인공 문법을 사용한 암묵 학습: EPAM IV를 사용한 모사*

Implicit Learning with Artificial Grammar : Simulations using EPAM IV

정 혜 선**
(Heisawn Jeong)

요 약 본 연구에서는 EPAM(Elementary Perceiver and Memorizer) IV를 사용하여 인공 문법이 사용된 암묵적 학습에서의 인간 수행을 모사하였다. 암묵 학습(implicit learning) 과제에서 참가자들은 인공 문법(artificial grammar)을 사용해 만들어진 '문법적' 문자열과 무선적으로 만들어진 '비문법적' 문자열을 학습하였는데, 이 때 비문법적 문자열보다 문법적 문자열의 학습이 더 우수하였다. 또한 참가자들은 이전에 본 적이 없었던 새로운 문자열에 대해서도 그 문법성을 판단할 수 있었다. 단순 기억 시스템인 EPAM IV에 항목 내 군집화(within-item chunking) 기능을 추가하여 암묵 학습 과제에서의 인간 수행을 모사한 결과, EPAM IV 또한 무선적인 문자열보다 문법적인 문자열을 보다 잘 학습하였고, 비문법적 문자열과 문법적 문자열을 구별할 수 있었다. 이러한 결과는 인공 문법을 사용한 암묵 학습 과제에서의 수행이 규칙 추상화보다는 군집화(chunking)에 근거한 재인 기억을 바탕으로 이루어짐을 시사한다.

Abstract In implicit learning tasks, human participants learn grammatical letter strings better than random letter strings. After learning grammatical letter strings, participants were able to judge the grammaticality of new letter strings that they have never seen before. EPAM (Elementary Perceiver and Memorizer) IV, a rote learner without any rule abstraction mechanism, was used to simulate these results. The results showed that EPAM IV with a within-item chunking function was able to learn grammatical letter strings better than random letter strings and discriminate grammatical letter strings from non-grammatical letter strings. The success of EPAM IV in simulating human performance strongly indicated that recognition memory based on chunking plays a critical role in implicit learning.

Key words Implicit learning, Artificial grammar, Simulation, EPAM IV

I. Introduction

People are sensitive to the regularities and patterns in the world. Research on implicit learning has tried to explain how people become sensitive to the underlying rules of the letter strings generated by artificial grammars. The key to answering this question lies in the form of knowledge acquired during implicit learning. Several researchers have proposed that recognition memory underlies human participants' performance in implicit learning tasks. For example, Miller [9] proposed that chunks are formed when participants memorize grammatical letter strings. Servan-Scriber and Anderson [18] also proposed that implicit learning involves a form of chunking and that grammatical judgments are performed based on a hierarchical

* This research was carried out in part with the financial support from Hallym University. The initial research would not have been possible without the help and encouragement from Howard Richman and Herbert Simon. For questions, please contact the author at the address below.
** Department of Psychology, Hallym University
Okchun-dong 1, ChunChon, Kangwon-do, 200-702
Korea (South).
Phone: 033-240-1377
Fax: 033-256-3424
e-mail: heis@hallym.ac.kr
연구세부분야: 인지과학; 인지심리학

network of chunks [see also 3, 10]. On the other hand, other researchers believed that some sort of rule abstraction occurs during implicit learning. For example, Reber [11] proposed that there exists an unconscious co-variation detection process that yields non-verbalizable, abstract, and grammatical knowledge. The process was likened to the language acquisition process during which abstract rules (i.e., grammars) are acquired through repeated exposure to linguistic stimuli. It was proposed that as children acquire grammar, people implicitly abstract the underlying rules of the letter strings when they are repeatedly exposed to them [7, 11, 12, & 14].

What is the form of knowledge acquired during implicit learning that allows people to discriminate grammatical from non-grammatical strings? Do human participants abstract rules, or do they simply store examples during learning? This question has been difficult to answer because even when people abstract rules, they often store information about the examples. In addition, even if people simply stored examples during implicit learning, rules can be abstracted based on the examples stored in memory when they were informed about the existence of the rules and asked to perform the grammatical judgment task. Separating the processes of rule abstraction and exemplar storage has been a difficult job because they are closely interconnected in human minds. This paper attempted to answer this question by simulating human implicit learning performance using EPAM (Elementary Perceiver And Memorizer) IV. EPAM IV is a pure rote learning system that does not have any rule abstraction mechanism. It only learns to recognize and to associate stimuli presented to it. If a rote learning system such as EPAM IV could simulate human performance in implicit learning tasks, it would suggest that implicit learning is based on the memory of the examples rather than the rules abstracted during learning. Conversely, if EPAM IV fails to simulate human performance, it would suggest that human participants' performance probably requires more than the stored memories of the examples.

II. Human Performance: Results from Reber (1967)

This paper reports on the EPAM IV's simulation of human performance reported in Reber [11]. He generated a set of letter strings using a finite-state grammar (see Figure 1). These letter strings were called grammatical letter strings and were of 6 to 8 letters in length. He also generated random letter strings that did not follow any rules, but consisted of the same letters used to generate grammatical letter strings (i.e., P, S, T, V, & X). The length of the random letter strings were made slightly shorter than the grammatical letter strings from 4 to 6 letters in length[1](see Table 1 for the grammatical letter strings and random letter strings used in this study). In Experiment 1, he asked participants to study grammatical letter strings in the experimental condition and random letter strings in the control condition. The letter strings were presented in seven sets of four letter strings in each condition. In the given set, each letter string was presented one by one for 5 seconds. Afterwards, participants were asked to write down all the letter strings in the set. They repeated this procedure until they reached a criterion of two consecutive correct reproductions of the set. Participants in both conditions showed a big initial drop in errors, but only participants in the experimental condition showed a consistent decline in errors, whereas the errors in the

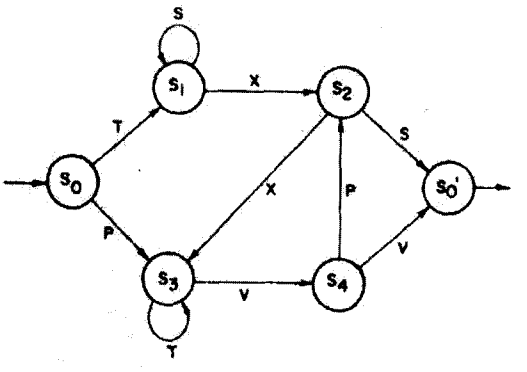

Figure 1. The finite state grammar used to generate grammatical letter strings in this study. It is identical to the one used in Reber except for the mappings of the letters.

1) Random letter strings contain more information than grammatical letter strings. This makes it more difficult to learn the random strings than grammatical strings. The length of the random strings was held to 4, 5, and 6 letters in order to make their information value equivalent to that of the grammatical letter strings. See Reber [10] and Miller [9] for more details on the information value of the grammatical and random strings.

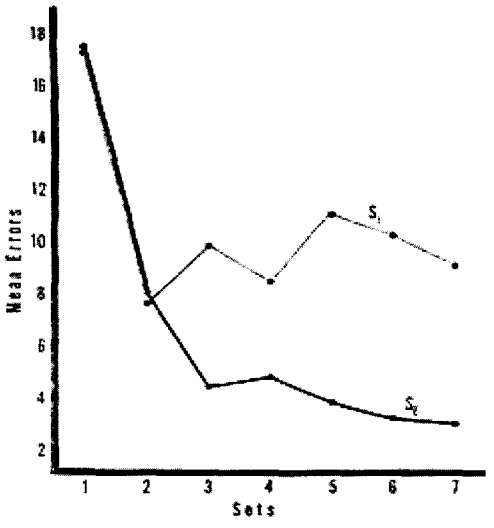

Figure 2. Mean number of errors to criterion on each of the seven learning sets in Experiment 1 of Reber (1967). S_g represents participants' performance in the experimental condition where they studied grammatical letter strings, and Sr represents participants' performance in the control condition where they studied random letter strings.

control group did not show such a consistent decline after the initial drop (see Figure 2).

In Experiment 2, Reber [11] first asked participants to study a set of grammatical letter strings as in Experiment 1. Afterwards, he told participants that the letter strings they just studied were formed by a set of grammatical rules. Although participants were informed about the existence of the rules, no further information was given about the rules themselves. Participants were merely told to consider the letter strings as grammatically permissible examples. Participants were then given a transfer task called grammaticality judgment task or well-formedness task, in which they were provided with a set of new letter strings and asked to judge whether these strings were grammatical (i.e., they are formed by the same rules used to create the study strings) or non-grammatical (i.e., they violate these rules)[2]. Participants judged

2) This task can be considered as a near transfer task. A far transfer task would be a task where human participant learn with strings expressed in one letter set and then make grammaticality judgments on items made up using a different letter set as was in Reber [12].

Table 1. Letter strings used in the simulation of learning

Grammatical letter strings		
PTTTVV	TSSSSXS	PVPXTVPS
PVPXVV	TSSXXVV	TSSSSSXS
TSSSXS	TSXXVPS	TSSSXXVV
TSXXVV	TSXXTVV	TSSXXTVV
TXXVPS	TXXTTVV	TSXXTVPS
PTTTTVV	TXXTVPS	TSXXTTVV
PTTTVPS	PTTTTVPS	TXXVPXVV
PTVPXVV	PTTVPXVV	TXXTTVPS
PVPXTVV	PTVPXTVV	TXXTTTVV
PVPXVPS		
Random letter strings		
VTSV	PXVPS	XPVXPT
TTVT	SSSSP	XSXSPS
TSVS	VVVPS	PXXPTV
TSXS	SXVPS	XVXSST
SSST	PXXPT	VXPTPT
TTPVS	PVVVP	VSPSXX
PSTPS	XTSTPS	XSXPST
SSPVS	SVTVPX	VVTPTT
TPSXT	PXPVXT	VVTXTV
SSVSV		

grammatical letter strings as grammatical 78% of the time and judged non-grammatical letter strings as grammatical 21% of the time. This means that they were able to discriminate grammatical letter strings from ungrammatical letter strings even though they were new letter strings that they had never seen before (see also Reber [13]; Reber & Lewis [15]). What was simulated in this study were these two results reported in Reber (1967): (a) better learning of the grammatical letter strings than random letter strings and (b) ability to discriminate grammatical letter strings from random letter strings.

III. Simulations with EPAM IV

EPAM is a system built on a theory of human perception and memory processes. It was first programmed for a computer by E. A. Feigenbaum in 1959 and has demonstrated an excellent fit with experimental data from a wide variety of psychological tasks. EPAM describes and explains a wide range of

human perceptual and memory processes. The phenomena surrounding verbal learning were selected and tried at first, but the program has been gradually extended to include other tasks. EPAM has been used successfully to account for the observed effects in paired-associate or serial anticipation verbal learning paradigms, speed of presentation of stimuli, inter-list and intra-list similarity, familiarization, one-trial versus multi-trial learning, and so forth. Although EPAM has been used to model mostly verbal learning in a standard experimental paradigm so far, in principle, it can be used to model a wide variety of phenomena such as categorization or conceptual learning tasks similar to implicit learning tasks examined in this article (see [4, 17] for more information on EPAM IV). Over the years, EPAM has undergone several revisions and has been progressively extended to new tasks and domains. The modifications made to EPAM over the years reflect the strategic learning that allows it to perform new tasks and did not significantly alter its basic mechanisms. The current version, EPAM IV, is a version extended to account for expert memory and was used to simulate human implicit learning performance in this study.

The EPAM theory was developed to identify a basic set of mechanisms that would give a unified account of diverse perceptual and memory phenomena. From EPAM'S first versions to the present one, the core of the EPAM's system consists of a small short-term memory storage that can hold a few familiar chunks of knowledge and a long-term semantic memory accessed from a structure called discrimination net. The discrimination net is a tree structure in which the top node in a net, the root node, is the ancestor of all the nodes below it. The top node has children, and its children have children, all the way down to the leaf nodes at the bottom of the net. Leaf nodes have no children of their own, but link the net to semantic memory, serving as an interface between the discrimination net and semantic memory. The successive nodes in the discrimination net contain tests on the values of the stimulus attributes that select the subsequent node to which it will be passed (e.g., 'Is this letter T?'). A leaf node contains no tests, but instead stores an image of the stimulus together with links to structures in semantic memory that contain additional information about it.

When EPAM[3] learns new items, it adds new nodes in the discrimination net. Both LTM and the discrimination net expand and are modified by learning processes, where the role of the discrimination nets is to provide a growing "index" that gives access to semantic memory in LTM. EPAM is capable of two kinds of learning: (a) it learns to recognize new stimuli and to discriminate among stimuli previously judged to be the same by adding new tests and branches to its discrimination net and (b) it stores new information about stimuli by elaborating the images at leaf nodes and the associative structure in LTM. When an object from the outside world is recognized, EPAM sorts it through the net using tests that are associated with each node, performing a series of tests till it reaches a leaf node.

III.a. Simulations of Learning

When EPAM IV studies a new stimulus, it builds discrimination nets. When the discrimination net for the given set of letter string is completed, its study routine returns to nil. Thus, in simulations using EPAM, it was assumed that EPAM IV studied the stimulus to the level equivalent to the learning criterion of two consecutive correct reproductions used in Reber (1967) when its study routine returns zero. In addition, EPAM records the number of routines and learning time to complete the net in the EPAM-clock. Thus, the number of routines and learning time in the EPAM IV-clock until EPAM IV completed the net on each set was used as a measure of learning. The results showed that with grammatical letter strings, EPAM IV took 89 seconds (8 study routines) to complete the net on the first set, but 63 sec (6 routines) on the second set. With random strings, EPAM IV took 63 sec (9 routines) on the first set, but 58 sec (6 routines) on the second set (see Figure 3). Thus, as was in human data, EPAM IV showed an initial improvement with both types of stimuli.

Although EPAM IV provided a decent approximation

3) Note that the term EPAM is used without version information, it is used in a generic sense and do not indicate a specific version of EPAM in this paper.

to human performance, its performance was not satisfactory. EPAM did not show much improvement after the initial drop. In addition, there was no overall advantage of grammatical letter strings over random letter strings: EPAM IV took more time and more study routines to learn the grammatical letter strings over seven sets (a total of 420 sec and 43 routines) than the random letter strings (a total of 378 sec and 41 routines). Thus, it seems that pure rote learning, as was implemented in EPAM IV at the time, was not enough to simulate the advantage that grammatical letter strings have over random letter strings.

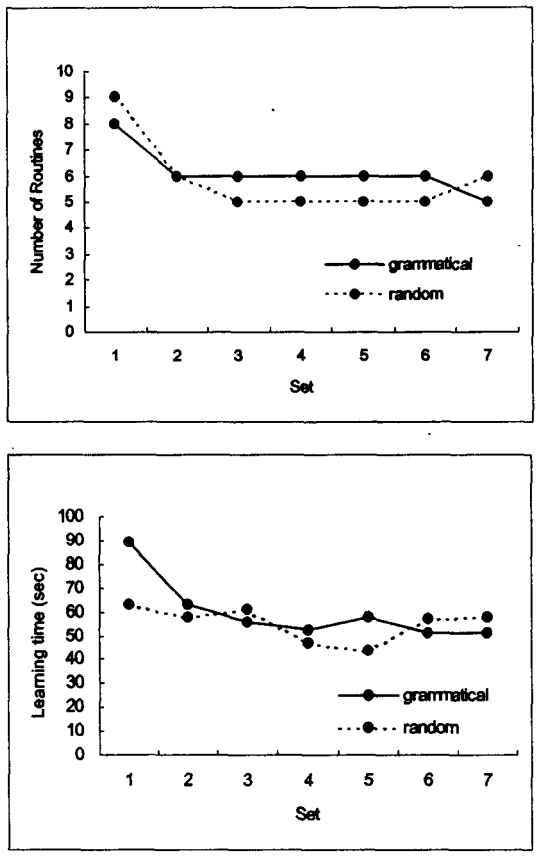

Figure 3. EPAM's study routine and learning time in simulation 1 without PARSE.

One of the key characteristics of human memory is chunking [1, 2]. The idea of chunking was first introduced by Miller [8]. Chunking is a grouping process by which nearby letter strings or numbers are grouped together. For example, the 26 letters of the alphabet are often encoded into seven chunks: abcd, efg, hijk, lmnop, qrst, uvw, and xyz [6]. Chunking is a natural (perhaps automatic) tendency to process stimuli by parts. Once chunks are formed, people treat them as if they were one item. The same thing may happen when people study letter strings used in implicit learning tasks. Although the grammar such as the one in Figure 1 is quite difficult to figure out, some of the patterns in the grammatical letter strings become noticeable after human participants are repeatedly exposed to the letter strings. People often report noticing repetition of letters (e.g., VVV or SSSSS) or some familiar sequence of letters in the string (e.g., VPS). Several researchers have pointed out the importance of such bigram and trigram frequencies [3, 10, & 18]. Although EPAM IV has a capacity to handle chunking through a LTM structure called retrieval structure, it does not have a mechanism to deal with this type of within-item chunking. The failure of EPAM IV to simulate human performance to a satisfactory level might be due to EPAM IV's lack of within-item chunking. If this were the case, a chunking procedure that identifies familiar letter sequences in the string would improve EPAM IV's performance.

To test this idea, a simple loop called PARSE was added (see Appendix). PARSE performs preliminary chunking, grouping adjacent identical letters in the string (e.g., VVV or SS). Once it chunks a sequence, EPAM IV puts it in a separate leaf node. When EPAM IV encounters the same sequence in another letter string later, EPAM does not need to build the image again. It only needs to point to the node where the chunk is stored. When PARSE was used with the same stimuli and procedures, EPAM's learning pattern changed. As was in the previous simulation, there was an initial improvement from set 1 to set 2 in both grammatical and random letter strings. The size of improvement, however, became bigger this time. In addition, EPAM IV showed a consistent improvement till the last study set with grammatical letter strings, but not with random letter strings (see Figure 4). As a result, the overall learning

time was shorter with the grammatical letter strings than with the random letter strings this time. In total, EPAM IV took 427 sec (43 routines) to learn the grammatical letter strings, but 455 sec (45 routines) to learn the random letter strings.

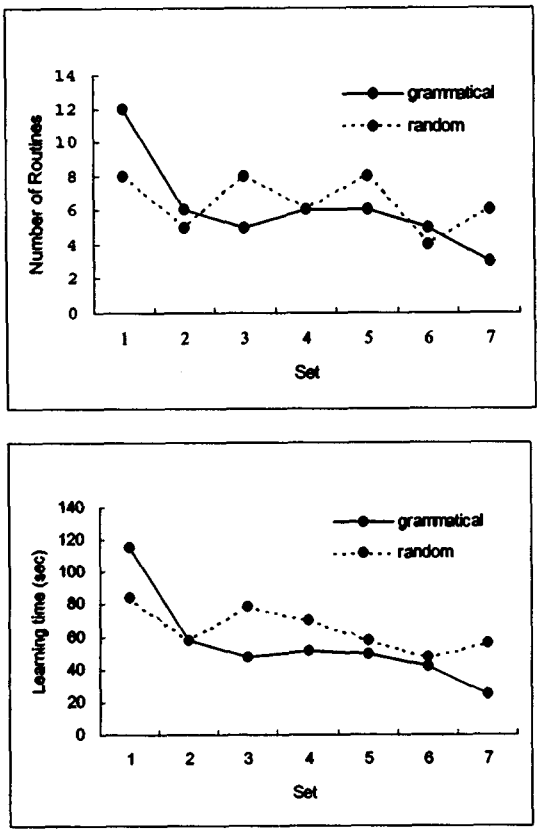

Figure 4. EPAM's study routine and learning time in simulation 1' with PARSE.

Although PARSE was quite primitive, it improved EPAM IV's performance substantially. EPAM IV was able to simulate Reber's [11] results that demonstrated that people learned grammatical letter strings better than random letter strings. Since EPAM IV operates on simple recognition memory with a primitive chunking procedure, the success of EPAM IV strongly support the hypothesis that people store examples rather than abstract rules during implicit learning.

III.b. Simulation of Grammaticality Judgment.

This section reports on EPAM IV's performance of the grammaticality judgment task. When EPAM recognizes something, it sorts stimuli through its discrimination net in order to gain access to the information about them that is stored in long-term semantic memory. When EPAM IV reaches the leaf node, it puts its content into a chunk box. The recognition process in EPAM is by no means perfect. EPAM IV can point a leaf node in response to a new test string, falsely recognizing it as an old item. In simulating grammaticality judgment task, a criterion of judging grammatical to test strings that pass all the tests and goes down to the leaf node and non-grammatical to the rest of the test strings was adopted. The frequency of which EPAM IV pointed a leaf node in response to new letter strings and the number of tests EPAM IV performed to recognize transfer letter strings were used as measures of recognition memory, because if a test item is more familiar to EPAM IV, EPAM IV performs more tests to recognize it.

After EPAM learned the study items in the same way as it did in the previous simulation with PARSE, EPAM was presented with test strings that were never studied before. Half of the test items were formed by the same grammar used to create the learning items, and the other half were constructed by taking 22 grammatical letter strings and replacing one letter with a letter that would make the string non-grammatical (The construction of non-grammatical letter strings followed the procedure in Reber & Lewis [15] because they provided more explicit criterion than Reber [11]; see Table 2). EPAM IV produced more leaf nodes in response to the grammatical test strings (12 leaf node out of 22 responses) than to the non-grammatical test strings (8 out of 22), demonstrating 60% of correct response rate. EPAM IV also performed more tests to recognize a test item when it was grammatical (4.27 test per an item) than when it was non-grammatical (3.96 test per an item). Although the size of the effect was not as large as in experimental studies that have shown a 70% and above correct response rate [11, 13], note that the actual outcome of the grammaticality judgment task depends on

Table 2. Letter strings used in the simulation of grammaticality judgment

Letter strings used in learning		
PVV	PVPXVV	TSXXVPS
TXS	TSSSXS	PTVPXTVV
TXSX	PTTTVPS	PVPXTVPS
PTVPS	PVPXVPS	TSSSSSXS
PTTTVV	TSSXXVV	TXXVPXVV
Grammatical test letter strings		
PTVV	PTVPXVV	PTTVPXVV
PVPS	PVPXTVV	TSSSXXVV
PTTVV	TSSSSXS	TSSXXTVV
TSSXS	TSXXTVV	TSXXTVPS
TXXVV	TXXTTVV	TSXXTTVV
TSXXVV	TXXTVPS	TXXTTVPS
TXXVPS	PTTTTVPS	TXXTTTVV
PTTTTVV		
Ungrammatical test letter strings[a]		
TPXS	TXXTVS	TSSSSXV
PTXVV	TVPXVPS	PTSXXVPS
PTVSS	PSXXTVV	TXPTTVPS
TTTVOS	PXPXTVV	TSXXPTVV
TPXXVV	PPTTVPS	TSSPSSXS
PTSVPS	TSSSPXS	PTTTTVXS
PTTVXS	PVPXTPS	PVPXTTVS
TSSPXS		

[a] Offending letter is underlined.

the decision criterion and strategies adopted. With a less stringent criterion, EPAM's performance would improve further. In addition, more important than the size of the effect is the existence of the effect in the right direction. The results of the simulation showed that it was possible to demonstrate the advantage of grammatical letter strings purely based on recognition memory, suggesting that recognition memory alone was sufficient to explain human participants' performance in a grammaticality judgment task.

V. Discussion

One of the major questions in implicit learning literature has been the form of knowledge learned during implicit learning, that is, whether rules are abstracted or whether examples (or fragments of examples) are stored. It has been difficult to establish reliably whether human participants rely on complex rules or memory of examples during the grammaticality judgment task. Because EPAM IV is a rote learner that does not have any capability to abstract rules, it can serve as a useful tool in investigating whether any kind of rule abstraction is involved in implicit learning. In this study, with the help from the preliminary within-item chunking procedure that groups adjacent identical letter strings together, EPAM IV successfully simulated human performance both at learning and in the grammaticality judgment task. EPAM was able to study grammatical letter strings faster than random letter strings and to differentiate grammatical and non-grammatical test strings based on recognition memory. Thus, the success of EPAM IV, along with the simulation by Servan-Schreiber & Anderson [18], supports the idea that recognition memory based on chunking is sufficient to explain human performance in implicit learning tasks [3, 18]. This kind of result is in line with human experimental results that indicate that the stored memories of examples are involved in implicit learning, especially memory of the fragments or chunks [5, 16].

The success of EPAM IV in simulating human data does not constitute sufficient proof of the theoretical mechanisms behind it since a given set of data can be accounted for by a number of theories or models. However, this problem is inherent in any model or theory building. The results from this study and other similar work should be considered as converging evidence supporting memory based theories of implicit learning processes.

References

[1] Chase, W. G., & H. A. Simon (1973). Perception in chess. Cognitive Psychology, 4, 55-81.

[2] de Groot, A. D. (1965). Thought and choice in chess. Mouton: The Hague.

[3] Dulany, D.S., Carlson, R. A., & Dewey, G.1. (1984). A case of syntactical learning and judgment: How conscious and how abstract? Journal of Experimental Psychology: General, 113(4),

541-555.

[4] Feigenbaum, E., & Simon, H.A. (1984). EPAM-like models of recognition and learning. Cognitive Science, 8, 305-336.

[5] Johnstone, T., & Shanks, D. R. (2001). Abstractionist and processing accounts of implicit learning, Cognitive Psychology, 42, 61-112.

[6] Klahr, D., Chase, W. G., & Lovelace, E. A. (1983). Structure and process in alphabetic retrieval. Journal of Experimental Psychology: Learning, Memory, and Cognition, 9, 462-477.

[7] Manza, L., & Reber, A. S. (1997). Representing artificial grammars: Transfer across stimulus forms and modalities. In D. C. Berry (Ed.), How implicit is implicit learning? Oxford University Press: Oxford, England.

[8] Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 63, 81-97.

[9] Miller, G. A. (1958). Free recall of redundant strings of letters. Journal of Experimental Psychology, 56, 485-491.

[10] Perruchet, P., & Pacteau, C. (1990). Synthetic grammar learning: Implicit rule abstraction or explicit fragmentary knowledge? Journal of Experimental Psychology: General, 119(3), 264-275.

[11] Reber, A. S. (1967). Implicit learning of artificial grammar. Journal of Verbal Learning and Verbal Behavior, 77, 317-327.

[12] Reber, A. S. (1969). Transfer of syntactic structure in synthetic language. Journal of Experimental Psychology, 81, 115-119.

[13] Reber, A. S. (1976). Implicit learning of synthetic language: The role of instructional set. Journal of Experimental Psychology: Human Learning and Memory, 2, 88-94.

[14] Reber, A. S. (1989). Implicit learning and tacit knowledge. Journal of Experimental Psychology: General, 118(2), 219-235.

[15] Reber, A. S., & Lewis, S. (1977). Implicit learning: An analysis of the form and structure of a body of tacit knowledge. Cognition, 5, 333-361.

[16] Redington, M., & Chater, N. (1996). Transfer in artificial grammar learning: A reevaluation. Journal of Experimental Psychology: General, 125(2), 123-138.

[17] Richman, H. B., Staszewski, J. J., & Simon, H. A. (1995). Simulation of expert memory using EPAM IV. Psychological Review, 102(2). 305-330.

[18] Servan-Schreiber, E., & Anderson, J. R. (1990). Learning artificial grammar with competitive chunking. Journal of Experimental Psychology: Learning, Memory, and Cognition, 16(4), 592-608.

Appendix: PARSE

```
parse
(defun parse (object)
    "groups adjacent and same letter together (study (parse object) *net*)"
  (loop for tail on (cdr object)
     with type = (get-value object 'type)
     with new-object = nil
     for last-letter = nil then current-letter
     for current-letter = (first tail)
     if (eq current-letter last-letter)
     do (cond ((atom (car new-object))
               (let ((little-list (list nil (car new-object) current-letter)))
               (put-value little-list type 'type)
               (setf new-object (cons little-list (cdr new-object)))))
('else (setf new-object (cons (append (car new-object) (lost current-
                                                        letter))
                              (cdr new-object)))))
     else do (setf new-object (cons current-letter new-object))
     finally (return (cons (dar object) (reverse new-object)))))
```