6)))
      (cond ((= x 5)
        (mod-chunk nonreferent1 name nonreferent3b number 6 center a middle b outside
c type different next nonreferent1c)
        (mod-chunk overall1 firstpick nonreferent3b)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 6)
        (mod-chunk nonreferent1 name nonreferent1c number 7 center g middle h outside
i type mirror next nonreferent2c)
        (mod-chunk overall1 firstpick nonreferent2c)
```

```
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 7)
        (mod-chunk nonreferent1 name nonreferent2c number 8 center g middle h outside
i type mirror next nonreferent3c)
        (mod-chunk overall1 firstpick nonreferent2c)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 8)
        (mod-chunk nonreferent1 name nonreferent3c number 9 center j middle k outside
l type answer next nonreferent1d)
        (mod-chunk overall1 firstpick nonreferent3c)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 9)
        (mod-chunk nonreferent1 name nonreferent1d number 10 center a middle b outside
c type different next nonreferent2d)
        (mod-chunk overall1 firstpick nonreferent1d)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 10)
        (mod-chunk nonreferent1 name nonreferent2d number 11 center a middle b outside
c type different next nonreferent3d)
        (mod-chunk overall1 firstpick nonreferent2d)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 11)
        (mod-chunk nonreferent1 name nonreferent3d number 12 center j middle k outside
l type answer next nonreferent1e)
        (mod-chunk overall1 firstpick nonreferent3d)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 12)
        (mod-chunk nonreferent1 name nonreferent1e number 13 center a middle b outside
c type different next nonreferent2e)
        (mod-chunk overall1 firstpick nonreferent1e)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 13)
        (mod-chunk nonreferent1 name nonreferent2e number 14 center j middle k outside
l type answer next nonreferent3e)
        (mod-chunk overall1 firstpick nonreferent2e)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 14)
        (mod-chunk nonreferent1 name nonreferent3e number 15 center a middle b outside
c type different next nonreferent1f)
        (mod-chunk overall1 firstpick nonreferent3e)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 15)
        (mod-chunk nonreferent1 name nonreferent1f number 16 center a middle b outside
c type different next nonreferent2f)
        (mod-chunk overall1 firstpick nonreferent1f)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 16)
        (mod-chunk nonreferent1 name nonreferent2f number 17 center j middle k outside
l type answer next nonreferent3f)
        (mod-chunk overall1 firstpick nonreferent2f)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 17)
        (mod-chunk nonreferent1 name nonreferent3f number 18 center g middle h outside
i type mirror next nonreferent1g)
        (mod-chunk overall1 firstpick nonreferent3f)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 18)
        (mod-chunk nonreferent1 name nonreferent1g number 19 center j middle k outside
l type answer next nonreferent2g)
        (mod-chunk overall1 firstpick nonreferent1g)
        (mod-chunk strategy1 operations 6)))
        (cond ((= x 19)
        (mod-chunk nonreferent1 name nonreferent2g number 20 center a middle b outside
c type different next nonreferent3g)
        (mod-chunk overall1 firstpick nonreferent2g)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 20)
        (mod-chunk nonreferent1 name nonreferent3g number 21 center a middle b outside
c type different next nonreferent1h)
        (mod-chunk overall1 firstpick nonreferent3g)
        (mod-chunk strategy1 operations 6)))
```

```
       (cond ((= x 21)
        (mod-chunk nonreferent1 name nonreferent1h number 22 center g middle h outside
i type mirror next nonreferent2h)
        (mod-chunk overall1 firstpick nonreferent1h)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 22)
        (mod-chunk nonreferent1 name nonreferent2h number 23 center j middle k outside
l type answer next nonreferent3h)
        (mod-chunk overall1 firstpick nonreferent2h)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 23)
        (mod-chunk nonreferent1 name nonreferent3h number 24 center a middle b outside
c type different next nonreferent1i)
        (mod-chunk overall1 firstpick nonreferent3h)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 24)
        (mod-chunk nonreferent1 name nonreferent1i number 25 center j middle k outside
l type answer next nonreferent2i)
        (mod-chunk overall1 firstpick nonreferent1i)
        (mod-chunk strategy1 operations 6)))
        (cond ((= x 25)
        (mod-chunk nonreferent1 name nonreferent2i number 26 center g middle h outside
i type mirror next nonreferent3i)
        (mod-chunk overall1 firstpick nonreferent2i)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 26)
        (mod-chunk nonreferent1 name nonreferent3i number 27 center d middle e outside
f type shifted next nonreferent1j)
        (mod-chunk overall1 firstpick nonreferent3i)
        (mod-chunk strategy1 operations 6)))
        (cond ((= x 27)
        (mod-chunk nonreferent1 name nonreferent1j number 28 center j middle k outside
l type answer next nonreferent2j)
        (mod-chunk overall1 firstpick nonreferent1j)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 28)
        (mod-chunk nonreferent1 name nonreferent2j number 29 center g middle h outside
i type mirror next nonreferent3j)
        (mod-chunk overall1 firstpick nonreferent2j)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 29)
        (mod-chunk nonreferent1 name nonreferent3j number 30 center g middle h outside
i type mirror next nonreferent1k)
        (mod-chunk overall1 firstpick nonreferent3j)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 30)
        (mod-chunk nonreferent1 name nonreferent1k number 31 center a middle b outside
c type different next nonreferent2k)
        (mod-chunk overall1 firstpick nonreferent1k)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 31)
        (mod-chunk nonreferent1 name nonreferent2k number 32 center a middle b outside
c type different next nonreferent3k)
        (mod-chunk overall1 firstpick nonreferent2k)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 32)
        (mod-chunk nonreferent1 name nonreferent3k number 33 center j middle k outside
l type answer next nonreferent1l)
        (mod-chunk overall1 firstpick nonreferent3k)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 33)
        (mod-chunk nonreferent1 name nonreferent1l number 34 center a middle b outside
c type different next nonreferent2l)
        (mod-chunk overall1 firstpick nonreferent1l)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 34)
        (mod-chunk nonreferent1 name nonreferent2l number 35 center j middle k outside
l type answer next nonreferent3l)
        (mod-chunk overall1 firstpick nonreferent2l)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 35)
```

```
        (mod-chunk nonreferent1 name nonreferent31 number 36 center d middle e outside
f type different next nonreferent1m)
        (mod-chunk overall1 firstpick nonreferent31)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 36)
        (mod-chunk nonreferent1 name nonreferent1m number 37 center g middle h outside
i type mirror next nonreferent2m)
        (mod-chunk overall1 firstpick nonreferent1m)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 37)
        (mod-chunk nonreferent1 name nonreferent2m number 38 center a middle b outside
c type different next nonreferent3m)
        (mod-chunk overall1 firstpick nonreferent2m)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 38)
        (mod-chunk nonreferent1 name nonreferent3m number 39 center d middle e outside
f type answer next nonreferent1n)
        (mod-chunk overall1 firstpick nonreferent3m)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 39)
        (mod-chunk nonreferent1 name nonreferent1n number 40 center g middle h outside
i type mirror next nonreferent2n)
        (mod-chunk overall1 firstpick nonreferent1n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 40)
        (mod-chunk nonreferent1 name nonreferent2n number 41 center d middle e outside
f type shifted next nonreferent3n)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 41)
        (mod-chunk nonreferent1 name nonreferent3n number 42 center j middle k outside
l type answer next nonreferent1o)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 42)
        (mod-chunk nonreferent1 name nonreferent1o number 43 center a middle b outside
c type different next nonreferent2o)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 43)
        (mod-chunk nonreferent1 name nonreferent2o number 44 center a middle b outside
c type different next nonreferent3o)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 44)
        (mod-chunk nonreferent1 name nonreferent3o number 45 center j middle k outside
l type answer next nonreferent1p)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 45)
        (mod-chunk nonreferent1 name nonreferent1p number 46 center d middle e outside
f type shifted next nonreferent2p)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 46)
        (mod-chunk nonreferent1 name nonreferent2p number 47 center j middle k outside
l type answer next nonreferent3p)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 47)
        (mod-chunk nonreferent1 name nonreferent3p number 48 center g middle h outside
i type mirror next nonreferent1q)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
        (cond ((= x 48)
        (mod-chunk nonreferent1 name nonreferent1q number 49 center d middle e outside
f type shifted next nonreferent2q)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 49)
```

```
        (mod-chunk nonreferent1 name nonreferent2q number 50 center g middle h outside
i type mirror next nonreferent3q)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
        (cond ((= x 50)
        (mod-chunk nonreferent1 name nonreferent3q number 51 center j middle k outside
l type answer next nonreferent1r)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
       (cond ((= x 51)
        (mod-chunk nonreferent1 name nonreferent1r number 52 center j middle k outside
l type answer next nonreferent2r)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
      (cond ((= x 52)
        (mod-chunk nonreferent1 name nonreferent2r number 53 center d middle e outside
f type shifted next nonreferent3r)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6)))
      (cond ((= x 53)
        (mod-chunk nonreferent1 name nonreferent3r number 54 center d middle e outside
f type shifted next 0)
        (mod-chunk overall1 firstpick nonreferent2n)
        (mod-chunk strategy1 operations 6))))

;; run each question separately to prevent too much spreading activation

(defun question1 (x)
 (dotimes (i x)
 (referentparameters 0)
 (run)
 (reset)))

(defun question2 (x)
 (dotimes (i x)
 (referentparameters 1)
 (run)
 (reset)))

(defun question3 (x)
 (dotimes (i x)
 (referentparameters 2)
 (run)
 (reset)))

(defun question4 (x)
 (dotimes (i x)
 (referentparameters 3)
 (run)
 (reset)))

(defun question5 (x)
 (dotimes (i x)
 (referentparameters 4)
 (run)
 (reset)))

(defun question6 (x)
 (dotimes (i x)
 (referentparameters 5)
 (run)
 (reset)))

(defun question7 (x)
 (dotimes (i x)
 (referentparameters 6)
 (run)
 (reset)))

(defun question8 (x)
 (dotimes (i x)
```

```
 (referentparameters 7)
 (run)
 (reset)))

(defun question9 (x)
 (dotimes (i x)
 (referentparameters 8)
 (run)
 (reset)))

(defun question10 (x)
 (dotimes (i x)
 (referentparameters 9)
 (run)
 (reset)))

(defun question11 (x)
 (dotimes (i x)
 (referentparameters 10)
 (run)
 (reset)))

(defun question12 (x)
 (dotimes (i x)
 (referentparameters 11)
 (run)
 (reset)))

(defun question13 (x)
 (dotimes (i x)
 (referentparameters 12)
 (run)
 (reset)))

(defun question14 (x)
 (dotimes (i x)
 (referentparameters 13)
 (run)
 (reset)))

(defun question15 (x)
 (dotimes (i x)
 (referentparameters 14)
 (run)
 (reset)))

(defun question16 (x)
 (dotimes (i x)
 (referentparameters 15)
 (run)
 (reset)))

(defun question17 (x)
 (dotimes (i x)
 (referentparameters 16)
 (run)
 (reset)))

(defun question18 (x)
 (dotimes (i x)
 (referentparameters 17)
 (run)
 (reset)))


(clear-all)

;; I have different trace levels here depending on what I am looking at.

;;(sgp :pm t :era t :bll 0.45 :al 0.05 :blc 0.05 :rt 0.20 :act t :pmt t)
;;(sgp :pm t :era t :bll 0.45 :al 0.05 :blc 0.05 :rt 0.20 :pmt t :emt t :act t :pt t)
```

```
;;(sgp :pm t :era t :bll 0.45 :al 0.05 :blc 0.05 :rt 0.02 :pan 1 :an 1)
(sgp :pm t :era t :bll 0.45 :al 0.05 :blc 0.05 :rt 0.02 :an .05)



(chunk-type attention problem)
(chunk-type overall firstpick)
(chunk-type strategy operations original firstpick firstpicknum
            practice secondpick secondpicknum selection counter)
(chunk-type referent center middle outside next fire)
(chunk-type nonreferent name number center middle outside type next firstpick
manipulations)
(chunk-type match)
(chunk-type compare first second)
(chunk-type respond)
(chunk-type eliminate name)

(add-dm
   (overall1 isa overall firstpick nonreferent2a)
   (compare1 isa compare first 0 second 0)
   (attention1 isa attention problem 0)
   (match1 isa match)
   (strategy1 ISA strategy operations 5 original 5 firstpick nonreferent3a
firstpicknum 2 practice 0
               secondpick nonreferent2a secondpicknum 1 selection 0 counter 1)
   (referent1 ISA referent center a middle b outside c next overall1 fire 0)
   (nonreferent1 ISA nonreferent name nonreferent1a number 1 center a middle b outside
c type mirror next nonreferent2a)
   (nonreferent1a isa chunk)
   (nonreferent2a isa chunk)
   (nonreferent3a isa chunk)
   (nonreferent1b isa chunk)
   (nonreferent2b isa chunk)
   (nonreferent3b isa chunk)
   (nonreferent1c isa chunk)
   (nonreferent2c isa chunk)
   (nonreferent3c isa chunk)
   (nonreferent1d isa chunk)
   (nonreferent2d isa chunk)
   (nonreferent3d isa chunk)
   (nonreferent1e isa chunk)
   (nonreferent2e isa chunk)
   (nonreferent3e isa chunk)
   (nonreferent1f isa chunk)
   (nonreferent2f isa chunk)
   (nonreferent3f isa chunk)
   (nonreferent1g isa chunk)
   (nonreferent2g isa chunk)
   (nonreferent3g isa chunk)
   (nonreferent1h isa chunk)
   (nonreferent2h isa chunk)
   (nonreferent3h isa chunk)
   (nonreferent1i isa chunk)
   (nonreferent2i isa chunk)
   (nonreferent3i isa chunk)
   (nonreferent1j isa chunk)
   (nonreferent2j isa chunk)
   (nonreferent3j isa chunk)
   (nonreferent1k isa chunk)
   (nonreferent2k isa chunk)
   (nonreferent3k isa chunk)
   (nonreferent1l isa chunk)
   (nonreferent2l isa chunk)
   (nonreferent3l isa chunk)
   (d isa chunk)
   (e isa chunk)
   (f isa chunk)
   (g isa chunk)
   (h isa chunk)
   (i isa chunk)
   (j isa chunk)
```

```
   (k isa chunk)
   (l isa chunk)
   (m isa chunk)
   (n isa chunk)
   (o isa chunk)
   (p isa chunk)
   (q isa chunk)
   (r isa chunk)
   (s isa chunk)
   (t isa chunk)
   (u isa chunk)
   (v isa chunk)
   (w isa chunk)
   (center isa chunk)
   (middle isa chunk)
   (outside isa chunk))

;; here we look at the referent, and push a goal to start the matching process.


(P look-at-referent
   =goal>
      ISA         referent
      center      =centerRef
      middle      =middleRef
      outside     =outsideRef
      next        =next
      fire        =fire
    - fire        4

    =goal2>
      isa         attention
    problem       =problem

    =goal3>
      isa         match
==>
   !push!         =next

   !bind!         =newvar (1+ =fire)

   =goal>
     isa           referent
     fire          =newvar


   !output!   (=newvar)
   !output!   ("referent of problem ~s" =problem)
)


(P give-up-looking-at-referent
      =goal>
        ISA         referent
      center      =centerRef
      middle      =middleRef
      outside     =outsideRef
      next        =next
      fire        4
==>
      !pop!
      !pop!
)

;;; Here we look at all the referents in a general way and develop a strategy for the
first pick, which
;;; is our first selection.  In our strategy chunk, we have our operations number,
which is the number
;;; of times we are going to manipulate the referent in order to match it to the non-
referent.  We
```

```
;;; also have firstpicknum, which is a numerical representation of the non-referent.
I found it easier
;;; to send number calls back and forth from LISP than sending it word strings.

(P look-at-overall-get-strategy
   =goal>
      ISA          overall
    firstpick      =firstpick
   =goal2>
      ISA          strategy
      operations   =operations
    firstpick      =stratfirstpick
    firstpicknum    =nonreferentnum
    - counter       3
   =fact>
      isa          nonreferent
    type           =type

==>
    !eval! (nonreferentparameters =nonreferentnum)
    !eval! (practiceStrategy =operations)
    !output! ("nonreferentnumber ~s" =nonreferentnum)

   =goal4>
      isa          strategy
   firstpick        =firstpick

   !push!          =goal2
)

;; This fires if we have eliminated the other two non-referents (i.e. our counter
equals three) so we
;; know at that point it must be the last choice.


(P Process-of-elimanation-select
   =goal>
      ISA          overall
    firstpick      =firstpick
   =goal2>
      ISA          strategy
      operations   =operations
    firstpick      =stratfirstpick
    firstpicknum    =nonreferentnum
    selection       =select
    counter         3
   =fact>
      isa          nonreferent
    type           =type
    name           =name

==>
    =fact3>
     isa          eliminate
    name          =select

    =nextgoal>
       isa          eliminate
      name          =select

    !focus-on!          =nextgoal
    !output! =select
    !output! =type
)

;; here we actually make a selection based on the elimination of the other two non-
referents
;; and move on to the next problem


(P Make-selection-based-on-elimanation
```

```
      =goal>
        isa         eliminate
      name          =select
      =fact>
        isa         nonreferent
        name          =firstpick
       number         =number
      fact2>
         isa       attention
       problem     =problem

==>

     !output! ("selecting based on elimination nonreferent named ~s" =select)
     !pop!
     !pop!
    !eval! (referentparameters =problem)
    !eval! (nonreferentparameters =number)
    !eval! (set-base-levels (nonreferent1 3 0))

)

;; This is our manipulate production which loops and fires a certain number of times
;; depending on the number in the "operations" slot.  If operations hits zero it stops
firing.
;; Note, we are just concentrating on the nonreferent, not the referent, which is what
produces
;; decay of the referent.
;;; the bind function at the end decrements the operations
;;; slot by one each time, so it will not fire if
;;; operations is zero



(P manipulate
   =goal>
      ISA          strategy
      operations   =operations
    - operations   0
     firstpick     =firstpick
     firstpicknum  =num
   =goal2>
      ISA          nonreferent
      name         =firstpick
      center       =centerNON
      middle       =middleNON
      outside      =outsideNON
    =fact2>
      isa         match
==>
   !bind!          =newvar (1- =operations)
   =goal>
      operations   =newvar

   !output!    ("manipulating the referent while looking at ~S" =firstpick)
   !output!    ("maniplation number ~s" =operations)
)
;;; the bind function here decrements the operations
;;; slot by one each time, so it will not fire if
;;; operations is zero


;; this fires when we are done manipulating

(P done-manipulate
   =goal>
       isa        strategy
       operations   0
   =fact2>
        isa       match
 ==>
```

```
   !eval! (Set-Base-Levels (compare1 10 5))
   !push!          =fact2
)

;; try and remember the referent

(P attempt-match
   =goal>
      ISA          match
   =fact>
      ISA          referent
      center       =centerREF
      middle       =middleREF
      outside      =outsideREF
   =goal2>
      isa           strategy
    operations      =operations
    original        =original
    firstpick       =firstpick
   =goal3>
      isa           compare
==>
     !push!         =goal3
   !output! ("remembered referent clearly enough to attempt match")
   !output! =firstpick
   !output! =original
   !eval! (practiceStrategy =original)
)

;; this will match do partial matching of images
;; note we have center, middle, and outside.  Each of these
;; has a similarity that is set in the set similarities command
;; Mirror objects have higher similarity values to the referent.

(P match-images
     =goal>
      isa            compare
     =fact1>
       isa          strategy
      firstpick     =firstpick
     =fact2>
      isa          nonreferent
      name         =firstpick
      center       centerNON
     middle        middleNON
    outside        outsideNON
==>
    =nextgoal>
       isa          respond
    !push!          =nextgoal
    !output! =firstpick
)

;; Here is where we rule out a non-referent if the match wasn't good.  Basically this
will
;; fire if we decrease activation levels of the non-referent enough (through partial
matching)
;; so that the non-referent is basically "forgotten".  Not too much psychological
validity, but
;; this is the best I could come up with right now.


(P rule-out-non-referent
    =goal>
       isa              compare
    =goal2>
       isa              strategy
       firstpick        =firstpick
       secondpick       =second
       secondpicknum    =secondpicknum
       original         =original
```

```
      counter          =counter
    =goal3>
      isa              nonreferent
      name             =firstpick
      number           =number
==>

     !bind!          =newvar (1+ =counter)

     =goal2>
        isa        strategy
     firstpick     =second
    firstpicknum   =secondpicknum
     operations    =original
     original      =original
     counter       =newvar

     !eval! (nonreferentparameters =number)
     !eval! (practicenewnonreferent =original =secondpicknum)

     !pop!
     !pop!
     !pop!
     !pop!

    !eval! (set-base-levels (nonreferent1 3 0))
    !output! ("ruling out referent ~s"=firstpick)
    !output! ("setting non-referent parameters with var ~s"=number)
    !output! ("second pick number is ~s"=secondpicknum)
    !output! ("origial manipulations is ~s"=original)
)

;; This fires if we do remember the non-referent clearly enough and partial matching
confirms a match.

(P make-response
     =goal>
       isa        respond
     =goal2>
       isa        strategy
     firstpick    =firstpick
     =fact>
       isa        referent
     =fact2>
       isa        nonreferent
      name        =firstpick
      number      =number
      type        =type

     =fact3>
       isa        attention
      problem     =problem

==>
     !output!     ("the current pick is ~s"  =firstpick)
     !pop!
     !pop!
     !pop!
     !pop!
     !pop!
     !pop!
    !eval! (referentparameters =problem)
    ;;!eval! (nonreferentbase =problem)


    !bind!          =newvar (* 3 =problem)

    !eval! (nonreferentparameters =newvar)
    !output! =problem
    !output! =number
    !output! =type
```

```
    !output! =newvar
    !eval! (set-base-levels (referent1 6 0))
    !eval! (set-base-levels (nonreferent1 3 0))
)

;; if we could not remember, we need to reactivate
;; the base levels for referent1 so that we can bring
;; up the activation levels for us to remember the next
;; time around.  That is also why we focus on the
;; referent goal.

(p could-not-remember-referent
    =goal>
        isa      match
    =goal2>
        isa      strategy
   original      =original
   operations    =operations
   original      =original
   practice      =practice
==>
   !eval! (set-base-levels (nonreferent1 3 0))

   !bind!          =newvar (1+ =practice)

   !eval!         (practicereferent =newvar)
   !output!       ("practicing referent")

   =goal2>
     isa         strategy
   operations    =original

   =goal3>
       isa        referent


    !pop!
    !pop!

    !focus-on!    =goal3

)

;; if we could not remember, we need to reactivate
;; the base levels for referent1 so that we can bring
;; up the activation levels for us to remember the next
;; time around.  That is also why we focus on the
;; referent goal.

;; For right now, A, B, C is different - make sure it is for the NONreferent
;; D, E and F are shifted
;; G, H, I are mirror
;; J, K, L is the answer

;; We might in the end have to make these unique to every problem

(spp done-manipulate :strength 0)
(goal-focus referent1)
(Setsimilarities
  (a center .99)
  (b middle .99)
  (c outside .99)
  (a centerNON .60)
  (b middleNON .30)
  (c outsideNON .10)
  (d centerNON .6)
  (e middleNON .3)
  (f outsideNON .1)
  (g centerNON .2)
  (h middleNON .4)
  (i outsideNON .6)
```

```
(j centerNON .99)
(k middleNON .99)
(l outsideNON .99)
 (Nonreferent3a Nonreferent3b .99)
(Nonreferent3d Nonreferent1k .99)
(Nonreferent3c Nonreferent1i .99)
(Nonreferent2b Nonreferent1e .99)
(Nonreferent1b Nonreferent1d .99)
(Nonreferent3a Nonreferent1c .99)
(Nonreferent2a Nonreferent1b .99)
(Nonreferent1a Nonreferent2a .99)
(Nonreferent2a Nonreferent3a .99)
(Nonreferent3a Nonreferent1a .99)
(Nonreferent1b Nonreferent2b .99)
(Nonreferent2b Nonreferent3b .99)
(Nonreferent3b Nonreferent1b .99)
(Nonreferent1c Nonreferent2c .99)
(Nonreferent2c Nonreferent3c .99)
(Nonreferent3c Nonreferent1c .99)
(Nonreferent1d Nonreferent2d .99)
(Nonreferent2d Nonreferent3d .99)
(Nonreferent3d Nonreferent1d .99)
(Nonreferent1e Nonreferent2e .99)
(Nonreferent2e Nonreferent3e .99)
(Nonreferent3e Nonreferent1e .99)
(Nonreferent1f Nonreferent2f .99)
(Nonreferent2f Nonreferent3f .99)
(Nonreferent3f Nonreferent1f .99)
(Nonreferent1g Nonreferent2g .99)
(Nonreferent2g Nonreferent3g .99)
(Nonreferent3g Nonreferent1g .99)
(Nonreferent1h Nonreferent2h .99)
(Nonreferent2h Nonreferent3h .99)
(Nonreferent3h Nonreferent1h .99)
(Nonreferent1i Nonreferent2i .99)
(Nonreferent2i Nonreferent3i .99)
(Nonreferent3i Nonreferent1i .99)
(Nonreferent1j Nonreferent2j .99)
(Nonreferent2j Nonreferent3j .99)
(Nonreferent3j Nonreferent1j .99)
(Nonreferent1k Nonreferent2k .99)
(Nonreferent2k Nonreferent3k .99)
(Nonreferent3k Nonreferent1k .99)
(Nonreferent1l Nonreferent2l .99)
(Nonreferent2l Nonreferent3l .99)
(Nonreferent3l Nonreferent1l .99))
```

INTENTIONALLY LEFT BLANK

APPENDIX C

SIMPLE FINAL MODEL

INTENTIONALLY LEFT BLANK

SIMPLE FINAL MODEL

```
;; ACT-R model of mental manipulation by Troy Kelley and Frank Lee

(defparameter *answer* 0)
~
(defparameter *mirror* 1)
~
(defparameter *different* 2)
~
(defparameter *others* 3)
~
(defparameter *never* 4)
~

~
(defparameter *size* 'Seven)
~

~
(defparameter *start* nil)
~

~
(defparameter *returnAnswer* nil)

(defparameter *sizeList* '(Seven Nine Eleven))


;; This function runs the model multiple times
;;
;; variables
;; runs - number of runs requested
;; manipulate - the number of manipulations requested
;; out - whether to print out the data
;; prop - t = write out data in proportion / nil = write data in raw count
;;
;; given the numer of runs, the model will run the model that many times
;; for each combination of size, (seven nine eleven), and manipulation, 1 -
manipulation).
;; Hence, given runs = 20 and manipulate = 10, the model will 20 times of (3 x 10)
;; combinations of polygons and 1, 2, ..., 10, manipulations.
;;


(defun do-run (runs &key (manipulation 10) (out t) (prop nil))
  (let (answers index)
    (dolist (size *sizeList*)
      (setf *size* size)
      (setf answers (make-array (list manipulation 10) :element-type 'integer
:initial-element 0))
      (dotimes (i runs)
        (dotimes (j manipulation)
          (reset)
          (setf *start* nil)
          (mod-chunk-fct 'referent1 (list 'count (1+ j)))
          (run)
          (if (null *start*)
            (setf index *never*)
            (if (null *returnAnswer*)
              (setf index *others*)
              (setf index *returnAnswer*)))
          (setf (aref answers j index) (1+ (aref answers  j index)))
          (setf *returnAnswer* nil)
          )
        )
      (format out "Size: ~a~%" size)
      (write-out-header t)
      (if prop
        (write-out-proportion out answers)
        (write-out-raw out answers))

      (format out "~%")
      )
```

```
    )
  )
;; put column headers in the output
(defun write-out-header (out)
  (format out "answer~cmirror~cdiff~cothers~cnever~%" #\tab #\tab #\tab #\tab ))

;; write out the data as a proportion
(defun write-out-proportion (out answers)
  (let ((count (first (array-dimensions answers)))
        (answer (second (array-dimensions answers)))
        (sum 0))
    (dotimes (i count)
      (setf sum 0)
      (dotimes (k answer)
        (setf sum (+ (aref answers i k) sum)))
      (dotimes (j answer)
        (format out "~,2f~c" (/ (aref answers i j) sum) #\Tab))
      (format out "~%")))

)

;; write out data in as a raw count
(defun write-out-raw (out answers)
  (let ((count (first (array-dimensions answers)))
        (answer (second (array-dimensions answers))))
    (dotimes (i count)
      (dotimes (j answer)
        (format out "~A~c" (aref answers i j)  #\Tab))
      (format out "~%")))

)

;; This function simply sets the global variable *returnAnswer* with the
;; chunk that was retrieved. This variable is used to tally up the answers
;; that are retrieved in multiple runs of the model.

(defun set-answer (x)
  (cond ((equal x 'partsAnswer)
         (setf *returnAnswer* *answer*))
        ((equal x 'partsMirror)
         (setf *returnAnswer* *mirror*))
        ((equal x 'partsDifferent)
         (setf *returnAnswer* *different*))
        )
  )

;;The basic idea of manipulate-similarity is to change the similarities
;;between the pair of chunks, ANSWER and MIRROR and between ANSWER, DIFFERENT
;;as a function of the manipulation.
;;
;;variables:
;;manipulation - passed in argument (while it can be anything, it is
;;               assumed to be a positive whole number that matches
;;               to some abstract number of manipulation that is performed
;;               on the mental image
;;
;;scale        - it is initialized to manipulation * .05.
;;
;;
;;As one can see in the definition, depending the what the size of the source
;;polygon is (i.e. seven, nine, or eleven polygons) set-similarities-fct is
;;called with (+ BASE scale). For instance, when minipulate-similarity is called
;;with manipulation = 2, this sets scale = (2 * 0.5) = .1. If the *size* is equal
;;to seven polygons, then set-similarities is called with
;;(answer mirror (+ .65 .1))
;;(answer different (+.60 .1))
;;
;; As one can see as the number of manipulation increases, the similarities goes
;; to 1.0, i.e. they are indistinguishable

(defun manipulate-similarity (manipulation)
  (let ((scale (* manipulation .05)))
    (when (equal *size* 'seven)
      (set-similarities-fct (list
                             (list 'answer 'mirror (if (> (+ .65 scale) 1) 1.0 (+ .65
scale)))
```

```
                                (list 'answer 'different (if (> (+ .60 scale) 1) 1.0 (+
.60 scale))))))
    (when (equal *size* 'Nine)
      (set-similarities-fct (list


                                (list 'answer 'mirror (if (> (+ .8 scale) 1) 1.0 (+ .8
scale)))
                                (list 'answer 'different (if (> (+ .60 scale) 1) 1.0 (+
.60 scale))))))

    (when (equal *size* 'Eleven)
      (set-similarities-fct (list

                                (list 'answer 'mirror (if (> (+ .76 scale) 1) 1.0 (+ .76
scale)))
                                (list 'answer 'different (if (> (+ .62 scale) 1) 1.0 (+
.62 scale))))))
    ))

(clear-all)

(sgp :rt .05 :pm t :er t :egs 1 :blc 6 :pmt t :ans .1 :mp 3 :pas .1 :v nil)
;;(sgp :rt .05 :pm t :er t :egs 1 :blc 2 :pmt t :ans .1 :mp 10 :pas .1)
;;(sgp :rt .05 :pm t :er t :egs 1 :blc 2 :ans .1 :mp 10 :pas .1)


(chunk-type start referent)
(chunk-type referent image count)
(chunk-type nonreferent part count)
(chunk-type part image bit)
(chunk-type answer image)

(add-dm
 (start1 isa start referent referent1)
 (referent1 isa referent image source count 2)
 (partsAnswer isa part image source bit answer)
 (partsMirror isa part image source bit mirror)
 (partsDifferent isa part image source bit different)
 (answer isa chunk)
 (mirror isa chunk)
 (different isa chunk)
 (source isa chunk))


;; first production looks at the referent, then sets the start variable

"
(P start
"
  =goal>
"
     isa          start
"
     referent     =referent
"
  =referent>
"
     isa          referent
==>
  !eval!          (setf *start* t)

  !push!          =referent

)

;; This is the manipulate function that sets the similarities function

"
(P manipulate
"
   =goal>
"
      isa        referent
"
```

```
      count       =count
"
==>
   !eval!         (manipulate-similarity =count)
   =goal>
      count       nil
)

;; once we are done, this fires when we are done manipulating, we pop the top goal off
;; the stack

"
(P done-manipulate
"
   =goal>
"
      isa         referent
"
    count         nil
    image         =whole
   =fact1>
     isa          part
     image        =whole
     bit          answer
==>

  !eval! (set-answer  =fact1)

  !output!        (=fact1)

  !pop!
  !pop!

)

(goal-focus start1)
```

NO. OF COPIES ORGANIZATION

1 ADMINISTRATOR
DEFENSE TECHNICAL INFO CTR
ATTN DTIC OCP
8725 JOHN J KINGMAN RD STE 0944
FT BELVOIR VA 22060-6218

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL CS AS REC MGMT
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL CI LL TECH LIB
2800 POWDER MILL RD
ADELPHI MD 207830-1197

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL DD
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL D D R SMITH
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
ARMY AUDIOLOGY & SPEECH CTR
WALTER REED ARMY MED CTR
WASHINGTON DC 20307-5001

1 CODE 1142PS
OFFICE OF NAVAL RESEARCH
800 N QUINCY STREET
ARLINGTON VA 22217-5000

1 WALTER REED ARMY INST OF RSCH
ATTN SGRD UWI C (COL REDMOND)
WASHINGTON DC 20307-5100

1 COMMANDER
USA AEROMEDICAL RSCH LAB
ATTN LIBRARY
FORT RUCKER AL 36362-5292

NO. OF COPIES ORGANIZATION

1 CHIEF
ARMY RESEARCH INSTITUTE
AVIATION R&D ACTIVITY
ATTN PERI IR
FORT RUCKER AL 36362-5354

1 AIR FORCE FLIGHT DYNAMICS LAB
ATTN AFWAL/FIES/SURVIAC
WRIGHT PATTERSON AFB OH 45433

1 US ARMY NATICK RD&E CTR
ATTN STRNC YBA
NATICK MA 01760-5020

1 STRICOM
12350 RESEARCH PARKWAY
ORLANDO FL 32826-3276

1 GOVT PUBLICATIONS LIBRARY
409 WILSON M
UNIVERSITY OF MINNESOTA
MINNEAPOLIS MN 55455

1 DR RICHARD PEW
BBN SYSTEMS AND TECH CORP
10 MOULTON STREET
CAMBRIDGE MA 02138

1 DR ANTHONY DEBONS
IDIS UNIV OF PITTSBURGH
PITTSBURGH PA 15260

1 DR NANCY ANDERSON
DEPT OF PSYCHOLOGY
UNIVERSITY OF MARYLAND
COLLEGE PARK MD 20742

1 MR WALT TRUSZKOWSKI
NASA/GODDARD SPACE FLIGHT CTR
CODE 588.0
GREENBELT MD 20771

1 DR NORMAN BADLER
DEPT OF COMPUTER & INFO
SCIENCE
UNIV OF PENNSYLVANIA
PHILADELPHIA PA 19104-6389

1 HQDA (DAPE ZXO)
ATTN DR FISCHL
WASHINGTON DC 20310-0300

NO. OF COPIES ORGANIZATION

1 COMMANDER
US ARMY AVIATION CENTER
ATTN ATZQ CDM S
(MR MCCRACKEN)
FT RUCKER AL 36362-5163

1 COMMANDER
US ARMY SIGNAL CTR &
FT GORDON
ATTN ATZH CDM
FT GORDON GA 30905-5090

1 DIRECTOR
US ARMY AEROFLIGHT DYNAMICS
MAIL STOP 239-9
NASA AMES RESEARCH CENTER
MOFFETT FIELD CA 94035-1000

1 DR SEHCHANG HAH
DEPT OF BEHAVIORAL SCIENCES
& LEADERSHIP
BUILDING 601 ROOM 281
US MILITARY ACADEMY
WEST POINT NY 10996-1784

1 US ARMY RESEARCH INSTITUTE
ATTN PERI IK (D L FINLEY)
2423 MORANDE STREET
FORT KNOX KY 40121-5620

1 US MILITARY ACADEMY
MATHEMATICAL SCIENCES CTR
OF EXCELLENCE
DEPT OF MATH SCIENCES
ATTN MDN A MAJ M D PHILLIPS
THAYER HALL
WEST POINT NY 10996-1786

1 CECOM
ATTN PM GPS COL S YOUNG
FT MONMOUTH NJ 07703

1 CECOM
SP & TERRESTRIAL COMMCTN DIV
ATTN AMSEL RD ST MC M
H SOICHER
FT MONMOUTH NJ 07703-5203

1 US ARMY INFO SYS ENGRG CMND
ATTN ASQB OTD F JENIA
FT HUACHUCA AZ 85613-5300

NO. OF COPIES ORGANIZATION

1 US ARMY RESEARCH OFC
4300 S MIAMI BLVD
RSCH TRIANGLE PK NC 27709

1 US ARMY SIMULATION TRAIN
& INSTRMNTN CMD
ATTN J STAHL
12350 RESEARCH PARKWAY
ORLANDO FL 32826-3726

1 DARPA
3701 N FAIRFAX DR
ARLINGTON VA 22203-1714

1 UNIV OF TEXAS
1 HICKS & ASSOCIATES, INC.
ATTN G SINGLEY III
1710 GOODRICH DR STE 1300
MCLEAN VA 22102

1 ARL HRED AVNC FLD ELMT
ATTN AMSRL HR MJ
(R ARMSTRONG)
PO BOX 620716 BLDG 514
FT RUCKER AL 36362-0716

1 ARL HRED AMCOM FLD ELMT
ATTN AMSRL HR MI (D FRANCIS)
BUILDING 5678 ROOM S13
REDSTONE ARSENAL AL 35898-5000

1 ARL HRED AMCOM FLD ELMT
ATTN ATTN AMSRL HR MO (T COOK)
BLDG 5400 RM C242
REDSTONE ARS AL 35898-7290

1 ARL HRED USAADASCH FLD ELMT
ATTN AMSRL HR ME (K REYNOLDS)
ATTN ATSA CD
5800 CARTER ROAD
FORT BLISS TX 79916-3802

1 ARL HRED ARDEC FLD ELMT
ATTN AMSRL HR MG (R SPINE)
BUILDING 333
PICATINNY ARSENAL NJ 07806-5000

1 ARL HRED ARMC FLD ELMT
ATTN AMSRL HR MH (C BIRD)
BLDG 1002 ROOM 206B
FT KNOX KY 40121

NO. OF
COPIES ORGANIZATION

1 ARL HRED CECOM FLD ELMT
ATTN AMSRL HR ML (J MARTIN)
MYER CENTER RM 2D311
FT MONMOUTH NJ 07703-5630

1 ARL HRED FT BELVOIR FLD ELMT
ATTN AMSRL HR MK (P SCHOOL)
10170 BEACH ROAD ROOM 12
FORT BELVOIR VA 22060-5800

1 ARL HRED FT HOOD FLD ELMT
ATTN AMSRL HR MV HQ USAOTC
(E SMOOTZ)
91012 STATION AVE ROOM 111
FT HOOD TX 76544-5073

1 ARL HRED FT HUACHUCA FLD ELMT
ATTN AMSRL HR MY (B KNAPP)
GREELY HALL (BLDG 61801 RM 2631)
FORT HUACHUCA AZ 85613-5000

1 ARL HRED FLW FLD ELMT
ATTN AMSRL HR MZ (A DAVISON)*
3200 ENGINEER LOOP STE 166
FT LEONARD WOOD MO 65473-8929

1 ARL HRED NATICK FLD ELMT
ATTN AMSRL HR MQ (M R FLETCHER)
NATICK SOLDIER CTR BLDG 3 RM 341
AMSSB RSS E
NATICK MA 01760-5020

1 ARL HRED OPTEC FLD ELMT
ATTN AMSRL HR MR (M HOWELL)
ATEC CSTE OM
PARK CENTER IV RM 1040
4501 FORD AVENUE
ALEXANDRIA VA 22302-1458

1 ARL HRED SC&FG FLD ELMT
ATTN AMSRL HR MS (C MANASCO)
SIGNAL TOWERS RM 303A
FORT GORDON GA 30905-5233

1 ARL HRED STRICOM FLD ELMT
ATTN AMSRL HR MT (A GALBAVY)
12350 RESEARCH PARKWAY
ORLANDO FL 32826-3276

1 ARL HRED TACOM FLD ELMT
ATTN AMSRL HR MU (M SINGAPORE)
BLDG 200A 2ND FLOOR
WARREN MI 48397-5000

NO. OF
COPIES ORGANIZATION

1 ARL HRED USAFAS FLD ELMT
ATTN AMSRL HR MF (L PIERCE)
BLDG 3040 RM 220
FORT SILL OK 73503-5600

1 ARL HRED USAIC FLD ELMT
ATTN AMSRL HR MW (E REDDEN)
BLDG 4 ROOM 332
FT BENNING GA 31905-5400

1 ARL HRED USASOC FLD ELMT
ATTN AMSRL HR MN (F MALKIN)
HQ USASOC BLDG E2929
FORT BRAGG NC 28310-5000

1 ARL HRED HFID FLD ELMT
ATTN AMSRL HR MP
DR A KARRASCH
C/O BATTLE CMD BATTLE LAB
415 SHERMAN AVE UNIT 3
FORT LEAVENWORTH KS 66027-2300

ABERDEEN PROVING GROUND

2 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL CI LP (TECH LIB)
BLDG 305 APG AA

1 LIBRARY
ARL BLDG 459
APG-AA

1 ARL HRED ECBC FLD ELMT
ATTN AMSRL HR MM (R MCMAHON)
BLDG 459
APG-AA

ABSTRACT ONLY

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRL CS EA TP TECH PUB BR
2800 POWDER MILL RD
ADELPHI MD 20783-1197

INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 2000	Final

4. TITLE AND SUBTITLE

Developing an ACT-R Model of Mental Manipulation

5. FUNDING NUMBERS

AMS: 611102.74A0011
PR: 1L161102B74A
PE: 6.11.02

6. AUTHOR(S)

Kelley, T.D.; Wiley, P.W. (both of ARL); Lee, F.J. (CMU)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
Human Research & Engineering Directorate
Aberdeen Proving Ground, MD 21005-5425

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
Human Research & Engineering Directorate
Aberdeen Proving Ground, MD 21005-5425

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ARL-TR-2179

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In order to better understand the use of cognitive architectures within the Human Research and Engineering Directorate of the U.S. Army Research Laboratory, and with the intent of supporting the Land Warrior program, members of the Manned Systems Design Methods Team of the Integration Methods Branch developed an adaptive control of thought - rational (ACT-R) (Anderson & Lebiere, 1998) model of mental manipulation. The goal of the model was to reproduce errors made during mental manipulation, which were similar in type and number to those made by actual soldiers who took a paper-and-pencil test of mental manipulation as part of a cognitive assessment battery. The final model was able to replicate errors made by the soldiers in the mental manipulation test; however, further work and data collection are needed to make the model predictive of the types of errors soldiers could make if they were exposed to specific types of mental manipulation problems. Lessons learned from this initial application of ACT-R are also discussed.

14. SUBJECT TERMS

ACT-R
mental rotation
cognitive modeling

15. NUMBER OF PAGES
65

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	