Error Modeling in the ACT-R Production System

Christian Lebière
Department of Psychology
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
cl+@cmu.edu

John R. Anderson
Department of Psychology
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
ja+@cmu.edu

Lynne M. Reder
Department of Psychology
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
reder+@cmu.edu

Abstract

We describe how to extend the ACT-R production system to model human errors in the performance of a high-level cognitive task: to solve simple linear algebra problems while memorizing a digit span. Errors of omission are produced by introducing a cutoff on the latency of memory retrievals. If a memory chunk cannot gather enough activation to be retrieved before the threshold is reached, retrieval fails. Adding Gaussian noise to chunk activation produces a pattern quantitatively similar to subject errors. Errors of commission are introduced by allowing imperfect matching in the condition side of productions. The wrong memory chunk can be retrieved if its activation is large enough to allow it to overcome the mismatch penalty. This mechanism provides a qualitative and quantitative fit to subject errors. In conclusion, this paper demonstrates that human-like errors, sometimes thought of as the exclusive domain of connectionist models, can be successfully duplicated in production system models.

Introduction

ACT-R (Anderson, 1993) is a model of human cognition which assumes that a production system operates on a declarative memory. It is a successor to previous production system models (Anderson, 1976, 1983) and continues the emphasis on activation-based processes as the mechanism for relating the production system to the declarative memory. Different declarative memories have different levels of activation which determine their rate and probability of processing by the production rules. ACT-R is distinguished from the prior ACT theories in that the details of its design have been strongly guided by the rational analysis of Anderson (1990). Essentially it is a production system tuned to achieve optimal performance given the statistical structure of the environment.

Errors are a fundamental aspect of human cognition that symbolic systems have never been able to completely model. Symbolic systems have been able to model consistent errors by assuming that people have systematic bugs (Van Lehn, 1989) and errors of commission by assuming certain rules fail to apply. They have much more difficulty with the occasional slips or intrusions (Norman, 1981). These systems are sometimes thought of as too precise, too deterministic and too algorithmic to be able to exhibit the random, gradual degradation of performance exhibited by humans. A symbolic system works or it does not. When connectionist models became popular (Rumelhart and McClelland, 1986), one of their main attractions was that their holistic computation style could exhibit a capacity for human-like errors and graceful degradation of performance under noise or component failures. ACT-R is a hybrid system. Although its declarative elements are symbolic structures and its procedural rules implement an algorithmic matching process, the activation of the chunks is spread through a connection network. In this paper, we describe how to model errors with the ACT-R system by redefining (part of) the matching process as an activation-based constraint-satisfaction mechanism.

To be concrete, we will explain this with respect to the following task which is described more fully in Anderson, Reder, & Lebière (in preparation). Subjects were asked to memorize a digit span of 2, 4 or 6 digits, and then solve a linear equation before recalling the digits. The equation types are detailed in Table 1. Types (1) through (4) are the simple equations, solvable by a single transformation. Types (5) through (8) are the complex equations, which require two transformations.

Errors in the equation solving tasked increased either with the complexity of the equations or with memory load. Subjects solved 96.9% of the simple equations and 90.8% of the complex equations. They solved 95.2 % of the equations with a memory load of 2, 94.7% with a load of 4, and 91.7% with a load of 6. Both factors also impacted on the proportion of digit strings recalled correctly. After solving simple equations, subjects recalled 95.0% of the digit strings while they recalled 91.3% after solving complex equations. They recalled 95.5% of the two-element strings, 95.8% of the four-element strings, and 88.1% of the six-element strings. These factors were additive. For more details see Anderson et al (in preparation).

Table 1

x * a = b	(1)
x / a = b	(2)
a + x = b	(3)
a - x = b	(4)
x * m + a = b	(5)
x * m - a = b	(6)
x / m + a = b	(7)
x / m - a = b	(8)

Table 2

```
p solve_x/a=b
   =goal>
      isa   solve-equation
      lht   =term
      rht   =b
   =term>
      isa   term
      lho   x
      op    /
      rho   =rho
==>
   =product>
      isa   term
      lho   =b
      op    *
      rho   =rho
   =goal>
      lht   x
      rht   =product
```

```
p compute_a*b
   =goal>
      isa   solve-equation
      lht   x
      rht   =term
   =term>
      isa   term
      lho   =lho
      op    *
      rho   =rho
   =lho*rho>
      isa   times-fact
      arg1  =lho
      arg2  =rho
      product  =product
==>
   =goal>
      rht   =product
```

```
p output_digit
   =goal>
      isa   recall-span
      pointer  =index
   =item>
      isa   memory-span
      item  =value
      position  =index
      next  =next
==>
   =goal>
      pointer  =next
```

Basic Model

An ACT-R model of this task would contain three sets of productions: one to manipulate the equations, one to compute arithmetic solutions, and one to retrieve digits from the span. Simplified examples of each production set are included in Table 2. ACT-R productions are composed of the keyword '**p**', the name of the production, then a number of chunk retrievals (the left-hand side) separated by '==>' from a number of chunk actions (the right-hand side). Symbols starting with '=' are variables, others are constants. A chunk is composed of its name (usually a variable, followed by '>'), the keyword '*isa*', its type, then a number of slot-value pairs. In the lhs, slot values which are constants or previously bound variables are interpreted as constants, others as variables to be bound to the actual slot value. In the rhs, values are assigned to the slots. Previously bound chunks are modified, others are created.

The first chunk in the lhs of the production is the top-level goal. For *solve_x/a=b* and *compute_a*b*, that goal is to solve an equation. In *solve_x/a=b*, the left-hand side of the equation has to be a fraction with x as numerator. In the action part, a new term multiplying the denominator by the original right-hand term is created, and placed in the right-hand side of the equation while the left-hand side simplifies to x. *Solve_x/a=b* encodes the two-step process of multiplying each side by the denominator, then simplifying.[1]

*Compute_a*b* applies when x is isolated in the left-hand side of the equation and the right-hand side consists of a product of two numbers, such as after *solve_x/a=b* has fired. The actual product is then retrieved through an **indirect match**, so named because, unlike all previous matches, the chunk *=lho*rho* has not been previously bound but is retrieved from the content of its slots (i.e. slot *arg1* has value *=lho* and slot *arg2* has value *=rho*). In the action part of the production, the answer replaces the product term as solution of the equation.

Output-digit has a similar structure. Its goal is to retrieve the digit of position *=index* in the span, which it does through another indirect match. Each memory chunk contains three slots: its index, the value stored, and the index of the next memory in the sequence. The pointer in the top goal is then replaced by the index of the next memory.

Indirect retrievals, as occur in *Compute_a*b* and *Output-digit* prove to be critical to the ACT-R theory of performance on such tasks, particularly the theory of errors. They are the point at which information is retrieved from memory in the performance of the task. Our theory of errors will locate the problem in the failure of those retrievals.

Latency Threshold and Errors of Omission

Direct matches do not involve any search; they simply result in the expansion of a known chunk value. While there are often many ways to write a production system model using different data structures and productions (and therefore direct matches), indirect matches are more closely determined by the structure of the task. As the indirect matches are the critical elements to the predictions of the theory, ACT-R is really quite constrained in its predictions.

In ACT-R, every declarative chunk has associated with it an activation measure which is interpreted as the log odds that the chunk will be matched in the present context. The activation of a chunk is defined as the sum of the activations it receives from various source chunks which are in the focus of attention. The amount of activation received from a source is the product of the source activation times the strength of association

[1] As written, that rule is essentially hard-coded in the production, and a separate production is needed for each such rule. No long-term memory retrieval is needed. An alternative model is described in Table 4.

between them, called **Interactive Association (IA)**[2]. Thus, as an example, faced with the equation x=3*4, the chunk encoding 3*4=12 will receive activation to the degree that 3 and 4 are active as sources and to the degree they are strongly associated to the chunk.

The level of activation of a chunk determines not only its probability of retrieval but also the retrieval time. The latency is defined as inversely proportional to the exponential of the chunk activation (and the production strength). This means that direct retrievals, which gather high levels of activation, will take a relatively short time whereas indirect retrievals will take increasingly long as their activation level decreases. It seems natural to impose a cutoff time on how long a retrieval can take. If a chunk cannot gather enough activation, its retrieval will exceed the **latency threshold** and will fail. To introduce some randomness in this deterministic behavior, it is necessary to add some noise to the chunk activations. This Gaussian noise mechanism is a standard ACT-R feature. Increasing the noise or decreasing the activation threshold will increase the number of errors. At equal error frequencies, the former will increase the randomness of the errors while the latter will decrease it.

Finally, to ensure that this threshold applies equally in all situations, it is necessary to make one additional assumption regarding activation: that the total source level is constant. Anderson et al (in prep.) discuss the implications of this assumption and its relationship to the working memory limitations of Just & Carpenter (1992). Our model contains many sources of activation: the goal (as usual), the digits of the memory span, and the elements of the equation to solve. The total source level will be divided as follows: the source level of the goal is constant, and the rest is divided equally among span digits and equation elements. The more complex the equation and the longer the span, the lower the source level of their components.

Anderson et al (in prep.) show that this mechanism can yield the error pattern observed in the previous section. First, consider how it applies to the equation-solving task. The only indirect retrievals needed are for the arithmetic facts. A previous section reported that the error rate for simple equations is 3.1% and for complex equations 9.2%. One reason for this difference in error rates is that simple equations require one indirect retrieval of an arithmetic fact while complex equations require two. Another factor is that the extra components to the equation lower levels for all sources. Any given fact gathers less activation, and the probability that it fails to be retrieved increases. Similarly for the span effect on equation solutions: increasing the length of the span decreases the level of all sources, including the equation digits used to retrieve arithmetic facts.

[2]In the Bayesian framework of ACT-R, IA values can be interpreted as the log likelihood ratio measure of how much the presence of a chunk in the context will increase the probability that the other is needed. The exact equations and Bayesian estimation formula for this parameter can be found in section 4.2.2 of (Anderson, 1993).

Similar considerations apply to the span recall task. Equation complexity lowers the source level of span digits and increases the likelihood of misretrieval. Longer spans mean both lower source levels and more retrievals, which produces the larger drop-off from 4 to 6 digits than 2 to 4 digits. Finally, complexity and span effects are additive because increases in equation complexity and length of span both add to the total number of sources, which determines their source level.

Qualitative Error Patterns

The model as described simulates errors by failing to retrieve a chunk, thereby bringing the process (equation-solving or span recall) to a halt and failing to provide an answer. That is not, however, how most actual errors occur. Subjects sometimes give erroneous answers, but only rarely fail to give an answer or retrieve a full span.

Let us concentrate on errors in the equation-solving task. By far the most common error (125 or 61%) occurred in algebraic transformations (e.g. forgetting to invert a sign). The second-most common error category (17 or 8%) were arithmetic errors, where subjects retrieve the wrong fact in the addition or multiplication table. We found, like Siegler (1988), that close errors (e.g. 6*8=54) were generally more common than wild ones (e.g. 6*8=19). The remaining errors (63 or 31%) did not seem to fall into any particular pattern. It is interesting to further examine the algebraic errors. The six transformation rules necessary to solve the problems in Table 1, with their most frequent errors and their frequency, are described in Table 3.

It seems at first that the subjects simply forget to invert the operator, but the pattern does not extend to rules (5) and (6). In fact, there is a clear pattern of pairwise interaction: the most common error for rule (1) is the answer for rule (2) and vice versa, and similarly for rule (3) and (4), and (5) and (6). In particular, it is interesting to note that although rule (1) and (5) have the same solution, their most frequent errors are different. Altogether, these errors account for three quarters of the total. This suggests that they are produced by a misretrieval of the transformation rule to apply. Also, the error rate for the addition and subtraction rules are equal, and much larger than the rate for the multiplication and division rules (by a factor of 5 for the best subjects and 10 for all subjects).

Table 3

#	Equation	Solution	Error	Frequency
(1)	x + a = b	x = b - a	x = b + a	46 / 56
(2)	x - a = b	x = b + a	x = b - a	20 / 23
(3)	x * a = b	x = b / a	x = b * a	3/ 4
(4)	x / a = b	x = b * a	x = b / a	7 / 9
(5)	a + x = b	x = b - a	x = a - b	10 / 14
(6)	a - x = b	x = a - b	x = b - a [3]	10 / 24

[3]The most common error among all subjects was in fact "x=-a-b", but it was skewed by a particularly bad subject. Among other subjects, "x=b-a" was the most frequent.

The errors in the digit span task also were largely errors of commission. Some of these errors were systematic. Subjects showed a tendency to recall a digit from a near position in the span producing a generalization effect (Nairne, 1992). Also, digits were not equally frequent and subjects showed a tendency to intrude the more frequent digits.

Mismatch Penalty and Errors of Commission

Allowing more active but only partially matching chunks to be retrieved instead of the correct ones can account for these errors of commission. Partial matching has received support, both empirically and computationally in recent work of Reder (Reder & Cleeremans, 1990; Reder & Kusbit, 1991; Reder & Ritter, 1992; Reder, Richards & Stroffolino, in prep.). We changed ACT-R's pattern-matcher so that rather than retrieving all combinations of chunks that matched the condition of a production, it selected the most active chunk. Chunks accumulate activation from sources in the environment. If there is a mismatch of a chunk to a production pattern, its activation level will be reduced by a mismatch penalty. A mismatch takes the form of a contradiction between what is required by a slot value in the pattern and the actual slot value of the matching chunk. For instance, the pattern may be looking for 3+4 and the chunk 3+5=8 will have 5 where 4 is required. The mismatch penalty is proportional to the degree of mismatch between the desired and actual slot values. The IA values between desired chunks and the actual chunks are interpreted as their degree of similarity.

Let us first see how this mechanism can account for the pattern of arithmetic errors. When retrieving the sum of 2 and 5, both numbers are made sources and contribute activation to the correct fact: 2+5=7. Many other facts also receive activation from these and other numbers and might even gather more activation than the desired fact because of the pattern of sources and the Gaussian noise in the activation values. To favor close matches, IA values between numbers are set to reflect their similarity.[4] Therefore, the penalty for 2+6=8 will be less than the penalty for 2+1=3, because 6 is more similar to 5 than 1. All things being equal, the former will be more active than the latter and will have a better chance of being retrieved (if a misretrieval occurs), which explains the predominance of close matches.

To model transformation errors, we need to switch from the model described in a previous section where each transformation rule was encoded by its own production and applied without indirect retrieval to one with only one production retrieving a different algebraic rule for each transformation. A simplified version of such a production is shown in the left-hand side of Table 4.

Essentially, the production retrieves by an indirect match the declarative rule which best matches the left-hand term of the equation, then constructs a new right-hand term using the new operator provided by the rule. Sample

Table 4

p transform	Chunks encoding rules:
=goal>	
isa solve-equation	x+a-rule
lht =term	isa rule
rht =b	lho x
=term>	op +
isa term	rho a
lho =lho	new-op -
op =op	
rho =rho	x-a-rule
=rule>	isa rule
isa rule	lho x
lho =lho	op -
op =op	rho a
rho =rho	new-op +
new-op =new-op	
==>	x*a-rule
=new-term>	isa rule
isa term	lho x
lho =b	op *
op =new-op	rho a
rho =rho	new-op /
=goal>	
lht =lho	...
rht =new-term	

chunks which encode the transformation rules in the right-hand side of Table 3 are shown. For instance, the first rule encodes that - inverts +.

The pairwise nature of the errors is a direct consequence of the structure of the rules. Rules (5) and (6) in Table 3 apply to equations where the constant is the first part of the left-hand term and the variable the second part, which is the opposite of the other rules. That double mismatch will limit confusion between them and the other rules. Rules (1) and (2) have connections from a set of operators (+ and -) disjoint from those for rules (3) and (4) (* and /), which means that a given operator will either activate one pair of rules or the other pair. In addition, mismatches within these pairs of rules will be smaller than between them, because the IA values between + and - (* and /) are larger, reflecting the fact that these operators often appear in the same context together. This explains the pairwise pattern of errors. The higher error frequency involving + and - can be attributed to a fan effect. Since these two operators are involved in twice as many rules, their IA strengths to these rules are reduced. The final activation of these rules will then be diminished, and the probability of error increased.

The position effect in the digit-span recall task can be explained in the same way as the arithmetic errors: since each memory is accessed by its index, small differences in position will carry a lesser penalty than large ones. The frequency effect results because digits accessed more often will have a higher base level activation (Anderson, 1993) which will uniformly increase their probability of recall. Finally, since this mechanism is also activation-driven, the quantitative considerations that were discussed

[4] Specifically, $IA(i,j) = \exp(-|i-j|)$.

regarding the latency threshold mechanism also apply here.

Conclusion

We have introduced two ways to model errors in the ACT-R production system. Errors of omission occur when a chunk present in long-term memory cannot gather enough activation to be retrieved before a fixed latency threshold. We have shown that this mechanism provides a very good quantitative fit to the data. Most of the actual subject errors, however, involved retrieving the wrong data or performing an incorrect operation. These errors of commission can occur if we replace the usual semantics of the pattern-matcher with a new constraint-satisfaction semantics where a matching pattern is not a series of succeed/fail tests but a set of constraints to be maximized in terms of the activation measure. This explains the qualitative as well as quantitative pattern of subject errors.

There are independent motivations for introducing a partial-matching mechanism into ACT-R beyond accounting for errors of commission. In many situations softer matching is required. Often objects that we are looking for are not exactly like their pattern specification. For instance, people will wear hats or grow beards and we still need to recognize their identity. One might argue that only in formal domains like mathematics does a single mismatch disqualify an item from being useful.

Error modeling provides an additional advantage. ACT-R is a very general system which allows for many ways to model a particular task. Having to match not only the subjects' rule-like behavior but also their occasional errors provides an additional constraint on the form of the model. In this case, for example, we had to switch from a production-based representation of algebraic transformation rules to a declarative chunk-based representation because the former model could not produce the fan effects necessary to account for the difference in error frequencies among arithmetic operations.

By using ACT-R's concept of activation, not just as a heuristic measure of the likelihood of a match, but as a measure of the match itself, we have shown that the occasional, gradual pattern of errors characteristic of human performance can be effectively modeled by production systems.

Acknowledgments

This work was supported by Office of Naval Research grant N00014-90-J-1489 to John R. Anderson and National Science Foundation grant BNS-8908030 to Lynne M. Reder.

References

Anderson, J.R. (1976). *Language, Memory, and Thought.* Hillsdale, NJ: Erlbaum.

Anderson, J.R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

Anderson , J.R. (1990). *The Adaptive Character of Thought*. Hillsdale, NJ: Erlbaum.

Anderson, J.R. (1993). *Rules of the Mind*. Hillsdale, NJ: Erlbaum.

Anderson, J.R., Reder, L.M., & Lebière, C. (in preparation). *Working Memory and the ACT-R Theory.*

Just, M.A. & Carpenter, P.N. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, 99, 122-149.

Nairne, J.S. (1992). The loss of positional certainty in long-term memory. *Psychological Science*, 3, 199-202.

Norman, D. A. (1981). Categorization of action slips. *Psychological Review*, *88*, 1-15.

Reder, L.M., & Cleeremans, A. (1990). The role of partial matches in comprehension: The Moses illusion revisited. In A. Graesser & G. Bower,(Eds.), *The psychology of learning and motivation*, Vol. 25, New York: Academic Press, pp. 233-258.

Reder, L.M. & Kusbit, G.W. (1991). Locus of the Moses Illusion: Imperfect encoding, retrieval or match? *Journal of Memory and Language*, 30, 385-406.

Reder, L.M. & Ritter, F. (1992). What determines initial feeling of knowing? Familiarity with question terms, not with the answer. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 18, 435-451.

Reder, L. M., Richards, D. R., & Stroffolino, P. *Rapid feeling of knowing: an activation-based account.* (in preparation).

Rumelhart, D.E. & McClelland, J.L. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1. Cambridge, MA: MIT Press/Bradford Books.

Siegler, R.S. (1988). Strategy Choice Procedures and the Development of Multiplication Skill. *Journal of Experimental Psychology*, Vol. 117 No 3, pp. 258-275.

Van Lehn, K. (1989). Learning events in the acquisition of three skills. *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, 923-927.