Languages for Gestalts of Line Patterns

Mehdi Dastani

Institute of Information and Computing Sciences, Faculty of Mathematics and Computer Science, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, The Netherlands.

and

Remko Scha

Institute for Logic, Language, and Computation, University of Amsterdam. Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands.

E-mail: mehdi@cs.uu.nl , scha@hum.uva.nl

Any formal model of visual Gestalt perception requires a language for representing possible perceptual structures of visual stimuli, as well as a decision criterion that selects the actually perceived structure of a stimulus among its possible alternatives. This paper discusses an existing model of visual Gestalt perception that is based on Structural Information Theory. We investigate two factors that determine the representational power of this model: the domain of visual stimuli that can be analyzed, and the class of perceptual structures that can be generated for these stimuli. We show that the representational power of the existing model of Structural Information Theory is limited, and that some of the generated structures are perceptually inadequate. We argue that these limitations do not imply the implausibility of the underlying ideas of Structural Information Theory and introduce alternative models based on the same ideas. For each of these models, the domain of visual stimuli that can be analyzed properly is formally defined. We show that the models are conservative modifications of the original model of Structural Information Theory: for cases that are adequately analyzed in the original model of Structural Information Theory, they yield the same results.

Key Words: visual perception, Gestalt, structural information theory, minimum principle, perceptual structures.

1. INTRODUCTION

Every sensory pattern has many potential structures. Human perceivers usually prefer one of these structures. For example, the visual pattern illustrated in Figure

1-A may have, among other ones, the two structures illustrated in Figure 1-B and 1-C. In this case, human perceivers tend to prefer the structure that is illustrated in Figure 1-B.

FIG. 1. *Visual pattern A has two potential structures B and C. Structure B is perceptually preferred.*

Several studies have tried to articulate the principles that underlie the human perception of sensory patterns and to predict the perceived (i.e. perceptually preferred) structures. Early work in this area was initiated by (Wertheimer, 1923). Wertheimer proposed several factors such as similarity, continuity, and proximity, in terms of which the perceived structure of sensory patterns can be explained. For example, the proximity factor causes closely positioned sensory elements to be perceived as members of one perceptual group; the similarity factor causes similar sensory elements to be perceived as forming one perceptual group. The problem is, however, that no prioritization for these factors is proposed. Thus, whenever different factors predict different structures for a pattern, the perceived structure of that pattern cannot be explained. For example, the proximity and similarity factors predict two different structures for the pattern illustrated in Figure 2. Although in this pattern proximity wins from similarity (i.e. the perceived structure of this pattern is determined by proximity), one cannot generalize this observation and state that proximity always wins from similarity.

FIG. 2. *The proximity factor, rather than the similarity factor, dominates perceptual grouping.*

Koffka (Koffka, 1935) emphasized that many of the Gestalt factors may be viewed as instances of one rather general principle, called Prägnanz, which predicts that the simplest structure of a pattern, among its possible structures, is the one that is actually perceived. The Prägnanz principle does not state what kind of pattern structures should be generated, nor does it provide a measure to determine the simplicity of pattern structures. Leeuwenberg (Leeuwenberg, 1971) addressed these questions by developing the Structural Information Theory, henceforth SIT. The fundamental concept proposed by SIT is descriptive simplicity: the perceived structure of a pattern is reflected by the simplest description of that pattern. The simplest description of a pattern is defined as the description which captures the largest amount of perceptually relevant regularity in that pattern.

In Structural Information Theory, the perceptually relevant regularities of patterns are specified as hierarchical arrangements of identical pattern parts. In order to specify possible descriptions of a pattern and decide on the simplest one, Leeuwenberg and his colleagues proposed a coding model. In this coding model, which we will briefly explain in the next section, patterns are represented as one-dimensional symbol sequences, i.e. strings; different occurrences of a substring are considered as identical pattern parts. These sequential patterns are then analyzed by means of a number of operators. Each operator is responsible for capturing a specific type of perceptually relevant pattern regularity. To compare the simplicities of the resulting descriptions, a complexity measure, called information load, is introduced.

The current paper is concerned with the coding model proposed by Leeuwenberg and Van der Helm (Leeuwenberg, 1971; Van der Helm & Leeuwenberg, 1991). We follow the basic idea of SIT which states that the perceived structure of a pattern is reflected by the description of that pattern which captures the largest amount of perceptually relevant regularity. However, we discuss various inadequacies of the coding model that was proposed. In particular, we argue that the coding model is limited; it can be applied only to a small class of perceptual patterns that can be unambiguously encoded as sequential patterns. Moreover, we show that the coding model as it stands is ambiguous; several of its assumptions are not explicitly specified. Finally, we give examples to demonstrate that some of the pattern regularities used in the coding model are not perceptually relevant.

We present alternative coding models by developing formal languages with expressions which denote perceptual patterns, such that the formal structure of the expressions represents the perceptual structure of the denoted patterns. For each of these languages a complexity measure is defined that assigns natural numbers (complexity values) to its expressions. We show in detail that our coding models generalize and modify the original SIT proposal in a constructive way; i.e., to the extent that SIT's predictions seemed plausible, they are preserved.

The paper is structured as follows. In section 2, Structural Information Theory and its existing coding model is explained. The domain of perceptual patterns that can be analyzed by the coding model is discussed and some minor modifications of SIT are proposed. In section 3, we formalize the coding model of modified SIT by proposing a language with a formal syntax and semantics together with a complexity measure defined on its expressions. It is shown that the perceived structure of a sequential pattern, as predicted by modified Structural Information Theory, is represented by the expression of the proposed language that denotes the pattern and has the lowest complexity value. In section 4, the proposed language is generalized to represent all two-dimensional line patterns and their perceptual structures. A complexity measure is designed for this language and it is shown that the perceived structure of a one-dimensional line pattern as predicted by the modified SIT is the expression that denotes the pattern and that has the lowest complexity value. Section 5 discusses limitations of the current proposal and looks at its prospects for further development.

2. STRUCTURAL INFORMATION THEORY

Structural Information Theory (SIT) is a general theory of pattern perception which tries to explain why humans consistently perceive certain pattern structures. The explanation is based on the assumption that the human perceptual system is sensitive to certain types of structural regularities of patterns. A structural regularity of a pattern is a hierarchical arrangement of identical pattern parts. For example, the regularity of the string $abab$ may be defined in terms of the identity of the first and the third characters a, the second and the fourth characters b, and the first and the second substring ab. Note that these identities are hierarchically ordered: the identity of the substrings ab implies the identities of the a's and the b's.

According to SIT, only certain types of structural regularities are perceptually relevant. These regularities are specified by the ISA operators: Iteration, Symmetry and Alternation. These operators are conjectured to reflect innate principles of mental representation. A pattern can be described by means of the ISA operators in different ways. The resulting descriptions indicate different Gestalts (perceptual structures) of the pattern. In order to disambiguate the set of alternative structures and to decide on the perceived structures of the pattern, a complexity measure is introduced. The complexity measure is defined on pattern descriptions and it indicates the lack of perceptual regularity within a pattern. It is claimed that the description of a pattern with the lowest complexity value describes the pattern in the most simple and cognitively economical way, and thus indicates the perceived structure of the pattern. The idea that the simplest description of a pattern indicates the perceived structure of that pattern is called the simplicity principle. Of course, some patterns may not have a unique simplest description; more than one description with the same minimum complexity value may exist. These patterns are ambiguous; they do not have a unique perceived structure.

In this way, SIT relates descriptive simplicity (i.e. simplicity of pattern descriptions) to phenomenal simplicity (i.e. simplicity of perceptual organizations). This distinguishes SIT from description theories which employ arbitrary operations to encode patterns in the most compact way, without claiming that the resulting descriptions indicate the phenomenal simplicity of the patterns (Li & Vitányi, 1993; Grünwald, 2000). For example, compression systems in computer science reduce the number of bits needed to store binary data. These compression systems employ various kinds of regularities, but there is no claim that the more compressed description reflects a meaningful structure of the binary data. SIT differs from these kinds of description theories in that its compression operators (ISA reductions) and complexity measure are perceptually motivated.

The set of ISA operators and the complexity measure, as originally proposed in (Leeuwenberg, 1971), have undergone several revisions until Van der Helm and Leeuwenberg put them on a formal basis (Van der Helm & Leeuwenberg, 1991). They introduced a criterion called accessibility which is based on a formal analysis of regularity and hierarchy. A number of experiments have tested predictions based on the simplicity principle. The predictions were, on the whole, confirmed by the experiments ((Van der Vegt, Buffart, & Van Leeuwen, 1989), (Boselie, 1988), (Boselie & Wouterlood, 1989), (Buffart, Leeuwenberg, & Restle, 1981), (Van Leeuwen & Buffart, 1989), (Van Leeuwen, Buffart, & Van der Vegt, 1988), and (Stins & Van Leeuwen, 1993)).

2.1. The Descriptions of Patterns by SIT

Although Structural Information Theory is intended as a general theory of pattern perception, it is built on a model for describing strings (character patterns). Strings are described by means of the Iteration, Symmetry, Right-Alternation and Left-Alternation operators. The domain of strings is not formally defined in the SIT literature. We now first provide a formal definition of this domain and then define the ISA operators.

Definition 2.1. Let A be a set of primitive symbols. The domain D_s of strings over A is defined as follows:

- $A \subset D_s$
- If $X_i \in D_s$ for $1 \leq i \leq n$, then $X_1 \cdots X_n \in D_s$
- If $X \in D_s$, then $(X) \in D_s$

In our examples, we will choose A to be the set of lower case letters of the English alphabet, i.e. $A = \{a, b, \ldots\}$. Note that the domain of strings does not only contain sequences of primitive symbols; sequences may also contain (possibly nested) parentheses.

Definition 2.2. Let $X, X_1, \ldots, X_n$ stand for any string in D_s; let m stand for any natural number greater than one. For $n \geq 1$, the Iteration, Symmetry, Right- and Left-Alternation operators are defined by the following schemata:

$$\begin{array}{ll}
Iteration: & m * (X) \ = \ \overbrace{X \cdots X}^{m\ times} \\
Symmetry: & S[(X_1)(X_2) \cdots (X_n)] \ = \ X_1 X_2 \cdots X_n X_n \cdots X_2 X_1 \\
Symmetry: & S[(X_1)(X_2) \cdots (X_n), (X)] \ = \ X_1 X_2 \cdots X_n X X_n \cdots X_2 X_1 \\
Right\ alternation: & < (X) > / < (X_1)(X_2) \cdots (X_n) > = \ X X_1 X X_2 \cdots X X_n \\
Left\ alternation: & < (X_1)(X_2) \cdots (X_n) > / < (X) > = \ X_1 X X_2 X \cdots X_n X
\end{array}$$

In this definition, two versions of the symmetry operator are defined to cover symmetrical strings with a pivot element (odd symmetry) and symmetrical strings without a pivot element (even symmetry). The following examples illustrate the description of strings by means of the ISA operators.

$$\begin{array}{lll}
Iteration: & 3 \star (a) & = \ aaa \\
Symmetry: & S[(a(b))(cd)] & = a(b)cdcd(b)a \\
Symmetry: & S[(a)(b)(cd), (e)] & = abcdecdba \\
Right\ Alternation: & < (a) > / < (b)(cd)(efg) > & = abacdaefg
\end{array}$$

The parentheses that occur in the arguments of the ISA operators indicate the constituents of these arguments. These constituents are called chunks. For example, in $S[(a)(bc)]$, the chunk bc is considered as one constituent of the first argument of the symmetry operator. Therefore, $S[(a)(bc)]$ and $S[(a)(b)(c)]$ describe respectively $abcbca$ and $abccba$. In the first string, two primitive elements are joined into one chunk while in the second string each primitive element is considered as a chunk by itself.

The descriptions consisting of applications of ISA operators to strings are called ISA forms. When a string as a whole cannot be described by an ISA operator, the string may be divided into several substrings, called clusters, such that each cluster can be described by the ISA operators. For example, the string $abcbcaefef$ may be analyzed as $S[(a)(bc)]\ 2 * (ef)$ when it is divided into the two clusters $abcbca$ and $efef$.

A string can be described by applying the ISA operators recursively since the arguments of an ISA operator are themselves strings. The nested structure of such a description assigns a hierarchical organization to the string that is being described. For example, a possible clustering and a nested structure of the string $aabbccccbbaa$

may be specified in the following way. At the first hierarchical level, the whole string can be analyzed by the symmetry rule and thus it can be considered as one cluster:

$$aabbccccbbaa \rightarrow S[(a)(a)(b)(b)(c)(c)]$$

The argument of the symmetry operator (i.e. $(a)(a)(b)(b)(c)(c)$) is itself a string which can be analyzed further. The structure of this argument determines the clusters at the next lower level. Because the string $(a)(a)(b)(b)(c)(c)$ cannot be analyzed by one ISA operator, it is divided into three clusters. Thus, at the second nested level the string can be analyzed as three clusters, i.e.

$$(a)(a)(b)(b)(c)(c) \rightarrow 2*((a))\ 2*((b))\ 2*((c))$$

The complete nested description of the string $aabbccccbbaa$ is as follows:

$$S[(2*((a)))\ (2*((b)))\ (2*((c)))]$$

The recursive application of ISA operators describes a string in terms of its regularities and therefore it reduces the number of primitive elements that are needed to describe the string. A description which cannot be analyzed further is called a final description. Different final descriptions indicate different Gestalts of a string. For example, $abS[(a)(b)]$ and $2*(ab)ba$ indicate different Gestalts of the string $ababba$.

2.2. Information Load

In order to decide on the perceived structures of patterns, a complexity measure called information load is introduced. Information load measures the complexity of pattern descriptions by adding two quantities. The first quantity is the number of occurrences of primitive elements in the description, and the second quantity is the number of occurrences of chunks that contain more than one primitive element. For example, the string $abcddcab$ may be described as $S[(ab)(c)(d)]$; this (final) description contains 4 primitive elements a, b, c and d and there is one chunk ab that contains more than one primitive element. Therefore, the information load of the description $S[(ab)(c)(d)]$ is $4 + 1 = 5$. Similarly, the string $abcdcdab$ may be described as $S[(ab)(cd)]$ which has four primitive elements and two chunks that contain more than one primitive element: ab and cd. The information load of this description is 6. Note that the chunked elements need not be primitive elements. For example, the string $fabccabeabccabef$ may be analyzed as $S[(f)(S[(ab)(c)](e))]$ in which the non-primitive element $S[(ab)(c)]$ constitutes one chunk together with primitive element e. In this description, there are five primitive elements and two chunks that contain more than one primitive element: $S[(ab)(c)](e)$ and ab. The information load for this description is 7. A detailed discussion of this complexity measure can be found in (Van der Helm, Van Lier, & Leeuwenberg, 1992). The following recursive definition is equivalent to their less formal description.

Definition 2.3. Let A be the set of primitive symbols, D_s be the domain of strings over A, and $t, t_1, \ldots, t_n$ stand for arbitrary SIT descriptions of strings from D_s in terms of the ISA operators. The information load function I^{sit}, which assigns natural numbers to SIT descriptions, can be recursively defined as follows:

- if $t \in A$, then $I^{sit}(t) = 1$
- $I^{sit}(m * (t)) \ = \ I^{sit}((t))$
- $I^{sit}(S[(t_1)\cdots(t_n)]) \ = \ I^{sit}((t_1)\cdots(t_n))$
- $I^{sit}(S[(t_1)\cdots(t_n),(t)]) \ = \ I^{sit}((t_1)\cdots(t_n)) + I^{sit}((t))$
- $I^{sit}(<(t)>/<(t_1)\cdots(t_n)>) \ = \ I^{sit}((t)) + I^{sit}((t_1)\cdots(t_n))$
- $I^{sit}(<(t_1)\cdots(t_n)>/<(t)>) \ = \ I^{sit}((t)) + I^{sit}((t_1)\cdots(t_n))$
- $I^{sit}((t)) = I^{sit}(t)$ if t contains only one element from A;
  $= I^{sit}(t) + 1$ otherwise
- $I^{sit}(t_1\cdots t_n) = \sum_{i=1}^{n} I^{sit}(t_i)$

The information load provides a measure for ordering the final descriptions of a pattern. A final description of a pattern which has the lowest information load is called a simplest description of that pattern. If there exists a unique simplest description for a pattern, that description is considered as indicating the perceived structure of the pattern. For example, $2*(ab)ba$ and $abS[(a)(b)]$, which are two possible final descriptions of the string *ababba*, have information loads 5 and 4, respectively. Therefore, the final description $abS[(a)(b)]$ is preferred over the final description $2*(ab)ba$, which implies that *ababba* is preferably perceived as consisting of *ab* and *abba* rather than as consisting of *abab* and *ba*. In contrast, if there exists no unique simplest description for a pattern, then the pattern will be considered as ambiguous, i.e. as having no definite perceptual structure. For example, $aS[(b),(a)]$ and $2*(ab)$, which are two possible final descriptions of the string *abab*, have the same information load 3. Therefore, the string *abab* is considered as ambiguous. SIT claims that if a pattern is not ambiguous and thus has a unique perceived structure, then there is a unique SIT description with minimum information load for the pattern, and vice versa. In the rest of this paper, we call this the uniqueness claim of SIT. In the following we write $E^{sit}(s)$ to refer to the set of final SIT descriptions of string s. The subset of $E^{sit}(S)$ that consists of SIT descriptions with minimum information load according to definition 2.3 is denoted by $Perceived_{sit}(s)$, i.e.

$$Perceived_{sit}(s) = \{e \in E^{sit}(s) \mid \forall e' \in E^{sit}(s)\ I^{sit}(e) \leq I^{sit}(e')\}$$

The set $Perceived_{sit}(s)$ is called the set of SIT-predicted structures of string s.

Definition 2.4. Let s be a string in D_s. The uniqueness claim of SIT can be summarized as follows:

$$|Perceived_{sit}(s)| = 1 \Leftrightarrow s \text{ is not ambiguous}$$

The explicit claim of SIT is that the perceived structures of a pattern are indicated by its simplest descriptions. In this way, Structural Information Theory formalizes the Prägnanz notion of Gestalt psychology in terms of descriptive economy: the information load of what is actually perceived is lower than the information load of what could have been perceived alternatively.

2.3. Encoding of Line Patterns in SIT (Representation Hypothesis)

SIT is presented as a general theory of pattern perception and it is claimed to be applicable to various domains of perceptual patterns (Leeuwenberg, 1971). A fundamental assumption of SIT is what may be called the representation hypothesis. According to this hypothesis, perceptual patterns can be encoded as sequential patterns (strings) in such a way that the ISA regularities of the sequential patterns reflect the perceptual regularities of the encoded patterns. The perceptual structure of patterns is thus assumed to be invariant under the encoding process. The strings which encode perceptual patterns are called primitive codes.

For example, a monophonic piece of music consisting of an uninterrupted sequence of tones of equal durations is represented as a sequence of numbers, each representing one pitch; an uninterrupted two-dimensional line pattern consisting of connected line segments is represented as a sequence of letters, each representing either the length of a line segment or the angle between two subsequent line segments. When line segments are not connected to each other, additional letters are introduced to denote invisible line segments. For three-dimensional line patterns, 'roll' and 'pan' angles are distinguished.

SIT is mainly applied to uninterrupted two-dimensional line patterns consisting of straight line segments. A primitive code for such a visual line pattern is constructed by choosing an appropriate starting point, tracing the line segments, and concatenating symbols representing the successive lengths and their relative angles. Examples of such representations are illustrated in Figure 3.

FIG. 3. *Encoding line patterns as strings.*

2.4. Limitations of the Coding Model

We have our doubts about the representation hypothesis of SIT even with respect to line patterns. In Structural Information Theory and in its first computational model (Van der Helm & Leeuwenberg, 1986), where the ISA reduction process is partially implemented, the subjects of analysis are primitive codes (strings) instead of visual line patterns. The step needed to represent an arbitrary visual line pattern as a primitive code is, however, not a trivial one: for an arbitrary visual line pattern, different primitive codes are possible.

The multiplicity of primitive codes can be explained by considering a line pattern as a graph where line segments are edges and junctions of line segments are nodes. A primitive code of a line pattern is then a full path (a path that contains all edges and each edge visited exactly once) in its corresponding graph. It may be the case that in the graph of a line pattern no full path exists at all. In such a case, additional invisible line segments can be introduced such that full paths come to exist. However, as the graph of a line pattern may contain loops, or nodes with degrees greater than two (i.e. more than two edges ending in one node), several different full paths, with corresponding primitive codes, may exist.

For example, consider the line pattern in Figure 4-A and note that the perceived structure of this pattern consists of two squares and one diamond. For this pattern many traces are possible depending on where the trace starts, how it continues, and if invisible line segments are added. One possible trace of the line segments, without introducing invisible line segments, is illustrated in Figure 4-C. Another trace is illustrated in Figure 4-B, where an invisible line segment (dotted line segment) is added.

The problem with multiple primitive codes of visual line patterns is that different primitive codes of a visual line pattern do not necessarily result in the same set of final codes. This implies that the simplest description computed from an arbitrary primitive code of a line pattern may not represent the perceived structure of the visual line pattern. In other words, a wrong choice of primitive code results in a wrong prediction of perceived structure. For example, the perceived structure of Figure 4-A cannot be generated from the primitive code shown in Figure 4-C since in this primitive code the perceived diamond is split into two parts.

FIG. 4. *Two possible encodings of one line pattern.*

The process of finding an appropriate primitive code for an arbitrary visual line pattern is not a trivial process. The fact that in the Leeuwenberg tradition this process is done by hand and is not included in the algorithm is an essential limitation. One suggestion to solve this problem is obvious: the simplest description of a pattern should be selected from the set of final descriptions that can be generated from all possible primitive codes of that pattern. It should be noted that such a solution implies that the computational complexity of the polynomial algorithm for finding the perceived structure of strings, proposed by Van der Helm and Leeuwenberg, becomes exponential (Van der Helm & Leeuwenberg, 1986).

There is also a more fundamental problem with this solution. Consider again the visual line pattern shown in Figure 4-A. The perceived structure of this line pattern can be generated from the primitive code 4-B, but not from the primitive code 4-C. Nevertheless, primitive code 4-C yields a final description which has a lower information load than any of the final descriptions of primitive code 4-B. This observation implies that the simplest description generated from the set of all primitive codes does not necessarily represent the perceived structure of a visual line pattern.

All this does not imply, however, that the simplicity principle is a problematic concept. The problem might reside in the representation hypothesis: the Gestalt of the line pattern in Figure 4-A may not be the Gestalt of a linear sequence of its line segments. Representing the line pattern in Figure 4-A as a sequence of its line segments may introduce artefacts in the resulting analysis. We therefore make the following conjecture.

Conjecture 1. The representation hypothesis of SIT, which states that arbitrary line patterns can be represented as sequential patterns in such a way that the SIT-predicted structure of the sequential patterns determines the SIT-predicted structure of the encoded line patterns, is not correct. ■

The above considerations suggest that there is one constraint which must be satisfied by a line pattern to make the representation hypothesis applicable: the line pattern should have a set of demonstrably equivalent primitive codes. This constraint is satisfied by line patterns in which the maximum number of line segments that meet at one point is two and for which there exist exactly two points at which only one line segment ends. We call these patterns linear line patterns. They can be characterized by graphs in which each edge represents one line segment, there are exactly two nodes with degree one, and all other nodes have degree two.

FIG. 5. *Examples of linear line patterns.*

In Figure 5, some examples of linear line patterns are shown. It should be observed that in linear line patterns there are exactly two starting points from which the sequence of concatenated line segments can be traced. These two starting points result in two strings which are reflections of each other. It is easy to check that the application of each ISA operator to a string and to its reflection results in the same constituent structure and the same information load for both strings.

2.5. A Modification of the ISA Operators

Even if we restrict the scope of SIT to linear line patterns, some problems remain. In this section, we discuss these problems and propose a modification of the ISA operators to avoid them. First of all, there is a formal problem which arises because SIT treats lengths of line segments and angles between them in exactly the same way. Because of this, the primitive code of a linear line pattern with a repetitive structure cannot be obtained by applying the iteration operator to an appropriate substring. To illustrate this problem, consider the linear line pattern shown in Figure 3-B. Although this pattern has a repetitive structure, its primitive code $a\alpha a\beta a\alpha a$ cannot be generated by the iteration operator since the subpattern $a\alpha a$ is iterated once more than the connecting angle β. In order to solve this problem, we distinguish characters that encode lengths of line segments from characters that encode angles, and use this distinction in defining the iteration operator. In particular, we use $a, b, \ldots$ to encode lengths of line segments and $\alpha, \beta, \ldots$ to encode angles. Based on this distinction we define the iteration operator as $m * (XY) = \overbrace{XY \cdots YX}^{m\ timesX}$, where X is the primitive code of a linear line pattern and Y is the angle that connects them. Thus, the subpattern X is iterated m times while the angle Y is iterated $m-1$ times.

The second problem is an empirical one. We cannot subscribe to the SIT claim that all ISA regularities are equally relevant when we consider the class of linear line patterns. In particular, we have a problem with this claim with respect to the alternation regularity which describes a pattern as consisting of two subpatterns. For instance, the expression $< (X) > / < (X_1) \cdots (X_n) >$ describes the pattern $XX_1XX_2 \cdots XX_n$ as consisting of two subpatterns (X) and $(X_1) \cdots (X_n)$. This implies that the regularity of the subpattern $(X_1) \cdots (X_n)$ is assumed to be visible and thus relevant for the overall perceived structure of the pattern.

FIG. 6. *The regularity among alternating elements is not perceived.*

We can construct many examples of linear line patterns which do not support this claim. For example, consider the linear line pattern illustrated in Figure 6. This pattern can be described as $< (A) > / < (A_1)(A_2)(A_3)(A_4)(A_2)(A_5) >$. The second argument $(A_1)(A_2)(A_3)(A_4)(A_2)(A_5)$ contains a regularity since the subpattern $(A_2)(A_3)(A_4)(A_2)$ can be described as having a symmetry structure, i.e. $S[((A_2))((A_3)(A_4))]$. However, our experience suggests that the regularity of this subpattern in Figure 6 is perceptually not relevant. Therefore, we modify the alternation operator in such a way that the regularity among alternating elements cannot be captured anymore. This can be done by isolating alternating elements and considering them as separate subpatterns to which the alternation operator is applied. This modification results in $n+1$-ary alternation operators where each alternating element is considered as one argument (see definition 2.6). Following this formulation of the alternation operators, the linear line pattern illustrated in Figure 6 can be described by the right alternation operator as:

$$< (A) > / < (A_1), (A_2), (A_3), (A_4), (A_2), (A_5) >$$

The third problem is a subtle issue concerning the alternation operators. These operators are sensitive with respect to the regularity of angles between the constant subpattern X and the alternating subpatterns $X_1, \ldots, X_n$. For example, consider $a\alpha b_1\beta_1 a\alpha b_2\beta_2 a\alpha b_3$ to be the primitive code of a linear line pattern. The linear line pattern can be described by the modified right alternation operator in the following two ways:

1- $< (a) > / < (\alpha b_1\beta_1), (\alpha b_2\beta_2), (\alpha b_3) >$
2- $< (a\alpha) > / < (b_1\beta_1), (b_2\beta_2), (b_3) >$

Both analyses are perceptually equivalent since they provide the same constituent structure of the linear line pattern, i.e. the linear line pattern consists of occurrences of line segment a that are alternated by line segments b_1, b_2, and b_3. Nevertheless, the information loads of the two analyses are different, such that the second analysis is preferred over the first perceptually identical analysis. In order to avoid this non-intuitive property, we require that the constant subpattern of the alternation structure does always end with a line segment rather than an angle. This implies that the right alternation operator works as in formulation 1 above. The same argument holds for the left alternation operator.

Moreover, in the original version of SIT parentheses are used in the primitive codes to indicate the chunking structure of encoded line patterns in terms of constituting line patterns. Since in the modified version of SIT the characters that encode line segments and angles are distinguished, the parentheses can in principle be applied to parts of primitive codes that start and end with an angle. However, these primitive codes do not encode line patterns (line patterns start and end with line segments) such that the parentheses do not indicate the chunking structure in terms of constituting line patterns. For this reason, we consider (YXY') and $Y(X)Y'$ as perceptually equivalent for any primitive code X and angles Y and Y'; also primitive codes (YX) and $Y(X)$ and primitive codes (XY') and $(X)Y'$ are assumed to be perceptually equivalent. This implies that primitive code $a\alpha b_1\beta_1 a\alpha b_2\beta_2 a\alpha b_3$ can be described by the right alternation operator as follows:

3- $< (a) > / < \alpha(b_1)\beta_1, \alpha(b_2)\beta_2, \alpha(b_3) >$

The information load of formulation 3 differs from the information load of formulation 1. In particular, the information load of each $(\alpha b_i\beta_i)$ is higher than the information load of each corresponding $\alpha(b_i)\beta_i$ since the first is a chunk consisting of more than one element. This is, however, a deliberate divergence from the original version of SIT because we assume here that the chunking structure is only relevant with respect to line patterns. Note that when the alternating arguments are primitive codes of linear line patterns with more than one line segment, then the information loads of formulations 1 and 3 are the same.

The last problem is the sensitivity of the odd symmetry operator with respect to the regularity of the two angles that connect the symmetrical parts to the pivot element. For example, consider $a\alpha a\beta b\beta a\alpha a$ to be the primitive code of a linear line pattern. In the original SIT, this linear line pattern can be described by the odd symmetry operator according to the following two formulations:

1') $S[(a)(\alpha)(a)\ ,\ \beta(b)\beta]$
2') $S[(a)(\alpha)(a)(\beta)\ ,\ (b)]$

Again, although both analyses are perceptually identical (i.e. the linear line pattern consists of two symmetrical halves, which are encoded by $a\alpha a$, and the pivot element, which is encoded by b), their information loads are different. As with the alternation structures, we require that the symmetrical halves of the odd symmetry structure does always end with a line segment, and not with an angle. This implies that the odd symmetry operator works as formulation 1'. In the following, we first provide a formal definition of the domain of primitive codes of linear line patterns and then define the modified version of the ISA operators.

Definition 2.5. Let A_{seg} be the set of primitive elements that denote lengths of line segments, and A_{ang} be the set of primitive elements that denote angles: $A_{seg} \cap A_{ang} = \emptyset$. The domain D_c of primitive codes of linear line patterns over $A_{seg} \cup A_{ang}$ is defined as follows:

- $A_{seg} \subset D_c$
- If $X_1, \ldots, X_n \in D_c$ and $Y_1, \ldots, Y_{n-1} \in A_{ang}$, then $X_1 Y_1 \cdots Y_{n-1} X_n \in D_c$
- If $X \in D_c$, then $(X) \in D_c$

In our examples, we will choose A_{seg} to be the set of lower case letters of the English alphabet, i.e. $A_{seg} = \{a, b, \ldots\}$, and A_{ang} to be the set of lower case letters of the Greek alphabet, i.e. $A_{ang} = \{\alpha, \beta, \ldots\}$. Note that the domain of primitive codes of linear line patterns does not only contain sequences of primitive symbols; primitive codes may also contain (possibly nested) parentheses.

Definition 2.6. Let m stand for any natural number greater than one; let $X, X_1, \ldots, X_n$ stand for arbitrary elements of D_c; and let $Y, Y_1, \ldots, Y_n, Y'_1, \ldots, Y'_n$ stand for arbitrary elements of A_{ang}. For $n \geq 1$, the ISA operators are modified as follows.

- $m * (XY) = \overbrace{XY \cdots YX}^{m\ times\ X}$

- $S[(X_1)(Y_1)\cdots(Y_{n-1})(X_n)\ ,\ Y] = X_1Y_1\cdots Y_{n-1}X_nYX_nY_{n-1}\cdots Y_1X_1$

- $S[(X_1)(Y_1)\cdots(Y_{n-1})(X_n)\ ,\ Y_1'(X)Y_2'] = X_1Y_1\cdots Y_{n-1}X_nY_1'XY_2'X_nY_{n-1}\cdots Y_1X_1$

- $<(X)>/<(Y_1)(X_1)(Y_1')\ ,\ldots,\ (Y_n)(X_n)>= XY_1X_1Y_1'X\cdots XY_nX_n$

- $<(X_1)(Y_1')\ ,\ldots,\ (Y_{n-1})(X_n)(Y_n')>/<(X)>= X_1Y_1'X\cdots XY_{n-1}X_nY_n'X$

In this definition, two cases for the symmetry operator are distinguished. The first case generates primitive codes of linear line patterns that have even symmetry structure, i.e. the symmetrical halves are connected to each other by an angle. The second case generates primitive codes of linear line patterns that have odd symmetry structure, i.e. the symmetrical halves are connected to each other by a pivot element.

In the sequel, we use MSIT to refer to the modified version of Structural Information Theory that is based on the domain of primitive codes and the modified ISA operators. The descriptions of primitive codes by means of the modified ISA operators will be called MSIT descriptions. We now need to redefine the information load function such that it can be applied to MSIT descriptions. The following definition of information load is in accordance with its original idea and counts the number of primitive elements and the number of chunks that contain more than one primitive element in pattern descriptions.

Definition 2.7. Let $t, t_1, \ldots, t_n$ stand for arbitrary MSIT descriptions and let $Y, Y_1, \ldots, Y_n, Y_1', \ldots, Y_n'$ stand for arbitrary elements of A_{ang}. The information load function I^{msit}, which assigns natural numbers to MSIT descriptions, can be defined as follows:

- $I^{msit}(t) = 1$ if $t \in A_{seg} \cup A_{ang}$
- $I^{msit}(m * (tY)) \ = \ I^{msit}(t) + I^{msit}(Y) + 1$
- $I^{msit}(S[(t_1)(Y_1)\cdots(Y_{n-1})(t_n)\ ,\ Y]) \ = \ I^{msit}((t_1)Y_1\cdots Y_{n-1}(t_n)) + I^{msit}(Y)$
- $I^{msit}(S[(t_1)(Y_1)\cdots(Y_{n-1})(t_n)\ ,\ Y_1'(t)Y_2']) =$
  $I^{msit}((t_1)Y_1\cdots Y_{n-1}(t_n)) + I^{msit}(t) + I^{msit}(Y_1') + I^{msit}(Y_2') + 1$
- $I^{msit}(<(t)>/<(Y_1)(t_1)(Y_1')\ ,\ldots,\ (Y_n)(t_n)>) \ =$
  $I^{msit}((t)) + \sum_{i=1}^{n} I^{msit}((t_i)) + \sum_{i=1}^{n} I^{msit}(Y_i) + \sum_{i=1}^{n-1} I^{msit}(Y_i')$
- $I^{msit}(<(t_1)(Y_1')\ ,\ldots,\ (Y_{n-1})(t_n)(Y_n')>/<(t)>) \ =$
  $I^{msit}((t)) + \sum_{i=1}^{n} I^{msit}((t_i)) + \sum_{i=1}^{n-1} I^{msit}(Y_i) + \sum_{i=1}^{n} I^{msit}(Y_i')$
- $I^{msit}((t)) = I^{msit}(t)$ if t contains only one element from $A_{seg} \cup A_{ang}$;
  $= I^{msit}(t) + 1$ otherwise
- $I^{msit}(t_1Y_1\cdots Y_{n-1}t_n) = \sum_{i=1}^{n} I^{msit}(t_i) + \sum_{i=1}^{n-1} I^{msit}(Y_i)$

In this definition, the information loads for the iteration operator and the odd symmetry operator are one higher than the sum of the information loads of their arguments. Two considerations are involved in this decision. First, according to the definition of ISA operators and their modifications the argument of the iteration operator and the pivot element of the symmetry operator are considered to be chunks.[1] Second, according to the modified version of ISA operators, these arguments contain always more than one element from $A_{seg} \cup A_{ang}$. As the information load of a chunk containing more than one primitive element is one higher than the sum of the information loads of the primitive elements, the information loads for the iteration and the second symmetry operators should be one higher than the sum of the information loads of their arguments. Note that the parentheses around Y_i and Y_i' on the righthand side of equations are removed since the information load of one element from A_{ang} within parentheses is the same as the information load of that element without parentheses.

It is important to note that the definitions 2.3 and 2.7 of information loads are equivalent when we consider the description of primitive codes used in definition 2.7 as the descriptions of strings used in definition 2.3. For example, consider the MSIT description $m * (a\alpha)$. The information load of $m * (a\alpha)$ is the same in both definitions, i.e. $I^{sit}(m * (a\alpha)) \overset{2.3}{=} I^{sit}((a\alpha)) \overset{2.3}{=} I^{sit}(a\alpha) + 1 \overset{2.3}{=} 3 \overset{2.7}{=} I^{msit}(a\alpha) + 1 \overset{2.7}{=} I^{msit}((a\alpha)) \overset{2.7}{=} I^{msit}(m * (a\alpha))$. The reader should verify this for other MSIT descriptions. We can now articulate a reformulation of SIT's original empirical claim.

Conjecture 2. The representation hypothesis of SIT is correct for linear line patterns, if we employ MSIT descriptions and the modified information load function. In other words, for any linear line pattern L with primitive code s, the SIT-predicted structures of L, denoted by the set $Perceived_{msit}(s)$, are represented by the MSIT expressions that describe s and have minimum complexity value, i.e.

$$Perceived_{msit}(s) = \{e \in E^{msit}(s) \mid \forall e' \in E^{msit}(s) \; I^{msit}(e) \leq I^{msit}(e')\}$$

where $E^{msit}(s)$ is the set of final MSIT descriptions of primitive code s. Moreover, the uniqueness claim can be reformulated as follows:

$$|Perceived_{msit}(s)| = 1 \Leftrightarrow \; L \text{ is not ambiguous}$$

■

3. $\mathcal{LSP}$: A LANGUAGE FOR GESTALTS OF SEQUENTIAL PATTERNS

In this section, we finish our reformulation of SIT by describing the formal language $\mathcal{LSP}$ (Language for Gestalts of Sequential Patterns). $\mathcal{LSP}$ is essentially equivalent to MSIT, but employs a more uniform and explicit notation. The $\mathcal{LSP}$ expressions denote sequential patterns in general and primitive codes of linear line patterns in particular. They are defined inductively, resulting in nested structures.

[1]For example, the information load of $2 * (ab)$ is 3 since ab is considered to be one chunk, and the information load of $S[(a), (cd)]$ is 4 since cd is considered to be one chunk.

The nested structure of a $\mathcal{LSP}$ expression indicates a hierarchical organization of the denoted primitive code, and thus a perceptual organization of the encoded pattern.

In order to represent symmetry structures, two distinct operators, called Even-symmetry (Sym_e) and Odd-symmetry (Sym_o), are introduced in $\mathcal{LSP}$. The first represents symmetrical patterns that do not contain a pivot element, and the second represents symmetrical patterns that do contain a pivot element. Also, we introduce the chunk operator ($Chunk$) to represent the chunk structure of patterns and the concatenation operator (Con) to represent the concatenation structure of patterns.

Definition 3.1. [Syntax of $\mathcal{LSP}$] Let A_{seg} be the set of characters that encode lengths of line segments, A_{ang} be the set of characters that encode angles, and $\mathbb{N}$ be the set of natural numbers. Let $Iter$, Sym_e, Sym_o, Alt_r, Alt_l, $Chunk$, and Con be operator names, corresponding with Iteration, Even-symmetry, Odd-symmetry, Right-alternation, Left-alternation, Chunk, and Concatenation, respectively. Then, the expressions of $\mathcal{LSP}$ are recursively defined as follows:

1. $A_{seg} \subset \mathcal{LSP}$
2. If $t \in \mathcal{LSP},\ y \in A_{ang}$, and $m \in \mathbb{N}$, then
   $Iter(t, y, m) \in \mathcal{LSP}$
3. If $t, t_1, t_2 \in \mathcal{LSP},\ y, y' \in A_{ang}$, then
   $Sym_e(t, y) \in \mathcal{LSP}$ and
   $Sym_o(t_1, t_2, y, y') \in \mathcal{LSP}$
4. If $t, t_1, \ldots, t_n \in \mathcal{LSP},\ y_1, \ldots, y_n, y'_1, \ldots, y'_n \in A_{ang}$, then
   $Alt_r(t, t_1, \ldots, t_n, y_1, \ldots, y_n, y'_1, \ldots, y'_{n-1}) \in \mathcal{LSP}$ and
   $Alt_l(t, t_1, \ldots, t_n, y_1, \ldots, y_{n-1}, y'_1, \ldots, y'_n) \in \mathcal{LSP}$
5. If $t \in \mathcal{LSP}$, then
   $Chunk(t) \in \mathcal{LSP}$
6. If $t_1, \ldots, t_n \in \mathcal{LSP}$ and $y_1, \ldots, y_{n-1} \in A_{ang}$, then
   $Con(t_1, y_1, \ldots, y_{n-1}, t_n) \in \mathcal{LSP}$

Again, for our examples we will choose A_{seg} to be the set of lower case letters of the English alphabet, i.e. $A_{seg} = \{a, b, \ldots\}$, and A_{ang} to be the set of lower case letters of the Greek alphabet, i.e. $A_{ang} = \{\alpha, \beta, \ldots\}$. The following expressions are examples of $\mathcal{LSP}$ expressions:

- $Con(a, \alpha, b, \beta, c)$
- $Iter(Sym_e(a, \alpha), \beta, 3)$
- $Sym_e(Alt_l(a, Con(b, \alpha, c), d, \beta_1, \beta_2, \gamma), \delta)$
- $Iter(Sym_o(Chunk(Con(a, \alpha, b)), Con(c, \beta, d), \gamma_1, \gamma_2), \lambda, 2)$

The semantics of $\mathcal{LSP}$ expressions is defined in terms of the corresponding modified ISA operators (definition 2.6) and the domain of primitive codes (definition 2.5). However, the recursive application of the ISA operators is problematic with respect to the symmetry operator. The problem is that the first argument of the symmetry operators should be of the form $(X_1) \cdots (X_n)$ (a sequence of primitive codes within parentheses). This means that the evaluation of the $\mathcal{LSP}$ expression t in $Sym_e(t, y)$ and $Sym_o(t, t', y, y')$ should provide a primitive code of the form

$(X_1)\cdots(X_n)$. For this reason, we add parentheses around all primitive elements that occur in the value of t and that are not already within parentheses. We thus define a new symmetry operator S^* as follows:

$$\begin{array}{lcl} S^*[X_1\cdots X_n\ ,\ Y] & = & S[(X_1')\cdots(X_n')\ ,\ Y] \\ S^*[X_1\cdots X_n\ ,\ Y(X)Y'] & = & S[(X_1')\cdots(X_n')\ ,\ Y(X)Y'] \end{array}$$

where for $1 \leq i \leq n$ it holds that $X_i = X_i'$ if $X_i \in A_{seg} \cup A_{ang}$; otherwise $X_i = (X_i')$. Note that adding parentheses around primitive elements does not increase the information load.

Definition 3.2. [Semantics of $\mathcal{LSP}$] Let D_c be the domain of primitive codes as in definition 2.5, $t, t_i \in \mathcal{LSP}$ for $1 \leq i \leq n$, and $y, y', y_j, y_j' \in A_{ang}$ for $1 \leq j \leq n$. Then, the semantic function $[\![\cdot]\!]$, which maps $\mathcal{LSP}$ expressions to primitive codes from D_c, is defined as follows.

1. If $t \in A_{seg} \cup A_{ang}$, then $[\![t]\!] = t$
2. $[\![Iter(t,y,m)]\!] = m * (\ [\![t]\!][\![y]\!]\ )$
3. $[\![Sym_e(t,y)]\!] = S^*[\ [\![t]\!]\ ,\ [\![y]\!]\ ]$
4. $[\![Sym_o(t_1,t_2,y,y')]\!] = S^*[\ [\![t_1]\!]\ ,\ [\![y]\!]([\![t_2]\!])[\![y']\!]\ ]$
5. $[\![Alt_r(t,t_1,\ldots,t_n,y_1,\ldots,y_n,y_1',\ldots,y_{n-1}')]\!] =$ $<([\![t]\!])>/<[\![y_1]\!]([\![t_1]\!])[\![y_1']\!]\ ,\ldots,\ [\![y_n]\!]([\![t_n]\!])>$
6. $[\![Alt_l(t,t_1,\ldots,t_n,y_1,\ldots,y_{n-1},y_1',\ldots,y_n')]\!] =$ $<([\![t_1]\!])[\![y_1']\!]\ ,\ldots,\ [\![y_{n-1}]\!]([\![t_n]\!])[\![y_n']\!]>/<([\![t]\!])>$
7. $[\![Chunk(t)]\!] = (\ [\![t]\!]\ )$
8. $[\![Con(t_1,y_1,\ldots,y_{n-1},t_n)]\!] = [\![t_1]\!][\![y_1]\!]\ldots[\![y_{n-1}]\!][\![t_n]\!]$

The following example illustrates the semantic evaluation of a $\mathcal{LSP}$ expression.

$$[\![Iter(Sym_e(a,\alpha),\beta,3)]\!] = \quad 3*([\![Sym_e(a,\alpha)]\!][\![\beta]\!]) = 3*(S^*[[\![a]\!],[\![\alpha]\!]]\beta) =$$
$$3*(S^*[a,\alpha]\beta) = 3*(S[(a),\alpha]\beta) = 3*((a)\alpha(a)\beta) = (a)\alpha(a)\beta(a)\alpha(a)\beta(a)\alpha(a)$$

Two $\mathcal{LSP}$ expressions are extensionally equal if and only if they both denote the same primitive code. Extensionally equal $\mathcal{LSP}$ expressions constitute different Gestalts of one linear line pattern. Structural Information Theory can be described in terms of the $\mathcal{LSP}$ language together with a complexity function I^{lsp}, which assigns a natural number to every $\mathcal{LSP}$ expression. This function is designed in such a way that the $\mathcal{LSP}$ expressions and their corresponding MSIT expressions have the same information load.

Definition 3.3. Let $t, t_i \in \mathcal{LSP}$ for $1 \leq i \leq n$; let $y, y', y_j, y_j' \in A_{ang}$ for $1 \leq j \leq n$; and let $m \in \mathbb{N}$. The information load of a $\mathcal{LSP}$ expression is the natural number that is assigned to it by the I^{lsp} function. The I^{lsp} function is defined as follows:

1. $I^{lsp}(t) = 1$ if $t \in A_{seg} \cup A_{ang}$
2. $I^{lsp}(Iter(t,y,m)) = I^{lsp}(t) + I^{lsp}(y) + 1$
3. $I^{lsp}(Sym_e(t,y)) = I^{lsp}(t) + I^{lsp}(y)$
4. $I^{lsp}(Sym_o(t_1,t_2,y,y')) = I^{lsp}(t_1) + I^{lsp}(t_2) + I^{lsp}(y) + I^{lsp}(y') + 1$

5. $I^{lsp}(Alt_r(t, t_1, \ldots, t_n, y_1, \ldots, y_n, y'_1, \ldots, y'_{n-1})) =$
$I^{lsp}(Chunk(t)) + \sum_{i=1}^{n} I^{lsp}(Chunk(t_i)) + \sum_{i=1}^{n} I^{lsp}(y_i) + \sum_{i=1}^{n-1} I^{lsp}(y'_i)$
6. $I^{lsp}(Alt_l(t, t_1, \ldots, t_n, y_1, \ldots, y_{n-1}, y'_1, \ldots, y'_n)) =$
$I^{lsp}(Chunk(t)) + \sum_{i=1}^{n} I^{lsp}(Chunk(t_i)) + \sum_{i=1}^{n-1} I^{lsp}(y_i) + \sum_{i=1}^{n} I^{lsp}(y'_i)$
7. $I^{lsp}(Chunk(t)) = I^{lsp}(t)$ if t contains only one element from $A_{seg} \cup A_{ang}$;
$= I^{lsp}(t) + 1$ otherwise
8. $I^{lsp}(Con(t_1, y_1, \ldots, y_{n-1}, t_n)) = \sum_{i=1}^{n} I^{lsp}(t_i) + \sum_{i=1}^{n-1} I^{lsp}(y_i)$

In this definition, the information loads for $Iter$ and Sym_o expressions are one higher than the sum of the information loads of their constituents. This decision is based on the same considerations that were involved in defining the information load function for the modified ISA operators (see definition 2.7). Moreover, the information load of the right and left alternation expressions is defined in terms of the sum of the information loads of their arguments that are reformulated as chunk expressions. The arguments are reformulated as chunk expressions because these arguments in the original and modified ISA operators are considered to be chunks. We reformulate these arguments, rather than increasing the information loads of alternation expressions (as done with $Iter$ and Sym_o), because the arguments can be primitive elements in which case the information loads should not be increased.

In the following, we write $E^{lsp}(s)$ to refer to the set of $\mathcal{LSP}$ expressions that denote primitive code s in D_c, i.e. $E^{lsp}(s) = \{e \in \mathcal{LSP} \mid [\![e]\!] = s\}$. The subset of $E^{lsp}(s)$ consisting of $\mathcal{LSP}$ expressions with minimum complexity value according to definition 3.3 is denoted by $Perceived_{lsp}(s)$, i.e.

$$Perceived_{lsp}(s) = \{e \in E^{lsp}(s) \mid \forall e' \in E^{lsp}(s)\ I^{lsp}(e) \leq I^{lsp}(e')\}$$

The set $Perceived_{lsp}(s)$ is called the $\mathcal{LSP}$-predicted structures of primitive code s.

THEOREM 3.1. *For any linear line pattern L with primitive code s, the SIT-predicted structures of L are represented by the expressions from $Perceived_{lsp}(s)$.*

Proof. For an arbitrary linear line pattern L and its primitive code s the SIT-predicted structures of L are determined by $Perceived_{msit}(s)$ (see conjecture 2). We now need to show that for any primitive code s, $Perceived_{msit}(s)$ and $Perceived_{lsp}(s)$ represent perceptual structures that pairwise have the same constituent structure and have the same information load, i.e. there exists a bijective function $f : Perceived_{lsp}(s) \to Perceived_{msit}(s)$ such that:

1. $\forall e \in Perceived_{lsp}(s)$: e and $f(e)$ have the same constituent structure,
2. $\forall e \in Perceived_{lsp}(s)$: $I^{lsp}(e) = I^{msit}(f(e))$

The function $[\![\cdot]\!]$, specified in definition 3.2, is such a bijective function which maps $E^{msit}(s)$ to $E^{lsp}(s)$ for any arbitrary $s \in D_c$.
1) Function $[\![\cdot]\!]$ preserves the constituent structure of corresponding elements of $E^{msit}(s)$ and $E^{lsp}(s)$ since the number of arguments of the modified ISA operators that are primitive codes of linear line patterns, and the number of arguments of the corresponding $\mathcal{LSP}$ expression that denote primitive codes of linear line patterns, are the same.

2) The following equalities show that function $[\![\cdot]\!]$ preserves the information load of corresponding elements of $E^{msit}(s)$ and $E^{lsp}(s)$ as well.

$I^{lsp}(t \in A_{seg} \cup A_{ang}) = 1 = I^{msit}([\![t]\!] \in A_{seg} \cup A_{ang})$

$I^{lsp}(Iter(t, y, m)) = I^{lsp}(t) + I^{lsp}(y) + 1 = I^{msit}([\![t]\!]) + I^{msit}([\![y]\!]) + 1 =$
$I^{msit}(m * ([\![t]\!][\![y]\!]))$

$I^{lsp}(Sym_e(t, y)) = I^{lsp}(t) + I^{lsp}(y) = I^{msit}([\![t]\!]) + I^{msit}([\![y]\!]) = I^{msit}(S[[\![t]\!], [\![y]\!]])$

$I^{lsp}(Sym_o(t_1, t_2, y, y')) = I^{lsp}(t_1) + I^{lsp}(t_2) + I^{lsp}(y) + I^{lsp}(y') + 1 =$
$I^{msit}([\![t_1]\!]) + I^{msit}([\![t_2]\!]) + I^{msit}([\![y]\!]) + I^{msit}([\![y']\!]) + 1 =$
$I^{msit}([\![t_1]\!]) + I^{msit}(([\![y]\!][\![t_2]\!][\![y']\!])) = I^{msit}(S[[\![t_1]\!], [\![y]\!]([\![t_2]\!])[\![y']\!]])$

$I^{lsp}(Alt_r(t, t_1, \ldots, t_n, y_1, \ldots, y_n, y'_1, \ldots, y'_{n-1})) =$
$I^{lsp}(Chunk(t)) + \sum_{i=1}^{n} I^{lsp}(Chunk(t_i)) + \sum_{i=1}^{n} I^{lsp}(y_i) + \sum_{i=1}^{n-1} I^{lsp}(y'_i) =$
$I^{msit}(([\![t]\!])) + \sum_{i=1}^{n} I^{msit}(([\![t_i]\!])) + \sum_{i=1}^{n} I^{msit}([\![y_i]\!]) + \sum_{i=1}^{n-1} I^{msit}([\![y'_i]\!]) =$
$I^{msit}(< ([\![t]\!]) > / < [\![y_1]\!]([\![t_1]\!])[\![y'_1]\!], \ldots, [\![y_n]\!]([\![t_n]\!]) >)$

$I^{lsp}(Alt_l(t, t_1, \ldots, t_n, y_1, \ldots, y_{n-1}, y'_1, \ldots, y'_n)) =$
$I^{lsp}(Chunk(t)) + \sum_{i=1}^{n} I^{lsp}(Chunk(t_i)) + \sum_{i=1}^{n-1} I^{lsp}(y_i) + \sum_{i=1}^{n} I^{lsp}(y'_i) =$
$I^{msit}(([\![t]\!])) + \sum_{i=1}^{n} I^{msit}(([\![t_i]\!])) + \sum_{i=1}^{n-1} I^{msit}([\![y_i]\!]) + \sum_{i=1}^{n} I^{msit}([\![y'_i]\!]) =$
$I^{msit}(< ([\![t_1]\!])[\![y'_1]\!], \ldots, [\![y_{n-1}]\!]([\![t_n]\!])[\![y'_n]\!] > / < ([\![t]\!]) >)$

For Chunk expressions, there are two cases.
Case 1: t contains more than one elements from $A_{seg} \cup A_{ang}$
$I^{lsp}(Chunk(t)) = I^{lsp}(t) + 1 = I^{msit}([\![t]\!]) + 1 = I^{msit}(([\![t]\!]))$
Case 2: otherwise
$I^{lsp}(Chunk(t)) = I^{lsp}(t) = I^{msit}([\![t]\!]) = I^{msit}(([\![t]\!]))$

$I^{lsp}(Con(t_1, y_1, \ldots, y_{n-1}, t_n)) = \sum_{i=1}^{n} I^{lsp}(t_i) + \sum_{i=1}^{n-1} I^{lsp}(y_i)$
$\sum_{i=1}^{n} I^{msit}([\![t_i]\!]) + \sum_{i=1}^{n-1} I^{msit}([\![y_i]\!]) = I^{msit}([\![t_1]\!][\![y_1]\!] \cdots [\![y_{n-1}]\!][\![t_n]\!])$ ■

The following theorem assumes the uniqueness claim and states that under this assumption if a linear line pattern is not ambiguous, i.e. if a linear line pattern has a unique perceived structure, then there is a unique $\mathcal{LSP}$ expression with minimum complexity value that denotes the linear line pattern and vice versa.

THEOREM 3.2. *Under the uniqueness claim assumption and for a linear line pattern L with primitive code s :*

$$|Perceived_{lsp}(s)| = 1 \Leftrightarrow L \textit{ is not ambiguous}$$

Proof. Conjecture 2 and Theorem 3.1 state that the SIT-predicted structures of any linear line pattern L with primitive code s are represented by $Perceived_{msit}(s)$ and $Perceived_{lsp}(s)$, respectively. This implies $|Perceived_{msit}(s)| = |Perceived_{lsp}(s)|$ for any linear line pattern L with primitive code s. Then, according to the uniqueness claim L is not ambiguous if and only if $|Perceived_{msit}(s)| = 1$. This concludes that any linear line pattern with primitive code s is not ambiguous if and only if $|Perceived_{lsp}(s)| = 1$. ∎

4. A REPRESENTATION SYSTEM FOR TWO-DIMENSIONAL LINE PATTERNS

According to SIT, perceptual structures of sensory patterns can be described in terms of ISA regularities among pattern parts, which are in turn defined in terms of the identity of pattern parts. Recall that for the class of line patterns, lengths of line segments and angles between them were considered as primitive pattern parts. In section 2.4, we showed that the linearisation of patterns proposed by SIT can only be applied properly to the limited class of linear line patterns. In the current section, we propose an alternative to the linearisation approach and extend the SIT approach to a more general class of line patterns.

In the sequel, we limit our analyses and examples to connected line patterns. The class of connected line patterns can be characterized by graphs in which each graph edge represents one line segment and all graph nodes can be reached from all other graph nodes through a path in the graph. Note that there are no constraints on the degrees of the nodes and that the class of connected line patterns includes the class of linear line patterns. A simple example of a connected line pattern is illustrated in Figure 7. Although the analyses and examples are limited to connected line patterns, the representation system that we introduce is designed to represent the perceptual structures of the general class of line patterns, i.e. line patterns in which line segments need not to be connected. The reason we limit our analyses and examples to connected line patterns is the nonavailability of an empirically supported complexity measure for the general class of line patterns. This issue is explained later in more detail.

In order to apply SIT directly to line patterns, we define ISA regularities in terms of spatial relations between line segments. We represent a line segment as a pair $< pos_1, pos_2 >$, where pos_1 and pos_2 are the two-dimensional positions of its two end points. The spatial relations between line segments are in turn defined in terms of Euclidean transformations (i.e. Euclidean rotation, translation, reflection, and their compositions) between their two-dimensional position values. A line pattern is represented as a set of position pairs. Moreover, in order to capture the special kind of symmetrical patterns in which not individual line segments but chunks of line segments are reflected, we mark chunked line segments by labelling them with *chk*. This corresponds to the use of parentheses in the domain of primitive codes of linear line patterns.

Definition 4.1. Let $A_{posp} = \{< pos_1, pos_2 > \mid pos_1, pos_2 \in \mathbb{R} \times \mathbb{R}\}$ be the set of position pairs. The domain D of two-dimensional line patterns over A_{posp} is defined as follows:

- if $x \in A_{posp}$, then $\{x\} \in D$
- if $x_1, \ldots, x_n \in D$ and $n \geq 2$, then $\bigcup_{i=1}^{n} x_i \in D$
- if $\{x_1, \ldots, x_n\} \in D$ and $n \geq 2$, then $\{chk(x_1, \ldots, x_n)\} \in D$

Line segments can be described in terms of each other by means of Euclidean translation, rotation and reflection transformations. As in the original version of SIT, the identity of line segments is based on their lengths: two line segments are identical if they have the same length. Since the lengths of line segments are invariant under Euclidean transformations, two line segments are identical if they can be described in terms of each other by means of Euclidean transformations. The difference between this approach and the original one is the removal of the angles from the coding system and their replacement by Euclidean transformations. The sequential tracing of the line segments is not a part of the encoding process anymore. In this way, we avoid the problems and limitations of SIT's original and modified coding model.

FIG. 7. *An example of a connected line pattern.*

For example, consider the square illustrated in Figure 7. It is represented by the following set of position pairs (i.e. line segments):

$$\{< p_1, p_2 > \, , \, < p_2, p_3 > \, , \, < p_3, p_4 > \, , \, < p_4, p_1 >\}.$$

The line segment $< p_2, p_3 >$ can be described in terms of the line segment $< p_1, p_2 >$ and a rotation with respect to the center of the square. The segment $< p_3, p_4 >$ can be described in terms of $< p_2, p_3 >$ and the same rotation transformation, and finally, $< p_4, p_1 >$ in terms of $< p_3, p_4 >$ and the same rotation transformation. The iterative application of the same transformation exemplifies the kind of pattern regularity which will be captured by the representation system proposed here. It is in a sense a two-dimensional analogue of the repetition operation in SIT's domain of one-dimensional primitive codes. In order to define two-dimensional pattern regularity, we need to define the application of Euclidean transformations to line patterns.

Definition 4.2. The application of a Euclidean transformation τ to a two-dimensional line pattern L is defined as $\tau(L) = \{\tau(< p_i, p_j >) \mid \, < p_i, p_j > \in L\}$; the application of τ to a line segment $< p_1, p_2 >$ is defined as $\tau(< p_1, p_2 >) = \, < \tau(p_1), \tau(p_2) >$; finally, the application of τ to a two-dimensional position $p = (x, y)$, which results in another two-dimensional position $p' = (x', y')$, will be written as $\tau(p) = p'$. Moreover, we write τ^i to indicate a nested application of τ with depth i. For example, $\tau^4(p) = \tau(\tau(\tau(\tau(p))))$ for the two-dimensional position p. For $i = 0$, this yields the identity transformation: $\tau^0(p) = p$.

4.1. Iteration Structure

The first class of regular structures for two-dimensional line patterns is the class of iteration structures. A line pattern L has the iteration structure if it can be described as the union of m subsets $L_1, \ldots, L_m$ such that all subsets can be obtained by iterative application of a Euclidean transformation τ to one of the subsets L_1, i.e., $L_i = \tau^{i-1}(L_1)$.

FIG. 8. *Position-based iteration structure.*

An example of this type of pattern was given above in Figure 7. Another example is illustrated in Figure 8 which is a triangle iterated twice. It is represented by the following set of position pairs:

$$\{< p_1, p_2 >, < p_2, p_3 >, < p_3, p_1 >, < p_3, p_4 >, < p_4, p_5 >, < p_5, p_3 >\}$$

This set of position pairs can be described in terms of 2 subsets of position pairs such that one can be described in terms of the other and a Euclidean translation. These subsets and the Euclidean translation are as follows:

$L_1 = \{< p_1, p_2 >, < p_2, p_3 >, < p_3, p_1 >\}$
$L_2 = \{< p_3, p_4 >, < p_4, p_5 >, < p_5, p_3 >\}$
ω is the translation transformation defined on positions $p = (p_x, p_y)$ as follows:
$\omega(p_x) = p_x + |x_2 - x_1|$
$\omega(p_y) = p_y$

Note that L_2 is obtained by applying the translation transformation ω to L_1. In general, given a Euclidean transformation ω composed of translation and rotation, and a line pattern represented by L, the following set that contains m occurrences of L represents a line pattern that can be analyzed as having the iteration structure:

$$\bigcup_{i=0}^{m-1} \omega^i(L)$$

4.2. Symmetry Structure

The second class of regular structures for two-dimensional line patterns is the class of symmetry structures. A line pattern has the symmetry structure if it can be partitioned into two subsets such that one subset can be obtained by applying a Euclidean reflection transformation to the other subset.

FIG. 9. *A visual pattern with symmetry structure.*

For example, consider the pattern illustrated in Figure 9. This pattern has the symmetry structure, i.e. the two triangles are reflections of each other. It is represented by the following set of position pairs:

$$\{< p_1, p_2 >, < p_2, p_3 >, < p_3, p_1 >, < p_3, p_4 >, < p_4, p_5 >, < p_5, p_3 >\}$$

This set of position pairs can be partitioned into two subsets of position pairs such that one can be described in terms of the other and a Euclidean reflection transformation:

$L_1 = \{< p_1, p_2 >, < p_2, p_3 >, < p_3, p_1 >\}$

$L_2 = \{< p_3, p_4 >, < p_4, p_5 >, < p_5, p_3 >\}$
ψ is the reflection transformation defined on positions $p = (p_x, p_y)$ as follows:
$\psi(p_x) = 2 * x_2 - p_x$
$\psi(p_y) = p_y$

Note that L_2 is obtained by applying the reflection transformation ψ to L_1. In general, given a Euclidean reflection transformation ψ and a set L representing a line pattern, the following set represents a line pattern that can be analyzed as having the symmetry structure:

$$L \bigcup \psi(L)$$

4.3. Chunk Structure

The notion of chunk is introduced to define a special kind of symmetrical pattern in which not individual primitive elements but chunks of primitive elements are reflected. The reflection of chunked elements preserves their mutual spatial relation. In fact, the reflection of chunked elements is similar to their translation relative to the reflection axis. For example, Figure 10 can be described as consisting of two polygons in which $< p_1, p_2 >$ and $< p_2, p_3 >$ are chunked, and $< p_5, p_6 >$ and $< p_6, p_7 >$ are chunked. The mutual spatial relation between the segments of the chunks in both polygons is preserved under the reflection transformation. Note that this kind of grouping operation is commonly used in computer graphic programs. It is needed when some parts of symmetrical halves are intended not to be mirror images of each other. These parts may be text objects, a human face, or other natural pictures.

FIG. 10. *The spatial relation between chunked line segments is preserved in the symmetrical halves.*

As mentioned, we indicate the chunks of position pairs by labelling them with *chk*. Thus, the following set of position pairs represents the line pattern illustrated in Figure 10:

$$\begin{aligned} \{ \quad & chk(< p_1, p_2 >, < p_2, p_3 >),\ < p_1, p_4 >,\ < p_3, p_4 >, \\ & chk(< p_5, p_6 >, < p_6, p_7 >),\ < p_4, p_5 >,\ < p_4, p_7 > \}. \end{aligned}$$

4.4. Alternation Structure

Alternation is regularity of patterns that are inherently sequential. In particular, a sequential pattern has alternation regularity if it consists of one sequential subpattern alternated by other sequential subpatterns. Therefore, alternation regularity makes only sense with respect to the subclass of linear line patterns that are inherently sequential, and not the general class of line patterns. For example, the primitive code $a\beta_1 b\beta_2 c\beta_3 a\beta_4 d$ has an alternation structure since the primitive code a is alternated by the codes $\beta_1 b\beta_2 c\beta_3$, and $\beta_4 d$. This structure is described by applying the modified right alternation operator, i.e. $< (a) > / < \beta_1(b\beta_2 c)\beta_3, \beta_4(d) >$. Note in this case that the alternation regularity captures only the regularity of the length of the line segment a.

Since in our coding model the representations of iterating parts are not interspersed among the representations of alternating parts (set representation versus string representation), one may argue that no additional alternation operator is needed to capture pattern regularity; alternation regularity of patterns, which is iteration regularity, can be captured by the iteration operator. However, because we aim at generating the structural descriptions for linear line patterns as well, we include alternation structure and add an alternation operator in our coding model. It is important to note that the alternation operator makes only sense with respect to linear line patterns and, therefore, it will be applied only to linear line patterns. In contrast to sequential patterns such as strings, left and right alternations will not be distinguished since the representations of alternating elements determine where they are placed in the two-dimensional space and thus do not depend on the representation of the iterating elements.

A linear line pattern has the alternation structure if it can be described as $2 * m$ subsets of line segments such that $m-1$ subsets can be obtained by applying $m-1$ Euclidean transformations composed of translations and rotations to one of the subsets. Each of the other m subsets is an alternating element. Consider the linear line pattern illustrated in Figure 11.

FIG. 11. *An example of a linear line pattern which has alternation structure.*

This pattern is represented by the following set of position pairs:
$\{< p_1, p_2 >, < p_2, p_3 >, < p_3, p_4 >, < p_4, p_5 >, < p_5, p_6 >, < p_6, p_7 >, < p_7, p_8 >, < p_8, p_9 >, < p_9, p_{10} >, < p_{10}, p_{11} >, < p_{11}, p_{12} >\}$
This set of position pairs can be described in terms of 2*3 subsets L_1, L_2, L_3, L_4, L_5 and L_6 such that L_2 and L_3 can be described in terms of L_1 and two Euclidean translation transformations, and L_4, L_5 and L_6 are the alternating parts. These subsets are as follows:

$L_1 = \{ < p_1, p_2 >, < p_2, p_3 >\}$
$L_2 = \{ < p_5, p_6 >, < p_6, p_7 >\}$
$L_3 = \{ < p_9, p_{10} >, < p_{10}, p_{11} >\}$
$L_4 = \{ < p_3, p_4 >, < p_4, p_5 >\}$
$L_5 = \{ < p_7, p_8 >, < p_8, p_9 >\}$
$L_6 = \{ < p_{11}, p_{12} >\}$

In general, given Euclidean transformations $\omega_1, \ldots, \omega_{m-1}$ composed of translations and rotations, a set L representing a linear line pattern, and each set L_i representing a linear line pattern, the following set represents a linear line pattern that can be analyzed as having the alternation structure:

$$L \cup \bigcup_{i=1}^{m-1} \omega_i(L) \ \cup \ \bigcup_{i=1}^{m} L_i$$

It is important to note that not all line patterns that can be described in this way are linear line patterns having the alternation structure. But, all linear line patterns that have the alternation structure can be described in this way.

4.5. Composition Structure

Line patterns for which there is no regularity of the above kinds among their parts are assumed to have a composition structure. The composition structure corresponds to the concatenation structure of the original and modified versions of SIT. For example, the visual pattern illustrated in Figure 12 has a composition structure. Of course, the parts of this line pattern have regular structures, but the pattern as a whole does not. Every line pattern can be described as having a composition structure.

FIG. 12. *An example of a line pattern which should be described as having the composition structure.*

In general, if the sets $L_1, \ldots, L_n$ represent line patterns, the following set represents a line pattern that can be analyzed as having the composition structure:

$$\bigcup_{i=1}^{n} L_i$$

4.6. $\mathcal{LCP}$: A Language for Gestalts of Two-dimensional Line Patterns

In this section, the above considerations about regularities of two-dimensional line patterns are embodied in a coding language whose expressions represent the Gestalts of two-dimensional line patterns. This coding language, called $\mathcal{LCP}$ (Language for Gestalts of Line Patterns), is defined in terms of a set of position pairs A_{posp}, a set $\Omega \subset A_{trans}$ of Euclidean transformations composed of translations and rotations, a set $\Psi \subset A_{trans}$ of Euclidean reflection transformations, and a set of operator names that specify structural regularities. The set Ω is defined to capture regularities of position pairs for iteration structures and the set Ψ is defined to capture regularities of position pairs for symmetry structures. Note that we assume constants denoting all relevant Euclidean transformations, just as SIT assumes constants denoting all relevant lengths and angles. I.e., we do not introduce formal notation for describing Euclidean transformations in terms of a finite set of primitives; for the purpose of the current paper, such details are irrelevant.

Definition 4.3. [Syntax of $\mathcal{LCP}$] Let A_{posp} be the set of position pairs, A_{trans} be the set of Euclidean transformations, $\Omega \subset A_{trans}$ is the set of transformations composed of translations and rotations, and $\Psi \subset A_{trans}$ is the set of reflection transformations. Let also *Iter*, *Sym*, *Alt*, *Comp*, and *Chunk* be operator names corresponding to iteration, symmetry, alternation, composition and chunk, respectively. The $\mathcal{LCP}$ expressions are defined as follows:

1. $A_{posp} \subset \mathcal{LCP}$,
2. If $t, t_1, \ldots, t_n \in \mathcal{LCP}$, $m \in \mathbb{N}, \omega, \omega_1, \ldots, \omega_{n-1} \in \Omega$, and $\psi \in \Psi$, then
   - $Iter(t, \omega, m) \in \mathcal{LCP}$,
   - $Sym(t, \psi) \in \mathcal{LCP}$,
   - $Alt(t, t_1, \ldots, t_n, \omega_1, \ldots, \omega_{n-1}) \in \mathcal{LCP}$,
   - $Chunk(t) \in \mathcal{LCP}$,
   - $Comp(t_1, \ldots, t_n) \in \mathcal{LCP}$.

The syntax of $\mathcal{LLP}$ expressions indicates how a line pattern is perceived in terms of its constituent line patterns. The $\mathcal{LLP}$ expressions denote two-dimensional representations of line patterns in D (see definition 4.1). The semantics of the $\mathcal{LLP}$ expressions can be recursively defined as follows.

Definition 4.4. [Semantics of $\mathcal{LLP}$] Let A_{posp} be a set of position pairs, A_{trans} be a set of Euclidean transformations, and D be the set of two-dimensional representations of line patterns over A_{posp} as in definition 4.1. Then, the semantic function $|[\cdot]|$, which maps $\mathcal{LLP}$ expressions to representations of two-dimensional line patterns in D, is defined as follows:

1. if $t \in A_{posp} \cup A_{trans}$, then $|[t]| \ = t$,
2. if $t, t_i \in \mathcal{LLP}$ for $1 \leq i \leq n$, $m \in \mathbb{N}$, $\omega, \omega_1, \ldots, \omega_{n-1} \in \Omega$, and $\psi \in \Psi$, then
   - $|[Iter(t, \omega, m)]| = \bigcup_{i=0}^{m-1} |[\omega^i]|(\ |[t]| \ )$,
   - $|[Sym(t, \psi)]| = |[t]| \bigcup |[\psi]|(\ |[t]| \ )$,
   - $|[Alt(t, t_1, \ldots, t_n, \omega_1, \ldots, \omega_{n-1})]| = |[t]| \cup \bigcup_{i=1}^{n-1} |[\omega_i]|(\ |[t]| \ ) \cup \bigcup_{i=1}^{n} |[t_i]|$,
   - $|[Chunk(t)]| \ = \ chk(|[t]|)$,
   - $|[Comp(t_1, \ldots, t_n)]| = \bigcup_{i=1}^{n} |[t_i]|$.

As we have noticed, the application of a reflection transformation to a chunked line pattern is not trivial. In fact, the spatial relations between chunked line segments are invariant under the reflection transformation. In order to preserve the spatial relations under the reflection transformation, we define the application of the reflection transformation to a chunked line pattern as a double reflection transformation: the chunked line pattern is reflected with respect to the actual reflection axis and, subsequently, the resulting chunked line pattern is reflected with respect to a reflection axis which is parallel to the actual reflection axis and goes through the center of chunked line pattern. The second reflection axis will be called internal. This double reflection transformation guarantees that the spatial relations between the chunked line segments will be preserved. It is illustrated in Figure 13.

FIG. 13. *The process of reflecting patterns containing chunked subpatterns. Chunked subpatterns are first reflected according to the actual reflection axis R and then according to their internal reflection axis S.*

In Figure 13-A, the chunked line pattern consists of two L-shaped line patterns. This chunked pattern is reflected in Figure 13-B according to the actual reflection axis R. Finally, in Figure 13-C the reflected chunked pattern is reflected again according to its internal reflection axis S. In general, the application of transformations to chunked patterns can be defined as follows:

Definition 4.5. Let Mid-reflect$(|[\psi]|, |[t]|)$ generate a reflection transformation that reflects the line pattern $|[t]|$ according to its internal reflection axis, i.e. a reflection axis which is parallel to the axis used by the reflection transformation $|[\psi]|$, and goes through the center of $|[t]|$. Let also ψ and ω represent respectively a reflection transformation and a transformation composed of translation and rotation. Then, the application of transformations to $chk(|[t]|)$ is defined as:

- $|[\psi]|(\ chk(|[t]|)) = chk(|[\psi]|(\text{Mid- reflect}(|[\psi]|, |[t]|)(|[t]|)))$

- $|[\omega]|(chk(|[t]|)) = chk(|[\omega]|(|[t]|))$.

In the first clause the line pattern $chk(|[t]|)$, which is considered as a chunk, is reflected twice. This results in a visual pattern that is similar to a translation of $chk(|[t]|)$. This is exactly what we wished to achieve with the reflection of the chunked patterns.

Finally, for a transformation τ the expression $|[\tau]|(|[t]| \cup |[t']|)$ obeys the distribution law, i.e.

$$|[\tau]|(|[t]| \cup |[t']|) = |[\tau]|(|[t]|) \cup |[\tau]|(|[t']|)$$

4.7. An Example of a $\mathcal{LLP}$ Expression and its Denotation

As an example of a line pattern and its corresponding $\mathcal{LLP}$ expressions consider the two-dimensional line pattern in Figure 14. The perceived structure of this pattern can be described as consisting of two clusters: two squares and one diamond. The following is a $\mathcal{LLP}$ expression that reflects this Gestalt of the line pattern.

$$Comp(Iter(Iter(< p_1, p_2 >, \omega_r, 4), \omega_t, 2), Sym(Sym(< p_4, p_6 >, \psi_v), \psi_h))$$

In this expression, ω_r is a 90-degree rotation transformation with respect to the center of the left square, ω_t is a horizontal translation transformation from position p_1 to p_7, ψ_v and ψ_h are the vertical and horizontal reflection transformations with respect to the reflection axes along positions $p_5 - p_6$ and $p_4 - p_7$, respectively.

FIG. 14. *A visual pattern perceived as consisting of two squares and one diamond.*

$$|[Comp(Iter(Iter(< p_1, p_2 >, \omega_r, 4), \omega_t, 2), Sym(Sym(< p_4, p_6 >, \psi_v), \psi_h))]| =$$

$$|[Iter(Iter(< p_1, p_2 >, \omega_r, 4), \omega_t, 2)]| \bigcup |[Sym(Sym(< p_4, p_6 >, \psi_v), \psi_h)]| =$$

$$\begin{aligned}
&\textstyle\bigcup_{i=0}^{1} |[\omega_t^i]|(|[Iter(< p_1, p_2 >, \omega_r, 4)]|) \\
&\bigcup |[Sym(< p_4, p_6 >, \psi_v)]| \bigcup |[\psi_h]|(|[Sym(< p_4, p_6 >, \psi_v)]|) =
\end{aligned}$$

$$\begin{aligned}
&\textstyle\bigcup_{i=0}^{1} |[\omega_t^i]|(\bigcup_{j=0}^{3} |[\omega_r^j]|(|[< p_1, p_2 >]|)) \\
&\bigcup |[< p_4, p_6 >]| \bigcup |[\psi_v]|(|[< p_4, p_6 >]|) \\
&\bigcup |[\psi_h]|((|[< p_4, p_6 >]|) \bigcup |[\psi_v]|(|[< p_4, p_6 >]|)) =
\end{aligned}$$

$$\begin{aligned}
&\textstyle\bigcup_{i=0}^{1} |[\omega_t^i]|(\{|[\omega_r^0]|(|[< p_1, p_2 >]|), |[\omega_r^1]|(|[< p_1, p_2 >]|), \\
&\qquad\qquad |[\omega_r^2]|(|[< p_1, p_2 >]|), |[\omega_r^3]|(|[< p_1, p_2 >]|)\}) \\
&\bigcup |[< p_4, p_6 >]| \bigcup |[\psi_v]|(|[< p_4, p_6 >]|) \\
&\bigcup |[\psi_h]|(|[< p_4, p_6 >]|) \bigcup |[\psi_h]|(|[\psi_v]|(|[< p_4, p_6 >]|)).
\end{aligned}$$

4.8. The Information Load of $\mathcal{LLP}$ Expressions

A line pattern may have several Gestalts each of which is represented by a $\mathcal{LLP}$ expression. In order to disambiguate the Gestalts of a line pattern and select its perceived structure, we may follow the Gestalt disambiguation idea of SIT by proposing a complexity measure for $\mathcal{LLP}$ expressions. However, the definition of such a measure must be supported by empirical research which is outside the scope of this paper. Nevertheless, since the class of linear line patterns is a subclass of line patterns and because SIT provides an empirically supported definition of complexity measure for linear line patterns (i.e. the information load), we may consider the subclass of $\mathcal{LLP}$ expressions that denote linear line patterns and reformulate the empirically supported definition of information load, which is provided by SIT, for this class of $\mathcal{LLP}$ expressions.

Definition 4.6. Let A_{posp} be the set of position pairs, A_{trans} be the set of Euclidean transformations, $t, t_i \in \mathcal{LLP}$ for $1 \leq i \leq n$, $\psi \in \Psi \subset A_{trans}$, and $\omega, \omega_1, \ldots, \omega_{n-1} \in \Omega \subset A_{trans}$. The information load I^{llp} of $\mathcal{LLP}$ expressions that denote linear line patterns can be recursively defined as follows:

- $I^{llp}(t) = 1$ for $t \in A_{posp} \cup A_{trans}$
- $I^{llp}(\ Iter(t, \omega, m)\ ) = I^{llp}(t) + I^{llp}(\omega) + 1$
- $I^{llp}(\ Sym(t, \psi)\ ) = I^{llp}(t) + I^{llp}(\psi)$
- $I^{llp}(\ Alt(t, t_1, \ldots, t_n, \omega_1, \ldots, \omega_{n-1})\ ) = I^{llp}(t) + \sum_{i=1}^{n} I^{llp}(t_i) + \sum_{i=1}^{n-1} I^{llp}(\omega_i) + n$
- $I^{llp}(\ Chunk(t)\ ) = I^{llp}(t)$ if t contains only one element from $A_{posp} \cup A_{trans}$;
  $= I^{llp}(t) + 1$ otherwise
- $I^{llp}(\ Comp(t_1, \ldots, t_n)\ ) = \sum_{i=1}^{n} I^{llp}(t_i) + n - 1.$

Empirical experimentations with the class of connected line patterns or even with the more general class of unconnected line patterns should establish a reliable definition of complexity measure that can be used for this representation system. In the following, we write $E^{llp}(s)$ to refer to the set of $\mathcal{LLP}$ expressions that denote a line pattern s in D, i.e. $E^{llp}(s) = \{e \in \mathcal{LLP} \mid [\![e]\!] = s\}$. The subset of $E^{llp}(s)$ consisting of elements with minimum complexity value according to definition 4.6 is denoted by $Perceived_{llp}(s)$, i.e.

$$Perceived_{llp}(s) = \{e \in E^{llp}(s) \mid \forall e' \in E^{llp}(s)\ I^{llp}(e) \leq I^{llp}(e')\}$$

The set $Perceived_{llp}(s)$ is called the $\mathcal{LLP}$-predicted structures of line pattern s. We show here that the SIT-predicted structures of a linear line pattern with a two-dimensional code s are represented by the expressions in $Perceived_{llp}(s)$. The proof is based on equivalent $\mathcal{LSP}$ and $\mathcal{LLP}$ expressions.

Definition 4.7. A $\mathcal{LSP}$ expression t^{lsp} and a $\mathcal{LLP}$ expression t^{llp} are equivalent, written as $t^{lsp} \sim t^{lsp}$, if and only if they denote the same linear line pattern, induce the same constituent structure (hierarchical clustering of the pattern), and have the same information load.

The proof is constructed inductively on the basis of equivalence between $\mathcal{LSP}$ and $\mathcal{LLP}$ expressions and a strict translation of line segments as position pairs and of angles between them as Euclidean transformations.

Theorem 4.1. *For any linear line pattern L with two-dimensional representation s, the SIT-predicted structures of L are represented by expressions from $Perceived_{llp}(s)$.*

Proof. Theorem 3.1 states that the SIT-predicted structures of a linear line pattern with primitive code s are represented by expressions in $Perceived_{lsp}(s)$. Let $s^{lsp} \sim s^{llp}, t^{lsp} \sim t^{llp}$, and $t_i^{lsp} \sim t_i^{llp}$ for $1 \leq i \leq n$ be equivalent expressions denoting linear line patterns S, T and T_i, having the information loads k_s, k_t and k_{t_i}, respectively. Moreover, let $\tau \in A_{trans}$ be a Euclidean transformation, and $y, y_i, y_i' \in A_{ang}$ for $1 \leq i \leq n$ be angles between line segments. The proof is based on a correspondence between $\mathcal{LSP}$ and $\mathcal{LLP}$ operators. For each pair of corresponding operators we show that they denote the same linear line pattern, induce the same constituent structure (have the same number of arguments), and result in the same information load when they are applied to equivalent expressions. This implies that $Perceived_{sit}(s)$ and $Perceived_{lsp}(s)$ represent perceptual structures that pairwise have the same constituent structure and have the same information load.

Iteration operator: A linear line pattern L with iteration structure has a primitive code of the form: $Xy \cdots yX$, where X is the primitive code of a line pattern T and y is the angle between subsequent occurrences of T. As argued in section 4.1, the linear line pattern L can be analyzed in terms of the two-dimensional representation of T and a Euclidean transformation, composed of a translation (possibly a zero-distance translation) and a rotation (possibly a zero-degree rotation) with respect to a rotation point. According to definitions 3.2 and 4.4, there are expressions $Iter(t^{lsp}, y, m) \in \mathcal{LSP}$ and $Iter(t^{llp}, \tau, m) \in \mathcal{LLP}$ which denote representations of L. Following definitions 3.3 and 4.6, the information load of both expressions is $k_t + 2$. Finally, these expressions induce the same constituent structure since iteration operators in both expressions have one pattern as an argument and the arguments are equivalent expressions.

Symmetry operators: We distinguish two cases: symmetrical linear line patterns with or without pivot element. A linear line pattern L with the symmetry structure without pivot element has a primitive code of the form: XyX^{-1}, where X is the primitive code of linear line pattern T, X^{-1} is the primitive code of the reflection of T, and y is the angle between them. As argued in section 4.2, the linear line pattern L can be analyzed in terms of the two-dimensional representation of T and a Euclidean reflection transformation with respect to the reflection axis between T and its reflection. According to definitions 3.2 and 4.4, there are expressions $Sym_e(t^{lsp}, y) \in \mathcal{LSP}$ and $Sym(t^{llp}, \tau) \in \mathcal{LLP}$ which denote representations of L. Following definitions 3.3 and 4.6, the information load of both expressions is $k_t + 1$. These expressions induce the same constituent structure since symmetry operators in both expressions have one pattern as an argument and the arguments are equivalent expressions.

A linear line pattern L with the symmetry structure with pivot element has a primitive code of the form: $XyX'y'X^{-1}$, where X and X' are the primitive codes of linear line patterns T and S, respectively. The linear line pattern L can be analyzed as a two-dimensional composition of two subpatterns: the symmetrical part and the pivot element. According to definitions 3.2 and 4.4 there are expressions $Sym_o(t^{lsp}, s^{lsp}, y, y') \in \mathcal{LSP}$ and $Comp(Sym(t^{llp}, \tau), s^{llp}) \in \mathcal{LLP}$ which denote representations of L. Following definitions 3.3 and 4.6, the information load of both expressions is $k_t + k_s + 2$. These expressions induce the same constituent structure since they contain two equivalent expressions as arguments.

Alternation operators: A linear line pattern L with the right-alternation structure has a primitive code of the form: $Xy_1X_1y'_1X\cdots Xy_nX_n$, where X and X_i are the primitive codes of linear line patterns T and T_i, respectively. As argued in section 4.4, the linear line pattern L can be analyzed in terms of the two-dimensional representations of T, T_i, and Euclidean transformations $\tau_1, \ldots, \tau_{n-1}$, each composed of a translation (possibly a zero-distance translation) and a rotation (possibly a zero-degree rotation) with respect to a rotation point. According to definitions 3.2 and 4.4 $Alt_r(t^{lsp}, t_1^{lsp}, \ldots, t_n^{lsp}, y_1, \ldots, y_n, y'_1, \ldots, y'_{n-1}) \in \mathcal{LSP}$ and $Alt(t^{llp}, t_1^{llp}, \ldots, t_n^{llp}, \tau_1, \ldots, \tau_{n-1}) \in \mathcal{LLP}$ denote the representations of L. Following definitions 3.3 and 4.6, the information load of both expressions is $k_t + \sum_{i=1}^{n} k_{t_i} + 2n - 1$. These expressions induce the same constituent structure since they contain $n+1$ equivalent expressions as arguments. The same can be shown for the left alternation structures.

Chunk operator: According to definitions 3.2 and 4.4, there are expressions $Chunk(t^{lsp}) \in \mathcal{LSP}$ and $Chunk(t^{llp}) \in \mathcal{LLP}$ which denote the representations of the same linear line pattern. Following definitions 3.3 and 4.6, the information load of both expressions is k_t if t^{lsp} and t^{llp} contain one primitive element, and $k_t + 1$ otherwise. Finally, these expressions induce the same constituent structure since in these expressions the arguments are equivalent expressions.

Concatenation operator: A linear line pattern L with the concatenation structure has a primitive code of the form: $X_1y_1 \ldots y_{n-1}X_n$, where X_i is the primitive code of linear line patterns T_i for $1 \leq i \leq n$. As argued in section 4.5, the linear line pattern L can be analyzed in terms of the two-dimensional representations of $T_1, \ldots, T_n$. Definitions 3.2 and 4.4 imply that $Con(t_1^{lsp}, y_1, \ldots, y_{n-1}, t_n^{lsp}) \in \mathcal{LSP}$ and $Comp(t_1^{llp}, \ldots, t_n^{llp}) \in \mathcal{LLP}$ denote representations of L. Following definitions 3.3 and 4.6, the information load of both expressions is $\sum_{i=1}^{n} k_{t_i} + n - 1$. Finally, these two expressions induce the same constituent structure since they have n pattern arguments, and the arguments are pairwise equivalent. ■

The following theorem assumes the uniqueness claim and states that under this assumption if a linear line pattern is not ambiguous, then there is a unique $\mathcal{LLP}$ expression with minimum complexity value that denotes the linear line pattern and vice versa.

THEOREM 4.2. *Under the uniqueness claim assumption and for a linear line pattern L with two-dimensional code s :*

$$|Perceived_{llp}(s)| = 1 \Leftrightarrow L \textit{ is not ambiguous}$$

Proof. Theorem 3.1 states that the SIT-predicted structures of a linear line pattern with primitive code s' are represented by expressions in $Perceived_{lsp}(s')$ and theorem 4.1 states that the SIT-predicted structures of the linear line pattern with its two-dimensional code s are represented by expressions in $Perceived_{llp}(s)$. This implies that $|Perceived_{lsp}(s')| = |Perceived_{llp}(s)|$ for any line pattern L with primitive code s' and two-dimensional code s. Theorem 3.2 states that any linear line pattern L with primitive code s' is not ambiguous if and only if $|Perceived_{lsp}(s')| = 1$.

This concludes that any linear line pattern with two-dimensional code s is not ambiguous if and only if $|Perceived_{llp}(s)| = 1$. ■

5. CONCLUSION AND FUTURE RESEARCH

We are far from a precise all-encompassing model of human visual perception. Any model must either be limited to a small subset of phenomena or a specific mechanism, or it must sacrifice preciseness. Or it has to do what we have done here: address a very stylized caricature of the actual phenomena. In choosing for the 'caricature' approach, we have been inspired by the competence/performance-distinction in linguistics, which is completely artificial but which has been very fruitful.

In this paper, we have developed two formal languages each of which represents a certain class of Gestalts for a certain class of perceptual patterns. The first language is designed for linear line patterns and covers those Gestalts that can be described in terms of regularities of line attributes such as lengths and relative angles. The second language is designed for the larger class of line patterns and covers Gestalts that can be described in terms of translations, rotations, and reflections of the constituent line segments. It should be emphasized that the proposed languages cannot represent Gestalts that depend on the proximity effect or on other visual attributes such as color, texture and thickness. These limitations will be addressed in future research.

The approach presented here does not account for proximity-based Gestalts. The proximity-based Gestalt of a pattern depends on relative distances between constitutive pattern elements: relatively close elements tend to be perceptually grouped. SIT is only concerned with regularity-based Gestalts and does not account for the interaction between proximity and regularity. Proximity-based Gestalts differ from regularity-based Gestalts since they cannot be selected by means of information complexity. The information complexity of the proximity-based Gestalt of a pattern can be equal to the information complexity of a regularity-based Gestalt which describes the pattern as having a random grouping structure. Thus, it may be the case that the proximity-based Gestalt of a pattern is preferred to a regularity-based Gestalt that has a lower information complexity. For a detailed discussion on the interaction between proximity and regularity see (Dastani, 1998).

In this paper, we have considered only those Gestalts that are based on regularity of spatial relations among line segments. The proposed $\mathcal{LLP}$ language can be extended to capture the Gestalt of visual patterns which are based on regularity among values of different visual and spatial attributes such as color, size, and texture (Dastani, 1998). For such extensions primitive visual elements can be defined as n-tuples of attribute values, instead of tuples of position values, and regularity among primitive visual elements can be captured in terms of n-tuples of transformations that can be applied to visual elements.

$\mathcal{LLP}$ is designed for the general class of line patterns. For linear line patterns we have reformulated the empirically supported information load, which is provided by SIT, and shown that it can be applied to select the perceived regularity-based Gestalts of linear line patterns represented by $\mathcal{LLP}$ expressions. The plausibility of the claim that this reformulation of information load can also be applied to the class of linear line patterns is based on empirical experimentation, reported by SIT, with line patterns that can be encoded as strings. Nevertheless, we believe that this reformulation of the information load needs to be verified by future experimentation. A possible research direction would therefore be to set up experiments to verify the empirical validity of the proposed information load for connected line patterns and perhaps to establish a new empirically supported information load for the general class of line patterns.

It is clear that we have not been concerned with the experimental assessment of the predictions of our models. Again, linguistics has served as a role model here: for the time being, accounting for rather obvious uncontroversial intuitions that the researcher may have about his own perception, has constituted enough of a challenge. To the extent that we have come up with notions that have some initial plausibility, it would be desirable to complement this work with more systematic experimentation.

It should also be noted that in the system we described, two-dimensional patterns are analyzed in terms of two-dimensional regularities. But humans tend to project three-dimensional interpretations on their input patterns. Several phenomena which are related to this tendency are therefore ignored by the current version. Moreover, we have described Gestalts as being determined solely by regularity, ignoring the role of recognizing previously experienced patterns. It is clear that this must be factored in. This may be done by interpreting our complexity measure in an information-theoretic fashion: recurring patterns receive a lower complexity value because they are 'expected'. Finally, analyzing Gestalts as static codes is only a preliminary step in the direction of a cognitively realistic description. A Gestalt is not characterized by one code, but by a stable group of transformations which map codes onto alternative but compatible codes.

Acknowledgement We would like to thank Reinhard Suck and Peter van der Helm for their valuable comments and suggestions.

References

Boselie, F. (1988). Local versus global minima in visual pattern completion. *Perception & Psychophysics*, *43*, 431-445.

Boselie, F., & Wouterlood, D. (1989). The minimum principle and visual pattern completion. *Psychological Research*, *51*, 93-101.

Buffart, H., Leeuwenberg, E., & Restle, F. (1981). Coding theory of visual pattern completion. *Journal of Experimental Psychology: Human Perception and Performance*, *7*, 241-274.

Dastani, M. (1998). *Languages of perception.* Ph.D. thesis, University of Amsterdam, The Netherlands.

Grünwald, P. (2000). Model selection based on minimum description length. *Journal of Mathematical Psychology*, *44*, 133-152.

Koffka, K. (1935). *Principles of Gestalt psychology.* New York: Harcourt, Brace and World.

Leeuwenberg, E. (1971). A perceptual coding language for visual and auditory patterns. *American Journal of Psychology*, *84*, 307-349.

Li, M., & Vitányi, P. (1993). *An introduction to Kolmogorov complexity and its applications.* Springer Verlag.

Van der Helm, P., & Leeuwenberg, E. (1986). Avoiding explosive search in automatic selection of simplest pattern codes. *Pattern Recognition*, *19*, 181-191.

Van der Helm, P., & Leeuwenberg, E. (1991). Accessibility: A criterion for regularity and hierarchy in visual pattern code. *Journal of Mathematical Psychology*, *35*, 151-213.

Van der Helm, P., Van Lier, R., & Leeuwenberg, E. (1992). Serial pattern complexity: Irregularity and hierarchy. *Perception*, *21*, 517-544.

Van der Vegt, J., Buffart, H., & Van Leeuwen, C. (1989). The structural memory: A network model for human perception of serial objects. *Psychological Research*, *50*, 211-222.

Stins, J., & Van Leeuwen, C. (1993). Context influence on the perception of figures as condition upon perceptual organization strategies. *Perception & Psychophysics*, *53 (1)*, 34-42.

Van Leeuwen, C., & Buffart, H. (1989). Facilitation of retrieval by perceptual structure. *Psychological Research*, *50*, 202-210.

Van Leeuwen, C., Buffart, H., & Van der Vegt, J. (1988). Sequence influence on the organization of meaningless serial stimuli: economy after all. *Journal of Experimental Psychology: Human Perception and Performance*, *14*, 481-502.

Wertheimer, M. (1923). Untersuchungen zur Lehre von der Gestalt. *Psychologische Forschung*, *4*, 301-350.

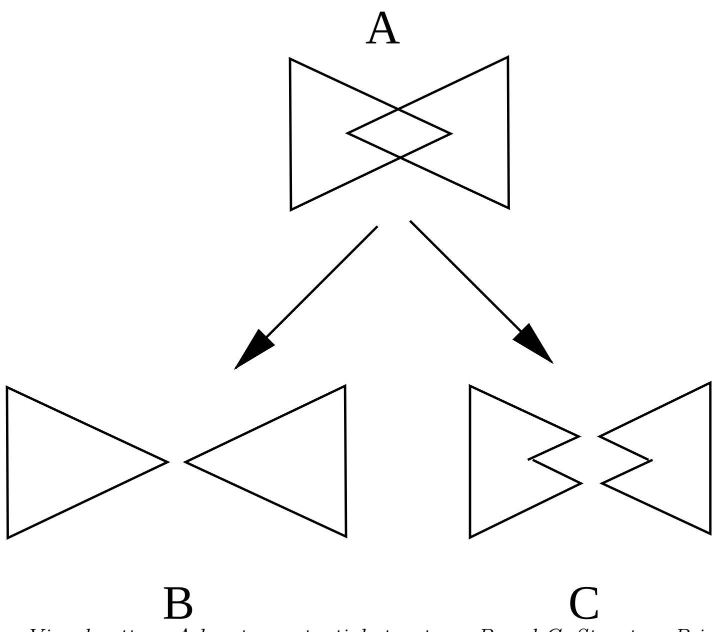


FIG. 1. *Visual pattern A has two potential structures B and C. Structure B is perceptually preferred.*

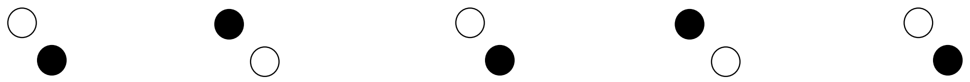


FIG. 2. *The proximity factor, rather than the similarity factor, dominates perceptual grouping.*

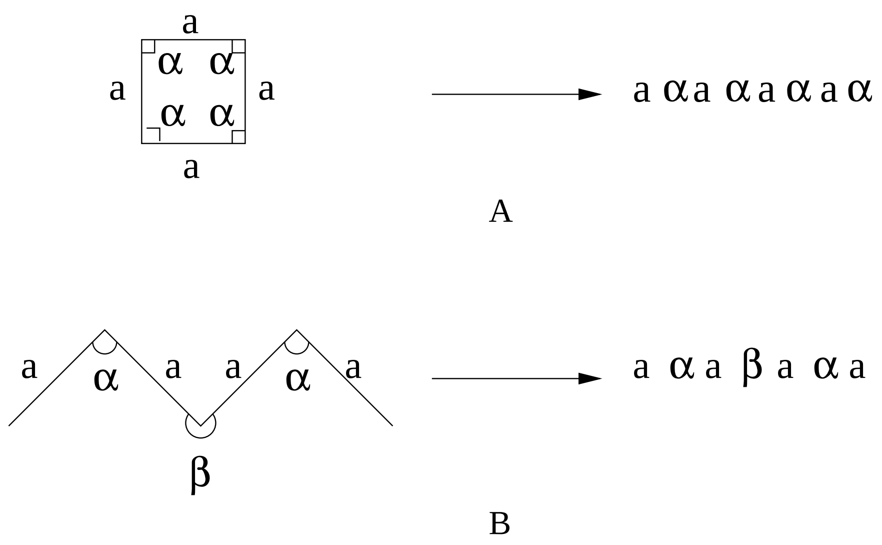
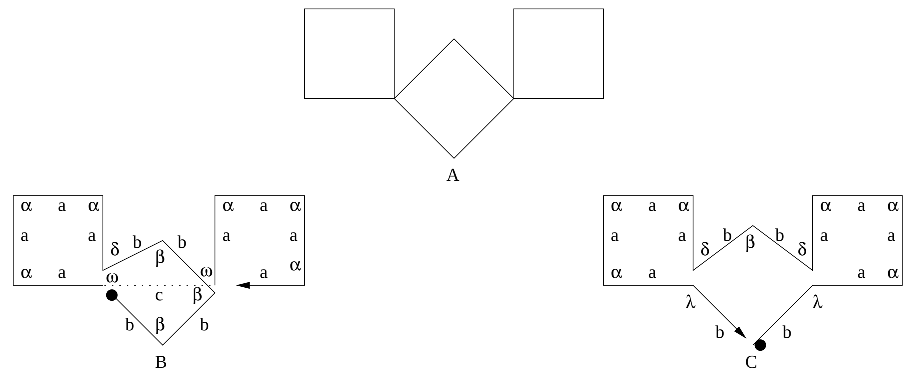


FIG. 3. *Encoding line patterns as strings.*



Primitive code of B:

b β b β b β b δ a α a α a α a ωc ωa α a α a α a

Final Code of B:

S[S[(b),(β)],(β)] δ S[S[S[(a),(α)],(α)],S[(ω),(c)]]

Information Load(Final Code of B)=9

Primitive code of C:

b λ a α a α a α a δ b β b δ a α a α a α a λ b

Final Code of C:

S[(b)(λ)S[S[(a),(α)],(α)](δ)(b),(β)]

Information Load(Final Code of C)=8

FIG. 4. *Two possible encodings of one line pattern.*

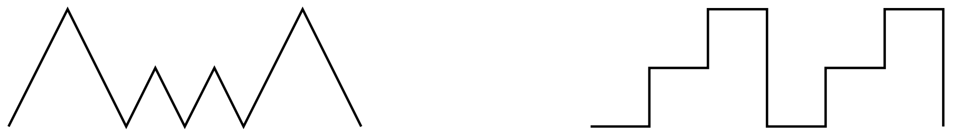


FIG. 5. *Examples of linear line patterns.*

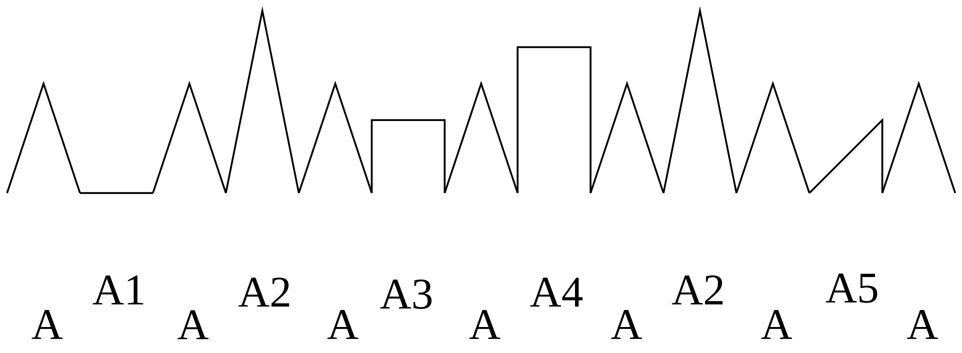


FIG. 6. *The regularity among alternating elements is not perceived.*

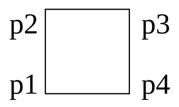


FIG. 7. *An example of a connected line pattern.*

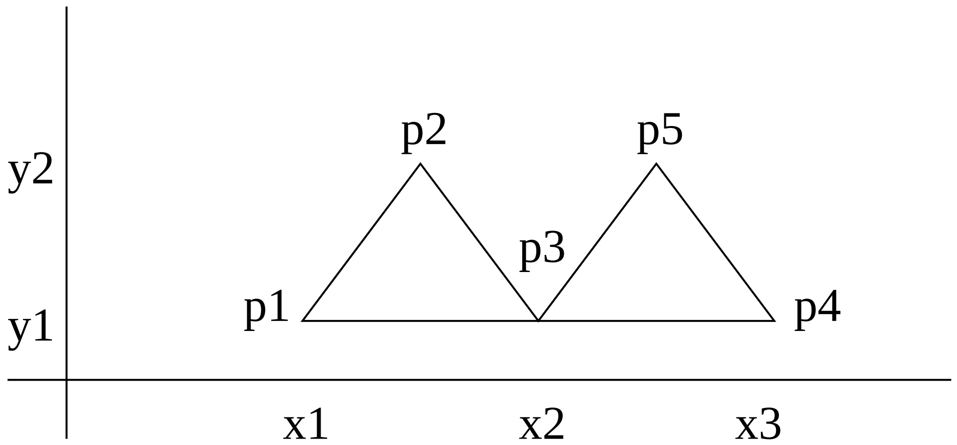


FIG. 8. *Position-based iteration structure.*

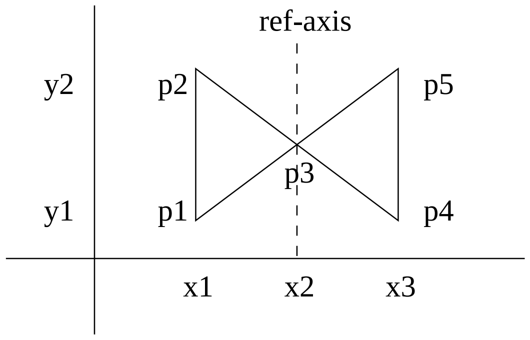


FIG. 9. *A visual pattern with symmetry structure.*

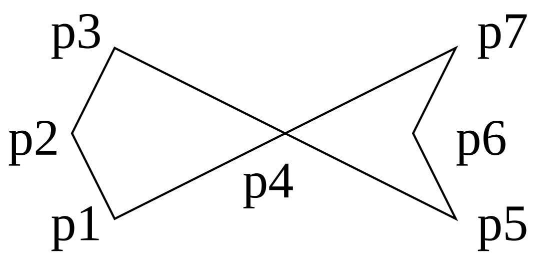


FIG. 10. *The spatial relation between chunked line segments is preserved in the symmetrical halves.*

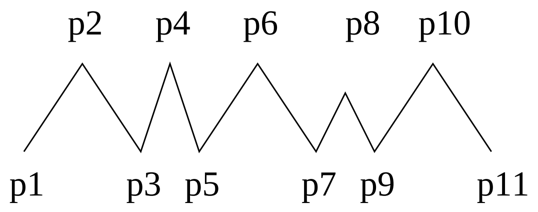


FIG. 11. *An example of a linear line pattern which has alternation structure.*

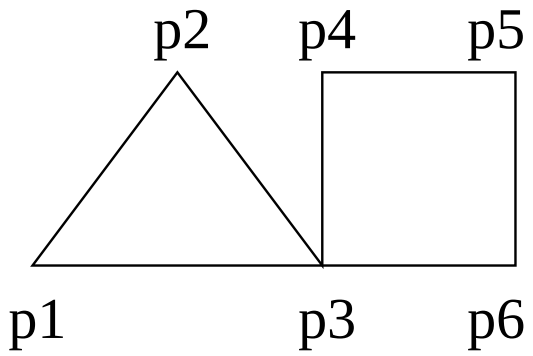


FIG. 12. *An example of a line pattern which should be described as having the composition structure.*

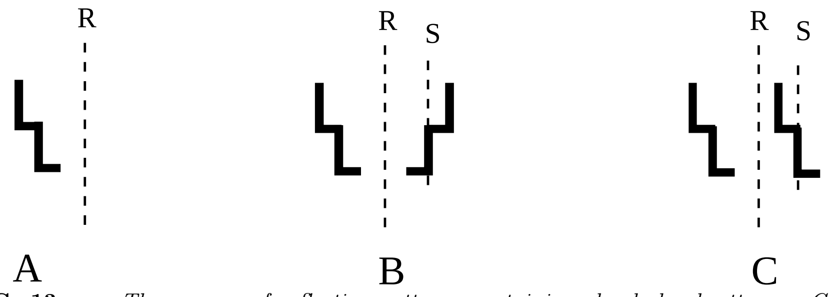


FIG. 13. *The process of reflecting patterns containing chunked subpatterns. Chunked subpatterns are first reflected according to the actual reflection axis R and then according to their internal reflection axis S.*

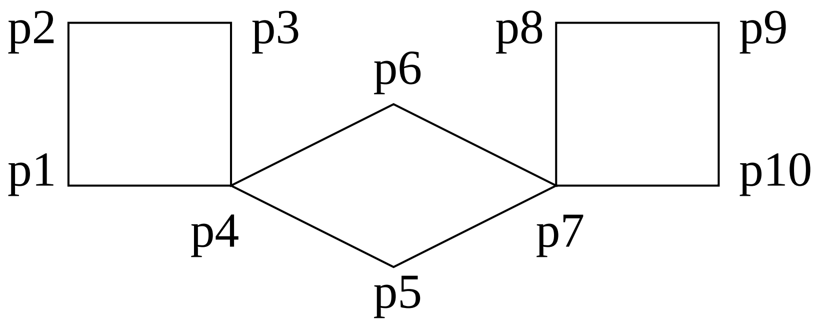


FIG. 14. *A visual pattern perceived as consisting of two squares and one diamond.*