Perspective Text Analysis: Tutorial to Vertex

Bierschenk, Inger; Bierschenk, Bernhard

2011

©*Kognitionsvetenskaplig forskning – Cognitive Science Research,* ISSN 0281-9864, 2011, 100
Copenhagen University & Lund University

Perspective Text Analysis
Tutorial to Vertex

Inger Bierschenk
Bernhard Bierschenk[1]

Abstract The present work is based on the Kantian (AaO)-axiom and conceptualised as a textbook. As scientific method in the true sense of the notion science, the Vertex version of Perspective Text Analysis (PTA) represents a completely new approach to text-based studies. The inter-lingual character of Vertex has been tested and established in the context of six different languages, four belonging to the German family and two to the Roman family. The actual presentation concerns the English version. Vertex comprises a strict measurement of the textual angles, which are used for a non-linear description of the verbal flow, whose evolutionary dynamics shapes a language space. The core of Vertex is introduced by means of instructions to a stepwise procedure with the aim to guide the user in text processing, string calculation, and geometric representation of the dimensions of intention and orientation. The evolved textual shapes and their transformation into energy landscapes are discussed in relation to their dynamics and terminological validity in communicating conceptual information of societal significance.

Text as Organisation and Structure

Approaching Texts

A great majority of scientific text analyses is initiated on the basis of the researcher's assumptions of what should be expressed in a certain text within a certain area. In a broad social science perspective there are as many methods of text analyses as there are projects or fields of research, that is, methods with, for example social, psychological, economic and political constraints. This is due to the fact that the researcher or research group develop their own theories of the "structure" of a field. The work is aimed at affirming or, in the single cases, rejecting this structure by the method used. A common name for a lot of such methods is content analysis.

Each content analysis implies some kind of pre-defined classification scheme applied on the text by an analyser with more or less reliable result. The entire coding process is time consuming at least by larger text materials, and once performed it is hard to apply anew, among other things because the analyser during processing quite simply is not able to keep in mind the classifications made at an earlier stage. It is even more difficult to perfectly agree with another analyser. Consequently, to this methodology is connected a stock of reliability tests, which make the result a question of probabilities, in the statistical sense.

Since the 70's and for several years we worked with great amounts of interview materials (4000 pages!), consisting of answers to unbound questions (Bierschenk & Bierschenk, 1976). We know what it means to develop coding rules and write a manual and have all the rules and decisions tested for reliability between coders (I. Bierschenk, 1977). With this experiential background we would like to state that the final result of all this effort is the feeling of having learnt a lot about spontaneous text production and about different processes involved in the various steps of analysis. But it became more and more difficult to

[1] Correspondence and requests for materials should be addressed to Bernhard Bierschenk, Department of Psychology at Lund University, Box 213, SE-221 00 Lund, Sweden. Additional information on theory and method development may be found at the URL address http://www.sites.google.com/site/aaoaxiom/tutorials

see how this layout or textual face could be capable of reflecting something internal, that is, structure.

It is a known fact that linguists and social scientists are very different in their way of approaching texts, at least traditionally seen. To a linguist, the text includes its context, while to a social scientist, the context may imply the wider frame of interpretation, that is, the text is included in the context. Somewhere within those borderlines the linguist and the social scientist meet under a joint flag, namely the structure concept, which by both is conceived of as an inner quality of the text, but which they try to capture by looking at the face, that is, the organisation. As a consequence, the same basic idea lies behind content analysis and semantics.

Text as Organisation

Typical of organisation is the definition of units and their interrelations, often described in terms of levels. Identification and analysis is done by means of a rule system departing from a base or kernel and associating the units into a certain order so that the procedure of reaching complete overview will be as efficient as possible. Depending on which complex relations the organisation stands for, the path from base to top may be more or less linear. There are parallel orderings and intermediate levels, more or less important nodes to pass. Sometimes decisions have to be made, for example, because some units may have the same function but different labels or vice versa. Other units are just slot fillers in the building and function as barriers. Common to organisations of various kind is that they are visual and tactile, at least they have visual representatives making hidden parts easily reachable, for example by reference and inference.

One consequence of the social science view on text as organisation is to regard the text as part of a kind of context formed by a collocation of organisations, which we might call an agency. Seen in this way one text and its agent is not an autonomous whole but is linked, let be in arbitrary order, to an agency of texts, whose common denominator the analysis shall detect. Seen in a narrower linguistic context the organisation is a grammatical sentence and its parts. Some parts are main components, some subordinated. We are convinced that most social scientists use the notion structure when they mean order and relations, just like linguists have done since modern grammarians in the 1950's proposed the description of a sentence as syntactic structures.

How then can we explain the mixing up of organisation and structure? At least in traditional linguistics a kernel is assumed from which expansions and transformations can be observed. Transformation is a structural concept, but in sentence analysis it is used for positional change (e.g., the passive construction). The idea to prove structural development via syntactic transformation caused a flow of psycho-linguistic and psychological tests in the 60's and 70's. One hypothesis tested was whether the degree of transformation on the kernel sentence could be associated with degree of difficulty in understanding, measured for example as processing time in reading the sentence. This was not the case. To conclude – organisational change cannot guarantee structural control whether the steering component is text or not.

Text as Structure

We think that anyone who has some experience of an organisation knows that the informal ways are much more important than every formal step or stair. In places which are involving people there is also a dynamics whose operations are both rule-breaking and unexpected. Connected to a dynamic functioning are, above all, open channels, the possibility for positional change, and sensitivity to the spirit (or Geist). These streams in the system are necessary for development to come about. As a whole, it seems though as if the best result

would be gained if at least some positions were filled by certain key functions, which do not change, in order to clarify the organisation's pathways and goals, that is, position and function support each other. So, the structure is only partly visible and often just by chance through organisational keys. But since they are stable, we can get at the pulse, the movement, the power without which the organisation is nothing more than an empty shell.

A text analysis whose fundamental governing concept is structure has to keep track of the functions and channels through which the steering component (the source) is operating. In several social science models we find the concept of actor, sometimes standing for individual (in the sense of a single representative) or organisation. Typical of an actor is only representation; function and position are not separated. It follows that the concept of role is central in such a model. The same idea is to be found in the linguistic model. A so called grammatical role builds on positional thinking (where position = role) and that the various role representatives are similar in the sense of classification; they must be of the same type to fit into the role. Quite natural, the analysis of a Gestalt is based on pre-required knowledge of the scenarios, the frames for the roles (I. Bierschenk, 1984).

In a structure model a perspective has to be discerned such that the source and the operator can be separated during the angular displacement process. This presupposes not only representation but above all individuality. To make clear in what way a perspective is present in a text, we have to replace actor by agent. Agent stands for an individual, autonomous way of steering a process. Depending on what the process is about or where in the flow the steering is visible, its source manifests itself in the most appropriate shape. Thus a sub component or part may stand for a whole, which means that the whole in all its parts is not known during processing. An important consequence of this model is that the agent function is bound to the first position in a functional schema. A schema differs from a frame insofar as any textual information may take the function of agent. In this way it is possible to control the co-operation between the visible and non-visible or unknown, which the Perspective Text Analysis (PTA) detects.

PTA uses the functionally bound position to detect whether it is being filled or not. When filled, there are variables or visible representative agents. When empty, this means that the agent hides, but at the same time it opens up a peep-hole to a space or, referring to our earlier example, a channel, which elicits a flowing of information through the text, since in that case the hidden unknown agent (X-agent) performs its government from another place. Through the hole it can be detected.

The bound position becomes a window through which a text producer chooses to show up or not, or through which a Geist may stream, impossible to grip in advance, but fully developed and conceivable when it has reached the end. The variable agents, also termed textual agents, have the function in common that they regulate the streams descending from the X-agent. Any other similarities between them are not possible to state a priori. Thus it is the functional use of the agent that paves the way for perspective information. The relationship between textual agents, text producer and some spirit or super-ordinate idea does not build on classes but on categories, in the Kantian sense.

PTA and Vertex

PTA has been presented since the 1980's (e.g., B. Bierschenk, 1984, 1991, 1993; I. Bierschenk, 1992, 1999, 2011a; B. Bierschenk & I. Bierschenk, 1993; B. Bierschenk, I. Bierschenk & H. Helmersson, 1996). The experiences made by a number of users form the background to the methodological progress (I. Bierschenk, 2011b). Version Vertex comprises a strict measurement of angular displacements, which are used for non-linear description of a language flow, visualized as language spaces (B. Bierschenk, 2001, 2005; I. Bierschenk & B. Bierschenk, 2004, 2011). The geometrical essentials are outlined in B. Bierschenk (2011).

Text Processing

You are now invited to follow a procedure, which will show what happens to a text when processed with the Vertex system. The procedures have been divided into fifteen steps under the following main elements: (1) Text processing, (2) Calculation, (3) Geometric representation, and (4) Naming (transformation and extraction of terms). Furthermore, the tutorial is based on eighteen tables and four figures.

The comprehensive significance of a text of any length or any subsection of it relates to its style of timing and spacing. Moreover, a style reflects the degree of implicitness or depth of a text. First of all you have to acquaint yourself with the piece of text that you will be handling in your exercises. The following text portion has been taken from research materials, handed in by two doctoral students of Economics and Business Administration:

Just think of the common attitude today, and that does not go for local government employees only, most people think I have got my salary, why should I bother to come up with ideas as to how the local authority could save money, I do not care a damn. It is the same reasoning here.

A community official stands for the verbal production. Hence, it is quite authentic and has the typical verbal characteristics that can be expected from interviews. Distinctive of the presented paragraph is partly the absence of any frame factor, partly the softness in the moulding of particular points of view as well as the points of observation. Moreover, independent of the length of a text, e.g., one sentence of length or more, a text represents always wholeness. Now, you are invited to closely follow the coding procedure.

Step 1: Transposition

As a first measure, we will guide you through the working procedures stepwise. Hence, in order to transpose text it is important to set up a table. The result of this practise should look like the layout of Table 1.

Table 1
Transposition

Row	*Text*	*Row*	*Text*	*Row*	*Text*	*Row*	*Text*
1	[.]	18	government	35	to	52	not
2	*	19	employees	36	come	53	care
3	Just	20	only	37	up	54	a
4	think	21	,	38	with	55	damn
5	of	22	most	39	ideas	56	.
6	the	23	people	40	as	57	*
7	common	24	think	41	to	58	It
8	attitude	25	I	42	how	59	is
9	today	26	have	43	the	60	the
10	,	27	got	44	local	61	same
11	and	28	my	45	authority	62	reasoning
12	that	29	salary	46	could	63	here
13	does	30	,	47	save	64	.
14	not	31	why	48	money	65	[*]
15	go	32	should	49	,		
16	for	33	I	50	I		
17	local	34	bother	51	do		

Transposition means that you have to change the layout of the clause. Thereby you put the text on its end. If transposition is taken as a measure for an anticipated algorithmic processing of produced textual elements, processing strings of graphemes can be made operational by searching between two sentence markers or between two clause markers.

As you can see in Table 1, each word gets its own row and also each punctuation mark. In addition, you have to mark the beginning and end of the text with [.] when such a punctuation mark is not naturally present. Within these end points there are in-between borders, which you mark with a starter (*) if there is no naturally present marker, e.g., (,). If the text would not continue at row 57, you would have to insert end of text (. [*]).

The transposed text has now been expanded with 4 rows, that is, 2 at the start ([.]*…), 1 at the beginning of the second sentence (* It…) and 1 at the very end (. [*]), which is signalling the end of text, since no further strings are following. Hence, the text is now transposed to 65 numbered rows. Each row is filled with strings of graphemes, some consist of just one grapheme, e.g. (,) and (I), others with several. In the following, when we talk about what is contained in the rows, we talk about strings.

Step 2: Dictionary Coding

Next step will be to identify strings that you are able to recognise as being part of the English dictionary. These a priori classified markers are listed in our empirical dictionary which you can inspect in Table 2.

Table 2

Generation of an empirical dictionary

Verb (40)	*Prep (60)*	*Prep (70)*	*Prep (80)*	*[2]Clause (01)*	*[1]Sentence (00)*
think	of	with	for	and	.
does	to			that	
go				,	
have				why	
got				as	
should				how	
bother					
come					
could					
save					
do					
care					
is					
reasoning					

[1]Marker of Sentence of 1st degree (Sentence Marker, SM)

[2] Marker of Sentence of 2nd degree (Clause Marker, CM)

When you have identified the lexical strings, you need to tag them with a code, which you will have to use in identifying all the unidentified strings, which will be identified by the algorithmic coding in Step 4 below. You find the Code system in Table 3. The codes comprise a system with two-digit numbers in the interval from 00 to 90.

Table 3
Code system

Identification	*Symbol*	*Description*	*Code*
Sentence Marker	[.]	Technical insertion of a period	00
Clause Marker	that	Naturally occurring Clause Marker	01
Agent	A_X	Contextual or conditional restrictions	10
Agent	A_X	Experiential specifications	20
Agent	A	Explicit	30
Agent	*	Implicit and unconditional	30
Verb	ω	Nucleus of kernel-sentence	40
Objective	O	p_0=Without Pointer and explicit	50
Objective	*	p_0=Without Pointer and implicit	50
on-Objective	-	p_1=Pointer *on*	60
with-Objective	-	p_2=Pointer *with*	70
for-Objective	-	p_3=Pointer *for*	80
Phrase	-	Verbless strings following a Clause Marker	90
Technical	[*]	Insertion before a closing period	01
Sentence Marker	.	Naturally occurring Sentence Marker	00

The Code system has been developed by B. Bierschenk and I. Bierschenk and was introduced in 1976. The codes allow for the denotation of direction by tens and for the denotation of the organisationally bound orientation by units. Meanings and marks will be clear to you as the coding goes on. The insertion of technical markers (*) is made on the place where a functional clause is implicit (as illustrated under Step 3).

Comment: The system recognises four types of markers, namely (1) Sentence Markers (SM), (2) Clause markers (CM), (3) Prepositions (types *on*, *with*, and *for*), and also (4) Verbs.

Now we hope that you have a coding sheet, e.g., an Excel-sheet, in front of you, which you can easily expand horizontally and vertically as you go on. Expand with two columns before the strings to make space for row numbering and the codes. Mark the previously introduced clause with the respective code on the row to the left of the text as shown in Table 4.

Comment: Punctuation marks are the sentence markers (. ? !) and the clause markers (, ; : -). Other clause markers are function words, as e.g. conjunctions. Note that there is a difference between *the employed* (inflected verb form) and *the employees* (noun form). The system requires that you code the basic sense and turn inflected forms (i.e. verbs and participles) into verbs. Thus the form *reasoning* is a verb irrespective of whether it may be a noun as well. This principle has consequences in the coding, since the verb is the key string to identify the unity clause.

Begin with looking up and identify the lexical strings. When you are ready you should have 35 coded strings. The empty slots will be filled with codes through the algorithmic processing. In the continued process you will see that there might be more clauses (the term is Functional Clause) than denoted by the clause markers, just depending on the verb identification. The solution to the complete dictionary coding is given in Table 4.

Table 4
Dictionary coding

Row	*Code*	*String*	*Row*	*Code*	*String*	*Row*	*Code*	*String*
1	00	[.]	23	-	people	45	-	authority
2	01	*	24	40	think	46	40	could
3	-	Just	25	-	I	47	40	save
4	40	think	26	40	have	48	-	money
5	60	of	27	40	got	49	01	,
6	-	the	28	-	my	50	-	I
7	-	common	29	-	salary	51	40	do
8	-	attitude	30	01	,	52	-	not
9	-	today	31	01	why	53	40	care
10	01	,	32	40	should	54	-	a
11	01	and	33	-	I	55	-	damn
12	01	that	34	40	bother	56	00	.
13	40	does	35	60	to	57	01	*
14	-	not	36	40	come	58		It
14	40	go	37	-	up	59	40	is
16	80	for	38	70	with	60	-	the
17	-	local	39	-	ideas	61	-	same
18	-	government	40	01	as	62	40	reasoning
19	-	employees	41	60	to	63	-	here
20	-	only	42	01	how	64	00	.
21	01	,	43	-	the	65	01	[.]
22	-	most	44	-	local			

Step 3: Coding of the Implicit Functional Clause

To make sure that the continued coding will be correct you have to identify all functional clauses, even those which are not marked so far. This is the kernel:

Agent + Verb + Objective

This principle is given as a template in Table 5.

Table 5
Implicit functional clause

Code	*Function*
01	*
30	*
40	**Verb**
50	*
01	*
30	*
40	**Verb**
50	*
01	*

A functional clause shall have a verb (only one !) and one or more strings before and after it. If no strings are present before or after, you insert a dummy (*) to mark the missing string, which at a later stage will be substituted with specific information. The table illustrates how a compound verb expression expands to form two functional clauses: As soon as there is a verb there must be a clause marker between this verb and the next following, and also there has to be a place for an Agent position.

The processing must begin at the very end of the clause or text and must work its way upward to the very beginning of the paragraph. Hence its last sentence marker is always the initialising period. The necessity of this

kind of processing has been determined empirically (B. Bierschenk & I. Bierschenk, 1986a, b). To demonstrate the principle of a bottom up processing we start with the dictionary coding of Table 4 above as a basis. You can follow the process via Table 6 in Step 4. Here, you can observe how functional clauses are working. We will now guide you through the algorithmic coding by using the patterns of Table 6 and comment on the solutions.

Table 6
Algorithmic coding

Row	*Code*	*String*	*Row*	*Code*	*String*	*Row*	*Code*	*String*
1	00	[.]	30	30	*	59	40	could
2	01	*	31	40	have	60	50	*
3	30	Just	32	50	*	61	01	*
4	40	think	33	01	*	62	30	*
5	60	of	34	30	*	63	40	save
6	60	the	35	40	got	64	50	money
7	60	common	36	50	my	65	01	,
8	60	attitude	37	50	salary	66	30	I
9	60	today	38	01	,	67	40	do
10	01	,	39	01	why	68	50	not
11	01	and	40	30	*	69	01	*
12	01	that	41	40	should	70	30	*
13	30	*	42	50	I	71	40	care
14	40	does	43	01	*	72	50	a
15	50	not	44	30	*	73	50	damn
16	01	*	45	40	bother	74	00	.
17	30	*	46	50	*	75	01	*
18	40	go	47	01(60)	to	76	30	It
19	80	for	48	30	*	77	40	is
20	80	local	49	40	come	78	50	the
21	80	government	50	50	up	79	50	same
22	80	employees	51	70	with	80	01	*
23	80	only	52	70	ideas	81	30	*
24	01	,	53	01	as	82	40	reasoning
25	30	most	54	01 (60)	to	83	50	here
26	30	people	55	01	how	84	00	.
27	40	think	56	30	the	85	01	[*]
28	50	I	57	30	local			
29	01	*	58	30	authority			

Step 4: Algorithmic Coding

Concerning the algorithmic coding, again, our suggestion is that you use an Excel-sheet, in which you can add empty rows and columns gradually, which will be required as the coding proceeds.

(1) Starting at the bottom of Table 6, you find the string *(here)* before you reach the verb, which is an Objective of the p_0-type and which you tag with code 50. So, what you do now is to expand with two more rows. Closest to the verb you mark the place (row 81) for the Agent

with (*) and give it code 30. The next row (80) above shall be marked with a clause marker (*) and be given the code 01. It is illustrative as to the verb code (on row 82) and how the last sentence is expanding because of missing explicitness in the string sequencing. In the first place (*reasoning*) is a verb and not a noun. Thus the sentence has two verbs, which means two functional clauses. The pattern to have in mind for the processing of this implicitness is the one just presented in Table 5.

(2) The next functional clause is explicit, which means that (*the same*) on row (78-79) are Objective strings of the 50-type and (*It*) on row 76 is a 30-type string.

(3) Taking a step upwards, you find two 50-strings (*a damn*) but not easily an Agent string, because the verb (*do*) on row 67 takes the string (*not*) as its 50-string (row 68). What to do? Well, again you have to mark an expansion with (*) on row 69 for implicit clause, and, following the pattern, likewise an Agent dummy (row 70) together with its code.

(4) Next step is to do a full process coding of the implicit functional clause. Because, look at what comes into view! Remember the pattern: Two verbs (rows 59 and 63) in a sequence means three empty rows for the A-dummy, a clause marker, and O-dummy in this order upwards. Now, you can complete the coding around the two verbs, and then take a step further upwards.

(5) You find here on row 54 the p_1-pointer, (*to*), between the two clause markers (*as* and *how*). Thus in the first coding step, this pointer got the lexical code (60). But in this connection, (pattern 01+60+01), it is neutralised and gets the marker code (01) in the second step.

(6) In connection with (*come*) on row 49 you find a p_2-pointer (*with*) which is steering the following string, i.e. give code (70) to both. The Objective part is differentiated, such that (*up*) is an ordinary p_0-type. Further there is an implicit Agent (row 48), so following the kernel you have to insert an Agent dummy. This step has as consequence that the pointer on row 47 is refunctionalised to clause marker in the position close to the verb (pattern 60+*+40).

(7) The verb *(bother)* has an indicator of the p_0-type (row 46), which you treat as before. Since it is missing the A-code on row 44 you have to expand and do the correct coding.

(8) Above the next verb *(should)* there is an Agent missing (row 40), which you must insert, but a 50-code on row 42 is explicit. The coding is complete when you have marked the implicit clause, i.e. insertion of clause marker (01) between (*I*) and A-dummy (row 43).

(9) Think over and do the coding for the sequence around the verbs *(got)* (row 35) and *(have)* (row 31) as an exercise. The have-clause is totally implicit, so do what you have to do. The got-clause is partly explicit and must also be completed. This sequence has now been expanded with five rows, right?

(10) Before you step up to the next implicit clause, you pass a completely explicit clause (the verb *think*). The only addition you need to introduce is a clause marker (row 29).

(11) Now you step up to the verb *(go)* on row 18, where you see on row19 an example of the p_3-pointer, i.e., an 80-type of Objective. As usual you expand with codes for clause marker and Agent above the verb (rows 16-17).

(12) Likewise for Agent on row 13 in connection to *(does)* you need to insert a Agent dummy.

(13) Finally there is on row 5 a p_1-pointer of the (60) type to the verb *(think)*. When you have processed the last 30-code, the coding is complete.

At this point you have learnt how the functional clause is marked and how the text is expanded because of missing explicitness in the string sequencing. Moreover, it has been necessary to insert a number of dummies for implicit A- and O-strings around the verb, which shall be used in making visible the perspective of the text. The coded text now consists of 85 rows. You will proceed by working with a number of functional clauses, which will be called Block, and displacement procedures.

Step 5: Block Coding and Supplementation

In this phase you will be able to acquaint yourself with block coding and displacement procedures. For this step you will need two extra columns, preferably to the right of the strings. You see the design below in Table 7. The first concerns the blocks. The other column you need is for the bookkeeping of the displacements. Each clause marker at the upper border of an AaO unit indicates a block. Please number the blocks from the beginning of the text and insert the digits in the reserved column just in front of the clause marker or first string of the clause. If two or more clause markers are following each other, you insert the block number in front of the first clause marker.

As you can see in Table 7, the number of blocks (B) is 15. They are important when it comes to computing the copies needed for the replacement of dummies. The assignment of copies to dummies will follow this principle:

A-dummy gets its copy from above, O-dummy from below.

In case the A-dummy is preceded by clause border (CM), the copy of the preceding 30-string is taken. If it is preceded by sentence border (SM), the copies of both 30- and 50- (60/ 70/ 80) strings are taken. At the beginning of text or paragraph there are no immediately preceding strings, and this circumstance is, according to Table 3, denoted by the variable (X). The supplement for the O-dummy is always picked up by copying the following A- and O-strings. But in this case the sentence border is not to be crossed. End of sentence has the function of end of text, which replaces the O-dummy with the variable (Y). As a consequence, the dummies to be displaced are observed in positions with specified surrounded strings.

In looking at the coded text, we can observe that the strings can be described by means of patterns with and without dummy-markers (*). Thus the first A-case of Table 7 is an explicit A-string followed by a verb, which indicates the pattern (word + verb). Block 2 contains an A-dummy to be supplemented. This Agent is preceded by three clause markers and followed by the verb. Thus the pattern for this case is (CM + * + verb), which means that its copy is to be taken from Block 1, namely the explicit string [Just].

The O-pattern in the first block is of the p_1-type (Verb+60-prep+Word), i.e. verb followed by the preposition *of*, which is a case of the p_1-type, plus strings. In Block 3, like the preceding one, you identify the case for (CM+*+Verb), which requires a further Agent displacement, which is resulting in a deeper embedding [[Just]] of the copied string, while the O-case is a preposition case of the p_3-type, i.e. (80-prep+Word).

Before we continue with calculations, it is important to remember that the text consists of strings, which will be used when it comes to denote the final results, as demonstrated in the Excursus chapter. Step 6 will show you how patterns just described are formalized into a system of messengers and connected to the clocking mode of a string (B. Bierschenk, 2001).

Table 7
Block coding and supplementation

Row	*Code*	*String*	*B*	*Supplement*	*Row*	*Code*	*String*	*B*	*Supplement*
1	00	[.]			43	01	*	8	
2	01	*	1		44	30	*		B7=most people
3	30	Just			45	40	bother		
4	40	think			46	50	*		B9=most people+up+ideas +the local authority+money
5	60	of			47	01(60)	to	9	
6	60	the			48	30	*		B8=most people
7	60	common			49	40	come		
8	60	attitude			50	50	up		
9	60	today			51	70	with		
10	01	,	2		52	70	ideas		
11	01	and			53	01	as	10	
12	01	that			54	01(60)	to		
13	30	*		B1=Just	55	01	how		
14	40	does			56	30	the		
15	50	not			57	30	local		
16	01	*	3		58	30	authority		
17	30	*		B2=Just	59	40	could		
18	40	go			60	50	*		B11=the local authority+money
19	80	for			61	01	*	11	
20	80	local			62	30	*		B10=as to how the local authority
21	80	govern ment			63	40	save		
22	80	employees			64	50	money		
23	80	only			65	01	,	12	
24	01	,	4		66	30	I		
25	30	most			67	40	do		
26	30	people			68	50	not		
27	40	think			69	01	*	13	
28	50	I			70	30	*		B12=I
29	01	*	5		71	40	care		
30	30	*		B4=most people	72	50	a		
31	40	have			73	50	damn		
32	50	*		B6=most people+I	74	00	.		
33	01	*	6		75	01	*	14	
34	30	*		B5=most people	76	30	It		
35	40	got			77	40	is		
36	50	my			78	50	the		
37	50	salary			79	50	same		
38	01	,	7		80	01	*	15	
39	01	why			81	30	*		B14=It
40	30	*		B6=most people	82	40	reasoning		
41	40	should			83	50	here		
42	50	I			84	00	.		
					85	01	[*]		

Step 6: Identification and Assignment of Messengers

Messengers are emerging at the second level of pattern processing. It follows that messengers are second order patterns, which are characterised by local properties. Consequently, local properties communicate the steering and control conditions of the

functional clause. Step 6 means coding the blocks by tagging the condition which is steering the clocking case. The validity of the clocking assumption is based on the hypothesis that distance is a function of rotational acceleration and that spinors have the capacity to carry the rotation of segments of varying complexity. Since any phase transition from virtual to material states results in a decrease in symmetry and since spinors offer an efficient notation in the form of directed numbers (Hestenes, 1986/1993, pp. 11-12, 51), they can be used to control broken symmetries and to establish prismatic textual surfaces. Furthermore, the measurement in Radians (Rad) in Table 8 is given by [arc $\alpha = 2\pi(i\phi/360)$] and [arc $\beta = 2\pi(i\theta/360)$]. Hestenes (p. 75) emphasises that the exponential function and its series expansion requires that angle is measured in radians.

As column name in Table 9 below you may use Messenger (M). The (M) assignment will be done by using the messengers given in Table 8. As you can read out from the table, each side of the verb has 9 messengers and consequently 9 basic steering and control conditions, which are all inclusive. Each messenger has been empirically defined and verified experimentally. Which case is applicable has been marked with the notion (A1-A9) for the messengers of the Agent and (O1-O9) for messengers of the Objective.

Table 8
Empirically defined messengers

	Left-hand Side of FC	*Rad*[1]		*Right-hand Side of FC*	*Rad*[1]
A	*= before the Verb*	*[iϕ/2]*	***O***	*= after the Verb*	*[iθ/2]*
1	$SM_p+\varnothing_A+\omega$	0	1	$\omega+\varnothing_O+SM$	0
2	$SM+CM+\varnothing_A+\omega$	0.785	2	$\omega+\varnothing_O+CM+Phrase+CM$	0.785
3	$CM+Phrase+\omega$	1.57	3	$\omega+Prep+\varnothing_O+SM$	1.57
4	$CM+Prep+Word+\omega$	2.36	4	$\omega+Prep+\varnothing_O+...+CM$	2.36
5	$Word+\omega$	3.14	5	$\omega+Word$	3.14
6	$Word+Prep+\omega$	3.87	6	$\omega+On\text{-}prep+Word$	3.87
7	$Word+Prep+\ldots+\omega$	4.71	7	$\omega+With\text{-}prep+Word$	4.71
8	$CM+\varnothing_A+\omega$	5.50	8	$\omega+For\text{-}prep+Word$	5.50
9	$SM_s+\varnothing_A+\omega$	6.28	9	$\omega+\varnothing_O+CM$	6.28

SM_p= at the beginning of a paragraph; SM_s= at the beginning of a sentence;
In the coding procedure, the dummy symbol is substituted with *

The first A-case is of the A5-type, which matches the pattern ([.]*Word+ω) of Block 1, which means sentence starting with a technical sentence marker plus a technical clause marker followed by one or more strings, which are followed by a verb. Thus mark the case with A5 and put the tag on the clause marker. It corresponds to a string rotation before the verb (ω).

The assignment has been supplemented with the proper clocking mode of the messengers for illustrative purposes. This illustration should make the connection between the empirically defined messengers of Table 8 and the rotation dynamics completely clear.

You may for example compare the A-string of Block 1 with the one in Block 2, where it is a dummy, preceded by three clause markers and followed by a verb. This pattern corresponds to case A8, thus the first clause marker of the second block should be tagged with A8, which signals an exceedingly different clocking mode. The O-pattern in the first block is of the O6-type, i.e. verb followed by a preposition case of p_1-type plus strings. Now you tag the pattern with O6, which you put on the verb string. In the next step it will be clear that the rotation of the verb is calculated as part of the O-field. In Block 2 you find an O-pattern which corresponds to the first pattern in the first block (ω+word), namely an explicit string connected to the verb. Mark the pattern with O5.

Table 9
Assignment of messengers

Row	*Code*	*String*	*B*	*M*	*Row*	*Code*	*String*	*B*	*M*
1	00	[.]			44	30	*		
2	01	*	1	A5	45	40	bother		O9
3	30	Just			46	50	*		
4	40	think		O6	47	01(60)	to	9	
5	60	of			48	30	*		A8
6	60	the			49	40	come		O5
7	60	common			50	50	up		
8	60	attitude			51	70	with		O7
9	60	today			52	70	ideas		
10	01	,	2	A8	53	01	as	10	A5
11	01	and			54	01(60)	to		
12	01	that			55	01	how		
13	30	*			56	30	the		
14	40	does		O5	57	30	local		
15	50	not			58	30	authority		
16	01	*	3	A8	59	40	could		O9
17	30	*			60	50	*		
18	40	go		O8	61	01	*	11	A8
19	80	for			62	30	*		
20	80	local			63	40	save		O5
21	80	government			64	50	money		
22	80	employees			65	01	,	12	A5
23	80	only			66	30	I		
24	01	*	4	A5	67	40	do		O5
25	30	most			68	50	not		
26	30	people			69	01	*	13	A8
27	40	think		O5	70	30	*		
28	50	I			71	40	care		O5
29	01	*	5	A8	72	50	A		
30	30	*			73	50	damn		
31	30	have		O9	74	00	.		
32	50	*			75	01	*	14	A5
33	01	*	6	A8	76	30	It		
34	30	*			77	40	is		O5
35	40	got		O5	78	50	the		
36	50	my			79	50	same		
37	50	salary			80	01	*	15	A8
38	01	,	7	A8	81	30	*		
39	01	why			82	40	reasoning		O5
40	30	*			83	50	here		
41	40	should		O5	84	00	.		
42	50	I			85	01	[*]		
43	01	*	8	A8					

In Block 3, like the preceding one, you identify case A8, while the O-case is a preposition case of p_3-type, i.e. O8. In Block 4 there is an A5 and an O5 pattern, and after it a new A8. The O-case in Block 5 can be identified as a verb followed by dummy and clause marker, which is denoted O9. Next block starts with another A8, which you now know the pattern of. As to the Object case of Block 6, it is of the O5 type consisting of several strings after the verb. You can see how the nesting of strings gives a series of implicit (structurally present) Agents; Blocks 7, 8, and 9 all take the A8. Block 7 further has an O5, as also Block 9 (*up*). As Object case of Block 8 you mark O9 because of the dummy.

In Block 9 we can identify a pattern with preposition of the p_2-type and string, i.e. O7, which you mark at the pointer position. Block 10 takes an A5 and an O9, the following an A8 and an O5. A5 and O5, the typical explicit clause pattern, comes next. In Block 13 we have an ($*_A$) and thus an A8 plus an explicit 50 code, O5. Now the last sentence runs automatically, beginning with A5, O5, and ending with A8, O5.

Calculation

Step 7: String Rotation

Once again, identified displacements can be updated and the changes in the winding of strings can be related to componential pairing. In computing the dynamics of developing mono- or multi-layered composites, involved super-string rotations have to be carried out in agreement with the patterns of winding super-strings, given in Table 10.

Table 10
Patterns of winding super-strings

Pattern	*Property*	*Magnitude*
Messenger	Virtual	Basic Value (W=1/1)
+Word	Physical	Curling Value (W1/10)
+Grapheme	Material	Valve Value (W=1/100)

With reference to the given pattern of winding super-strings, it implies the application of the following rules:

(1) The winding (W) of the virtual string is taken as basis (W=1/1). Every layered segment of this string is treated as an expression of the resonating property of a super-string. Since it is an expression of the string's contextual circumstances, its surface oriented curling plays a complementary role.
(2) The curling is accounted for by adding (+) the fraction of (W=1/10). The curling value is based on the observation that no more than ten words are making up a particular super-string. This value is added to the basic winding value of the string.
(3) In addition, the valve fraction is computed as (W=1/100). This fraction is multiplied with the number of participating graphemes. Geometrically conceived, they have a share in the development of a particular super-string.

By means of Table 11, you will learn how to decide upon the rotational speed of the strings and their acceleration. The calculation of the string rotations will be done by using the radians given in Table 8. You are now going to start calculation as illustrated in Table 11. We take the last sentence as our example because of the processing of dummies. The string *It* takes the value 0.314 (string within component) + (0.0314*2) (grapheme within string), which gives the value (=0.3768). To this value you add the base value of the component (A5=3.14).

Your computing should give you the sum (=3.5168). In the same way you are going to calculate O5. Please notice, that you always assign the sentence marker to the nearest preceding component and the clause markers (i.e., the clause openers) to the nearest following

component. The sentence is a good illustration of how to compute implicit strings. Because of the verb *reasoning* extra rows are added containing clause marker and ($*_A$), which trigger the pattern A8.

Table 11
String rotation

Row	*Code*	*String*	*M*	*Sum*	*Row*	*Code*	*String*	*M*	*Sum*
1	00	[.]			44	30	*		-3.90607
2	01	*	A5		45	40	bother	O9	
3	30	Just		3.5796	46	50	*		-11.4004
4	40	think			47	01(60)	to	A8	
5	60	of	O6		48	30	*		-5.59128
6	60	the			49	40	come	O5	
7	60	common			50	50	up		3.9564
8	60	attitude			51	70	with	O7	
9	60	today		7.3143	52	70	ideas		6.0759
10	01	,	A8		53	01	as	A5	
11	01	and			54	01(60)	to		
12	01	that			55	01	how		
13	30	*		5.093017	56	30	the		
14	40	does	O5		57	30	local		
15	50	not		3.9878	58	30	authority		4.3424
16	01	*	A8		59	40	could	O9	
17	30	*		0.965102	60	50	*		0.467059
18	40	go	O8		61	01	*	A8	
19	80	for			62	30	*		3.1002877
20	80	local			63	40	save	O5	
21	80	government			64	50	money		4.0506
22	80	employees			65	01	,	A5	
23	80	only		10.12	66	30	I		3.8308
24	01	*	A5		67	40	do	O5	
25	30	most			68	50	not		3.925
26	30	people		4.4274	69	01	*	A8	
27	40	think	O5		70	30	*		3.542757
28	50	I		3.9564	71	40	care	O5	
29	01	*	A8		72	50	a		
30	30	*		3.395861	73	50	damn		
31	40	have	O9		74	00	.		4.71
32	50	*		-1.73949	75	01	*	A5	
33	01	*	A8		76	30	It		3.5168
34	30	*		1.050653	77	40	is	O5	
35	40	got	O5		78	50	the		
36	50	my			79	50	same		4.3646
37	50	salary		4.4274	80	01	*	A8	
38	01	,	A8		81	30	*		3.624687
39	01	why			82	40	reasoning	O5	
40	30	*		-0.025445	83	50	here		
41	40	should	O5		84	00	.		4.5216
42	50	I		3.9878	85	01	[*]		
43	01	*	A8						

As you already know, the value of the Agent of the preceding block (30) is displaced and inserted in the 30-slot of the second block. For example, an arrow between the two envisioned gliding plans (B. Bierschenk, 2001, p. 4) makes evident, that the clocking initiates displacements, which imply that the Agent becomes shaded. You may control the complexity and the complete calculation of the string rotations by inspecting Table 11. Thereby you can notice how certain blocks, for example Blocks 7-9 on the rows 48 to 72 leave holes, which form fading path for the Agent as well as sinking path for the Objective.

After having completed this procedure you have made all preparations required for a 3D-presentation. In the next step, you will be working with setting up the necessary matrices. We suppose that you are familiar with the idea that a text may be represented in a 3D mode, a step which you are to prepare in the following.

Step 8: Strand Rotation

On the basis of the summed values of the composites of Table 11 you will see how strands are forming.

Table 12
Strand rotation

Pair Number	*Rad α-strand*	*Rad β-strand*	*Interval Number*	*Case Number*
1	3.5796	7.3143	1	1
2	5.093017	3.9878	2	1
3	0.965102	10.12	2	2
4	4.4274	3.9564	3	1
5	3.395861	-1.73949	3	2
6	1.050653	4.4274	3	3
7	-0.025445	3.9878	4	1
8	-3.90607	-11.4004	4	2
9	-5.59128	3.9564	4	3
10	-5.59128	6.0759	4	4
11	4.3424	0.467059	4	5
12	3.102877	4.0506	4	6
13	3.8308	3.925	5	1
14	3.542757	4.71	5	2
15	3.5168	4.3646	6	1
16	3.624687	4.5216	6	2

Strands are the expression of systematised sums and become the entries in Table 12, which is based on the parameters (1) Interval, (2) Number of variables [**A**=(α)] and [**O**=(β)] per interval, and (3) Rotations (radians). Table 12 contains the complete solution to the strand rotations. It is suitable to construct the table with the values of (**A**) and (**O**) in a parallel manner. As you can see you need 5 columns. In the set-up of the table the A-O unit is still central. All A-O pairs in the text are registered in the first column. By now you have also seen that there are sometimes more than one object in a clause, so therefore the pairs are 16 instead of 15 (number of blocks).

As you may have noticed, the number of intervals in this text is 6 but the number of A-O pairs varies. For example, in the fifth interval there are two β-variables but only one shaded α-variable. Hence, the fifth column denotes the serial order of the pairs within an interval.

Geometric Representation

Step 9: Developed Textual Spaces

In this step you will have to transfer the strands to some suitable graph program. (We have used SigmaPlot, Version 11). You are asked to construct one graph for each component. The order between variables within intervals (column 5 of Table 12) should be transferred to the x-axis and the number of intervals to the y-axis. The value of the component governs the development on the z-axis. The way the graphs are shaped after the data have been transferred to a typical SigmaPlot arrangement can be viewed in Figure 1.

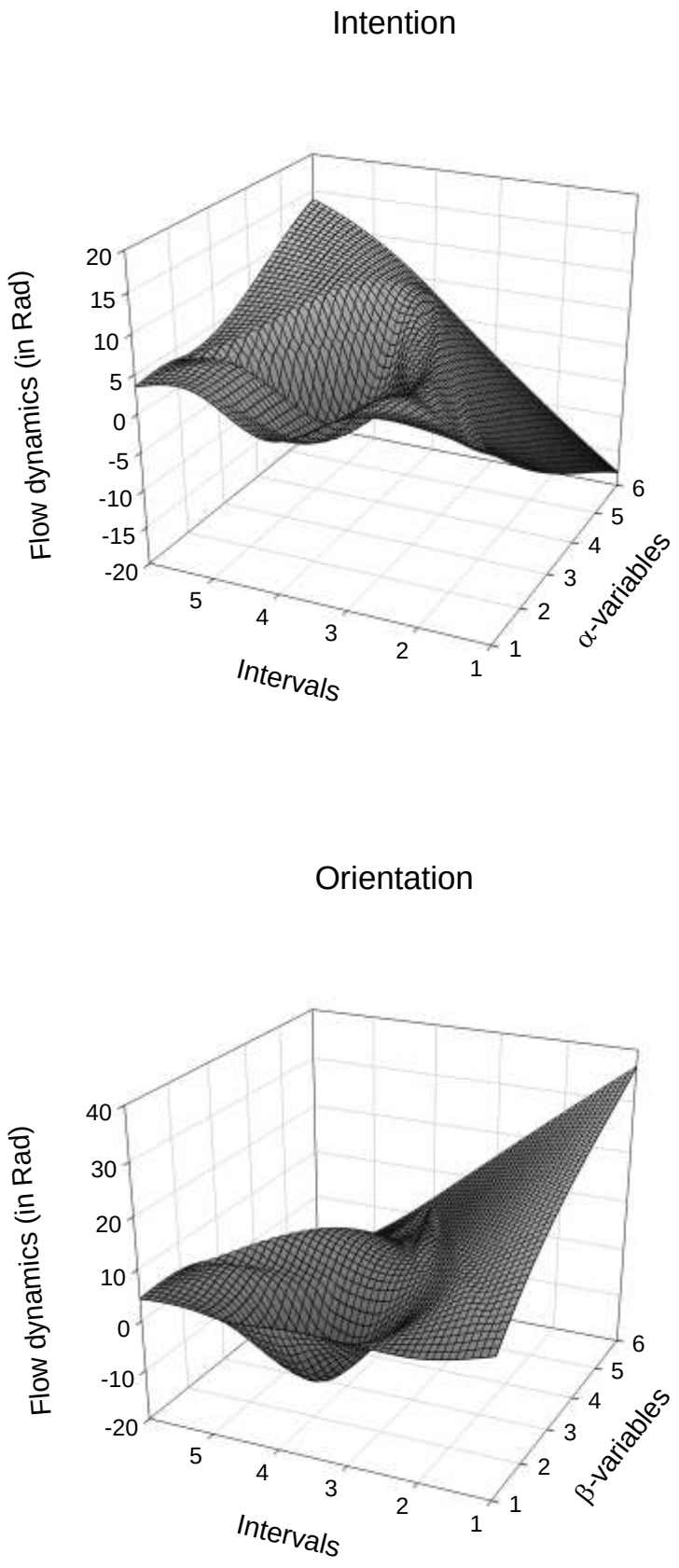


Figure 1 *Unfolded spaces*

Comment: A first comment to the shapes has to do with the fact that the transference of data from the table has been made from the left, like in ordinary reading. The reason for this way of working is simply that it seems most natural. It implies that reading the textual development out of the graphs has to start from the right instead.

Now you can begin familiarise yourself with the text the way it looks when it has been transformed into a graphical form and become similar to a waving fabric. In another connection we have named theses shapes texture, which seems to be a fitting term.

The *Intention* (A) component emerges with a dynamic watered shape (values fluctuating between (~+0.96 and ~+5.09) during the first three intervals. This movement is followed by a basin-like formation due to the first five cases of the fourth interval (lowest value is ~-5.59). Finally the texture smoothly moves up to the end (~+3.62).

The *Orientation* (O) component starts with a value around (~+7.31) in the first case of the first interval, which is pictured in the stretched edge in the lower right corner. At the second case of the second interval the highest value (~+10.12) can be seen as the texture starts going in waves. From the second case of the fourth interval the movement dives to (~-11.40), which is marked by the bowl-shape in the middle of the graph. A fairly deep fold is formed by the end of the same interval before it smoothly ends up in (~+4.52) at the edge of the bowl.

*Comment on (**A**) and (**O**):* It may be worthwhile commenting on (A) and (O) as unity. We would like to point out to you the complementary shapes of the graphs. Where there is dynamic movement in the Agent in the beginning, the development of the Objective is flat and where there is a sudden depth in the Objective, the Agent is shaded in several cases in a row. Thus, the fourth interval carries the focus of the text producer, i.e. the perspective.

Until now we have demonstrated that the open sphere of a language space is controlled by the displacement function, which means that this function is working in the direction of growing string-vectors. However, with respect to the viscous-elastic properties, determining the magnitude of shearing and straining, the function controls also rolling vectors, which can be observed to emerge in hyperbolic spaces. Hence, the effect of deformation becomes manifest through multi-layered entanglements, which are refracted through imperfect [AaO] units. For that reason, growing must be equated with rotational differences, which are the result of differences in energy, originally invested into perfect [AaO] units and the energy fused into imperfect units. Even more important is that this kind of differences appears at the thermodynamic level as waves, which relate to various forms of rotational acceleration.

Quite another question is what happens when the purpose is to get at the concentrated energy the text producer has invested. That will be our focus from step 10 on.

Step 10: Ordering of Variables within Intervals

The developed space of the text and the dynamics characteristic of the verbal flow show a picture of the text seen through a number of cooperating external measures. Quite naturally, the external measures cannot give an image of the internal properties of the text, i.e. the origin of the textual dynamics. We can name these properties with notions like pressure or focus. Thus it is a matter of concentration in the sense of energy consumption. A measure on the concentrated textual energy is based on a function for the way the text becomes folded and concentrated around certain places. To be able to discover such operator values we need a principle for grouping of variables, which means that the variables carry higher information values. This is another kind of values than the ones that give rise to the rotation.

With the point of departure in binary groups (G*), suggested by Connes (1994), numeric structures can be detected and made manifest. Connes' operator (Δ) function leads to remarkably simple measurements of leaves, branches and trees, which take into consideration coordinative interactions so that even small leaf can contribute to a determination of the appearance of a tree-structure.

At every moment the numerical value of the (Δ) function is half the difference between the effects of a reference value, resisting or non-resisting an operator value, which is taken as basis for fusion. By applying the operator (Δ/2) to the fusion processes, certain magnitudes become united. Since the procedure is using the two-fold matrix, the operator value is inserted in the upper left cell, while the contrasting reference value is to be found in the bottom right cell. The remaining cells are filled with zeros. In this way, the fusion algorithm identifies these cells with identity copies.

Thus, generating pairs implies that the values are enveloped by operations that are closing all open *sets*. Since this procedure is connected with Heisenberg's uncertainty principle (Greene, 1999, p. 424), it is, according to Mackenzie (1997), sufficient for generating the space of the entire standard model of elementary particle interaction. Hence, the attractors are resulting from [T=C⊗C], which constitutes the workspace. It follows that the concentration space must develop on two simple connection (C) matrices.

To summarise, the fusion algorithm works with a connection matrix, based on the association of two discrete points. It was noted that the algorithm has the obvious power of changing either point into the other. Since Connes' fusion operator works with only two points and their alter-egos, it means, according to Mackenzie (1997) that the algorithm is producing the extraordinary condition, namely that space can be represented with a pair of numbers on with classical arithmetic operations can be performed despite the fact that every point is twinned with an indistinguishable alter-ego. Therefore, the prominent aspect of progressive processing of a trace [T] is definable and constitutes an efficient folding of strings into complicated tree-structures. Finally, the relation expressed through the trace is the natural foundation of functional identity.

Now you need to set up a table again, one for each component and with two columns each. The sets should look like the groupings of the A- and O-components which are presented in Tables 13 and 14. You start from Table 12, which contains the Intervals. For each variable you create a row, and also for each marker of interval (punctuation mark, as defined in the Comment on p. 6). You have now one column for variable within interval and one for the value (radian). As soon as you find a value that exceeds the critical value you sort it out but save it for later use. When you have processed the entire text example and generated the solutions to the tables, they should look like Tables 13 and 14.

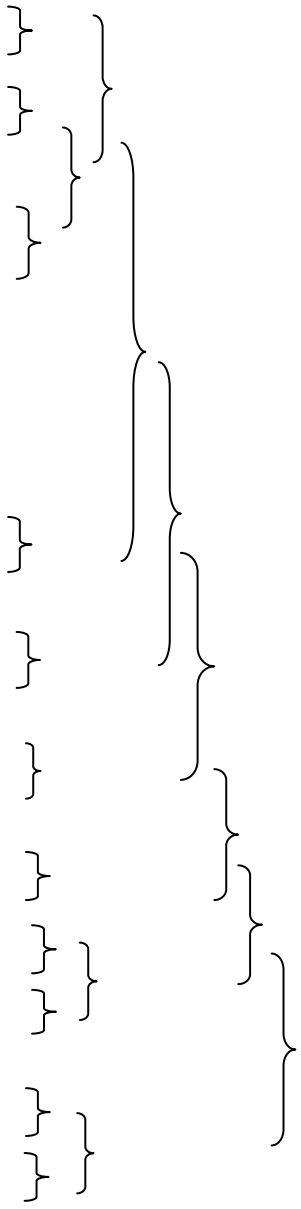
When you have finished this step you have for example got Table (13) with close values of the order (4, 5), (11, 12), (13, 14), (15, 16). All other values, namely (3, 6, 7, 8, 9, 10), have been sorted out for the present. For practical purpose, make space at the end of the table for these kinds of variables. In the same way, you will be able to set up Table 14 which gives you access to the grouping in the O-component.

The observed organisation implies that all differences should stay under the critical value of ($0 > \omega < 1$) in order for two values to form the basis for a groupoid. In this step you will compare the values within an interval with the critical value. If the mean value of a pair-wise grouping does not exceed the critical value, the value remains and is compared with the next value downwards in the table.

In the next step you will use the differences between radians within intervals to generate a transformation process out of the grouping, and emerging is a new kind of conservation principle.

Table 13
*Grouping in the **A**-component*

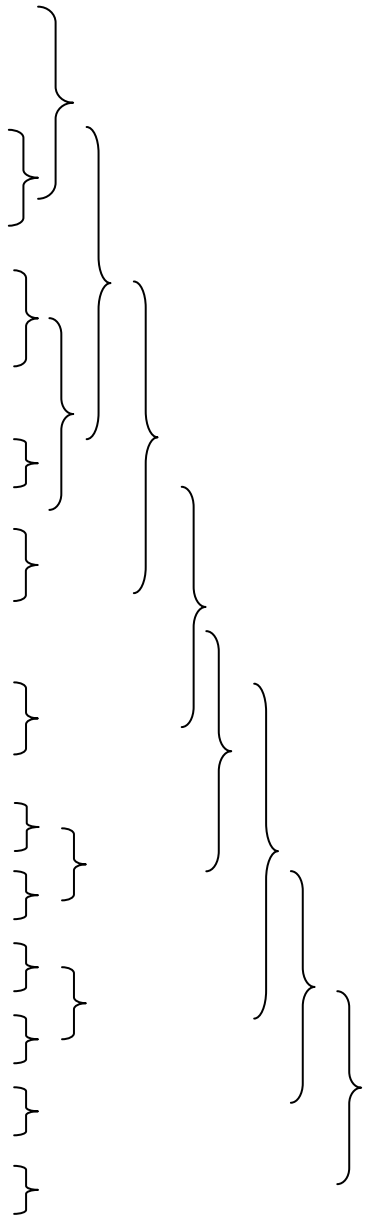
Variable	*Radian*
No	*α-strand*
.	
1	3.579600
,	
2	5.093017
3	
,	
4	4.427400
5	3.395861
6	
,	
7	
8	
9	
10	
11	4.3424
12	3.102877
,	
13	3.830800
14	3.542757
.	
15	3.516800
16	3.624687
.	
3	0.965102
,	
6	1.050653
,	
7	-0.025445
8	-3.90607
9	-5.59128
10	-5.59128



Comment: The introduced Zipper mechanism (B. Bierschenk, 2002) takes into account the time aspect of the acceleration, i.e. the dependency between variables that have been produced close to each other. Thus, according to this algorithm, the grouping is valid within the interval. Not until the interval is processed you may cross the border. In this way you eliminate universality, which would not be in line with the evolutionary theory behind the method. This principle will be developed further in the next section.

Table 14
*Grouping in the **O**-component*

Variable	*Radian*
No	*β-strand*
.	
1	
,	
2	3.987800
3	
,	
4	3.956400
5	
6	4.427400
,	
7	3.989700
8	
9	3.956400
10	
11	
12	4.0506
,	
13	3.9250
14	4.7100
.	
15	4.3646
16	4.5216
.	
11	0.467059
,	
5	-1.739490
,	
1	7.314300
,	
10	6.0759
,	
3	10.120000
,	
8	-11.4004



Step 11: Dimensioning of Time Dependent Nets

The convention you should use for generating a net is a coordinate system with a gradient of 45 degrees (rhombic form). You will need either paper and pencil or a drawing program that can set up the coordinates, as shown in Figure 2, because this gives you the best overview. The best position to start at is the upper left corner of the rhomb from where you can expand the net. For sure, you will get an approximate estimate of the size of the needed

net through the number of variables and dummies. But this is not enough. You must also give space for the possibility of closing them in a ring. You will soon realise that there are some construction principles you need to pay attention to.

Now continue by transferring the inmost bracket in the constructed tree (Step 10) into the approximated net. To number the dimensionality of the net, you had better start at the upper left corner and assign to the edge the value (0). The same start value applies for the upper line. Next position at the left dimension (vertical) thus will be (0:1). The left-right dynamics requires a further dimension (horizontal), which means that the first upper position gets the value (1:0). In order to determine the nesting in the O-component, you need to construct a new net like that of Figure 2.

To determine what the nesting looks like for the O-component you need to determine the time dependence between the variables of the binary groups (groupoids) within the periodic mesh system. For example you insert the value for (D) at the position (1:0) and the value for variable (β_2) at the position (0:1). Thereafter you need to transform the established bracket by marking their grouping with curved lines, ending in the node (T_1), which will occupy position (1:1). The value for (β_4) at position (0:2) and the value for variable (β_6) at position (0:3) are generating the node (T_2) at position (1:3). After that are the curved lines binding together point attractors, which attract the variables, with state attractors, which carry the singularities of the transformation. The transformation must take the form of a swallow's tail. If you do not manage to realise this form, you have made some mistake somewhere in the transformation process.

To repeat: with the given example, you have been able to notice that the construction begins in the second interval, because it contains the nearest distance between any two rotation values within the period. (T_2) is the root of the concentration process. After you have established the first singularities, you need to transit the border backwards to the first interval in order to reach the variable (β_2). Since this interval contains only one variable, but you need two for a groupoid to form, you retreat to the alter-ego (dummy) of the variable and mark it with the zero value. It follows that the singularity (T_1) is summing up the value of the groupoid. At this stage you are in the position to link (T_2) backwards to the first singularity (T_1). This link is stretching into the second dimension, where you now can establish the third singularity (T_3) at the position (2:3).

When you are ready with this second net, you have created two separate mesh systems, which now will be the basis for representing the fusion dynamics in forming energy concentrations in folded landscapes. As you may have realised, you mark a terminal for each and every variable and dummy. All terminals are to be found at the edges of a net. In conclusion, in the linking process you are asked to generate the singularities forwards and bind the resulting nodes backwards. The following principles should facilitate your work:

No line may cross one and the same cell twice
A line may not cross itself
The generated path must approximate the form of a ring

In the process you take the closeness in space and time among the point and state attractors into account. In looking at O-Mesh (Fig. 3) you will see that the alter-ego, when present, is mirroring the value of the variable of a groupoid. The construction of the A-Mesh (Fig. 2) does not differ from the principles just described. The most striking property of a net is the number of empty places. No net can consist of more than 74 % filled places (Wales, 2003, p. 12). Therefore, it is quite natural that you find holes in it, which lead to irregularities. And these are crucial for the upcoming analysis.

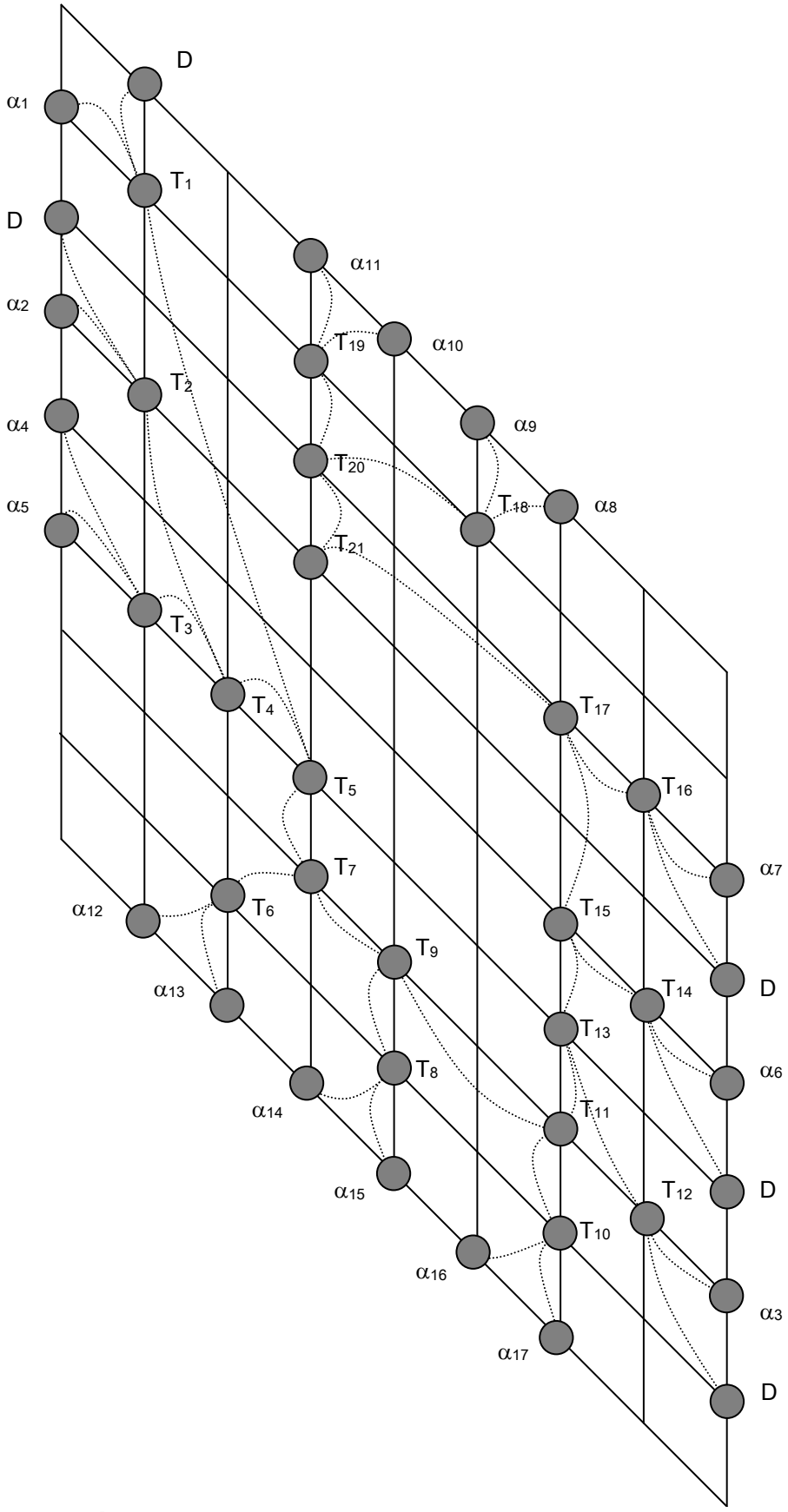


Figure 2 *A-mesh*

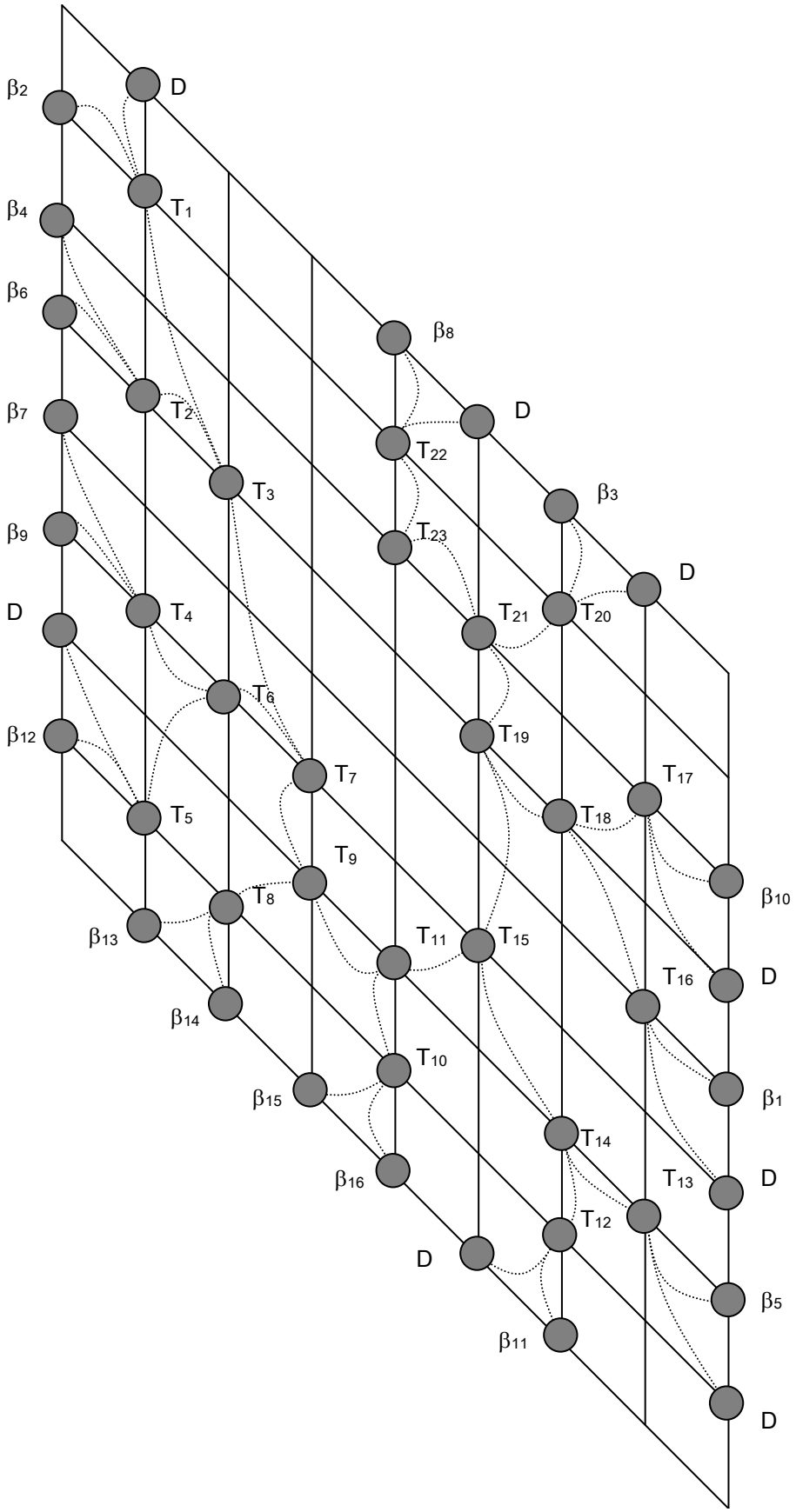


Figure 3 *O-mesh*

For assessing the meaning of the emergent state attractors of the Figures 2 and 3, it is essential to conceive the development of their ecological import as result of changes in their degree of complexity. Thus, coupling configurations, related to neighbourhood, prevent ambiguity, because time-dependent coupling, speed and reversible covalent interface smoothness are generating novel fitness conditions. This novelty is the result of the described connection dynamics, which is superior when compared to the functional architecture of classical clustering algorithms. Before that, though, you have to take step 12.

Step 12: Transfer of Coordinate Systems to Tables

Your next task is to transfer the net construction to table form. As you can notice in the Tables 15 and 16, you have to insert values for the empty places as well, i.e. (0). A zero rotation has the same effect in the table as a filler value. The necessary means will be one column for the nodes of the point and state attractors (Coordinates) and one for the empirical values (Radians).

Further, all intersections are not occupied by nodes, and the nodes are not evenly distributed over the net. On the other hand, you can also see that the configurations of the O- and A-components are surprisingly similar. The small variability that you may notice is of no import for the moment.

Table 15

Grid of the A-component

C	*Rad*	*C*	*Rad*	*C*	*Rad*	*C*	*Rad*	*C*	*Rad*
00	0	10	0	20	0	30	-5.59128	40	0
01	3.5796	11	3.5796	21	0	31	-5.59128	41	0
02	0	12	0	22	0	32	-15.08863	42	0
03	5.093017	13	5.093017	23	0	33	25.357856	43	0
04	4.4274	14	0	24	0	34	0	44	0
05	3.395861	15	7.823261	25	12.916278	35	16.495878	45	0
06	0	16	0	26	0	36	23.94115	46	31.314712
07	0	17	0	27	7.445277	37	0	47	7.373557
08	0	18	4.3424	28	3.102877	38	3.8308	48	3.542757

C	*Rad*	*C*	*Rad*	*C*	*Rad*	*C*	*Rad*
50	-5.59128	60	-3.90607	70	0	80	0
51	-9.49735	61	0	71	0	81	0
52	0	62	40.446486	72	-0.025445	82	-0.025445
53	0	63	0	73	0	83	0
54	0	64	40.471931	74	1.05063	84	1.05063
55	0	65	39.421301	75	0	85	0
56	0	66	38.456199	76	0.965102	86	0.965102
57	0	67	7.141487	77	0	87	0
58	3.5168	68	3.624687	78	0	88	0

Table 16
Grid of the O-component

C	*Rad*	*C*	*Rad*	*C*	*Rad*	*C*	*Rad*	*C*	*Rad*
00	0	10	0	20	0	30	0	40	-11.4004
01	3.9887	11	3.9887	21	0	31	0	41	-11.4004
02	3.9564	12	0	22	0	32	0	42	52.727859
03	4.4274	13	8.3838	23	12.3725	33	0	43	0
04	3.9897	14	0	24	0	34	0	44	0
05	3.9564	15	7.9461	25	11.9967	35	24.3692	45	0
06	0	16	0	26	0	36	33.5042	46	41.89904
07	4.0506	17	4.0506	27	8.635	37	0	47	8.8862
08	0	18	3.9250	28	4.71	38	4.364	48	4.5216

C	*Rad*	*C*	*Rad*	*C*	*Rad*	*C*	*Rad*
50	0	60	10.12	70	0	80	0
51	0	61	10.12	71	0	81	0
52	64.128859	62	0	72	6.0759	82	6.0759
53	54.008259	63	13.3902	73	0	83	0
54	0	64	0	74	7.3143	84	7.3143
55	40.618059	65	0	75	0	85	0
56	0	66	-1.272341	76	-1.7394	86	-1.7394
57	0	67	0.467059	77	0	87	0
58	0	68	0.467059	78	0	88	0

Step 13: Folding Landscapes

When you transfer the nodes and their values to a graph program (SigmaPlot, 2008, p. 148) you can see that the nodes refer to three dimensions, where the variables define the dimension (X), i.e., (x_1, x_2, x_3). The second dimension (Y), i.e., (y_1, y_1, $y_{1,}\ldots$, y_3) represents the layers and the third (Z) gives the values of the variables in running order. Even for this step you are advised to use a suitable graph program. When you have finished the transfer of your data, the program will ask you to specify the type of graph you prefer. The most appropriate is to choose 3D Mesh Plot according to Figure 4.

A folded landscape can be characterised from many perspectives. One is to regard the way in which they form into hills, mountains and valleys, i.e. from a natural geographic view. If you look at them like this, a simple description may offer itself:

The folded *A-landscape* depicts in the foreground a few concentrations, mainly under zero line. But on its right-hand side, it shows the formation of foothills with two peaks. A sloping formation is visible in the background, while three mountain peaks of equal height rise at the left-hand side.

The *O-landscape* shows in the foreground two deep valleys cutting into a mountain massif. Hence, a varied formation appears under the zero line. Further to the left emerges a mountain crest in front of a prolonged and narrow valley. Finally, another crest dominates the background. The O-landscape gives a more buckled and diverse impression than the A-landscape, which is more distinct.

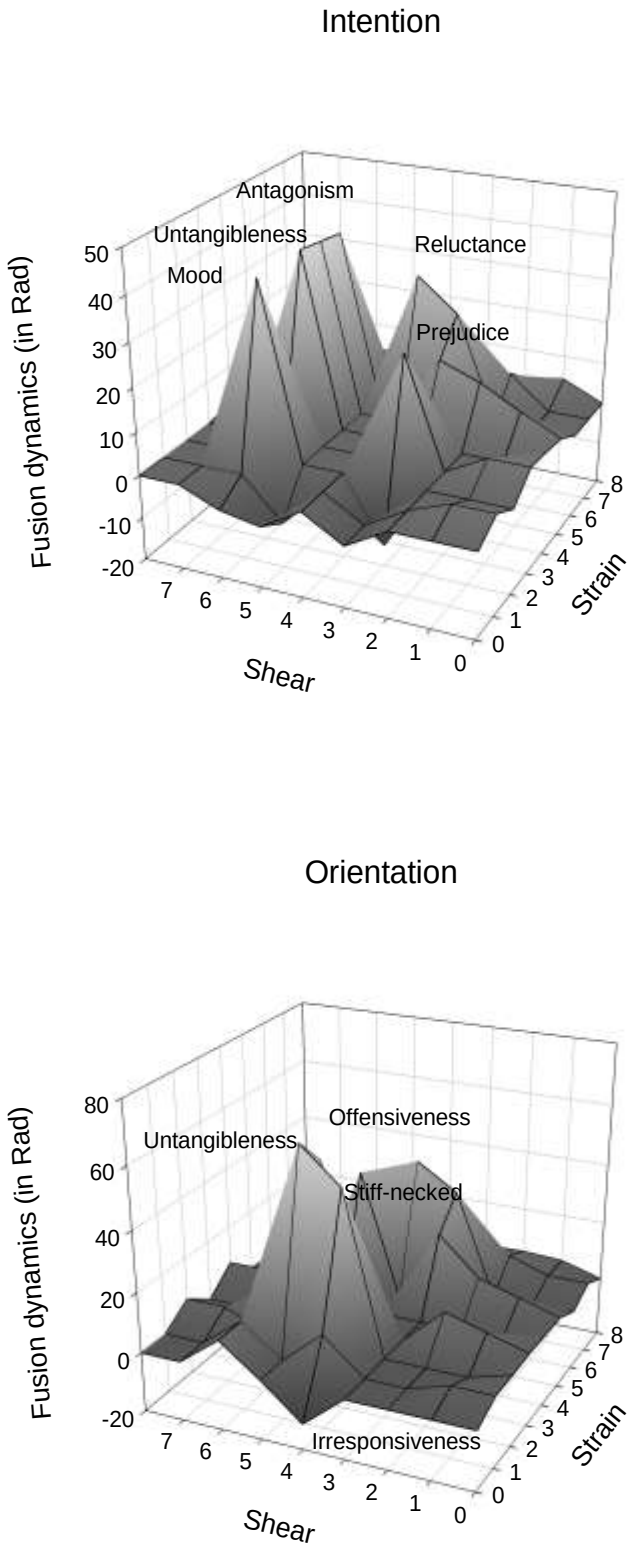


Figure 4 *Folded spaces*

Another description is the one that the land surveyors have, e.g. when measuring distance between various elevations in a landscape. Just like people do when they face an unknown landscape, you may, if you wish, specify various measuring points. Each topological point in the landscape is unambiguously defined through the nodes in the ground matrices of Figure 2 and Figure 3, which you are familiar with by now. If you use the unique denotations, it is easy for you to characterise the highest point, e. g. with (T_{21}), whose value is (~+64.13), the lowest point with (T_{22}), whose value is (~-11.40), and the end point of the process, which is to be found at (T_{23}=~+52.73). In this way unique points of reference will govern the continued discussion.

So far you have certainly observed that some information based on graphemes has not been used. Then, you may also understand that this analysis deviates markedly from all other more or less known text analysis methods. For sure you could stop the process here, just like other mountain climbers have done before several peaks in Himalaya (K…). But if you want to give name to (T…) and consequently some kind of meaning, you should go back to the surface of the text, because that is where the structure of the text is anchored, no doubt.

Excursus

Step 14: Naming the Singularities

For this process you are invited to follow on a supplementary and advanced study of the process of generating termini, starting with step 14. If you go back to Figure 4 you can see the result of the transformation process. You may get some feeling for the way in which the recommended procedure works with an example from Table 17.

You bind an O-string to the respective point attractor (edge value), which is the first step of this process. In the O-component you have for example the variable (β_4), which you connect to the string (*I*). Previously the curved line from the variable to the Singularity (T_2) had indicated that this string should be transformed to something else, which however no longer has any physical existence. For this reason you must take into account the second curved line, oriented towards (T_2). Since it connects variable (β_6) you need to get hold of the associated textual string, namely (*my salary*).

If the latter shall have any influence on the former it must result in something transformed (i.e., something new). Preliminary, you may settle on **Security** as the best approximation. Of course you may feel free to find an alternative description, which can catch the strength of this transforming step. However you reason, the result should be virtual, i.e. have no longer any direct correspondence with the physical context.

Step 15: Extraction of Descriptors

This step concerns the A-component and requires that you extract proper termini, which can describe the nodes of the A-mesh. The procedure of Table 18 will help you: You begin looking for the starting variable, which is a dummy (D) and follow the path to (α_1) you get to the singularity (T_{A1}). In order to find the corresponding path in the (**O**) mesh you switch to (β_1) and follow the path to (D).

Immediately before the (D), you will find the terminus for the description (T_{O16}), which is **Inclination**. Thus (T_{A1}) gets Inclination as its descriptor. In continuing, you transit from (D) to (α_2) of the other groupoid, which guides you towards (T_{A2}). Then you switch to the (**O**) mesh and follow the path of the corresponding (β_2) to (D). Thereby you have come in the position to extract the proper description for (T_{A2}). Just before this terminal (D) you find (T_{O1}) and extract its terminus **Denial** as proper description of (T_{A1}).

Table 17
Transformation of the β-variables

Node	*Value*	*Transformation*	*Node*	*Value*	*Transformation*
D	0		**T_{12}**	**0.467059**	**Dissonance**
2	3.9887	not	D	0	
T_1	**3.9887**	**Denial**	5	-1.73949	most people+I
4	3.9564	I	**T_{13}**	**-1.73949**	**Mood**
6	4.4274	my salary	*T_{12}*	*0.467059*	**Dissonance**
T_2	**8.3838**	**Security**	*T_{13}*	*-1.73949*	*Mood*
T_1	*3.9878*	**Denial**	**T_{14}**	**-1.272431**	**Adverseness**
T_2	*8.3838*	*Security*	*T_{11}*	*41.89904*	**Resentment**
T_3	**12.3725**	**Self-sufficiency**	*T_{14}*	*-1.272431*	*Adverseness*
7	3.9897	I	**T_{15}**	**40.618059**	**Antagonism**
9	3.9564	up	D	0	
T_4	**7.9461**	**Proposing**	1	7.3143	of the common attitude today
D	0		**T_{16}**	**7.3143**	**Inclination**
12	4.0506	money	D	0	
T_5	**4.0506**	**Income**	10	6.0759	ideas
T_4	*7.9461*	*Proposing*	**T_{17}**	**6.0759**	**Conception**
T_5	*4.0506*	*Income*	*T_{16}*	*7.3143*	*Inclination*
T_6	**11.9967**	**Negotiation**	*T_{17}*	*6.0759*	*Conception*
T_3	**12.3725**	**Self-sufficiency**	**T_{18}**	**13.3902**	**Prejudice**
T_6	**11.9967**	*Negotiation*	*T_{15}*	*40.618059*	*Antagonism*
T_7	**24.3692**	**Want for Nothing**	*T_{18}*	*13.3902*	*Prejudice*
13	3.925	not	**T_{19}**	**54.008259**	*Offensiveness*
14	4.71	a damn	D	0	
T_8	**8.635**	**Spite**	3	10.12	for the local government employees only
T_7	*24.3692*	*Want for Nothing*	**T_{20}**	**10.12**	**Trend**
T_8	*8.635*	*Spite*	*T_{19}*	*54.008259*	*Offensiveness*
T_9	**33.0042**	**Reluctance**	*T_{20}*	*10.12*	*Trend*
15	4.3646	the same	**T_{21}**	**64.128859**	**Untangibleness**
16	4.5216	here	D	0	
T_{10}	**8.8862**	**Sameness**	11	-11.4004	the local authority+money
T_9	*33.0042*	*Reluctance*	**T_{22}**	**-11.4004**	**Irresponsiveness**
T_{10}	*8.8862*	*Sameness*	*T_{20}*	*64.128859*	*Untangibleness*
T_{11}	**41.89904**	**Resentment**	*T_{22}*	*-11.4004*	*Irresponsiveness*
D	0		**T_{23}**	**52.727859**	**Stiff-necked**
11	0.467059	as how the local authority+money			

Table 18
Extraction of the termini from the O-mesh

A-component	*O-component*	*Terms/Names*	*Fusion*
Pendulum	*Destination*	*Extract*	*Value*
T_1: D → 1	T_{O16}	Inclination	*3.5796*
T_2: D → 2	T_{O1}	Denial	5.093017
T_3: 4 → 5	T_{O13}	Mood	7.823261
T_4: T_{A3} → T_{A2}	T_{O2}	Security	12.916278
T_5: T_{A4} → T_{A1}	T_{O3}	Self-sufficiency	16.495878
T_6: 11 → 12	T_{O8}	Spite	7.445277
T_7: T_{A6} → T_{A5}	T_{O5}	Income	23.94115
T_8: 13 → 14	T_{O8}	Spite	7.141487
T_9: T_{A8} → T_{A7}	T_{O9}	Reluctance	31.314712
T_{10}: 15 → 16	T_{O10}	Sameness	7.141487
T_{11}: T_{A10} → T_{A9}	T_{O11}	Resentment	38.456199
T_{12}: D→ 3	T_{O20}	Untangibleness	0.965102
T_{13}: T_{A12} → T_{A11}	T_{O15}	Antagonism	49.421301
T_{14}: D → 6	T_{O2}	Security	1.05063
T_{15}: T_{A14} → T_{A13}	T_{O13}	Mood	40.471931
T_{16}: D → 7	T_{O4}	Proposing	-0.025445
T_{17}: T_{A16} → T_{A15}	T_{O19}	Trend	40.446486
T_{18}: 8 → 9	T_{O22}	Irresponsiveness	-9.49735
T_{19}: D → 11	T_{O12}	Dissonance	-5.59128
T_{20}: T_{A19} → T_{A18}	T_{O18}	Prejudice	-15.08863
T_{21}: T_{A21} → T_{A17}	T_{O18}	Prejudice	25.357856

In continuing, you transit from (α_4) to (α_5) of the third groupoid, which guides you towards (T_{A3}). Then you switch to the (**O**) mesh and follow the path of the corresponding (β_4) to (β_5). This is a very long swing, which is resulting in the extraction of (T_{O13}). Hence **Mood** is the proper descriptor of (T_{A3}). The next shown pendulum swing starts with (T_{A3}) and then transits over to (T_{A2}), which is a closed loop and is therefore leading to the Singularity (T_{O2}).

Since (T_{O2}) is the terminus just before (T_{O3}) you extract as the proper name for (T_{A4}). You may finalise the emerging sub tree with the path of (T_{A5}), which starts in (T_{A4}) and ends in (T_{A1}). Consequently, you begin the path with (T_{O4}) and follow the swing, round the edge of (T_{O7}) over to variable (T_{O1}), which leads you to the Singularity (T_{O3}) which is **Self-sufficiency**. Hence the proper descriptor for (T_{A5}) is the terminus Self-sufficiency. In this way you continue the extraction process until the whole (**A**) mesh has got its terminological description. You find the solution to the entire mesh in Table 18.

Comment: Finally, it is worthwhile to remember that the AaO axiom stipulates that the Agent must get its description through the Objective, an axiom which has been validated empirically by the completion of this step.

References

Bierschenk, B. (1984). Steering mechanisms for knowability. *Cognitive Science Research*, 1. Lund University. (ERIC. ED 264 246)

Bierschenk, B. (1991). The schema axiom as foundation of a theory for measurement and representation of consciousness. *Cognitive Science Research*, 38. Lund University. (ED 338 650)

Bierschenk, B. (1993). The fundamentals of perspective text analys*is. Cognitive Science Research*, 45. Lund University.

Bierschenk, B. (2001). Geometric foundation and quantification of the flow in a verbal expression. *Cognitive Science Research*, 81. Copenhagen University & Lund University. (ED 459 193)

Bierschenk, B. (2002). Real time imaging of the rotation mechanism producing interview-based language spaces. *Cognitive Science Research*, 83. Copenhagen University & Lund University. (ED 465 812)

Bierschenk, B. (2005). Differentiated limits for knowability. *Cognitive Science Research*, 97. Copenhagen University & Lund University.

Bierschenk, B. (2011). Functional text geometry: The essentials of Perspective Text Analysis. *Cognitive Science Research*, 101. Copenhagen University & Lund University. (Lund University: Open Access)

Bierschenk, B., & Bierschenk, I. (1976). *Computer-based content analysis.* (Studia Psychologica et Paedagogica, 32) Lund: Gleerup.

Bierschenk, B., & Bierschenk, I. (1986a). Concept formulation. Part II. Measurement of formulation processes. *Cognitive Science Research*, 11. Lund University. (ED 275 160)

Bierschenk, B., & Bierschenk, I (1986b). Concept formulation: Part III. Analysis of mentality. *Cognitive Science Research*, 12. Lund University. (ED 275 161)

Bierschenk, B., & Bierschenk, I. (1993) Perspektivische Textanalyse [Perspective Text Analysis]. In: E. Roth (Ed.), *Sozialwissenschaftliche Methoden [Social Science Methods]* (pp.175-203). München: Oldenbourg Verlag. (In German)

Bierschenk, B., & Bierschenk, I. (2011, July). Functional text geometry (1-5). [On-line]. Available: http://www.sites.google.com/site/aaoaxiom/tutorials

Bierschenk, B., Bierschenk, I., & Helmersson, H. (1996). Die Ökologie des Sprachraums [The ecology of the language space]. In: W. Bos, & C. Tarnai (Eds.), *Computerunterstützte Inhaltsanalyse in den Empirischen Sozialwissenschaften. Theorie - Anwendung - Software [Computer aided content analysis in the empirical social sciences. Theory - Application - Software]* (pp. 11-31). München: Waxmann. (In German)

Bierschenk, I. (1977). Datorbaserad innehållsanalys: Kodningsmanual [Computer-based content analysis: Coding manual]. *Pedagogisk dokumentation, 52*. Lund University, Malmö: Department of Educational and Psychological Research.

Bierschenk, I. (1984). The schematism of natural language. *Cognitive Science Research*, 2. Lund University. Also published in O. Togeby (Ed.), *Papers from the 8th Scandinavian Conference of Linguistics* (pp. 73-78).

Bierschenk, I. (1992). An excursion into the ecological co-ordinates of language space. *Cognitive Science Research*, 43. Lund University, Department of Psychology.

Bierschenk, I. (1999). The essence of text. A dialogue on Perspective Text Analysis. *Cognitive Science Research*, 70. Copenhagen University & Lund University. (ERIC, ED 430 053)

Bierschenk, I. (2011a). Ett ekologiskt perspektiv på språk och textanalys. [An ecological perspective on language and text analysis]. *Cognitive Science Research*, 98. Copenhagen University & Lund University. (In Swedish) (Lund University: Open Access).

Bierschenk, I. (2011b). Applications of perspective text analysis. A thematic overview. *Cognitive Science Research*, 99. Copenhagen University & Lund University. (Lund University: Open Access).

Bierschenk, I., & Bierschenk, B. (2004). Diagnose der Leistungsheterogenität durch die Perspektivische Textanalyse: VERTEX [Diagnosing heterogeneity in achievement through the Perspective Text Analysis: VERTEX]. In: W. Bos, Lankes, E.-M., Plaßmeier, N., & Schwippert, K. (Eds.), *Heterogenität: Eine Herausforderung an die empirische Bildungsforschung [Heterogeneity: A Challenge to the empirical research in education]* (pp. 16-28). Münster: Waxmann. (In German)

Connes, A. (1994). *Noncommutative geometry*. New York: Academic Press.

Greene, B. (1999). *The elegant universe. Superstrings, hidden dimensions, and the quest for the ultimate theory*. New York: W. W. Norton & Company.

Hestenes, D. (1986/1993). *New foundations for classical mechanics*. Dordrecht: Kluwer Academic.

Mackenzie, D. (1997). Through the looking glass. In arithmetic 5 and 7 can be added in any order to yield 12. When order does matter, you have entered the strange, disorientating world of noncommutativity. *The Sciences, 37*(3), 32-37.

SigmaPlot (2008). *Exact graphs for exact science. User's manual* (Version 11). Chicago: SPSS Inc.

Wales, D. J. (2003). *Energy landscapes. With applications to clusters, biomolecules and glasses.* Cambridge: Cambridge University Press.

Accepted September 5, 2011