On the Computation of the Topological Entropy of Artificial Grammars

John R. Vokey
University of Lethbridge
https://orcid.org/0000-0002-8451-3166

Randall K. Jamieson
University of Manitoba
https://orcid.org/0000-0003-2505-223X

Author Note

This work was supported in part by a Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grant to RKJ. Both authors authors read and approved the final version of the manuscript. The authors have no conflicts of interest with respect to their authorship or the publication of this article. We thank both Rik Warren and Esther Van den Bos for comments on earlier drafts of this manuscript.

Abstract

Based on a proposal by Warren and Schroeder (2015), we provide a simple algorithm and computer code in the `R` programming language for the computation of the topological entropy (TE) from transition matrices created from their "subscripted element technique" as a measure of the complexity of artificial grammars (Bollt & Jones, 2000). We include a set of conventions for the transcription of grammar directed graphs to transition matrices that should provide for the consistent computation of the TE of artificial grammars.

Keywords: complexity, artificial grammars, Topological Entropy

On the Computation of the Topological Entropy of Artificial Grammars

Bollt and Jones (2000) proposed an information-theoretic Topological Entropy (TE) as a measure of the complexity of artificial grammars (AGs). Schiff and Katan (2014) demonstrated that the TE of an AG was highly correlated with mean percent correct grammaticality judgements over 56 different experiments in the literature involving 10 different AGs, despite wide variation in the training and testing techniques and materials used. The higher the TE or the complexity of the AG, the lower the percent correct, falling to chance (50%) with the most complex AG (cf. Van den Bos & Poletiek, 2008, 2015). More complex AGs appear to be more difficult to learn, in line with common intuition. Thus, TE appears to be a robust measure of the complexity of AGs, and, as such, a potentially useful tool for exploring the role of complexity in artificial grammar learning (AGL).

According to Bollt and Jones (2000), the TE, h, of an AG, g, for which w_n is the number of unique strings of length n, is defined as (cf. Robinson, 1998)

$$h(g) = \lim_{n\to\infty} \frac{log_e w_n}{n} \tag{1}$$

Generally, w_n increases exponentially as the string length, n, increases. TE can be interpreted as a measure of the rate of that increase (Warren & Schroeder, 2015). It is premised on the notion that an AG that can produce more unique items than another AG is thereby in that sense more complex than that other AG. It is computed as (cf. Robinson, 1998)

$$h(g) = log_e(\lambda_1) \tag{2}$$

for which λ_1 is the largest eigenvalue of the smallest, memoryless, $N \times N$ topological transition matrix, A (a matrix consisting of entries of 0 and 1), from the directed graph of the AG that encodes all of the unique transitions possible for that AG (larger transition matrices provide no new information and, thus, will return the same largest eigenvalue).

In its definitional form, TE is difficult to compute (Bollt & Jones, 2000; Schiff &

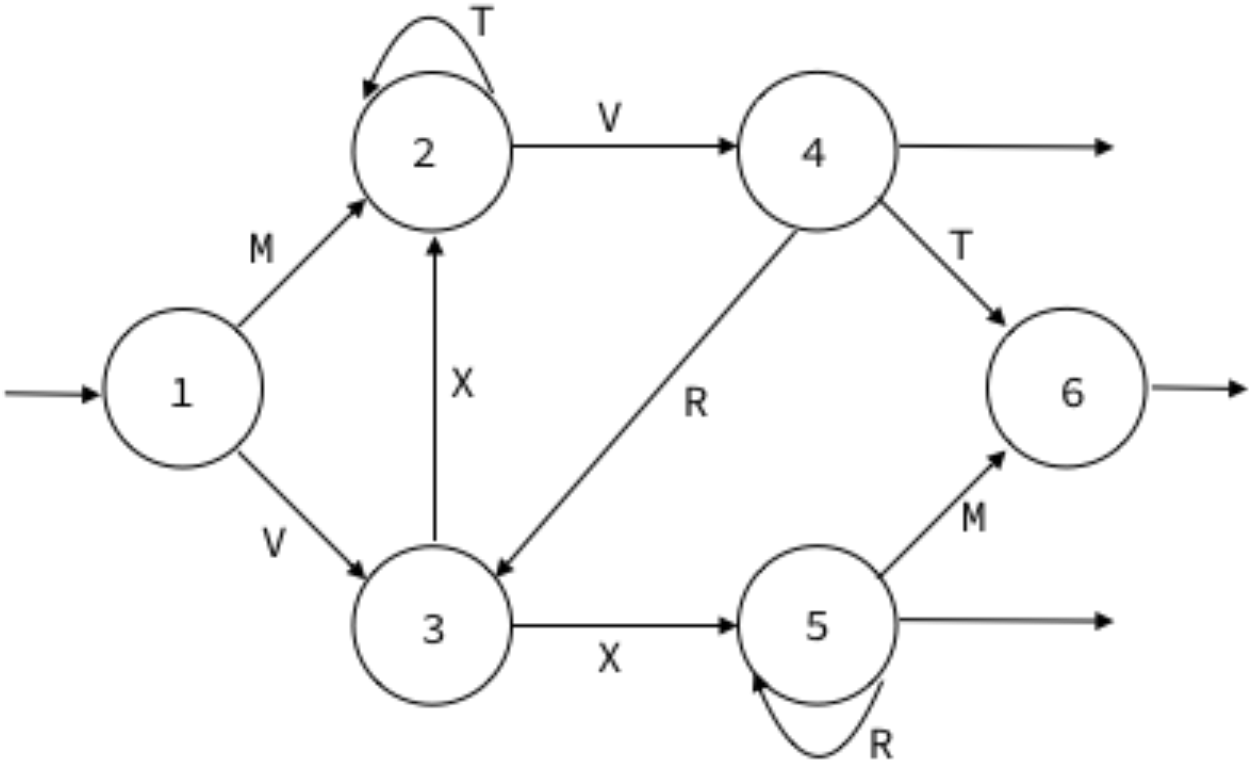

Figure 1. The directed graph of the artificial grammar from Reber and Allen (1978). The numbered nodes (or states) bracket transitions to other nodes generating letters or elements of a string. For example, transitioning from node 1 to node 2, node 2 to node 4, and node 4 to node 6, generates the string "MVT". Any such string that can be generated in this way is referred to as a grammatical string; any string that can't be generated in this way (e.g., "MRXVT") is referred to as a nongrammatical string.

Katan, 2014; Van den Bos & Poletiek, 2008, 2015; Warren & Schroeder, 2015). Bollt and Jones (2000) proposed their "lifted matrix" procedure to ease the construction of the transition matrix somewhat, and Schiff and Katan (2014) provided a script in Matlab that, along with the AGL StimSelect Matlab toolbox from Bailey and Pothos (2008), simplifies the procedure even more. Unfortunately, as discussed by Warren and Schroeder (2015), and documented in more detail subsequently, the process is still error-prone, resulting in inconsistent published TE values for the same nominal AGs.

Two problems

The problems appear to be two-fold, although both are related to the construction of the transition matrix for the grammar. First, there is the difficulty of translating the grammar *directed graph*, such as the one depicted in Figure 1 from Reber and Allen (1978) into the set of transitions from which the transition matrix will be constructed. As we will demonstrate, this problem appears to be more one of agreeing and adhering to a consistent set of conventions than any intrinsic set of difficulties. The second problem concerns the translation of the table of transitions into a transition matrix proper. For a grammar with k letters or elements, the "lifted matrix" method suggested by Bollt and Jones (2000) begins by populating a first-order $k \times k$ single-letter transition matrix (e.g., whether the letter 'X' is followed directly by the letter 'R', and so on). If that matrix does not encode all of the possible transitions of the AG, a second-order k^2 bigram $\times$ k^2 bigram transition matrix is populated, and so on until a transition matrix is obtained that captures every transition of the AG. The matrices become very large and sparse very quickly (with k letters, the first transition matrix is of order $k \times k$, for k^2 cells, the second $k^2 \times k^2$, for k^4 cells, and so on; fortunately, the matched rows and columns of the resulting matrix with all zero entries may be removed having no effect on the computation of the subsequent largest eigenvalue). The principal problem here is how one ascertains whether all and only all of the appropriate transitions have been captured in the transition matrix for the AG. The Matlab script of Schiff and Katan (2014) helpfully mechanises much of that process, but does not, unfortunately, guarantee that either of the two-fold issues raised here are resolved.

Warren and Schroeder (2015) solved the second problem in a simple way. They demonstrated that if every transition of the directed graph of an AG is labelled uniquely (rather than by the letter or element it generates, as in Bollt & Jones, 2000), then the natural logarithm of the maximum eigenvalue of the resulting transition matrix formed from that coding of the directed graph of the AG is in fact the TE of the AG as proposed by Bollt and Jones (2000). Warren and Schroeder (2015) refer to this method of construction of the

transition matrix as the "subscripted element technique". Transition matrices formed in this way are typically much smaller than those following the lifted matrix approach of Bollt and Jones (2000). Shown in Listing 1 is the Warren and Schroeder (2015) technique to compute the TE of any AG from a set of such unique transitions, coded as the simple function `computeTE` in the `R` programming language. Accepting the approach of Warren and Schroeder (2015) and our simple functional implementation of the same as a solution to the second problem, we now focus on the first problem: how to translate a directed graph of an AG into an agreed-upon list of transitions. In the process, we present a set of conventions to alleviate the ambiguity of the transition coding. We assert that the set of transitions used should be part of the documentation of any AG in the literature.

We start with the Reber and Allen (1978) AG directed graph shown in Figure 1. We do so because in part the TE of this AG is not (much) in dispute, and the AG itself is canonical in the literature. We can use it, however, to exemplify the common translation problems. According to Warren and Schroeder (2015), to compute TE of an AG we need first to list each of the transitions of the directed graph of the AG as unique entities. Most of these are simple as they correspond directly to the arcs or arrows through the directed graph of the AG, and, indeed, correspond to what many refer to as the "rules" of the AG (e.g., Van den Bos & Poletiek, 2008, 2015). The computation of TE requires that the transition matrix of the AG be *recursive* such that unique strings of any length may be generated. Most AGs in the psychological literature embody this feature by virtue of having at least one arc or arrow loop back to at least one of the earlier nodes. Note, in the directed graph shown in Figure 1, the transitions node 2 to node 2, node 3 to node 2, node 4 to node 3, and node 5 to node 5 provide for the needed recursion. However, a few of the AGs used in the literature do not have any recursive transitions (see, e.g., Kinder & Lotz, 2009, for two examples). To ensure the required recursion, it has been recommended as a convention that all transitions that exit the AG be encoded as looping back to the start node of the AG (Bollt & Jones, 2000; Schiff & Katan, 2014; Warren & Schroeder, 2015). For example, for the directed graph

shown in Figure 1, the exit transitions emanating from nodes 4, 5, and 6, would be encoded as transitioning back to node 1.

The `computeTE` function shown in Listing 1 assumes that each node is encoded as a 2-digit number (so up to 99 numerical nodes may be encoded), with nodes numbering less than 10 encoded with a leading "0".[1] So, the transition from node 1 to node 2 would be encoded as "0102". Each transition code is then followed by a comma and the letter or element corresponding to that transition, using the underscore character to denote looping back to the first node. For example, the transition from node 1 to node 2 for the grammar directed graph AG in Figure 1 would be encoded as "0102,M", and the exit transition from the AG of node 4 would be encoded as "0401,_", capturing the looping back to the start of the AG. The complete list of of the 13 transitions that completely encapsulates this AG according to these conventions may be found in Table 1.

Warren and Schroeder (2015) describe the lifted matrix procedure of Bollt and Jones (2000) to create topological transition matrices as both "cumbersome and error prone" (p. 90), and they provide examples, ironically, of such errors they found in Bollt and Jones (2000). To take just one, Bollt and Jones (2000) compute the TE of the Reber and Allen (1978) AG shown in Figure 1 to be 0.7324, computed as the natural logarithm of the largest eigenvalue of a 47×47 reduced topological transition matrix from an original 125×125 (5^3) lifted matrix [i.e., in terms of the Bollt and Jones (2000) procedure, it takes a minimum of trigram elements to encode every transition of the Reber and Allen (1978) grammar]. According to Warren and Schroeder (2015), at least four of the transitions included in the transition matrix of Bollt and Jones (2000) are impossible for this AG, and another four legal transitions are missed entirely. Further details may be found in Appendix A of Warren and Schroeder (2015). The computed value of TE for this AG, as derived from the 13×13 topological transition matrix, obtained with the Warren and Schroeder (2015) method (and

[1] The codes of the nodes do not need to be numerical; they can be any two-character code of numbers or letters (or any ASCII characters) or any combination of them (e.g. 'C7'), increasing the number of nodes substantially. The numerical codes were used here because most AGs in the literature use numbered nodes.

the `computeTE` function in Listing 1) is 0.7608.

Although the Schiff and Katan (2014) Matlab script appears to reduce errors quite substantially in producing the topological transition matrix from the list of transitions translated from the directed graph of the AG, it doesn't correct for errors in the translation itself. The same is true for the Warren and Schroeder (2015) procedure. For example, while commenting on the TE values Van den Bos and Poletiek (2008) computed using the lifted matrix procedure for the Reber (1967), TE = 0.48, and the Reber and Allen (1978), TE = 1.52, AGs, Warren and Schroeder (2015) noted that both were in error relative to the values of 0.6931 and 0.7608, respectively, that they had computed using their procedure. Unfortunately, although our computations using the `computeTE` function and those of Schiff and Katan (2014) agree with TE value for the Reber and Allen (1978) AG, both groups agree that the TE for the Reber (1967) AG is 0.602, *not* the value of 0.6931 reported by Warren and Schroeder (2015). Given that we used the procedure of Warren and Schroeder (2015) to generate the transition matrix from the transition table from the translation of the directed graph of the Reber (1967) AG, the difference must be in the translation of the AG to the transition table—and it is. As shown in Table 1 for the Reber (1967) AG, we translated the directed graph of the AG to have 11 transitions, including two ways to exit with an "S" ("0406,S" and "0506,S"). The only way we can compute the value reported by Warren and Schroeder (2015) is to translate the two paths terminating in an "S" by combining them into one path that also loops back to the start node ("0501,S").

Two conventions

To do so is to violate two conventions of the translation that to this point have been implicit. As our computations of the TE values for various AGs (mostly) otherwise match those of both Warren and Schroeder (2015) and Schiff and Katan (2014), it would appear that all three groups translate the directed graphs of these AGs to transition tables in much the same way, including the use of the two conventions that we will now make explicit.

1. Any transition from a node that adds a symbol to the output string must terminate on another node that is *not* the initial node. If no such other node exists in the directed graph of the AG, one must be added.

2. Any transition from a node to the initial node must *not* add a functional symbol to the output string (the underscore used here is simply a flag that no symbol is to be added to the output string).

Both of these conventions are exemplified in each of the translations of the AGs shown in Tables 1 and 2. These conventions are just that, conventions; we could just as easily agree to hold as conventions their exact opposites, including the convention to loop back to node 1 as most AGs in the literature already exhibit some degree of the required recursion. The point is that we need agreed-upon conventions to achieve consistent computation of TE values. Our computations of TE match those of Schiff and Katan (2014) for each of the 10 AGs they used, except one. For the Conway and Christiansen (2006) AG, we compute a TE of 0.6562, and can match the TE of 0.716 computed by Schiff and Katan (2014) only by violating both conventions. Instead of the 12 transitions for this AG shown in Table 1 terminating in the last two as "0407,M" and "0701,_", we can get the TE of Schiff and Katan (2014) only by collapsing these two transitions into one: "0401,M".

Shown in Table 2 are the transition tables as per the conventions and our computed TE values for the 10 AGs used by Van den Bos and Poletiek (2008). We were able to match only two (AG A and AG E) of the TE values they reported, and the differences for some of the non-matches were often substantial: for AG J, for example, Van den Bos and Poletiek (2008) reported a TE = 2.5761, whereas we computed it as 0.8587. Warren and Schroeder (2015) computed TEs for both AG A and AG B, and matched neither of the values computed by either Van den Bos and Poletiek (2008) or us. We were able to reproduce the two values computed by Warren and Schroeder (2015) by, again, violating the two conventions. Van den Bos and Poletiek (2015) used AG E and AG D as, respectively, their simple and complex AGs to investigate the role of complexity (TE) in various AGL tasks. By their calculations,

the simple AG E had a TE of 0.7131, and the complex AG D had a TE of 2.0496. However, by our transition tables and calculations, although we concur with the TE computed for AG E, we found a substantially smaller TE of 0.6823 for the allegedly more complex AG D. That is, by our calculations, the labels of the two AGs are reversed, and they differ very little in complexity. However, these differences are now moot. After contacting Van den Bos to point out these errors, Van den Bos (personal communication, May 10, 2019) noted that the transition tables used by Van den Bos and Poletiek (2008) contained unnecessary transitions; Van den Bos (personal communication, May 10, 2019) eliminated these transitions, recomputed the TE values for the 10 grammars of Van den Bos and Poletiek (2008), and now concurs with the TE values shown in Table 2 (Van den Bos & Poletiek, 2019).

Conclusion

Topological Entropy of Bollt and Jones (2000) provides for an overall measure of AG complexity that is substantially and robustly correlated with AGL performance (Schiff & Katan, 2014). On that basis, it could prove to be an important and powerful tool in the study of the role of complexity in AGL. Unfortunately, its computation to this point has been "cumbersome and error-prone" (Warren & Schroeder, 2015). With the development of the much simpler procedure of Warren and Schroeder (2015), our rendering of it as a simple function in `R`, and the adoption of a few conventions for the translation of the AG directed graphs to transition tables (which should make it easier to catch our errors), these problems should be ameliorated. We recommend that the conventions used and the transition tables of the directed graphs of the AGs created to produce the transition matrices be part of the documentation of any AG in the literature.

```
computeTE <- function(grammar){
  # computeTE
  # Based on the suggestion of Warren & Schroeder (2015)

  # John R. Vokey & Randall K. Jamieson, June 27, 2016

  # sample coded grammar from Reber & Allen (1978):

  # RebAll1978 <- c("0102,M","0103,V","0202,T","0204,V",
  #                 "0305,X","0302,X","0403,R","0406,T",
  #                 "0506,M","0505,R","0401,_","0501,_",
  #                 "0601,_")
  #'_' means return to the start node of the grammar

  # computeTE(RebAll1978) should return a TE of .761 for
  # the sample grammar

  n <- length(grammar) # assumes grammar is n lines of text
  tranMat <- matrix(0,n,n) # create the transition matrix
  for (theLine in 1:n){
    tranMat[theLine,] <- tranMat[theLine,]+
      (substr(grammar[theLine],3,4)== substr(grammar,1,2))
  }
  e <- Re(eigen(tranMat,only.values=TRUE)$values[1])
  return(list(TE=log(e),eig1=e,tranMat=tranMat))
}
```

Listing 1. The function `computeTE` in the `R` programming language to compute the topological entropy (TE) of AGs of Bollt and Jones (2000). It takes as input a vector of comma-delimited coded transitions of an AG and returns a list of three objects: the topological entropy (TE) and the transition matrix of that AG (tranMat), and the largest eigenvalue of that transition matrix (eig1). For example, the function call in `R` of `t <- computeTE(RebAll1978)` will return the TE of the grammar in `t$TE` and the transition matrix in `t$tranMat`.

References

Bailey, T. M., & Pothos, E. M. (2008). AGL StimSelect: Software for automated selection of stimuli for artificial grammar learning. *Behavior Research Methods*, *40*, 164–176. doi: 10.3758/BRM.40.1.164

Bollt, E. M., & Jones, M. A. (2000). The complexity of artificial grammars. *Nonlinear Dynamics, Psychology, and Life Sciences*, *4*(2), 153–168. doi: https://doi.org/10.1023/A:1009524428448

Brooks, L. R., & Vokey, J. R. (1991). Abstract analogies and abstracted grammars: Comments on Reber (1989) and Mathews et al. (1989). *Journal of Experimental Psychology: General*, *120*, 316–323. doi: https://doi.org/10.1037/0096-3445.120.3.316

Conway, C. M., & Christiansen, M. H. (2006). Statistical learning within and between modalities: Pitting abstract against stimulus-specific representations. *Psychological science*, *17*(10), 905–912. doi: https://doi.org/10.1111/j.1467-9280.2006.01801.x

Kinder, A., & Lotz, A. (2009). Connectionist models of artificial grammar learning: what type of knowledge is acquired? *Psychological Research PRPF*, *73*(5), 659–673. doi: https://doi.org/10.1007/s00426-008-0177-z

Knowlton, B. J., & Squire, L. R. (1996). Artificial grammar learning depends on implicit acquisition of both abstract and exemplar-specific information. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *22*(1), 169.

Mathews, R. C., Buss, R. R., Stanley, W. B., Blanchard-Fields, F., Cho, J. R., & Druhan, B. (1989). The role of implicit and explicit processes in learning from examples: A synergistic effect. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *15*, 1083–1100. doi: https://doi.org/10.1037/0278-7393.15.6.1083

Meulemans, T., & Van der Linden, M. (1997). Associative chunk strength in artificial grammar learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *23*, 1007–1028. doi: https://doi.org/10.1037/0278-7393.23.4.1007

Reber, A. S. (1967). Implicit learning of artificial grammars. *Journal of Verbal Learning and*

Verbal Behaviour, *5*, 855–863. doi: https://doi.org/10.1016/s0022-5371(67)80149-x

Reber, A. S. (1969). Transfer of syntactic structure in synthetic languages. *Journal of Experimental Psychology*, *81*, 115–119. doi: https://doi.org/10.1037/h0027454

Reber, A. S., & Allen, R. (1978). Analogy and abstraction strategies in synthetic grammar learning: A functionalist interpretation. *Cognition*, *6*, 189–221. doi: https://doi.org/10.1016/0010-0277(78)90013-6

Robinson, C. (1998). *Dynamical systems: stability, symbolic dynamics, and chaos*. CRC press. doi: https://doi.org/10.1112/s0024609397343616

Schiff, R., & Katan, P. (2014). Does complexity matter? Meta-analysis of learner performance in artificial grammar tasks. *Frontiers in Psychology*, *5*, 1–10. doi: https://doi.org/10.3389/fpsyg.2014.01084

Skosnik, P., Mirza, F., Gitelman, D., Parrish, T. B., Mesulam, M.-M., & Reber, P. J. (2002). Neural correlates of artificial grammar learning. *Neuroimage*, *17*(3), 1306–1314. doi: https://doi.org/10.1006/nimg.2002.1291

Van den Bos, E., & Poletiek, F. H. (2008). Effects of grammar complexity on artificial grammar learning. *Memory & Cognition*, *36*, 1122–1131. doi: https://doi.org/10.3758/mc.36.6.1122

Van den Bos, E., & Poletiek, F. H. (2015). Learning simple and complex artificial grammars in the presence of a semantic reference field: Effects on performance and awareness. *Frontiers in Psychology*, *6*, 1–7. doi: https://doi.org/10.3389/fpsyg.2015.00158

Van den Bos, E., & Poletiek, F. H. (2019, Nov 01). Correction to: Effects of grammar complexity on artificial grammar learning. *Memory & Cognition*, *47*(8), 1619–1620. Retrieved from `https://doi.org/10.3758/s13421-019-00946-0` doi: 10.3758/s13421-019-00946-0

Warren, R., & Schroeder, P. J. (2015). *Topological entropy measure of artificial grammar complexity for use in designing experiments on human performance in intelligence, surveillance, and reconnaissance (ISR) tasks* (Tech. Rep.). Air Force Research

Laboratory 711th Human Performance Wing Human Effectiveness Directorate Wright-Patterson Air Force Base, Oh 45433 Air Force Materiel Command United States Air Force.

Witt, A., & Vinter, A. (2012). Artificial grammar learning in children: abstraction of rules or sensitivity to perceptual features? *Psychological research*, *76*(1), 97–110. doi: https://doi.org/10.1007/s00426-011-0328-5

Table 1
Transition tables for each of the 10 AGs from Schiff & Katan (2014) and their corresponding TE values [in square brackets] as computed from the `computeTE` *function in Figure 1.*

Reber (1969) [0.560]	Mathews et al. (1989) [0.578]	Reber (1967) [0.602]	Skosnik et al. (2002) [0.603]	Brooks & Vokey (1991) [0.856]
1 0102,W	1 0102,F	1 0102,T	1 0102,P	1 0102,M
2 0105,N	2 0103,D	2 0103,V	2 0104,X	2 0103,V
3 0203,S	3 0302,J	3 0202,P	3 0204,J	3 0205,V
4 0206,S	4 0204,D	4 0303,X	4 0203,H	4 0207,X
5 0306,S	5 0304,H	5 0204,T	5 0405,T	5 0404,T
6 0405,N	6 0405,Q	6 0305,V	6 0406,V	6 0402,M
7 0507,P	7 0505,F	7 0403,X	7 0602,T	7 0403,V
8 0407,P	8 0406,M	8 0406,S	8 0306,P	8 0306,X
9 0510,N	9 0606,Q	9 0504,P	9 0605,H	9 0307,M
10 0208,W	10 0507,H	10 0506,S	10 0503,X	10 0507,X
11 0608,W	11 0608,J	11 0601,_	11 0307,J	11 0508,R
12 0710,N	12 0706,H		12 0507,V	12 0704,R
13 0809,S	13 0807,M		13 0701,_	13 0708,R
14 0811,P	14 0709,F			14 0607,V
15 0904,P	15 0809,J			15 0609,T
16 0907,P	16 0901,_			16 0709,T
17 0910,N				17 0808,V
18 1011,P				18 0801,_
19 1109,S				19 0810,M
20 1012,Z				20 0909,R
21 1112,Z				21 0901,_
22 1201,_				22 0910,X
				23 1001,_

Meulemans & Van der Linden (1997)[0.686]	Conway & Christiansen (2006) [0.716]	Knowlton & Squire (1996) [0.740]	Reber & Allen (1978) [0.761]	Witt & Vinter (2012) [0.916]
1 0102,F	1 0102,X	1 0101,X	1 0102,M	1 0102,B
2 0103,T	2 0103,V	2 0201,T	2 0103,V	2 0103,R
3 0204,V	3 0202,T	3 0103,V	3 0202,T	3 0204,Y
4 0302,M	4 0303,R	4 0302,J	4 0204,V	4 0404,Y
5 0304,X	5 0203,M	5 0304,X	5 0305,X	5 0405,G
6 0405,F	6 0204,X	6 0305,T	6 0302,X	6 0305,G
7 0406,R	7 0305,V	7 0404,J	7 0403,R	7 0505,G
8 0505,R	8 0502,R	8 0504,V	8 0406,T	8 0504,Y
9 0606,M	9 0306,T	9 0201,_	9 0506,M	9 0406,B
10 0506,T	10 0604,R	10 0401,_	10 0505,R	10 0507,R
11 0507,T	11 0407,M	11 0501,_	11 0401,_	11 0708,T
12 0608,X	12 0701,_		12 0501,_	12 0608,T
13 0706,R			13 0601,_	13 0401,_
14 0807,V				14 0501,_
15 0701,_				15 0601,_
16 0801,_				16 0701,_
				17 0801,_

Table 2

Transition tables for each of the 10 AGs from Van den Bos & Poletiek (2008) and their corresponding TE values [in square brackets] as computed from the `computeTE` *function in Figure 1.*

AG A [0.5543]	AG B [0.6019]	AG C [0.6753]	AG D [0.6823]	AG E [0.7131]
1 0102,Z	1 0102,Z	1 0102,Z	1 0102,Z	1 0102,Z
2 0103,N	2 0103,N	2 0103,N	2 0103,N	2 0103,N
3 0204,T	3 0204,T	3 0204,T	3 0204,T	3 0204,T
4 0306,R	4 0306,R	4 0306,R	4 0306,R	4 0306,R
5 0205,Q	5 0205,Q	5 0205,Q	5 0205,Q	5 0205,Q
6 0305,M	6 0305,M	6 0305,M	6 0305,M	6 0305,M
7 0407,P	7 0407,P	7 0407,P	7 0402,N	7 0407,P
8 0507,S	8 0507,S	8 0507,S	8 0407,P	8 0507,S
9 0508,W	9 0508,W	9 0508,W	9 0507,S	9 0508,W
10 0608,X	10 0603,J	10 0608,X	10 0508,W	10 0608,X
11 0709,R	11 0608,X	11 0702,W	11 0608,X	11 0709,R
12 0710,Q	12 0709,R	12 0709,R	12 0702,W	12 0710,Q
13 0810,M	13 0710,Q	13 0710,Q	13 0709,R	13 0810,M
14 0811,T	14 0810,M	14 0803,X	14 0710,Q	14 0811,T
15 0904,Z	15 0811,T	15 0810,M	15 0810,M	15 0904,Z
16 1005,J	16 0904,Z	16 0811,T	16 0811,T	16 0907,S
17 0912,J	17 1005,J	17 0904,Z	17 0904,Z	17 1005,J
18 1106,N	18 0912,J	18 1005,J	18 1005,J	18 1008,P
19 1012,N	19 1106,N	19 0912,J	19 0912,J	19 0912,J
20 1112,Z	20 1012,N	20 1106,N	20 1106,N	20 1106,N
21 1201,_	21 1112,Z	21 1012,N	21 1012,N	21 1108,J
	22 1201,_	22 1112,Z	22 1112,Z	22 1012,N
		23 1201,_	23 1201,_	23 1112,Z
				24 1201,_

AG F [0.7465]	AG G [0.7586]	AG H [0.8021]	AG I [0.8449]	AG J [0.8587]
1 0102,Z	1 0102,Z	1 0102,Z	1 0102,Z	1 0102,Z
2 0103,N	2 0103,N	2 0103,N	2 0103,N	2 0103,N
3 0204,T	3 0204,T	3 0204,T	3 0204,T	3 0204,T
4 0306,R	4 0306,R	4 0306,R	4 0306,R	4 0306,R
5 0205,Q	5 0205,Q	5 0205,Q	5 0205,Q	5 0205,Q
6 0305,M	6 0305,M	6 0305,M	6 0305,M	6 0305,M
7 0402,N	7 0407,P	7 0402,N	7 0407,P	7 0402,N
8 0407,P	8 0507,S	8 0407,P	8 0507,S	8 0407,P
9 0507,S	9 0508,W	9 0507,S	9 0508,W	9 0507,S
10 0508,W	10 0603,J	10 0508,W	10 0603,J	10 0508,W
11 0603,J	11 0608,X	11 0603,J	11 0608,X	11 0603,J
12 0608,X	12 0702,W	12 0608,X	12 0702,W	12 0608,X
13 0702,W	13 0709,R	13 0702,W	13 0709,R	13 0702,W
14 0709,R	14 0710,Q	14 0709,R	14 0710,Q	14 0709,R
15 0710,Q	15 0803,X	15 0710,Q	15 0803,X	15 0710,Q
16 0803,X	16 0810,M	16 0803,N	16 0810,M	16 0803,X
17 0810,M	17 0811,T	17 0810,M	17 0811,T	17 0810,M
18 0811,T	18 0904,Z	18 0811,T	18 0904,Z	18 0811,T
19 0904,Z	19 1005,J	19 0904,Z	19 0907,S	19 0904,Z
20 1005,J	20 0912,J	20 1005,J	20 1005,J	20 0907,S
21 0912,J	21 1106,N	21 1008,P	21 1008,P	21 1005,J
22 1106,N	22 1108,J	22 0912,J	22 0912,J	22 1008,P
23 1012,N	23 1012,N	23 1106,N	23 1106,N	23 0912,J
24 1112,Z	24 1112,Z	24 1012,N	24 1108,J	24 1106,N
25 1201,_	25 1201,_	25 1112,Z	25 1012,N	25 1108,J
		26 1201,_	26 1112,Z	26 1012,N
			27 1201,_	27 1112,Z
				28 1201,_