Languages of Perception

M. M. Dastani

Languages of Perception

ILLC Dissertation Series 1998-05

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
phone: +31-20-5256090
fax: +31-20-5255101
e-mail: illc@fwi.uva.nl

Languages of Perception

Academisch Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam,
op gezag van de Rector Magnificus
prof.dr J.J.M. Franse
ten overstaan van een door het college voor promoties ingestelde
commissie in het openbaar te verdedigen in de
Aula der Universiteit
op woensdag 9 december 1998 te 9:00 uur

door

Mohammad Mehdi Dastani

geboren te Fasa (Iran)

Promotores:

Prof.dr.ir. Remko Scha
Faculteit der Geesteswetenschappen
Universiteit van Amsterdam
Spuistraat 134
1012 VB Amsterdam

Dr. Peter van Emde Boas
Faculteit Wiskunde, Informatica, Natuur- en Sterrenkunde
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam

Copromotor:

Prof.Dr. Bipin Indurkhya
Department of Computer Science
Tokyo University of Agriculture and Technology
2-24-16 Nakacho, Tokyo 184-8588 Japan

Cover design by Sophie van der Sluis

Contents

Acknowledgments

I am very grateful to my supervisors, Remko Scha, Peter van Emde Boas and Bipin Indurkhya, for their invaluable advice, guidance, encouragement, and support.
Working closely with Remko Scha was very enjoyable. He was inspiring and stimulating, not only because of his sound criticisms and clear ideas, but also because of his enthusiasm and his artistic, social and cultural views. He played a key role in developing the ideas of this thesis.
Peter van Emde Boas was very generous in giving his time to discuss the mathematical issues of the thesis. I always felt comfortable to go to him and ask for his comments. His expertise and detailed feedback were indispensable for this research.
I would like to thank Bipin Indurkhya who encouraged me to carry on doing scientific research and who proved to be a invaluable source of inspiration. His expertise in analogy and context issues showed me different perspectives on my research. He was very patient in his supervision via intense e-mail correspondence from Japan.
A special thanks goes to Henk Zeevat who supported me in obtaining the research position at the Computational Linguistic Department. I appreciate the numerous discussions and brainstorms we had about all kinds of related issues.
I also would like to acknowledge Kwee Tjoe Liong who was always available and willing to provide me all kinds of background knowledge and practical details. It was really a pleasure to work with him in one room and to have many inspiring discussions.

The meetings and discussions with Cees van Leeuwen, Jos de Bruin, Dejuan Wang, and Yuri Engelhardt were always stimulating and productive. I enjoyed the gatherings and discussions with them. I really had a fun time sharing my working space with Mark van de Berg, Noor van Leusen, and Jos Skolnik, and I thank them for their cheerful company.
Of course, my family and friends were a great support to me during my research period. I thank them for backing me up emotionally in times of crisis.

Parts of this thesis were previously published in [BDP+93], [DOS93], [ID97], [Das96], [Das97a], [Das97b], [DIS97], [DI97], [DT98].

Amsterdam, November 1998.

Chapter 1

Introduction

In everyday life, we are confronted with visual information provided by the environment. This visual information may originate from the scenes of cities or forests, but also from television images, computer interfaces, and many other natural or artificial sources. In general, we have no difficulties in recognizing meaningful entities in the visual information we receive, and in organizing it coherently. For example, when we look at an urban neighborhood which we have never seen before, we perceive individual buildings and separate them from each other even when they are continuously bounded to each other. Also, in a natural environment, we easily perceive individual flowers, plants, or trees and discriminate them from each other even when one is partially hidden by the other. Although this ability seems to be effortless and direct, it is far from trivial to understand, describe, and model it.

An understanding of the human visual system would have a great impact in various scientific and technological fields. Besides its importance in theoretical study of the human mind, there is a growing practical interest in computational models of the visual system. Such models are, for example, essential for autonomous robots which receive visual information from the environment via a camera and have to analyze real world situations, construct hypotheses about objects in those situations, and interact with these objects. Also, with the upcoming multimedia systems and image databases, there is a great need for automatic systems that can analyze, classify, and annotate images according to the way that humans perceive them.
A model of the human visual system is not only useful for analyzing im-

ages, but it can also be of great interest for graphical design and generation of images for human use. In the fields of information design and information visualization the goal is to generate images that represent information such that a human viewer can understand and extract this information by looking at the images. This can only be the case when the image generator, either a human designer or an automatic image design system, has an understanding of the human visual system to ensure that the generated images will be perceived as intended.

In order to understand and model the human visual system, one should analyze visual information as it is presented to human visual sensors (human eyes) and describe how this information can be mapped into meaningful entities for which we have names and which we can place in a conceptual framework. We assume two steps in mapping visual information into meaningful entities. The first step concerns the low-level structuring of visual information. This step provides the constituent structure of visual information, i.e. it determines: A) constituents of visual information and B) how they are composed to build up larger wholes. In the second step, the visual constituents resulting from the first step should then be interpreted in some conceptual framework. The interpretation of visual constituents is based on many factors such as reasoning and past experiences. It should be noted that these two steps interact with each other: when the structured visual information from the first step cannot be placed into a conceptual framework coherently, the low-level structuring step should provide an alternative constituent structure for the visual input.

In this thesis, we will concentrate on the first step of structuring visual information and investigate the principles on the basis of which visual constituents are composed to form larger wholes. However, we do not discuss how primitive visual constituents are determined. In an ultimate theory, these should probably be pixels, line-segments, and/or edges between contrastive areas. But for the moment we will avoid commitments about this issue, by focusing on classes of pictures for which a particular type of higher-level units may be assumed as primitives. What we focus on in this thesis is the problem of gestalt perception: assuming a set of primitive elements, we try to account for the phenomena that pictures built up of these elements are perceived by humans as having a particular hierarchical constituent structure. In the study of gestalt perception primitive elements are assumed to be composed and structured unconsciously and directly according to some innate principles that are believed to underlie the human visual system. During the last century, there have been several formulations for these suggested innate principles. We start with a recent formulation of the innate principles of the human visual system and develop a mathematical model for gestalt perception. We discuss various aspects of gestalt perception and work out an application in which a model of human visual system is indispensable.

The organization of this thesis is as follows. In chapter 2, we briefly explain the gestalt approach as originally formulated by Wertheimer and discuss the construction rules that have been claimed to be based on the innate principles of the human visual system. Since Wertheimer, more fundamental and general principles have been proposed. An overview of the development of the gestalt tradition will be given. In particular, we will focus on the approach which was developed by Leeuwenberg. Leeuwenberg has proposed a theory, called Structural Information Theory, where the notions of regularity and complexity play essential roles. According to this theory, visual information can be perceived as having different structures, but the actually perceived structure is the most regular one. The regularity of perceptual structures is measured by the complexity of their descriptions: the more regularities are captured by a description, the less complex it is. In this way, Structural Information Theory claims that the preferred perceptual structure of visual information is the simplest structure. Leeuwenberg and Van der Helm have defined the notions of regularity and complexity in a formally precise way. Based on these notions, they have proposed a coding system to represent visual patterns and their perceptual structures. We will criticize this coding system by showing that it is not powerful enough to represent certain classes of visual information.

In chapter 3, we introduce formal coding languages consisting of expressions which represent the perceptual structures of one- and two-dimensional visual information. First, we consider one-dimensional string patterns. Examples of these patterns are $aaabbb$, $abba$, $abcklm$, $rzkw$, etc. Looking at these string patterns, one can describe them in various ways. However, people usually describe the first pattern as consisting of some a's followed by some b's, the second pattern as consisting of ab followed by its mirror image, the third pattern as consisting of abc and klm, and the last pattern as consisting of the individual letters r, z, k, and w. Although the first three string patterns can be described in the same way as the last string

pattern, they are usually described in terms of larger chunks of subpatterns. In order to represent these string patterns and their perceptual structures in a formally precise way, an algebraic coding language is specified. The expressions of this coding language are quite similar to those proposed by Leeuwenberg and Van der Helm. The difference is that we consider these expressions as terms of an algebra while Leeuwenberg and Van der Helm do not use such an algebraic framework.
We then continue by extending this algebraic coding language. The extension of this coding language is based on our intuition that the knowledge and accessibility of some relations defined on constituents of perceptual patterns in general, and string patterns in particular, influences the perceptual structures of string patterns. For example, the fact that in the western alphabet c is the successor of b which is in turn the successor of a and the fact that m is the successor of l which is in turn the successor of k effects the perception of the string pattern $abcklm$. As a result, the string pattern $abcklm$ is perceived as consisting of abc and klm.
Once a coding language for one-dimensional string patterns is specified, we proceed with studying two-dimensional visual patterns and discuss how these patterns and their perceptual structures can be represented. In particular, we consider two-dimensional visual patterns in terms of visual attribute values and claim that their perceptual structures can be described in terms of regularities in visual attribute values. Subsequently, a coding language for two-dimensional visual patterns is introduced. The expressions of this coding language represent the perceptual structure of two-dimensional visual patterns. The contribution of this chapter can be summarized as the development of the algebraic framework and, more importantly, the specification of the coding language for a large class of two-dimensional visual patterns that could not be covered by the coding system proposed by Leeuwenberg and Van der Helm.

In chapter 4, it is explained how the perceptual structure of two-dimensional visual patterns as represented by the expressions of our coding language can be computed. The computation of perceptual structures of two-dimensional visual patterns is considered as a parsing process. We use the specification of the coding language to define a set of parsing rules by means of which possible structures of visual patterns can be analyzed. Following the terminology of Leeuwenberg and Van der Helm, these parsing rules will be called coding rules. In order to parse a two-dimensional visual pattern, we start out with the unstructured representation of that visual pattern and proceed

with applying the coding rules to it repeatedly. The repeated application of coding rules to the unstructured representation of a two-dimensional visual pattern results in structured representations of that visual pattern which express possible perceptual structures of the parsed visual pattern. The preferred perceptual structure of a two-dimensional visual pattern is determined by selecting the simplest of these structures.
Later in chapter 4, the computational complexity of the parsing process is discussed. We show that the parsing process is deterministic but exponential in the size of the unstructured representation of the input patterns.
Finally, we discuss the proximity structure of two-dimensional visual patterns. The proximity structure of two-dimensional visual patterns is a perceptual grouping of visual elements such that relatively close visual elements form one proximity cluster. In the coding system proposed by Leeuwenberg and Van der Helm the proximity structure of visual patterns is disregarded. In order to integrate the factor of proximity in our approach, existing cluster methods can be used to determine the proximity clusters that are involved in a two-dimensional visual pattern. The proximity clusters involved in a pattern are then integrated in the initial representation of that pattern which is the subject of the parsing process. Since the parsing process starts with the proximity-based representation of a pattern, the resulting perceptual structures will express this proximity structure as well.

In most cases, visual patterns are perceived within a certain context. The context may be other simultaneously present visual patterns, an application in which visual patterns are used in a certain way, or the past experience of the human viewer. In chapter 5, the possible effects of context on the perceived structure of visual patterns will be discussed. We argue that a visual pattern presented in a certain context may be perceived differently than when the same pattern is perceived in isolation or in another context. The context effect will be discussed for both one- and two-dimensional visual patterns. To illustrate this effect, we consider proportional analogies containing one- and two-dimensional visual patterns. A proportional analogy consists of four perceptual patterns (elements) and follows the scheme that can be represented as: A is to B as C is to D, where A, B, C, and D are perceptual patterns. In particular, we consider proportional analogy as the context for the perceptual patterns involved. This context requires that A is related to B in the same way as C is related to D.
An example of proportional analogy that consists of one-dimensional string patterns is "*abccba* is to *ppqrpp* as *abccccba* is to *ppqrqrpp*". The analogical

relation posited between the string patterns is a context which effects the perceptual structures of these string patterns. In this example, the first string pattern in isolation may be perceived as consisting of two mirroring string patterns: *abc* and *cba*. However, because of the context effect which is created by the analogical relation, the string pattern *abccba* will be perceived as consisting of three string patterns: *ab*, *cc*, and *ba*; other perceptual structures do not satisfy the requirement imposed by the context of proportional analogy. We elaborate on proportional analogies and propose an algebraic model for them. In this model, the preferred perceptual structure of visual patterns in the context of proportional analogy can be determined.

As we have noticed at the beginning of this introduction, a model of the human visual system is not only interesting for analyzing visual input (images) to determine their perceptual structures, but it is also essential in generating visual output (images) for human use. For example, visual images like bar-charts, pie-charts, graphs, maps, etc. are used to represent and communicate information. In generating images for this purpose, one should in fact design them in such a way that the desired information can easily be perceived by human viewers. In order to guarantee that the human viewer correctly perceives the represented information, a model of the human visual system is necessary. In chapter 6, we concentrate on data visualization and define it as the inverse of the interpretation of visual patterns. Since the interpretation process of visual patterns includes the process of visual perception, data visualization should involve the inverse process of visual perception. In this way, the model of the human visual system is used to generate images instead of parsing them.
In visualizing data, quantitative and qualitative relations can be represented by means of different visual attributes like position, color, size, shape, texture, etc. These attributes may induce quantitative and qualitative perceivable relations among visual elements. For example, visual elements that have different saturation values but the same hue value, like different saturations of blue, are perceived as being in an ordinal relation (a qualitative relation), i.e. one perceives that a visual element is more saturated than another one. This ordinal relation can be used to represent a data relation like the older-than relation. In fact, the core idea of data visualization is to use these perceivable relations to represent data relations. We will study various visual attributes and discuss the perceivable relations that they may induce on visual elements. These visual relations differ from constituent relations (gestalts) since they do not represent grouping of visual elements,

but some quantitative or qualitative visual relations among them. In this way, this chapter covers a broader class of perceptual relations than it was covered in previous chapters.

Furthermore, it will be shown that certain data may be represented in various ways by different images, but only some of these images represent the data effectively. We argue that the effectivity of visual representations is partially related to the structural matching between data structure on the one hand and perceptual structure of representing images on the other hand. This view of effective data visualization will then be modeled in an algebraic framework.

Chapter 2

Gestalt Perception and Structural Information Theory

Human sensory systems acquire information from the surrounding environment and transform it into experiences and understanding of the environment. This process of transforming sensory inputs into structured experiences is called *perception*. In particular, perception can be considered as a parsing process by which a stimulus pattern is transformed into a structured representation of that pattern. In the following chapters of this thesis, we develop a formal model of visual gestalt perception which takes its point of departure in Leeuwenberg's *Structural Information Theory*. In this chapter, we briefly explain this theory and its background.

2.1 Gestalt Tradition

The gestalt tradition was initiated by Wertheimer. He observed that, although our environment can theoretically be interpreted as being composed of arbitrary elements, we experience it usually as being composed of certain elements within certain arrangements: *"When we are presented with a number of stimuli we do not as a rule experience "a number" of individual things, this one and that and that. Instead larger wholes separated from and related to one another are given in experience; their arrangement and division are concrete and definite"* [Wer23]. Wertheimer demonstrated his idea by an experiment in which subjects look at two light spots which are flashed on and off alternately. For a certain time-interval between flashes, a motion is perceived which is in fact not there. The factual or physical

constituents, which are *two* light spots, are not perceived.
Another example is illustrated by the visual pattern in Figure 2.1-A.

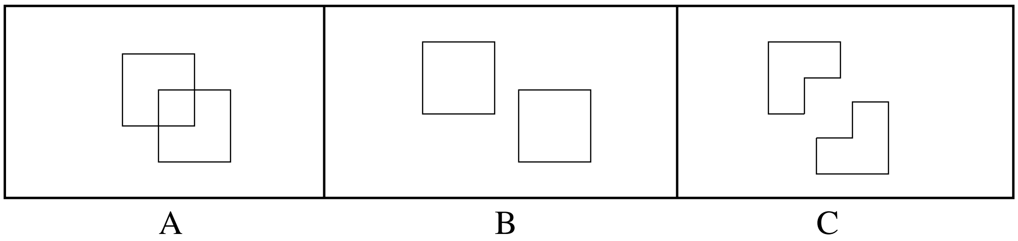


Figure 2.1: *Pattern A may have different organizations like those given in B and C.*

This pattern can be described as having different organizations as shown in Figure 2.1-B and 2.1-C. However, the perceived organization and therefore the preferred perceptual description of the pattern is the one shown in 2.1-B which implies that the pattern 2.1-A is perceived as consisting of two squares rather than two L-formed shapes.

The idea of Wertheimer has led to a psychological school, known as the *Gestalt* school, with the well-known slogan: *the whole is more than the sum of its parts*. An important agenda of the Gestalt research was to define the laws that explain the relation between patterns and their perceptual organizations. Wertheimer [Wer23] proposed a number of laws, known as *proximity, similarity, constant direction* (or *continuity*), *closure* and *habit* (or *past experience*).
The law of proximity states that elements which are positioned close together tend to be perceived as members of one perceptual group. For example, in Figure 2.2 the spatial distribution of the dots results in three perceptual groups of dots.

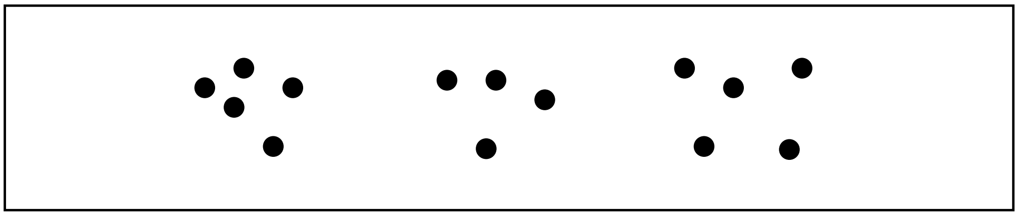


Figure 2.2: *The law of Proximity.*

The law of similarity states that similar elements tend to be perceived as forming one perceptual group. For example, in Figure 2.3 the pairs of circles and the pairs of black dots are perceived as perceptual groups.

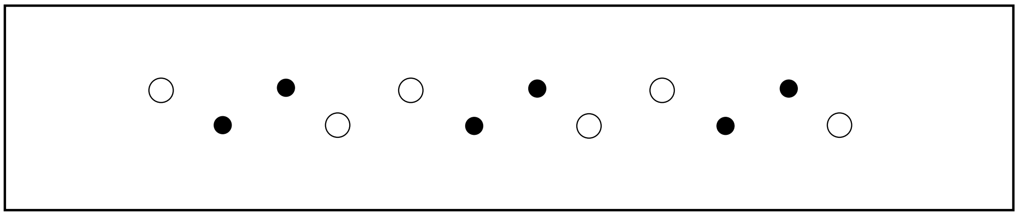


Figure 2.3: *The law of Similarity.*

The law of constant direction or continuity states that a line or a linear arrangement of elements is supposed to change its direction in a continuous fashion. In the dot-based pattern in Figure 2.4-A, the continuous change of direction, which is induced by the position of successive dots from left to right, is schematically illustrated.

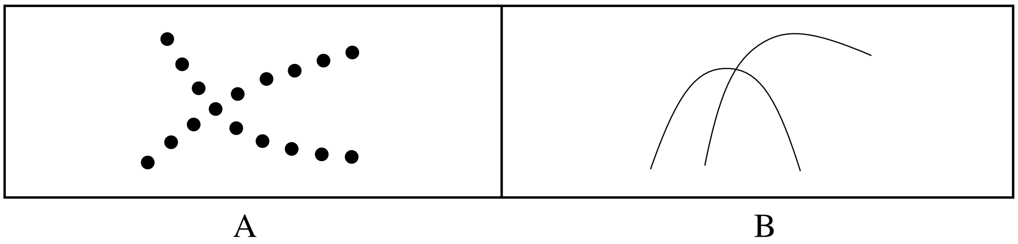


Figure 2.4: *The law of Constant Direction.*

The continuous change of direction between dots from left to right results in two groups of dots each of which forms a curve. One other example of the continuity law is illustrated by the pattern in Figure 2.4-B. In this pattern, theoretically four curves meet at one point. Although there may be four basic curves in this figure, only two specific intersecting curves are perceived. Each of these two intersecting curves consists of two basic curves that follow each other continuously.
The law of closure implies that self-enclosed subpatterns, like closed curves, determine which elements are grouped or related to each other. For example, in Figure 2.5 the three enclosed shapes, shown in 2.5-B, determine the perceptual organization of the pattern shown in 2.5-A.

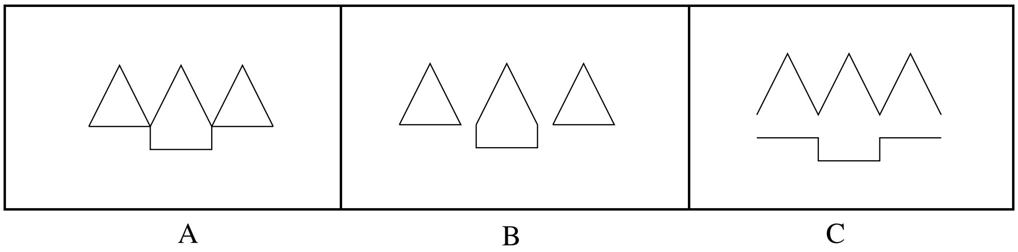


Figure 2.5: *The law of Closure.*

In contrast, the alternative grouping structure, shown in 2.5-C, is not the perceived organization of the pattern shown in 2.5-A.
Finally, the last law is concerned with the influence of habits and past experiences. This law implies that patterns are interpreted as consisting of a certain set of elements which have occurred in the past and thus are well known to the human perceiver. For example, the experience with a written language, like Arabic script, influences the perception of character groups that form the words of that script. For a person who is not familiar with such a script the grouping of characters is not definite such that different perceptual groupings of characters can be perceived.

When we want to analyze a particular pattern, the different gestalt laws may either support each other by predicting the same perceptual organization of a pattern, or they may be in conflict with each other in the sense that different laws predict different perceptual organizations of a pattern. For example, for the pattern shown in Figure 2.3 different gestalt laws such as proximity and similarity predict different perceptual organizations. For this pattern, the law of similarity clearly has a stronger effect than the law of proximity: the similarity-based grouping wins over the proximity-based grouping. One might think that there would be a strength ordering relation among different gestalt laws such that the perceptual organization of patterns can be determined unambiguously by the gestalt laws and their mutual strength ordering. However, such a simple ordering cannot be defined. For example, consider the pattern shown in Figure 2.6.

In this pattern, the similarity law loses its dominance from the proximity law since the spatial distance of non-similar elements is small enough. Although Wertheimer emphasizes the interaction between the various laws, he does not propose a model that may describe it.

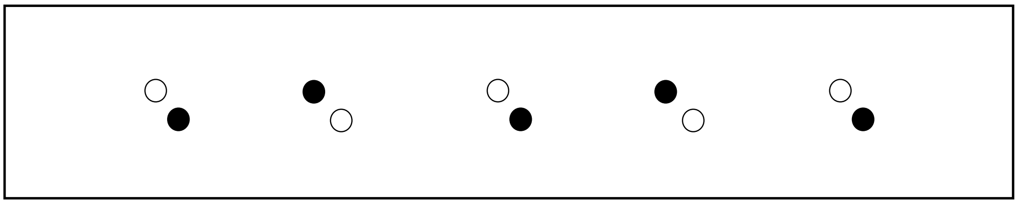


Figure 2.6: *The proximity law, and not the similarity law, dominates the perceptual grouping.*

Koffka [Kof35] tried to account for the interaction between the various gestalt laws. He claimed that the gestalt laws introduced by Wertheimer, do not reflect the basic principles of human perception. Instead, he postulates only one principle *Prägnanz* : *simplicity* or *regularity*. When different prioritizations of the gestalt laws assign different structures to a pattern, the law of Prägnanz predicts that the simplest or the most regular of these structures is the one that is actually perceived.

However, as Van der Helm and Leeuwenberg [VdHL91] have noticed, the formulation of the Prägnanz law neither states what kind of pattern descriptions should be generated, nor does it provide a measure which determines the simplicity or the regularity of pattern descriptions. Leeuwenberg and Van der Helm have attempted to answer these questions by proposing a coding language together with a complexity measure. The coding language is supposed to generate all possible structural descriptions of patterns. A pattern may then have various structural descriptions, each of which reflects a possible organization of that pattern. The complexity measure is then used to select the preferred perceptual organization of a pattern among all its possible organizations.

2.2 Structural Information Theory

The language and the complexity measure proposed by Leeuwenberg and Van der Helm [Lee71, VdHL91] is based on a theory of pattern perception called Structural Information Theory, henceforth SIT. The aim of SIT is to explain why patterns are perceived as they are, i.e. why a pattern is perceived as having a certain structure and not a different one. The explanation is based on the assumption that the human perceptual system is sensitive to certain kinds of structural regularities of patterns. A structural regularity of a pattern is a certain hierarchical arrangement of identical pattern parts. For example, the regularity of the string pattern *abab* may be defined in terms of the identity of the first and the third a's, the second and the fourth b's, and the identity of the first and the second substrings ab's. Note that these identities are in a hierarchical structure such that the identities of the substrings ab's at a higher hierarchical level implies the identities of the a's and the b's at a lower hierarchical level.

According to SIT, only certain kinds of structural regularities are perceptually relevant. These structural regularities, which determine the perceptual organizations or gestalts of patterns, can be specified by means of a set of operators, which are conjectured to reflect innate principles of mental representation. These operators are called ISA operators which stands for *Iteration, Symmetry* and *Alternation* operators.
A pattern can be parsed or encoded into different descriptions by means of ISA operators. The resulting descriptions express different gestalts of the encoded pattern. In order to disambiguate the set of alternative gestalts and to decide on the *preferred* perceptual description 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: the lowest complexity value corresponds to the largest number of perceptually motivated regularities. It is claimed that the description of a pattern with the minimum complexity value describes the pattern in the most simple and cognitively economical way, i.e. the description with the minimum complexity value captures the maximum amount of regularity within the pattern and therefore expresses the preferred perceptual structure of the pattern. The idea that the simplest description of a pattern expresses the preferred perceptual structure of that pattern is called the *simplicity principle.*

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 some description coding theories where any kind of pattern regularity is potentially used to encode patterns in the most compact way without claiming that the resulting codes express any phenomenal simplicity of the encoded patterns. For example, in computer technology compression systems reduce the amount of bits needed to store or memorize binary data. These compression systems employ various kinds of regularities of the binary data, but do not involve a claim that the simpler or the more compressed codes reflect meaningful structures of the encoded data. SIT differs from these kinds of description coding theories in that it is defined on the basis of empirically motivated compression operators (ISA operators) and a specific complexity measure.

The set of ISA operators and the complexity measure, originally proposed in [Lee71], has undergone several revisions until [VdHL91] where Van der Helm and Leeuwenberg put the ISA operators and the complexity measure on a formal and psychological basis. They introduced a criterion called *accessibility* which implies that the regularity and hierarchy induced by ISA operators in the preferred description of patterns correspond directly to the perceived regularity and hierarchy of patterns. The accessibility criterion is based on a formal analysis of regularity and hierarchy and it is used to justify the complexity measure and the perceptual relevance of regularities that are specified by the ISA operators.
During the last 20 years, Leeuwenberg and his co-workers have reported on a number of experiments that tested predictions based on the simplicity principle. These experiments were concerned with the disambiguation of ambiguous patterns. The predictions of the simplicity principle were, on the whole, confirmed by these experiments [Bos88, BW89, BLR81, VLB89, VLBVdV88].

2.2.1 A Coding System for SIT

Although Structural Information Theory is intended as a general theory of pattern perception, the coding system which is introduced to represent patterns and their perceptual structures is in fact designed for one-dimensional sequential patterns such as character strings. For this class of patterns, the coding system specifies a set of coding rules that are motivated by the ISA operators. These coding rules are used to parse one-dimensional patterns and generate their perceptual descriptions. These rules are called Iteration, Symmetry, Right-Alternation and Left-Alternation and they are defined in the following way.

Let X be a symbol sequence and n be an integer. The first coding rule is based on the iteration operator and it is specified as:
$XX \ldots X \rightarrow n \star (X)$,
i.e. a sequence $XX \ldots X$ consisting of n occurrences of a symbol sequence X can be replaced by $n * (X)$.

Let $Y, X_1, \ldots, X_n$ be symbol sequences. The second coding rule is based on the symmetry operator and it is specified as:
$X_1 X_2 \ldots X_n Y X_n \ldots X_2 X_1 \rightarrow S[(X_1)(X_2) \ldots (X_n), (Y)]$,
i.e. a sequence $X_1 X_2 \ldots X_n Y X_n \ldots X_2 X_1$ in which symbol sequences $X_1 X_2 \ldots X_n$ and $X_n \ldots X_2 X_1$ on both sides of the symbol sequence Y are reflections of each other, is replaced by $S[(X_1)(X_2) \ldots (X_n), (Y)]$.

Let $X, Y_1, \ldots, Y_n$ be symbol sequences. The third coding rule is based on the right alternation operator and it is specified as:
$XY_1 XY_2 \ldots XY_n \rightarrow < (X) > / < (Y_1)(Y_2) \ldots (Y_n) >$,
i.e. a sequence in which a symbol sequence X (the left-most symbol sequence) is alternated by the symbol sequences $Y_1, \ldots, Y_n$ is replaced by $< (X) > / < (Y_1)(Y_2) \ldots (Y_n) >$.

Let $X, Y_1, \ldots, Y_n$ be symbol sequences. The fourth coding rule is based on the left alternation operator and it is specified as:
$Y_1 X Y_2 X \ldots Y_n X \rightarrow < (Y_1)(Y_2) \ldots (Y_n) > / < (X) >$,
i.e. a sequence in which a symbol sequence X (the right-most symbol sequence) is alternated by the symbol sequences $Y_1, \ldots, Y_n$ is replaced by $< (Y_1)(Y_2) \ldots (Y_n) > / < (X) >$.

The following examples illustrate the application of these rules to sequential patterns.

Iteration rules:	aaa	$\rightarrow 3 \star (a)$
Symmetry rule:	$abcbca$	$\rightarrow S[(a)(bc), ()]$
Right-Alternation rule:	$abacdaefg$	$\rightarrow < (a) > / < (b)(cd)(efg) >$

In a symmetrical pattern, there may be a pivot element (odd symmetry) or not (even symmetry). These two cases are covered by the symmetry rule, because the second argument of the symmetry operator, which stands for a pivot element, may be an empty or a non-empty element. For example, $S[(a), (b)]$ corresponds with the pattern aba in which b is the pivot element while $S[(a)(b),()]$ corresponds with the pattern *abba* where a pivot element is absent.

The parentheses that occur in the arguments of ISA operators express the constituents, called *units*, of ISA arguments. In fact, the ISA operators are defined in terms of constituents of patterns, which are not necessarily primitive elements. For example, in $S[(a)(bc),()]$ the elements b and c together are considered as one constituent of the first argument of the symmetry operator. Therefore, $S[(a)(bc),()]$ and $S[(a)(b)(c),()]$ correspond respectively to *abcbca* and *abccba* which are two different symmetrical patterns. In the first symmetry (non-mirror symmetry), two primitive elements are considered as one unit while in the second symmetry (mirror symmetry) each primitive element is considered as one unit. In the following, parentheses are not used when units are only primitive elements. So, for example we will write $S[abc,()]$ instead of $S[(a)(b)(c),()]$.

The expressions resulting from applications of ISA rules to patterns are called *ISA forms*. When an ISA coding rule is not applicable to a pattern as a whole, that pattern can be divided into several subpatterns such that coding rules can be applied to the resulted subpatterns. These subpatterns will be called *chunks*. For example, the pattern $abcbcaefef$ may be analyzed as $S[(a)(bc),()]\ 2*(ef)$ which means that the pattern is divided into two chunks, i.e. $S[(a)(bc),()]$ and $2*(ef)$.

Furthermore, the ISA rules can be applied recursively to patterns; the arguments of an ISA operator are themselves patterns. The recursive application of coding rules to patterns results in an embedding structure of patterns. The embedding structure of a pattern is claimed to represent the perceived hierarchical organization of that pattern. For example, a possible chunking and an embedding structure of the pattern *aabbccccbbaa* may be specified in the following way. At the first hierarchical level, the whole pattern can be analyzed by the symmetry rule and thus it can be considered as one chunk:

$aabbccccbbaa \rightarrow S[aabbcc,()]$.

The argument of the symmetry operator (i.e. *aabbcc*) is itself a pattern which can be analyzed further. The structure of this argument determines the chunks at the next lower embedding level. Because the pattern *aabbcc* cannot be analyzed by one ISA operator, it is divided into three chunks. Thus, at the second embedding level the pattern can be analyzed as three chunks, i.e.

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

Consequently, the complete embedded description of the pattern $aabbccccbbaa$ is as follows:

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

The recursive application of ISA rules to patterns encodes pattern regularities and therefore it reduces the number of primitive elements in pattern descriptions. When no ISA rule is applicable to a pattern, i.e. when no further reduction is possible, the resulted pattern description is assumed to possess no regularity anymore. The resulting structured code is called a *final code*. Because of the application of different sequences of ISA rules to a pattern, different final codes for that pattern may result. These different final codes express different gestalts of the encoded pattern. For example, $abS[ab, ()]$ and $2*(ab)ba$ express different gestalts of the pattern $ababba$.

In order to decide on the preferred gestalts of patterns, a complexity measure, called *information load* (I), is introduced. Information load is defined on final codes of patterns and it measures the complexity of final codes by computing and adding two quantities. The first quantity is the number of occurrences of primitive elements in the final code, and the second quantity is the number of occurrences of units that contain more than one primitive element in the final code. For example, the pattern $abcddcab$ may be described as $S[(ab)(c)(d), ()]$; this final code contains 4 primitive elements a, b, c and d and there is one unit (ab) that contains more than one element. Therefore, the information load of the description $S[(ab)(c)(d), ()]$ is $4+1=5$. Similarly, the pattern $abcdcdab$ may be described as $S[(ab)(cd), ()]$ which has four primitive elements and two units that contain more than one element: (ab) and (cd). The information load of this description is 6. Note that the united elements need not be primitive elements. For example, the pattern $fabccabeabccabef$ may be analyzed as
$S[(f)(S[(ab)(c), ()](e)), ()]$
in which the non-primitive element $S[(ab)(c), ()]$ forms one unit together with (e). In this description, there are five primitive elements and two

units that contain more than one element: (ab) and $(S[(ab)(c),()](e))$. The information load for this description is then 7. The detailed definition and the calculation of this complexity measure is explained in [VdH94].

The information load provides a measure for ordering the final codes of a pattern. A final code of a pattern which has the lowest information load is called a *minimum code* of that pattern. If there exists a unique minimum code for a pattern, the minimum code is considered as the preferred perceptual description of that pattern. For example, $aS[(b),(a)]$ and $2*(ab)$, which are two possible final codes of the pattern *abab*, have information loads 3 and 2, respectively. Therefore, the final code $2*(ab)$ is preferred over the final code $aS[(b),(a)]$.

A pattern may lead to different final codes that have the same information load; i.e. there may be more than one minimum code for a pattern. In that case, there is no unique preferred description that can be assigned to such a pattern. A pattern that has different final codes with the same information load is an ambiguous pattern: the pattern has no definite perceptual structure. The following example illustrates an ambiguous pattern:

$$\begin{array}{lll} ababba & \leftrightarrow & 2*(ab)ba \quad \text{with } I=4 \\ ababba & \leftrightarrow & abS[ab,()] \quad \text{with } I=4 \end{array}$$

In summary, the explicit claim of SIT is that the preferred description of a pattern is the simplest. In this way, Structural Information Theory formalizes one aspect of the Prägnanz notion of gestalt psychology, viz. descriptive economy; *the information load of what is actually perceived is lower than the information load of what could have been perceived alternatively.*

2.2.2 A Partial Computational Model of SIT

One problem with Structural Information Theory has been the claim that SIT is computational intractable. Algorithms for calculating the simplest structural descriptions were thought to involve a combinatorial explosion. However, Van der Helm and Leeuwenberg [VdHL86] present a polynomial algorithm which computes the preferred perceptual description of sequential patterns.

Starting with a sequential pattern, the first step in this algorithm is the

construction of a directed acyclic graph for that pattern. The nodes in this graph are the positions between elements in the pattern. Two special nodes, the Start-node and the Final-node, represent the beginning and the end of the pattern, respectively. A link in the graph that points from node A to node B corresponds to a contiguous subpattern starting at position A and ending at position B of the pattern. Note that such a constructed graph for a sequential pattern represents all possible subpatterns of that pattern.
Subsequently, for each subpattern which corresponds to a link of the graph, the best way to describe that subpattern by just one ISA operator is recursively determined. This results a new graph in which each link is labeled with an ISA code describing the preferred organization of the subpattern associated with that link. Moreover, each link has a *length* which is equal to the information load of its corresponding ISA code.
For example, the acyclic graph illustrated in Figure 2.7-A represents all possible subpatterns of string pattern *abab*. The determination of ISA codes for each subpattern results in the graph illustrated in Figure 2.7-B.

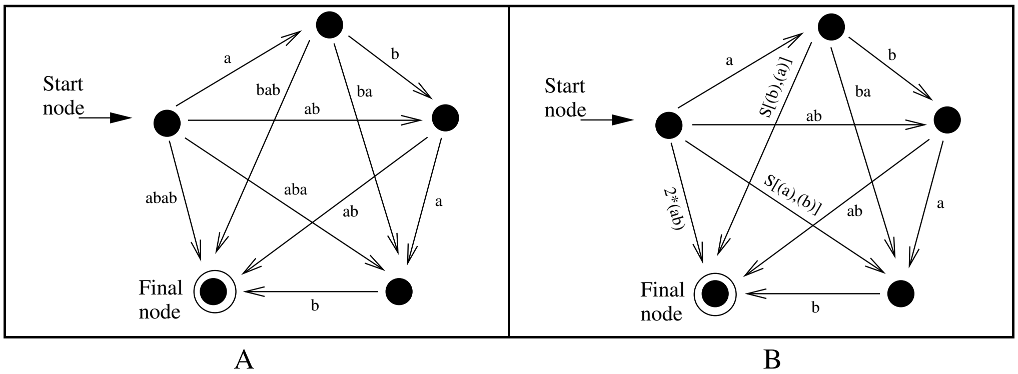
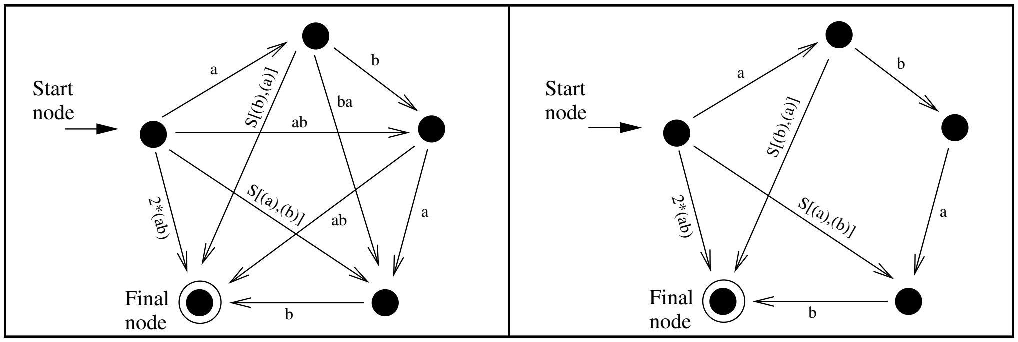


Figure 2.7: *The graph represents all possible subpatterns of pattern abab (A). The graph in which all subpatterns are described by one ISA code (B).*

In the graph, several paths start at the Start-node and end at the Final-node. These paths represent all possible final codes, i.e. different perceptual chunkings of the original pattern. Each path has a length which is the sum of the lengths of the links that constitute that path.

The second step in this algorithm simplifies the graph. This is done by removing the links that represent a subpattern with a length longer than 1 that cannot be analyzed by the ISA rules, i.e. the links that contain

more than one element and are not covered by one ISA code. Those links can be disregarded because equivalent paths with shorter or equal length remain. Notice that links corresponding to only one element (length 1) are not removed such that reduced graphs contain always paths which have the maximum complexities.
The reduced and the original graphs of a pattern are then equivalent with respect to the set of alternative organizations that can be generated for that pattern, i.e. all structurally non-identical paths of the original graph (non-identical organizations of the pattern) will occur in the reduced graph as well. For example, the original and the reduced graphs corresponding to pattern *abab* are given in Figure 2.8-A and Figure 2.8-B, respectively. Each path starting from the Start-node and ending in the Final-node corresponds to a different final code.



A B

Figure 2.8: *The graph representation for pattern abab (A) and its reduced version (B).*

The problem of calculating the minimum code can now be replaced by calculating the shortest path in the graph, using the shortest path algorithm introduced by Dijkstra [Dij59]. Notice that a shorter path does not mean a path with a smaller number of links. For example, consider the pattern $abaccefe$. Among others, there are two paths representing two different organizations of that pattern as they are shown in Figure 2.9.

The path with three links has the shortest length, equal to 5, while the path with two links has a length equal to 6. Therefore the path with three links indicates the preferred organization of the pattern.

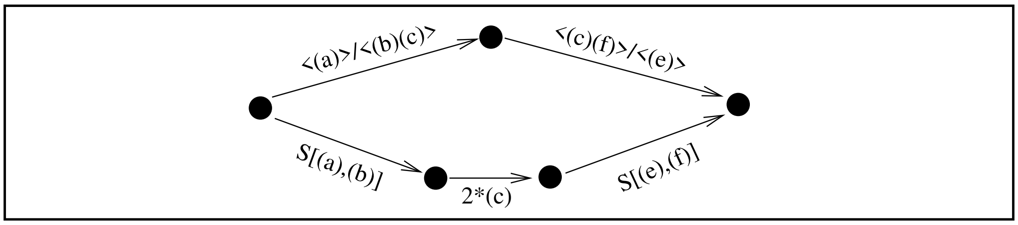


Figure 2.9: *The shortest path does not have the lowest number of links.*

A graph corresponding to a final code specifies a chunking at the highest organizational level. Therefore, the approach envisages a recursive structure between graphs. For example, the preferable path corresponding to *aabbccaabbcc* will be one single link representing one chunk, i.e. $2*(aabbcc)$ while the preferable path corresponding to *aabbcc* will be a path containing three links representing three chunks $2*(a)$, $2*(b)$ and $2*(c)$, respectively. The recursive structure of graphs is thought to represent all possible hierarchical organizations of the represented patterns.

2.3 Limitations of SIT

A fundamental assumption of SIT is what may be called the *encoding hypothesis*. According to this hypothesis, perceptual patterns can be encoded as one-dimensional string patterns such that the regularities of the encoding string patterns reflect the perceptual regularities of their corresponding encoded patterns. The perceptual structure of patterns is thus assumed to be invariant under the encoding process. The encoding string patterns are called *primitive codes*. The perceptual regularities of these encoding string patterns are determined as we explained in the previous section.

SIT is presented as a general theory of pattern perception and it is applied to various domains of perceptual patterns [Lee71]. In each of these domains the perceptual patterns are encoded as sequences of discrete symbols. For example, monophonic music consisting of an uninterrupted sequence of tones of equal durations is encoded as a sequence of numbers representing pitches; an uninterrupted two-dimensional line pattern consisting of connected line segments is encoded as a sequence of letters representing lengths of line segments and the angles between them. When line segments are interrupted and thus are not connected to each other, an "invisibility operator" is applied to generate invisible line segments between them. For

three-dimensional line patterns the angles are enriched with three dimensional directions.
However, SIT is mainly applied to uninterrupted two-dimensional line patterns consisting of straight-line segments (turtle graphics). A primitive code for such a visual line pattern is constructed by choosing an appropriate starting point and tracing the line segments and concatenating symbols that are assigned to the successive lengths and their relative angles. An example of such an encoding is illustrated in Figure 2.10.

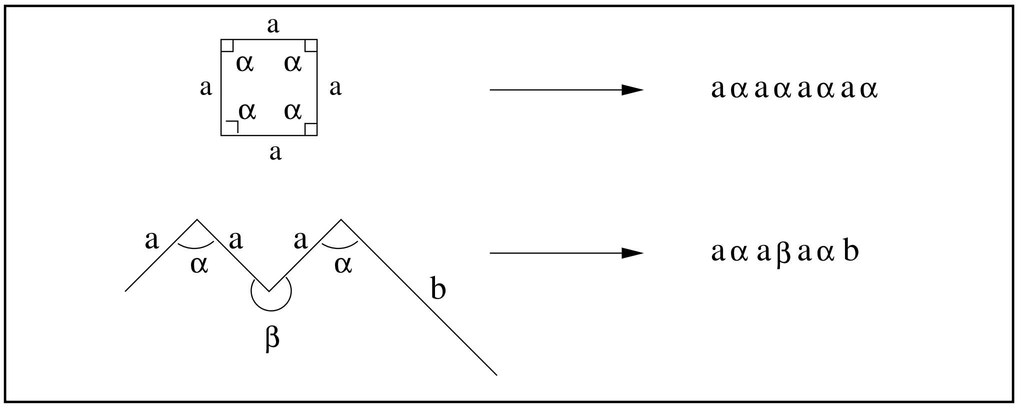


Figure 2.10: *Encoding of a line pattern into a string pattern.*

We have our doubts about the encoding hypotheses of SIT. In Structural Information Theory and in its first computational model [VdHL86], where the ISA reduction process is partially implemented, the subjects of analysis are primitive codes (string patterns) instead of visual line patterns. However, the step needed to encode an arbitrary visual line pattern into a primitive code is not a trivial step. In fact, given an arbitrary visual line pattern, several primitive codes may be possible. Each of these primitive codes may result in a different set of gestalts. For example, a visual line pattern in which lines cross each other can be considered as a graph such that each possible path in that graph represents a possible primitive code. In Figure 2.11, tracing line segments along different paths results in two different primitive codes.
It is important to note that different primitive codes of a visual line pattern do not necessarily provide the same set of gestalts of that pattern. For this reason, a primitive code of a visual line pattern may not provide the actual perceived gestalt of the pattern, i.e. the minimum code computed from a primitive code of a visual line pattern represents a gestalt which is

not the preferred gestalt of that pattern. This implies that a wrong choice of primitive code results in a wrong prediction of preferred gestalt. In order to illustrate this phenomenon, consider again the two primitive codes in Figures 2.11-A and 2.11-B. The primitive code in Figure 2.11-A does not provide the preferred gestalt that describes the visual line pattern as consisting of one straight-line and one triangle, while the second primitive code in Figure 2.11-B does provide such a gestalt.

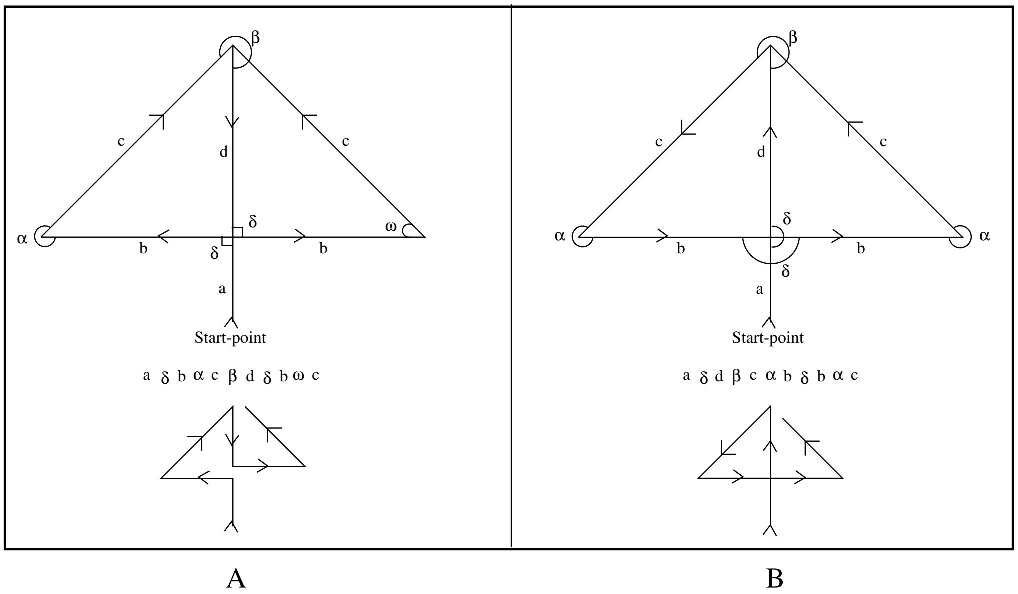


Figure 2.11: *Two possible encodings of one line pattern.*

Thus, the process of finding an appropriate primitive code for an arbitrary visual line pattern is not a trivial process. The fact that this process is done by hand in the Leeuwenberg tradition and is not included in the algorithm is an essential limitation.

One other related problem concerns the definition and the computation of perceptual regularities. Perceptual regularities are defined on and applied to primitive codes, which are one-dimensional string patterns. For example, the alternation regularity is defined as different occurrences of a substring that are alternated by other substrings within a *sequential order*. It is however by no means clear how this theory can be applied to visual patterns that consist of unconnected visual elements utilizing all kinds of visual attributes.

Because of these two problems (i.e. the problem that patterns have no unique primitive code and the problem that regularities are only for one-dimensional patterns), the domain of visual patterns that can be properly analyzed by SIT is limited to those line patterns for which only one primitive code is plausible. These restrictions result in a class of line patterns which we will call *linear line patterns*. Linear line patterns are turtle line drawings for which the turtle starts somewhere and moves in such a way that the line segments are connected and do not cross each other. In Figure 2.12, some examples of such patterns are shown.

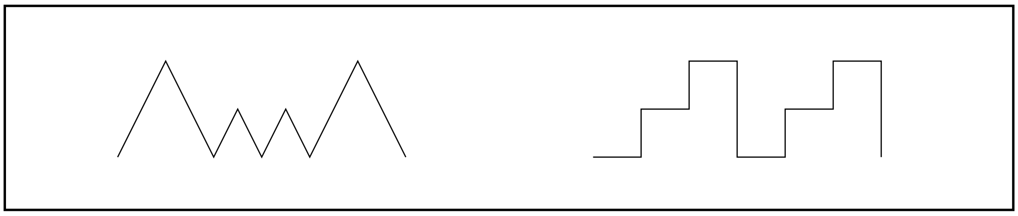


Figure 2.12: *Examples of linear line patterns.*

Finally, a serious shortcoming of SIT concerns the limited range of perceptual structures that it covers. In particular, there are two important sorts of perceptual structures that are not covered by SIT. The first sort of structures can be characterized as those that are based on regularity of different visual or spatial attributes of visual elements, and the second sort of structures are those that constitute the proximity gestalts.

In SIT, the perceptual grouping of pattern elements is determined by the ISA reduction rules that are defined in terms of the equality of pattern elements. However, in SIT the equality of visual elements is defined in terms of equality of all attribute values except the position value: two visual elements are equal if and only if they are equal in all attribute values except the position value. Thus, two visual elements that differ from each other according to an attribute value, other than the position value, cannot be considered as equal and therefore they cannot form one perceptual group. It may be clear that this is a serious limitation since visual elements in Figure 2.13 with different shapes but equal texture values form definitely one perceptual group. Therefore, a proper coding system should allow perceptual grouping based on different attributes.

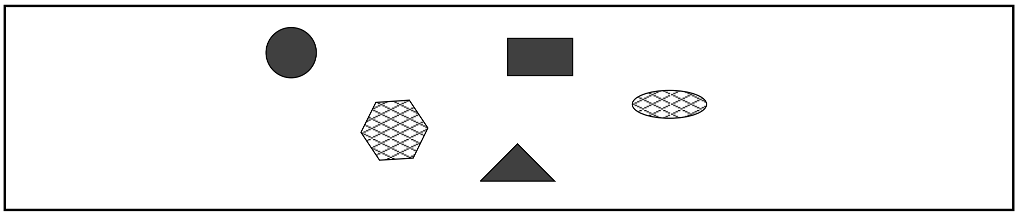


Figure 2.13: *Visual elements with different shape values but equal texture value form one group.*

Moreover, SIT covers only structural regularities like symmetry and iteration in terms of which some gestalt phenomena like good continuation, foreground/background, etc. can be explained. However, the influence of proximity is not covered. The proximity-based gestalt of a pattern is a perceptual grouping of its constitutive pattern elements which is determined based on *relative distances* between those pattern elements: relatively close elements tend to be perceptually grouped. Thus, according to the proximity-based gestalt the neighboring elements can be grouped such that the local organizations will be detected before the global ones are constructed. For example, consider the pattern shown in Figure 2.14.

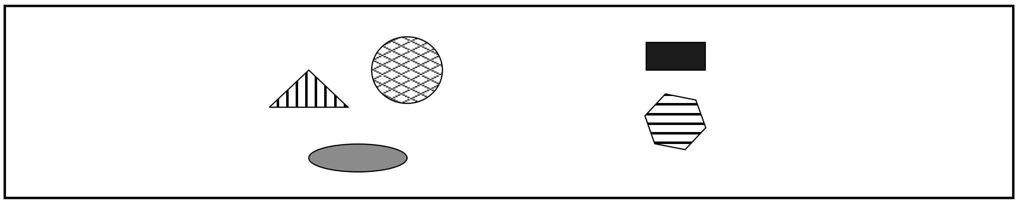


Figure 2.14: *Relatively close visual elements form one group.*

This pattern consists of a number of visual elements that do not constitute any structural regularity by means of any attribute. Although the involved visual elements do not constitute any structural regularity, the pattern is not perceived as having a random structure, but it is perceived as having a certain grouping structure.

Proximity-based gestalts differ from structural gestalts since they cannot be selected by means of reduction of information complexity. In other words, proximity-based gestalts do not reduce information needed to describe patterns. Therefore, the information complexity resulting from the proximity-based gestalt of a pattern may be equal to the information com-

plexity resulting from a structural gestalt which describes that pattern as having a random grouping structure. In fact, none of these two gestalts reduce the amount of information needed to describe that pattern.
Thus, it may be the case that the proximity-based gestalt of a pattern wins from its preferred structural gestalt while the first gestalt has a higher information complexity than the second gestalt. In order to illustrate this point, consider the pattern in Figure 2.15-A. This pattern shows a situation in which the proximity-based analysis (3 pairs of circles and two single circles, Figure 2.15-B) wins from the structural-based analysis (4 pairs of circles, Figure 2.15-C) while the first analysis results a higher complexity value (complexity = 2) than the second analysis does (complexity = 1). Therefore, the preferred gestalt of patterns, among all possible structural- and proximity-based gestalts, cannot be determined by measuring only the information complexity.

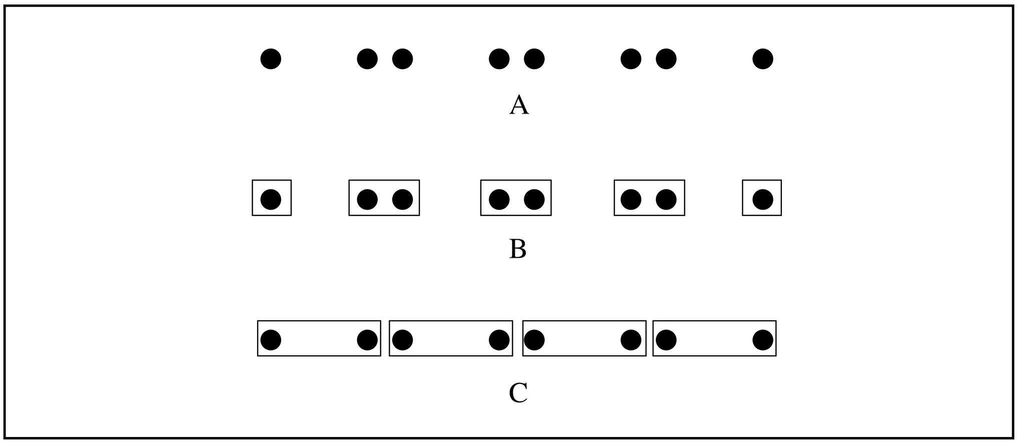


Figure 2.15: *The proximity (B) and the structural (A) analysis of the pattern (A).*

One may think that the structural regularities must *either* occur within one proximity group *or* between two or more proximity groups. For example, consider the patterns in Figure 2.16. In these patterns, the structural regularities are either within a proximity group or between two or more proximity groups.

Unfortunately, this heuristic is not quite valid since there are patterns which tend to be perceived as having structural gestalts constituted by *proper subsets* of elements from different proximity groups. However, the perceptual

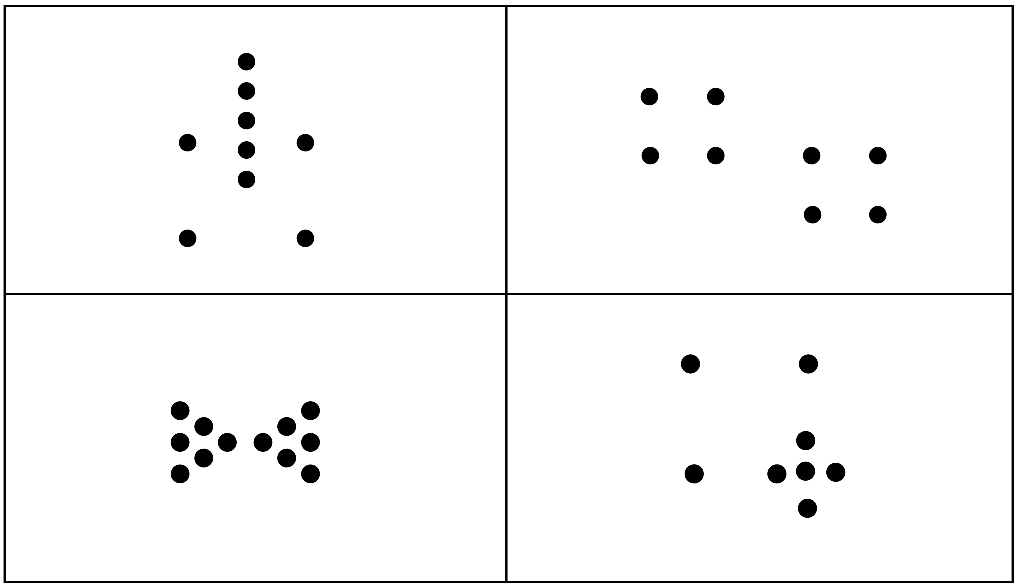


Figure 2.16: *The interaction between proximity and structural analyses.*

preference of these structural gestalts depends on the relative number of elements from the involved proximity groups that constitute these structural gestalts. In fact, if the number of elements from proximity groups that constitute a structural regularity becomes lower than the number of elements (from the same proximity groups) that do not constitute that structural regularity, then the structural regularity loses its perceptual preference. This phenomena is sometimes called the *goodness* of structural regularities or the goodness of gestalts. For example, consider the patterns illustrated in Figure 2.17.

The patterns in Figures 2.17-A and 2.17-C have symmetry and iteration regularities, respectively. However, although the patterns illustrated in Figure 2.17-B and 2.17-D can be perceived as having similar structural regularities as those illustrated in 2.17-A and 2.17-C respectively, the perception of their regularities are not as good as in 2.17-A and 2.17-C.
Notice that in these patterns the proximity effect is responsible for the goodness of structural regularities. For example, consider the influence of the proximity effect in the pattern 2.17-D. The perception of the two groups of four dots, which have the form of two rectangles say X and Y, is disturbed by the proximity effect that is caused by the two closely positioned dots. In order to perceive either X or Y, one need to perceive only one of these

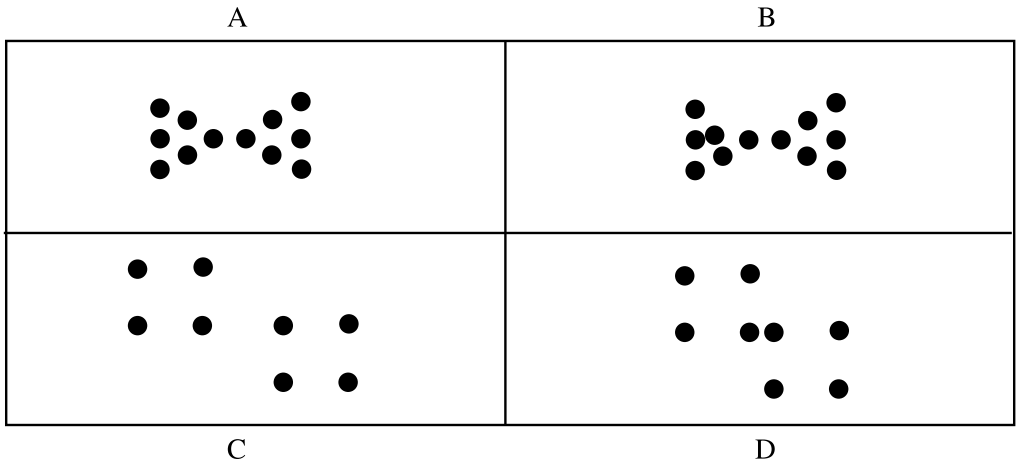


Figure 2.17: *The proximity factor effects the goodness of patterns.*

two dots. However, to perceive X or Y, the proximity effect enforces the perception of the second dot as well. This means that X or Y must be perceived from a group of five dots such that the goodness of their perceptions is decreased. We will not focus on the goodness of structural regularities but we note that any disturbance in regularities oppresses the goodness of perception of those regularities.

Although Structural Information Theory neither discusses the proximity gestalt nor claims to cover it, one may suggest that the proximity gestalt, like structural gestalts, can be explained by means of the minimum principle. According to this suggestion, proximity gestalts are pattern descriptions which describe patterns in terms of groups of pattern elements rather than in terms of individual pattern elements. In this way, it is assumed that human perceiver does not perceive individual elements of patterns but only groups of them. The information needed to describe a pattern is then the information of groups of elements of that pattern rather than the information of its individual elements. Since there are always fewer groups of elements than individual elements in a pattern, proximity gestalts can be considered as pattern descriptions which reduce the amount of information needed to describe patterns. However, this suggestion implies some abstraction of pattern descriptions to which the minimum principle should be applied. Without proposing an abstraction of pattern descriptions, we will introduce in chapter 4 a method which accounts for the interaction of structural and proximity gestalts.

In the next chapter, various coding systems are presented. First, we introduce coding systems that are designed to represent one-dimensional string patterns and their possible gestalts. Then, we introduce coding systems that are designed to represent two-dimensional visual patterns and their possible gestalts. The gestalts of two-dimensional visual patterns are defined in terms of visual and spatial attributes. In this coding system, there is no need to translate visual patterns first into one-dimensional string patterns and then compute their gestalts.

Chapter 3

Coding Languages for Gestalts of Patterns

In the previous chapter, we discussed the Structural Information Theory (SIT) which is a general predictive theory of pattern perception, i.e. a theory that predicts the preferred gestalts of perceptual patterns. We argued that the proposed coding system of SIT [VdHL91], which was designed to represent possible gestalts of perceptual patterns, can only represent the gestalts of string patterns and the gestalts of a small subclass of line patterns. In particular, we showed that the proposed coding system is not powerful enough to represent the gestalts of important classes of perceptual patterns like the class of two-dimensional visual patterns.

In this chapter, we introduce various algebraic coding languages. Each coding language consists of algebraic expressions that represent possible gestalts of a certain class of perceptual patterns. In order to specify the coding language for a certain class of perceptual patterns, we consider the class of perceptual patterns as the universe of an algebra for which the set of operators are motivated by the structural operators of SIT (i.e. ISA operators). Then, the set of algebraic terms (expressions) that are generated by such an algebra defines the coding language for the given class of perceptual patterns.

In section 1, we specify the coding language for one-dimensional string patterns. This coding language is an algebraic reformulation of the original version of the coding system proposed by Van der Helm and Leeuwenberg [VdHL91]. This coding language is specified by considering the class of one-dimensional string patterns as the universe of an algebra. The gestalts of these patterns are then defined as the structure of algebraic terms that represent the patterns.

In section 2, we generalize the notion of gestalt. This generalization is based on the assumption that the acquaintance and accessibility of certain operators, defined on patterns from a certain domain, influence the gestalts of those patterns. We elaborate on one-dimensional string patterns and illustrate the influence of string operators such as "successor" and "predecessor" on their gestalts.

In section 3, two-dimensional visual patterns and their gestalts are discussed. Two-dimensional visual patterns are defined in terms of values of visual attributes such as position, color, size, shape, orientation, texture, etc. The gestalts of two-dimensional visual patterns are then defined in terms of structures of their visual attribute values.

In section 4, this view of two-dimensional visual patterns and their gestalts is formalized by specifying an algebraic coding language for two-dimensional visual patterns. This coding language consists of algebraic expressions that represent possible gestalts of two-dimensional visual patterns. These expressions are the terms of an algebra for which the universe consists of n-tuples of visual attribute values, i.e. the universe consists of representations of two-dimensional visual patterns. We then show that this coding language does not cover some gestalts of two-dimensional visual patterns. In order to cover a broader range of gestalts of two-dimensional visual patterns, the coding language will be extended.

3.1 A Coding Language for String Patterns

In this section, an algebraic coding language for one-dimensional string patterns is introduced. This coding language consists of algebraic expressions that represent gestalts of one-dimensional string patterns. In order to specify the coding language, we introduce an algebra $<\mathcal{U}, \mathcal{F}>$, henceforth String-algebra. The universe of objects for the String-algebra is $\mathcal{U} = \mathcal{D} \cup \mathcal{N}$, where $\mathcal{D}$ is the domain of one-dimensional string patterns and $\mathcal{N}$ is the set of natural numbers; $\mathcal{F}$ is the set of ISA operators. The domain $\mathcal{D}$ of one-dimensional string patterns is defined as follows:

1. DEFINITION. *The domain $\mathcal{D}$ of one-dimensional string patterns is the set of all finite strings generated by the grammar $G = (V, T, P)$ where:*
$V = \{S, A\}$ is the set of non-terminal symbols,
$T = \{a, \ldots, z\}$ is the set of terminal symbols,
S is the start symbol, and
P is the following set of rewrite rules:

$$\{\ S \rightarrow A,$$
$$A \rightarrow AA,$$
$$A \rightarrow a \mid \ldots \mid z\ \}.$$

In order to represent the gestalts of patterns, the original coding system of SIT introduces four operators called Iteration, Symmetry, Right-alternation and Left-alternation. The original coding system of SIT does not contain an explicit operator to express pattern concatenation: the concatenation of patterns is expressed by the concatenation of their representations. In the algebraic version of the coding system of SIT all structures will be expressed explicitly by an operator. Therefore, an additional operator, called *Concatenation* operator (Con), is included in the algebraic version of the coding system.

Moreover, in the original coding system of SIT only one symmetry operator is introduced. This symmetry operator has two arguments: the first argument stands for the symmetrical part and the second argument stands for a possible pivot element. The symmetry operator generates symmetrical patterns with or without a pivot element by assuming an empty element: when the second argument of the symmetry operator is an empty element, the generated symmetrical pattern does not contain a pivot element, otherwise it does. Alternatively, for the algebraic version of the coding system we do not assume an empty element since an empty element has no perceptual relevance. Instead, we introduce two distinct symmetry operators, called *Even-symmetry* (Sym_e) and *Odd-symmetry* (Sym_o): The first generates symmetrical patterns that do not contain a pivot element and the second generates symmetrical patterns that do contain a pivot element.

2. DEFINITION. *Let $X = x_1 \cdots x_k$ be a string pattern, with $x_i \in \{a, \ldots, z\}$; $Y_1, \ldots, Y_m$ belong to the domain $\mathcal{D}$ of string patterns; and $n \in \mathcal{N}$. The set $\mathcal{F}$ of ISA operators is defined as follows:*

$$
\begin{array}{lcl}
Iter(X, n) & \rightarrow & X \cdots X \quad (n \textit{ times } X), \\
Sym_e(X) & \rightarrow & x_1 \cdots x_k x_k \cdots x_1, \\
Sym_o(X, Y_i) & \rightarrow & x_1 \cdots x_k Y_i x_k \cdots x_1, \\
Alt_r(X, < Y_1, \ldots, Y_m >) & \rightarrow & Y_1 X Y_2 X \cdots Y_m X, \\
Alt_l(X, < Y_1, \ldots, Y_m >) & \rightarrow & X Y_1 X Y_2 \cdots X Y_m, \\
Con(Y_1, \ldots, Y_m) & \rightarrow & Y_1 \cdots Y_m.
\end{array}
$$

The following derivations illustrate some algebraic expressions (gestalts) and the string patterns that are represented by them.

$Iter(Con(a, b, c), 2) \rightarrow$
$Iter(abc, 2) \rightarrow$
$abcabc.$

$Sym_e(Con(Iter(a, 2), b)) \rightarrow$
$Sym_e(Con(aa, b)) \rightarrow$
$Sym_e(aab) \rightarrow$
$aabbaa.$

$Alt_r(Sym_o(a, b), < p, s, w >) \rightarrow$
$Alt_r(aba, < p, s, w >) \rightarrow$
$abapabasabaw.$

The above set of ISA operators is not powerful enough to generate all gestalts that have been covered by the original coding system of SIT. This limitation is related to the symmetry operator. In the original coding system the symmetry operator is not only defined on primitive elements $\{a, \ldots, z\}$, but it is also defined on groups of elements considered as *units*. The symmetry operator is then supposed to preserve the sequential order of the united elements. For example, the pattern *abccab* is a symmetrical pattern in which elements a and b are considered as one unit. In order to express this kind of symmetry structures the original coding system of SIT employs parentheses to indicate the united elements. Thus, in the original coding system of SIT the structure of the pattern *abccab* is expressed as $S[(ab)(c), ()]$.
In order to extend the algebraic version of the coding system and cover

unit-based symmetry structures, one may define the symmetry operators not only on string patterns from $\mathcal{D}$, which are constituted by merely primitive elements, but also on some (pre-structured) string patterns in which the units of elements are explicitly expressed. Therefore, the domain $\mathcal{D}$ of one-dimensional string patterns should be extended with the set of unit-structured one-dimensional string patterns. This extended domain can simply be specified by introducing parentheses as additional terminal symbols in the above grammar G and adding an extra production rule that generates units of elements. This rule can be defined as follows:

$A \rightarrow (A)$

As a consequence of including units, the above set of ISA operators should be extended. In fact, the ISA operators should be defined on either primitive elements or united elements. Note in the above definition that the ISA operators were defined on string patterns that are constituted by terminal symbols $\{a, \ldots, z\}$. In order to redefine the ISA operators we first define the notion of *string objects* as encompassing the terminal symbols $\{a, \ldots, z\}$, and the units of elements.

3. DEFINITION. *Let $\mathcal{D}$ be the set of one-dimensional string patterns specified by the (extended) grammar G. The set $\mathcal{D}'$ of string objects is a subset of $\mathcal{D}$ defined as follows:*
1) If $x \in \{a, \ldots, z\}$, then $x \in \mathcal{D}'$,
2) If $x_1, \ldots, x_n \in \mathcal{D}'$, then $(x_1 \cdots x_n) \in \mathcal{D}'$.

The ISA operators should then be defined on string patterns that consist of string objects. Thus, in the above definition of ISA operators the element x_i $(1 \leq x_i \leq k)$ in the string pattern $X = x_1 \cdots x_k$ is a string object (i.e. $x_i \in \mathcal{D}'$) rather than a primitive element (i.e. $x_i \in \{a, \ldots, z\}$).

Finally, in order to express the unit structure of patterns explicitly, a new operator, called *Unit* operator, is added to the set of algebraic operators $\mathcal{F}$. Since the unit structure of string patterns is already expressed by the parentheses, the Unit operator may seem to be superfluous. However, the introduction of the Unit operator has the advantage that all pattern structures are expressed by explicit operators. Given the string objects $x_1, \ldots, x_m \in \mathcal{D}'$, the unit operator may be defined as follows:

$Unit(x_1 \cdots x_m) \rightarrow (x_1 \cdots x_m)$

Based on this version of the String-algebra, the following two derivations illustrate two algebraic expressions (gestalts) and the string patterns they represent.

$Sym_e(Con(Unit(Con(a,b)),c,d)) \rightarrow$
$Sym_e(Con(Unit(ab),c,d)) \rightarrow$
$Sym_e(Con((ab),c,d)) \rightarrow$
$Sym_e((ab)cd) \rightarrow$
$(ab)cddc(ab)$.

$Sym_e(Con(d,Unit(Con(Sym_e(Con(unit(Con(a,b)),c)),e)))) \rightarrow$
$Sym_e(Con(d,Unit(Con(Sym_e(Con(unit(ab),c)),e)))) \rightarrow$
$Sym_e(Con(d,Unit(Con(Sym_e(Con((ab),c)),e)))) \rightarrow$
$Sym_e(Con(d,Unit(Con(Sym_e((ab)c),e)))) \rightarrow$
$Sym_e(Con(d,Unit(Con((ab)cc(ab),e)))) \rightarrow$
$Sym_e(Con(d,Unit((ab)cc(ab)e))) \rightarrow$
$Sym_e(Con(d,((ab)cc(ab)e))) \rightarrow$
$Sym_e(d((ab)cc(ab)e)) \rightarrow$
$d((ab)cc(ab)e)((ab)cc(ab)e)d$.

A structural description of a pattern, or one of its gestalts, shows how the pattern is built out of other patterns. This corresponds to what is called a *term* of an algebra.

4. DEFINITION. *The class of structural descriptions over the String-algebra* $<\mathcal{U},\mathcal{F}>$, *denoted by* $\mathcal{G}_{<\mathcal{U},\mathcal{F}>}$, *where* $\mathcal{U}=\mathcal{D}\cup\mathcal{N}$, *can be recursively defined as follows:*

1) For all $t\in\mathcal{D}$, $t\in\mathcal{G}_{<\mathcal{U},\mathcal{F}>}$, *and*
2) If $f\in\mathcal{F}(n)$ *and* $t_1,\ldots,t_n\in\mathcal{G}_{<\mathcal{U},\mathcal{F}>}$, *then* $f[t_1,\ldots,t_n]\in\mathcal{G}_{<\mathcal{U},\mathcal{F}>}$.

In order to allow abstract descriptions of classes of patterns, rather than descriptions of individual patterns, one may introduce variables standing for terms. In this way, gestalts are considered as templates, i.e. structures that different individual patterns have in common. To do this, we extend the universe $\mathcal{U}$ of the String-algebra with a denumerably infinite set of variables $\mathcal{X}$, i.e. $\mathcal{U}=\mathcal{D}\cup\mathcal{N}\cup\mathcal{X}$. The above recursive definition of algebraic terms $\mathcal{G}_{<\mathcal{U},\mathcal{F}>}$ is then extended by the following clause:

- If $x \in \mathcal{X}$, then $x \in \mathcal{G}_{<\mathcal{U},\mathcal{F}>}$.

The terms of the String-algebra are either *primitive* or *complex*.

5. DEFINITION. *A primitive term is a single element from the set* $\{a, \ldots, z\} \cup \mathcal{X}$. *An ISA operator applied to a finite number of terms builds up a complex term.*

3.1.1 $\mathcal{SL}$: A Language for Gestalts of String Patterns

Based on the definition of String-algebra $< \mathcal{U}, \mathcal{F} >$ and the definition of algebraic terms, we define the algebraic coding language for the one-dimensional string patterns, henceforth *String-language* ($\mathcal{SL}$), as the class of all terms of the String-algebra, i.e. $\mathcal{SL} = \mathcal{G}_{<\mathcal{U},\mathcal{F}>}$.

6. DEFINITION. *The String-langugae $\mathcal{SL}$, specified by the String-algebra $< \mathcal{U}, \mathcal{F} >$, where $\mathcal{U} = \mathcal{D} \cup \mathcal{N} \cup \mathcal{X}$ is thus recursively defined as follows:*

- *For all $t \in \mathcal{D} \cup \mathcal{X}$, $t \in \mathcal{SL}$*
- *If $t \in \mathcal{SL}$ and $n \in \mathcal{N}$, then $Iter[t, n] \in \mathcal{SL}$*
- *If $t \in \mathcal{SL}$, then $Sym_e[t] \in \mathcal{SL}$*
- *If $t_1, t_2 \in \mathcal{SL}$, then $Sym_o[t_1, t_2] \in \mathcal{SL}$,*
- *If $t, t_1, \ldots, t_n \in \mathcal{SL}$, then*
  $Alt_r[t, < t_1, \ldots, t_n >] \in \mathcal{SL}$, *and*
  $Alt_l[t, < t_1, \ldots, t_n >] \in \mathcal{SL}$,
- *If $t_1, \ldots, t_n \in \mathcal{SL}$, then $Con[t_1, \ldots, t_n] \in \mathcal{SL}$, and*
- *If $t_1, \ldots, t_n \in \mathcal{SL}$, then $Unit[t_1 \cdots t_n] \in \mathcal{SL}$.*

The $\mathcal{SL}$ expressions refer to elements from the set of string patterns $\mathcal{D}$. Since $\mathcal{SL}$ expressions may contain variables and because variables stand for elements from $\mathcal{D}$, we define an assignment function $g : \mathcal{X} \rightarrow \mathcal{D}$ to assign elements from $\mathcal{D}$ to variables from $\mathcal{X}$. Now, the interpretation function (I) for the $\mathcal{SL}$ expressions can be defined.

7. DEFINITION. *Let* $< \mathcal{U}$; $\mathcal{F} >$ *be the String-algebra where* $\mathcal{U} = \mathcal{D} \cup \mathcal{N} \cup \mathcal{X}$, $\mathcal{D}' \subseteq \mathcal{D}$ *be the set of string objects, and* g *be an assignment function as defined above. The denotation of the* $\mathcal{SL}$ *expressions can then be defined by the interpretation function* I *in a bottom-up fashion as follows:*

- *if* $t \in \mathcal{D}$, *then* $I(t) = t$,
- *if* $t \in \mathcal{X}$, *then* $I(t) = g(t)$,
- *if* t *is of the form* $Iter[X, n]$ *where* $X \in \mathcal{D}$,
  then $I(t) = X \cdots X \qquad (n \text{ times } X)$,
- *if* t *is of the form* $Sym_e[X_1 \cdots X_n]$ *where* $X_i \in \mathcal{D}'$,
  then $I(t) = X_1 \cdots X_n X_n \cdots X_1$,
- *if* t *is of the form* $Sym_0[X_1 \cdots X_n, Y]$ *where* $X_i \in \mathcal{D}'$ *and* $Y \in \mathcal{D}$,
  then $I(t) = X_1 \cdots X_n Y X_n \cdots X_1$,
- *if* t *is of the form* $Alt_r[X, < Y_1, \ldots, Y_n >]$ *where* $X, Y_i \in \mathcal{D}$,
  then $I(t) = Y_1 X Y_2 X \cdots Y_n X$,
- *if* t *is of the form* $Alt_l[X, < Y_1, \ldots, Y_n >]$ *where* $X, Y_i \in \mathcal{D}$,
  then $I(t) = X Y_1 X Y_2 \cdots X Y_n$,
- *if* t *is of the form* $Con[X_1, \ldots, X_n]$ *where* $X_i \in \mathcal{D}$,
  then $I(t) = X_1 \cdots X_n$,
- *if* t *is of the form* $Unit[X]$ *where* $X \in \mathcal{D}$,
  then $I(t) = (X)$.

Based on the interpretation function, the extensional equality (which is not the structural identity) of $\mathcal{SL}$ expressions is defined. Two $\mathcal{SL}$ expressions are extensionally equal if and only if they both are interpreted as the same string pattern.

The variable-free $\mathcal{SL}$ expressions represent the gestalts of one-dimensional patterns. Consequently, extensionally equal expressions may constitute different gestalts of one pattern.

As a result, structural information theory can be described in terms of the $\mathcal{SL}$ language together with a complexity function, C, which assigns a natural number to each $\mathcal{SL}$ expression, i.e. $C(t) \in \mathbf{N}$, where $t \in \mathcal{SL}$.

8. DEFINITION. *Let t be a primitive $\mathcal{SL}$ expression, $T_1, \ldots, T_n$ be arbitrary $\mathcal{SL}$ expressions and f be any ISA operator except the Unit operator. Then, based on the empirically motivated suggestion proposed by Leeuwenberg and Van der Helm, the complexity function C can be defined as follows:*

$C(t) = 1$
$C(f(T_1, \ldots, T_n)) = \sum_{i=1}^{n} C(T_i)$
$C(Unit(T_1 \ldots T_n)) = \sum_{i=1}^{n} C(T_i) + 1$

The minimum principle states that the preferable gestalt of a pattern $t \in \mathcal{D}$ is the $\mathcal{SL}$ expression, among all extensionally equal $\mathcal{SL}$ expressions to which the pattern t is assigned by the interpretation function, that has the lowest complexity value.
Thus, the *being-a-gestalt-of* can be considered as a relation between on the one hand the set of one-dimensional string patterns $\mathcal{D}$ and on the other hand, the set of $\mathcal{SL}$ expressions. This relation indicates the alternative gestalts that may be assigned to string patterns. For each pattern the minimum principle selects algebraic $\mathcal{SL}$ expressions that represent its preferred gestalts.

3.2 Extending String-algebra with Domain Relations

The perceptual structures of patterns covered by SIT are based on certain kinds of regularities. These regularities are defined in terms of the identities of pattern constituents. For example, the regularity of the string pattern *abccab* is defined in terms of the identity relations between the first and the second occurences of *ab* and the first and the second occurences of *c*. These identity relations are reflected by the description of the pattern, i.e. $Sym_e(Unit(a, b), c)$.

However, given a certain domain of perceptual patterns other relations may be applicable to the pattern constituents as well. For instance, in the domain of string patterns the order of alphabet characters relates them to each other by means of relations such as *successor*, *predecessor*, etc. We claim that, like the identity relation, domain specific relations may induce perceptually motivated pattern regularities as well. In particular, we claim that the pattern regularity (descriptive simplicity), which is defined in terms of

domain specific relations, can be related to the perceptual regularity (phenomenal simplicity).
This issue is explicitly addressed in Van der Helm and Leeuwenberg [VdHL91]. They argue that such operators are cognitive and not perceptual, and therefore should not be incorporated in the perceptual coding principles. In this way, they suggest a strict seperation between perception and cognition. But, we do not follow the idea of a strict seperation between perception and cognition and feel that while operators requiring addition or otherwise complex operations may be left out, simple operators like *successor* and *predecessor* do end up playing a critical role in perception, and ought to be included in the gestalts.
This idea is based on the observation that in the domain of string patterns, patterns like *abc* or *zyx* have different perceptual status than a pattern like *kbr*. The first two patterns have special perceptual status since the pattern *abczyx* will be preferably perceived as consisting of two substrings *abc* and *zyx* instead of arbitrary substrings. The special status of these patterns can be explained in terms of their regular structures, which are defined in terms of domain relations. In this case, the string pattern *abc* has a regular structure since c is the successor of b which is in turn the successor of a. In other words, the regularity of the pattern *abc* is based on the domain specific successor relation which holds between the first and the second elements and between the second and the third elements. Similarly, the string pattern *zyx* is perceived as a regular pattern since x is the predecessor of y which is in turn the predecessor of z. These kinds of regularities will be called *domain specific regularities.* Thus, our conjecture is that the acquaintance with domain relations may influence the perception of patterns since these relations may reveal regularity.

Moreover, in the coding system proposed by Leeuwenberg and Van der Helm there is no account for the role that visual attributes like size, color, texture, etc. play in the gestalts of perceptual patterns. This is important since the constituents of perceptual patterns may be related to each other according to various visual attributes. For example, in the string pattern aktns pg the individual *non-identical* letters are related to each other according to their size values such that the size relation between individual letters effects the gestalt of this pattern. Although we believe that visual attributes play an important role in the gestalts of perceptual patterns, we ignore the effect of these attributes on the gestalts of string patterns and will discuss this issue in more detail when we study the gestalts of two-dimensional visual patterns.

Thus, at this point we do not follow Van der Helm and Leeuwenberg [VdHL91] who claim that the perceptually motivated structures are only identity structures that can be described by the ISA operators and will suggest an extension of perceptually motivated structures. For instance, in the above example the structure of the pattern *abc* should be analysed as the iteration of the successor relation among the three pattern elements.

Consequently, we will extend the String-algebra to cover domain specific regularities by modifying the iteration and the alternation operator. The modified version of the ISA operators will generate perceptually motivated regularities with respect to a certain predefined and empirically selected set of domain specific relations which we will call the set of *admissible relations*. We assume that these domain specific relations can be specified as one-place functions. Therefore, we use the term *admissible function* instead of admissible relation.

To modify the iteration operator such that it covers domain specific regularities, we redefine this operator to take an admissible function as additional argument. In order to modify the left and right alternation operators, we follow the original idea of these operators as proposed in SIT. According to SIT, the alternation operators express structures that comprise two interspersed structures such that one of them is an iteration structure. In fact, the regularity of an alternation structure is the regularity of the iteration structure involved. Since we modify the iteration operator by including an admissible function as one of its arguments, we modify the alternation operators in the same way by including an admissible function as one of their arguments. This additional argument will extend the domain of the embedded iteration structure and therefore the domain of alternation structures. Thus, to incorporate domain specific regularities into our algebraic framework, we let F_d be the set of admissible one-place domain specific functions that may play a role in the construction of gestalts, and augment the iteration and alternation operators with an extra argument that is an element of F_d. Note that the ISA operators are second-order operators now. Before we define the extended version of the String-algebra, we should define admissible functions. Although we have argued that admissible functions should be selected by empirical studies, we believe that identity (id), successor ($succ$), and predecessor ($pred$) have to be included in the set of admissible functions. These admissible functions are defined as follows:

$$\begin{array}{llllll} id(a) & = a, & id(z) & = z, & id(x(yz)) & = x(yz), \quad etc. \\ succ(a) & = b, & succ(z) & = a, & succ((xy)z) & = (yz)a, \quad etc. \\ pred(a) & = z, & pred(z) & = y, & pred((axt)) & = (zws), \quad etc. \end{array}$$

The extended String-algebra $< \mathcal{U}, \mathcal{F} >$, called *Extended-string-algebra* consists of the universe $\mathcal{U} = \mathcal{D} \cup \mathcal{X} \cup \mathcal{N}$ as defined above and the set $\mathcal{F} = F_{ISA} \cup F_d$ which are defined as follows.

9. DEFINITION. *Let $X = x_1 \cdots x_k$ be a string pattern from $\mathcal{D}$, where x_i is a string object; $Y_1, \ldots, Y_m$ be arbitrary string patterns from $\mathcal{D}$; $f \in F_d = \{id, succ, pred, \ldots\}$; and $n \in \mathcal{N}$. We write $f^n(X)$ to indicate that the one-place function f is applied n times to X. The set of extended algebraic ISA operators, namely $\mathcal{F}$, on $\mathcal{U}$ is defined to contain the following operators:*

$$\begin{array}{lll} Iter(X, f, n) & \rightarrow & X\ f(X) \cdots f^{n-1}(X), \\ Sym_e(X) & \rightarrow & x_1 \cdots x_k x_k \cdots x_1, \\ Sym_o(X, Y_i) & \rightarrow & x_1 \cdots x_k Y_i x_k \cdots x_1, \\ Alt_r(X, f, < Y_1, \ldots, Y_m >) & \rightarrow & Y_1\ X\ Y_2\ f(X) \cdots Y_m\ f^{m-1}(X), \\ Alt_l(X, f, < Y_1, \ldots, Y_m >) & \rightarrow & X\ Y_1\ f(X)\ Y_2 \cdots f^{m-1}(X)\ Y_m, \\ Con(Y_1, \ldots, Y_m) & \rightarrow & Y_1 \cdots Y_m, \\ Unit(Y_1 \cdots Y_m) & \rightarrow & (Y_1 \cdots Y_m), \\ f(X) & & As\ defined\ above. \end{array}$$

For example, the extended ISA-algebra can generate the following descriptions for the string patterns aaa, abc, $ccbbaa$, $abccba$, $a(bc)d(bc)a$, $afbkct$, and kbr.

$$\begin{array}{lll} aaa & = & Iter(a, id, 3) \\ abc & = & Iter(a, succ, 3) \\ ccbbaa & = & Iter(Iter(c, id, 2), pred, 3) \\ abccba & = & Sym_e(Iter(a, succ, 3)) \\ a(bc)d(bc)a & = & Sym_o(Con(a, Unit(Con(b, c))), d) \\ afbkct & = & Alt_l(a, succ, < f, k, t >) \\ kbr & = & Con(k, b, r) \end{array}$$

The expressions of the Extended-string-algebra specify perceptual chunkings of string patterns. Thus, according to the modified version of ISA operators the string *abc* has an iteration regularity such that *abc* is considered as one single chunk. Note that the modified version of the iteration operator does not specify any regularity in the pattern *kbr*; therefore its perceptual chunking consists of three subchunks, i.e. k, b, and r.

Note that the symmetry operators can be considered as a special case of the modified iteration operator since they can be considered as two times iteration of one symmetry half with respect to the reflection function. Although the modification of the ISA operators makes the symmetry operator superfluous, we do not exclude it from the Extended-string-algebra. The reason not to describe the symmetry operator in terms of the iteration operator is to avoid the generation of algebraic terms which have the form of iteration of three or more times with respect to the reflection relation, i.e. the Extended-string-algebra should not generate algebraic terms of the form $Iter(X, reflection, 3)$. These algebraic terms should then violate the claim of SIT, which states that the descriptive regularities are related to the phenomenal regularities.

The extension of the String-algebra with admissible functions changes the structure of algebraic terms and therefore their structural complexities. We assume that admissible functions increase the complexity of algebraic terms since patterns will be preferably perceived in terms of identical parts. In general, we assume that different admissible functions have different perceptual relevance. In this way, different admissible functions increase the complexity value of the expressions of the Extended-string-algebra differently. This assumption reflects the prediction that there is a preference ordering among admissible functions in terms of which string patterns are described.
Moreover, we assume that the application of admissible functions to complex expressions should increase the information complexity more than their applications to primitive expressions. This idea is based on the observation that the string pattern *bcdabc* is preferably perceived as $Con(Iter(b, succ, 3), Iter(a, succ, 3))$ rather than $Iter(Iter(b, succ, 3), pred, 2)$. However, the exact order and the complexity values of admissible functions and their applications should be established by empirical research.

Finally, the computational model of SIT can be modified in order to compute the domain specific regularities as well. In the original version of the computational model (see chapter 2 section 2.1.2) a graph is gener-

ated which represents all possible substrings of a given string pattern. In fact, each edge of the generated graph represents a possible substring of the given string pattern. Then, for each substring (edge of the graph) it is tested whether that substring can be described in terms of the ISA operators. Recall that this test examines whether various constituents of a substring are identical.
In order to modify this computational model to cover the domain specific regularities of string patterns, we should test whether a substring, represented by a graph edge, can be described in terms of the modified version of the ISA operators. As the only difference between the original and the modified versions of the ISA operators consists in the iteration and the alternation operators, the computational complexity is only influenced by the complexity of these operators. The difference between the original and the modified version of these operators is the additional tests according to domain specific functions. Thus, to describe a substring in terms of these two operators one should not only test whether various constituents of the substring are identical, but it should also be tested whether those constituents (that are tested for the identity relation) are in domain specific relations. This effects the computational complexity linearly with the number of domain specific relations, which means that the order of complexity remains the same.

3.3 Gestalts of Two-dimensional Visual Patterns

We follow the SIT idea that a pattern may have several potential structures and that its perceptual structure can be selected by applying the simplicity principle. However, we believe that the encoding system introduced by SIT is not satisfactory because it covers only the small subclass of line patterns that can be represented unambiguously as linear one-dimensional strings. In order to define a system for analyzing and computing the gestalt of visual patterns such that SIT's problems and limitations can be avoided and a larger class of visual patterns can be covered, we describe gestalts in terms of regularities of attribute values of two-dimensional *primitive visual elements*.
We emphasize primitive visual elements since gestalts of two-dimensional visual patterns may also be based on the regularities of values of the characteristic attributes of compound visual elements (i.e. a group of primitive

visual elements). A characteristic attribute of a compound visual element indicates a property of that element which is not a property of the constitutive primitive visual elements. In this thesis, we will ignore those gestalts that are based on the regularities of values of the characteristic attributes of compound visual elements.
Attributes of two-dimensional primitive visual elements are often divided into visual and spatial attributes. For example, color and texture are often classified as visual attributes while position and orientation are classified as spatial attributes. In the following, we do not use the classification of attributes into visual and spatial since it is not clear whether attributes such as shape or size should be classified as visual or spatial. Therefore, we will write *visual attribute* to indicate any attribute of two-dimensional primitive visual elements.

3.3.1 Attribute-based Structure

In this section, primitive visual elements such as points, lines, circles, etc. are considered in terms of their visual attributes. The perceptual grouping of visual patterns will then be defined in terms of *certain kinds* of regularities that may exist among the values of different attributes of their constitutive primitive visual elements. A perceptual grouping which is defined in terms of regularities of values of one or more attributes is said to be *induced* by those attributes. For example, in Figure 3.1 primitive visual elements are randomly positioned (no regularity in position values) and have different shapes and sizes, but may have the same texture value. The equality of texture values results in similarity of primitive visual elements and thereby induces a grouping among them. This grouping is said to be induced by the texture attribute. Similarly, in Figure 3.2 primitive visual elements are randomly positioned and have different sizes, but have either circle or square shapes and may have the same texture value. The shape equality and the texture equality result in two possible perceptual groupings. The perceptual grouping induced by the shape attribute results in two perceptual groups: one consists of primitive visual elements that have circle shapes and the other consists of primitive visual elements that have square shapes. The perceptual grouping induced by the texture attribute results also in two perceptual groups each of which consists of primitive visual elements that have the same texture value. Finally, for an example of perceptual grouping induced by the position attribute, consider the visual pattern illustrated in Figure 3.3. In this visual pattern, the L-shape elements 1, 3 and 6 may be perceived as one perceptual group.

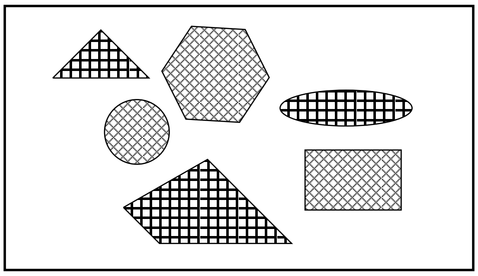


Figure 3.1: *An example of Texture based grouping.*

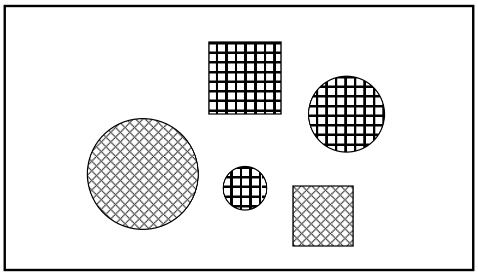


Figure 3.2: *An example of Shape and Texture based grouping.*

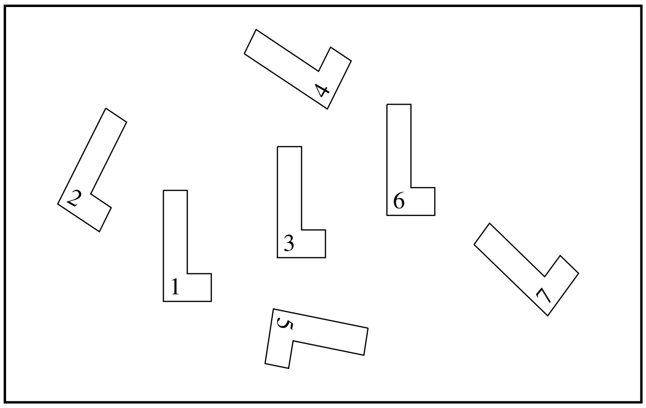


Figure 3.3: *The L-shape element 1, 3 and 6 form one perceptual group.*

Visual attributes may perceptually interact with each other. The interactions between different visual attributes influence the perceptual grouping of visual elements [Gar74, She91]. In order to study the interaction of attributes two psychological notions called *separability* and *integrality* of stimuli dimensions are used. Two dimensions are called separable when the perceptual grouping of elements is induced along one or the other dimension. For example, considering the shape and the texture attributes of primitive visual elements, a perceptual grouping may be induced by the shape attribute, by the texture attribute, or by both of them. In the last case, each attribute supports that perceptual grouping separately. In contrast, two dimensions are called non-separable (i.e. integral) when the perceptual grouping is not induced along one or the other dimension separately, but rather by the overall similarities of primitive visual elements caused by the combination of the integral dimensions. For example, using the hue, the saturation, and the brightness attributes of primitive visual elements, the induced grouping is not along one of these three attributes, but it is induced by the overall similarities of elements which is caused by the hue, the saturation, and the brightness attributes together. Similarly, in a two-dimensional space the horizontal and the vertical dimensions are non-separable since positional gestalts, constituted by primitive visual elements that differ in both x- and y-positions, are supported neither by only the horizontal dimension nor by only the vertical dimension. In order to illustrate the integrality of the horizontal and the vertical dimension consider the visual patterns in Figure 3.4-A and 3.4-B.

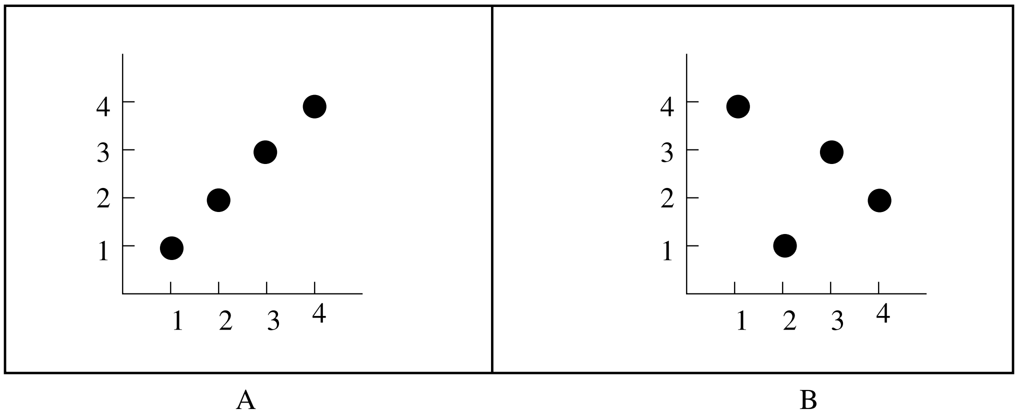


Figure 3.4: *The integrality of the horizontal and the vertical dimensions: A) black dots are perceptually grouped due to the regularity among x- and y-positions, B) black dots are not perceptually grouped due to the regularity among x-positions (y-positions are not regular).*

The regularities of the x- and the y-position values induce a perceptual grouping among elements in the visual pattern illustrated in Figure 3.4-A, while the regularity of only the x-position values does not induce a perceptual grouping among elements in the visual pattern illustrated in Figure 3.4-B. In general, it can be concluded that two or more integral attributes do not induce a perceptual grouping when there exists regularity among values of only some, but not all, of those integral attributes.

In the following, we assume that the combination of non-separable dimensions defines a complex space the elements of which are determined by the combined values of these dimensions. The perceptual grouping can then be based on the regularity of elements from such a complex space, i.e. the regularity of a certain combination of values of non-separable dimensions. The resulting perceptual groupings are then induced by combinations of non-separable attributes. These combinations of non-separable attributes may be considered as additional complex attributes. So, in order to cover grouping effects induced by combinations of non-separable dimensions, one should identify these combinations by introducing additional complex attributes and determine how the perceptual grouping is induced by them. In the rest of this chapter, we mainly focus on perceptual groupings that are induced by the horizontal and the vertical dimensions (integral dimensions) of the two-dimensional space and assume that other integral dimensions can be analyzed in a similar way.

3.3.2 Attribute-based Regularity

In the visual patterns illustrated in Figure 3.1 and 3.2, two or more primitive visual elements are claimed to be in one perceptual group because they have identical values for one or more attributes. However, consider the visual pattern illustrated in Figure 3.3. The perceptual grouping in this visual pattern is induced by the position attribute. Although the position values of the L-shape primitive elements 1,3, and 6 in this visual pattern are not identical, they form one perceptual group.

In this section, we argue that primitive visual elements may form perceptual groups because some regularities may exist among their attribute values. We consider the regularity of attribute values in terms of their functional dependencies. The functional dependency among attribute values can be characterized in terms of certain *admissible transformations*. An admissible transformation may then transform a subset of attribute values into another subset of values of the same attribute. Accordingly, one subset of attribute values can be described in terms of the second subset of attribute values and an admissible transformation. Thus, a set of attribute values may be represented in terms of a subset of those values and an admissible transformation. In this way, it can be explained that the L-shape elements 1, 3 and 6 in Figure 3.3 form one perceptual group because the position of the L-shape element 3 results from applying a translation transformation to the position of the L-shape element 1 and the position of the L-shape element 6 results from applying the same translation transformation to the position of the L-shape element 3.

We distinguish parameterized and non-parameterized transformations. A parameterized transformation transforms a subset of values of one attribute to another subset of values of the same attribute based on its parameter value. Consequently, different parameter values can be used to transform a subset of values of one attribute to different subsets of values of the same attribute. In contrast, a non-parameterized transformation does not depend on parameter values and therefore transforms one subset of values of an attribute to another subset of values of that attribute.
For example, the parameterized transformation $\omega_{att}^{(i)}$ transforms a value x of the attribute *att* by adding the parameter value to x, i.e.

$$\omega_{att}^{(i)}(x) = x + i.$$

The applications of $\omega_{att}^{(1)}$ and $\omega_{att}^{(2)}$ to the value v of the attribute *att* result two different values $v + 1$ and $v + 2$ of the attribute *att*, respectively. On the other hand, the non-parameterized transformation ψ_{att} transforms the value x of the attribute *att* by multiplying a constant value to it, i.e.

$$\psi_{att}(x) = x \times c.$$

The application of this transformation to the value v of the attribute *att* results always in value $v \times c$ of the attribute *att*.
In order to illustrate the use of transformations, consider the visual pattern in Figure 3.4-A. The x- and the y-positions of the primitive visual elements (black dots) involved in this pattern can be described in terms of the x- and the y-position of the left-most black dot which is $(1, 1)$. In fact, the positions of the three right-most black dots can be described in terms of the position of the left-most black dot and the parameterized position transformations

$$\omega_{xpos}^{(i)}(x) = x + i, \text{ and}$$
$$\omega_{ypos}^{(i)}(y) = y + i.$$

Note that the x- and the y-positions of the three right-most black dots result by applying these transformations to the position (1,1) and based on the parameter values 1,2, and 3.

In the following, we will use the three basic Euclidean transformations called *Translation, Rotation* and *Reflection* for the two-dimensional position attribute. Moreover, we distinguish two kinds of rotation transformations. The rotation transformations of the first kind, called *p-rotate transformations*, rotate primitive visual elements only with respect to their center positions and do not rotate the shape of primitive visual elements. The rotation transformations of the second kind, called *s-rotate transformations*, rotate primitive visual elements with respect to their positions and shapes. For example, visual patterns illustrated in Figure 3.5-A and 3.5-B result from p-rotate and s-rotate transformations, respectively.

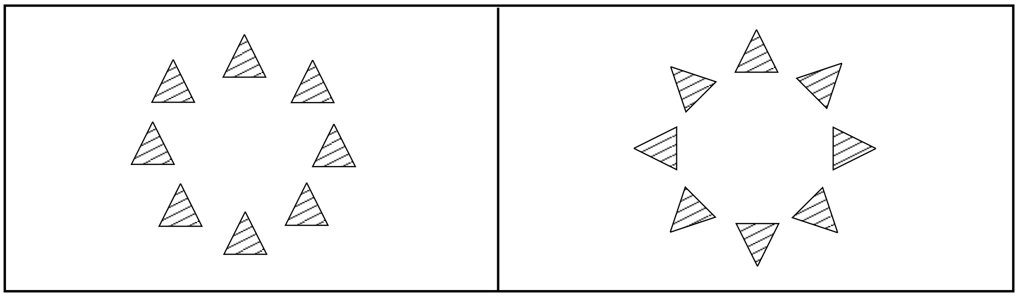


Figure 3.5: *Examples of p-rotation (A) and s-rotation (B) transformations.*

Finally, besides the distinction between parameterized and non-parameterized transformations, we introduce a second distinction between transformations which is orthogonal to the first one. According to the second distinction, the admissible transformations can be divided into two kinds: *abstract* and *concrete transformations*. An abstract transformation is an intensional description of functional dependencies between values of an attribute, while a concrete transformation is an extensional description of functional dependencies between values of an attribute. An abstract transformation can thus be represented by a formal expression which expresses the functional dependencies between attribute values in terms of abstract mathematical operations and relations, while a concrete transformation can only be represented by a set of tuples each of which relates attribute values to each other and thereby expresses their functional dependencies.
For example, the above mentioned parameterized transformation $\omega_{att}^{(i)}$ and the non-parameterized transformation ψ_{att} are abstract transformations which are represented by the formal expressions $x + i$ and $x \times c$, respectively. An example of a non-parameterized concrete shape transformation is the set $\{(rectangle, circle), (square, triangle), (triangle, line)\}$, and an example of a parameterized concrete shape transformation is the set

$\{(1, circle, rectangle), (2, circle, circle), (3, circle, square)\}$. The distinction between abstract and concrete transformations will be used for several reasons as we will explain in the rest of this chapter.

Note also that we speak of admissible transformations without discussing the criteria that will determine when a transformation is admissible. In general, we assume that admissible transformations should be empirically selected and thus should be known beforehand.

3.4 A Coding System for Visual Patterns

In the previous section, we considered two-dimensional visual patterns as consisting of primitive visual elements like points, lines, circles, etc. It is explained that primitive visual elements can be defined in terms of their visual attribute values. Then, the regularity of visual patterns was analyzed in terms of regularity of attribute values of their constitutive primitive visual elements. The regularity of attribute values is considered in terms of functional dependencies among them. These functional dependencies can be characterized by admissible transformations.

In this section, we represent a visual pattern as a set of n-tuples. Each n-tuple consists of values of n different visual attributes and represents a primitive visual element. For example, considering only the position and the shape attribute, a primitive visual element which has circle shape and is placed at the position $(1, 1)$ is represented as $<< 1, 1 >, circle >$. The $<>$ brackets around the x- and the y-position indicate the integrality of these attribute values.

The perceptual regularity of two-dimensional visual patterns will be defined in terms of functional dependencies among values of identical attributes of the n-tuples that represent the primitive visual elements of that pattern. These functional dependencies are characterized by admissible transformations that transform values of identical attributes to each other. Since primitive visual elements are represented by means of values of n attributes, the functional dependencies between n-tuples can be characterized by an n-tuple of admissible transformations. Each transformation in the n-tuple is defined on values of one attribute.

For example, let the non-parameterized abstract x-position transformation ψ_{xpos}, the non-parameterized abstract y-position transformation ψ_{ypos}, and the non-parameterized concrete shape transformation ψ_{shape}, be respectively defined as follow:

$\psi_{xpos}(x) = x + 1$,
$\psi_{ypos}(y) = y + 1$, and
$\psi_{shape}(circle) = \{(circle, square)\}$.

Then, the non-parameterized transformation ψ is a 3-tuple, defined as

$$\psi =<< \psi_{xpos}, \psi_{ypos} >, \psi_{shape} >.$$

Now, the transformation ψ characterizes the functional dependencies between two primitive visual elements that are represented by n-tuples $<< 1, 1 >, circle >$ and $<< 2, 2 >, square >$ since the second n-tuple results by applying ψ to the first n-tuple, i.e.

$$\begin{aligned} &<< \psi_{xpos}, \psi_{ypos} >, \psi_{shape} > (<< 1, 1 >, circle >) \ = \\ &\qquad << \psi_{xpos}(1), \psi_{ypos}(1) >, \psi_{shape}(circle) >=<< 2, 2 >, square >. \end{aligned}$$

Note the application of the 3-tuple of transformations (ψ) to a 3-tuple of attribute values. In general, given $v_{att_1}, \ldots, v_{att_n}$ as values of $att_1, \ldots, att_n$, respectively, and given transformations $\tau_{att_1}, \ldots, \tau_{att_n}$ defined on the same attributes, the application of $< \tau_{att_1}, \ldots, \tau_{att_n} >$ to $< v_{att_1}, \ldots, v_{att_n} >$ is defined as follows:

$$< \tau_{att_1}, \ldots, \tau_{att_n} > (< v_{att_1}, \ldots, v_{att_n} >) =< \tau_{att_1}(v_{att_1}), \ldots, \tau_{att_n}(v_{att_1}) >.$$

In the following, we define perceptual regularities of two-dimensional visual patterns as functional dependencies among visual attribute values that can be described in terms of ISA operators. We discuss different classes of regularities of two-dimensional visual patterns. Each class of regularities gives rise to a certain structure which is described by a specific ISA operator.

3.4.1 Iteration Structure

The first kind of perceptual structures of visual patterns covers iteration structures. A visual pattern has the iteration structure if its representation consists of k subsets of n-tuples such that they can be described in terms of k successive applications of an admissible parameterized transformation to the n-tuples contained in one of the subsets. An admissible parameterized transformation is defined on a natural number (its parameter value) and its successive applications to the n-tuples that are contained in one subset generate all subsets of n-tuples. The idea is that the order of successive natural numbers, used as the parameter values in the successive applications of an admissible transformation, imposes an ordering on the generated subsets of n-tuples. For example, consider the pattern in Figure 3.6 in which only position and shape attributes are taken into account.

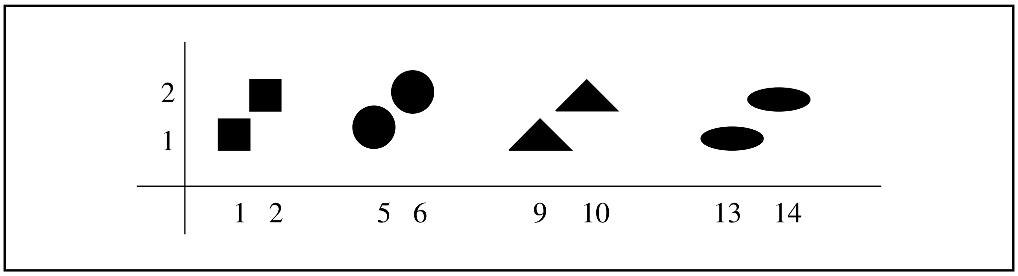


Figure 3.6: *Position based iteration structure.*

This pattern has an iteration structure. The x- and the y-positions of all pairs of similar primitive elements can be generated by four successive applications of two abstract parameterized transformations to the x- and the y-positions of the square pair: one translation transformation $\omega^{(i)}_{xpos}$ for the x-position attribute and one identity transformation $\omega^{(i)}_{ypos}$ for the y-position attribute. Moreover, the shape transformation $\omega^{(i)}_{shape}$ is a concrete parameterized transformation. These transformations are respectively defined as follows:

$$\omega^{(i)}_{xpos}(x) = x + (i-1),$$
$$\omega^{(i)}_{ypos}(y) = y, \text{ and}$$
$$\omega^{(i)}_{shape}(square) = \{ \quad (1, square, square), (2, square, circle), (3, square, triangle), (4, square, oval) \ \}.$$

Let $<< 1,1 >, square >$ and $<< 2,2 >, square >$ be two 3-tuples that represent the two squares illustrated in Figure 3.6, and $\omega^{(i)}_{xpos}, \omega^{(i)}_{ypos}$, and $\omega^{(i)}_{shape}$ as defined above. Then, the whole visual pattern illustrated in Figure 3.6 can be represented as the following set:

$$\{ \quad << \omega^{(i)}_{xpos}, \omega^{(i)}_{ypos} >, \omega^{(i)}_{shape} > (<< 1,1 >, square >), \\ << \omega^{(i)}_{xpos}, \omega^{(i)}_{ypos} >, \omega^{(i)}_{shape} > (<< 2,2 >, square >) \ | \ i = 1, \ldots 4\}.$$

In general, given visual attributes $att_1, \ldots, att_n$, an n-tuple of admissible parameterized transformations $\omega^{(i)} =< \omega^{(i)}_{att_1}, \ldots, \omega^{(i)}_{att_n} >$ consisting of parameterized transformations defined on attributes $att_1, \ldots, att_n$, and n-tuples $e_1, \ldots, e_p$ where $e_j =< v^j_{att_1}, \ldots, v^j_{att_n} >$ consisting of n values of attributes $att_1, \ldots, att_n$, the following set represents a visual pattern which has iteration structure:

$$\{\omega^{(i)}(e_1), \ldots, \omega^{(i)}(e_p) \ \mid \ i = 1, \ldots, k\}.$$

The application of transformation $\omega^{(i)}$ to an n-tuple (e_j) is defined above.

3.4.2 Symmetry Structure

The second kind of perceptual structures of visual patterns covers symmetry structures. A visual pattern has a symmetry structure if its representation consists of two subsets of n-tuples such that the position values of the n-tuples contained in one subset are reflections of the position values of the n-tuples contained in the second subset relative to a reflection axis. The values of non-positional attributes of the n-tuples contained in one subset are identical to the values of the same attributes of the reflected n-tuples contained in the second subset. In this way, one subset of n-tuples can be described in terms of the second subset and an n-tuple of transformations. The position transformations are non-parameterized Euclidean reflection transformations and all other non-positional transformations are non-parameterized identity transformations.
For example, the visual pattern shown in figure 3.7 is a pattern with symmetry structure. Note that in this symmetrical pattern primitive visual elements that are reflection of each other are identical in all attribute values except the position value. The structure of this kind of two-dimensional symmetrical patterns will be called *strict symmetry*.

The concept of the symmetry structure can be generalized to cover a larger class of symmetrical patterns. This generalization is based on the idea that the symmetry structure of two-dimensional visual patterns can still be perceived even when the values of non-positional attributes of reflected primitive visual elements are not identical (the positions of the primitive visual elements have to be reflections of each other).
Like visual patterns that have the strict symmetry structure, visual pat-

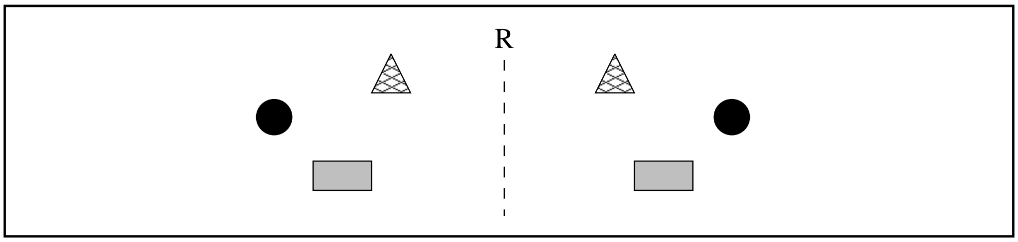


Figure 3.7: *A visual pattern with strict symmetry structure.*

terns with the general case of symmetry structure can be represented in terms of two subsets of n-tuples such that the position values of the n-tuples contained in one subset are reflections of the position values of the n-tuples contained in the second subset relative to a reflection axis. But, unlike visual patterns that have the strict symmetry structure, the values of non-positional attributes of the n-tuples contained in one subset may differ from the values of identical non-positional attributes of the reflected n-tuples contained in the second subset. In this way, one subset of n-tuples can be described in terms of the second subset and an n-tuple of transformations including Euclidean reflection transformation. In the following, we consider the strict symmetry as a special case of symmetry structure.
For an example of the general case of symmetry structure consider the visual pattern illustrated in Figure 3.8. In this visual pattern, the size [1] of each primitive visual element from the right-side symmetry half is twice the size of its reflected visual element from the left-side symmetry half.

Considering only the x-position, the y-position, the shape, and the size attribute, this pattern can be represented as two subsets, each consisting of three 4-tuples. The x- and the y-position values of the 4-tuples contained in one subset can be determined by applying the Euclidean reflection transformation relative to the reflection axis $x = 5$ to the x- and the y-position values of the 4-tuples contained in the second subset. This implies two abstract non-parameterized transformations ψ_{xpos} and ψ_{ypos} (for respectively the x- and the y-position) defined as follows:

$\psi_{xpos}(x) = 10 - x$
$\psi_{xpos}(y) = y$

[1]The sizes of different primitive visual elements can be compared to each other by comparing the sizes of the rectangles that contain those primitive visual elements.

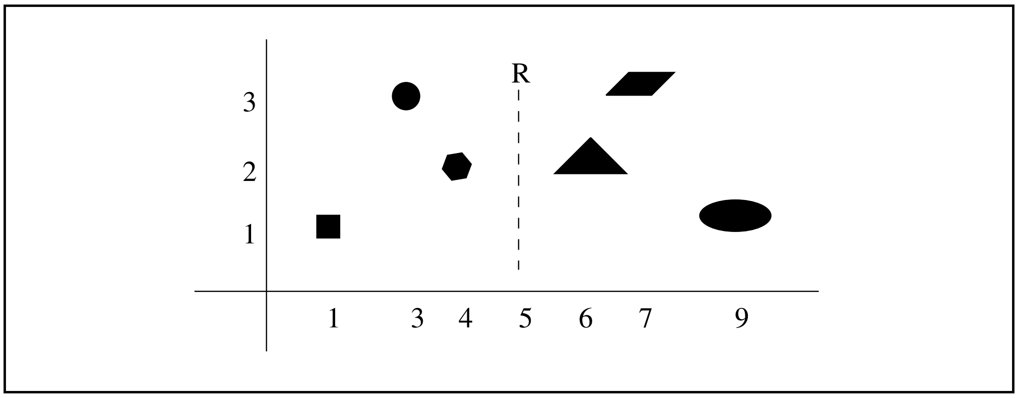


Figure 3.8: *A visual pattern with symmetry structure.*

Moreover, the shape values of the 4-tuples contained in one subset (the subset that represents the right-side symmetry half) can be determined by applying a non-parameterized concrete transformation ψ_{shape} to the shape values of the 4-tuples contained in the second subset. The shape transformation ψ_{shape} is defined as follows:

$$\psi_{shape}(S) = \{(hexagon, triangle), (circle, parallelogram), (square, oval)\}.$$

Finally, the size values of the 4-tuples contained in one subset (the subset that represent the right-side symmetry half) can be determined by applying a non-parameterized abstract transformation ψ_{size} to the size values of the 4-tuples contained in the second subset. The size transformation ψ_{size} is defined as follows:

$$\psi_{size}(S) = 2 \times S.$$

Let $<< 1, 1 >, square, 1 >$, $<< 3, 3 >, circle, 1 >$, and $<< 4, 2 >, hexagon, 1 >$ be 4-tuples that represent the three primitive visual elements from the left-side symmetry half of the visual pattern illustrated in Figure 3.8, and non-parameterized transformations $\psi_{xpos}, \psi_{ypos}, \psi_{shape}$, and ψ_{size} as defined above. Then, the whole visual pattern illustrated in Figure 3.8 can be represented as the following set:

$$
\begin{array}{llll}
\{ & <<1,1>, square, & 1 & >, \\
 & <<3,3>, circle, & 1 & >, \\
 & <<4,2>, hexagon, & 1 & >, \\
<<\psi_{xpos}, \psi_{ypos}>, \psi_{shape}, \psi_{size}> (& <<1,1>, square, & 1 & >), \\
<<\psi_{xpos}, \psi_{ypos}>, \psi_{shape}, \psi_{size}> (& <<3,3>, circle, & 1 & >), \\
<<\psi_{xpos}, \psi_{ypos}>, \psi_{shape}, \psi_{size}> (& <<4,2>, hexagon, & 1 & >)\}.
\end{array}
$$

In general, given visual attributes $att_1, \ldots, att_n$, an n-tuple of admissible non-parameterized transformations $\psi =< \psi_{att_1}, \ldots, \psi_{att_n} >$ consisting of non-parameterized transformations defined on attributes $att_1, \ldots, att_n$, and n-tuples $e_1, \ldots, e_{k/2}$ where $e_j =< v^j_{att_1}, \ldots, v^j_{att_n} >$ consisting of n values of attributes $att_1, \ldots, att_n$. Then, the following set represents a visual pattern which has the symmetry structure:

$\{e_1, \ldots, e_{k/2}, \ldots, \psi(e_1), \ldots, \psi(e_{k/2})\}$.

3.4.3 Unit Structure

The notion of unit is introduced to define a special kind of symmetrical patterns in which not individual primitive elements but units of primitive elements are reflected. The reflection of a unit of primitive visual elements preserves the spatial relation between united elements such that the reflection of a unit of elements is similar to a translation of the united elements relative to the reflection axis. For example, the two L-shape elements in Figure 3.9 are united elements such that their mutual spatial relation is preserved under the reflection transformation. Note that in graphics programs, such as superpaint, this kind of grouping operation is common.

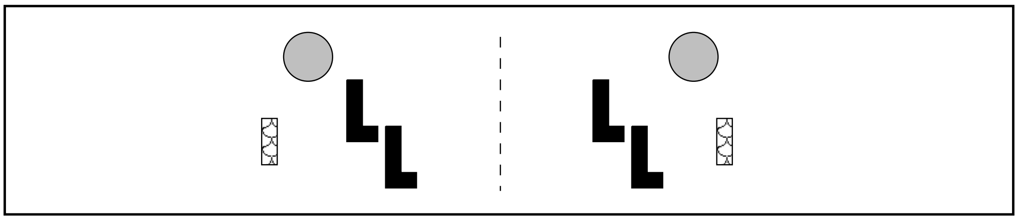


Figure 3.9: *The spatial relation between united L-shape elements is preserved in the symmetrical halves.*

The notion of unit makes sense not only in these artificially constructed symmetrical patterns, but also in visual patterns in which some parts or primitive visual elements in symmetrical halves are intended not to be mirror images of each other. These parts or primitive visual elements may be text objects, a human face, or any other natural pictures. In Figure 3.10, a symmetrical pattern is illustrated in which the text object is considered as a unit and therefore not reflected.

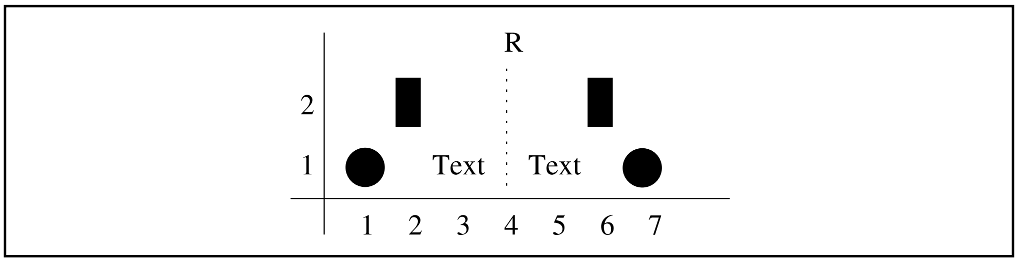


Figure 3.10: *A symmetrical pattern in which the text object is considered as a unit element.*

In the following, we assume that units of primitive visual elements are given beforehand by placing their representing n-tuples within $\langle\ \rangle$ brackets. Note the difference between these brackets and those that are used for n-tuples. Thus, considering only the x-position, the y-position, and the shape attribute, and given $<< 1,1 >, circle >$, $<< 2,2 >, square >$, $<< 3,1 >, Text >$, $<< 5,1 >, Text >$, $<< 6,2 >, square >$, and $<< 7,1 >, circle >$ as n-tuples that represent primitive visual elements in the visual pattern illustrated in Figure 3.10, the following set of n-tuples represents a visual pattern in which text elements are unit elements:

$$\{ \quad << 1,1 >, circle >, \quad << 2,2 >, square >, \quad \langle << 3,1 >, Text > \rangle, \\ << 6,2 >, square >, \quad << 7,1 >, circle >, \quad \langle << 5,1 >, Text > \rangle, \ \}.$$

3.4.4 Alternation Structure

In the original version of SIT, where the domain of patterns is restricted to the domain of sequential string patterns, there is an additional perceptually motivated structure called alternation structure. We have explained that the alternation structure is based on a sequence of string elements in which one string element is alternating with other string elements. For example, the string pattern *akpacsgaq* would be considered as having an alternation structure which can be represented by the expression $Alt_l(a, < kp, csg, q >)$.

It is important to note that the alternation structure of patterns can only be perceived if there is a sequence order among constitutive elements. In the case of string patterns the neighborhood of string elements (each element has two neighbors) imposes a sequence order among them.

However, since in the case of two-dimensional visual patterns there is no pre-defined sequence order among constitutive primitive visual elements, one may think that the alternation structure cannot be defined on two-dimensional visual patterns. In order to show that this suggestion is not valid, consider the visual pattern illustrated in Figure 3.11.

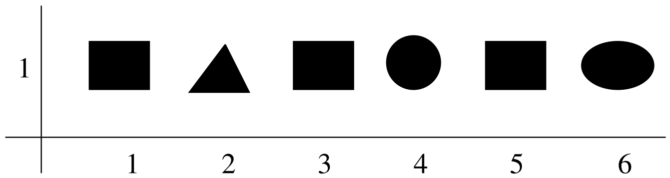


Figure 3.11: *An example of two-dimensional visual pattern which may be considered as having alternation structure.*

In this visual pattern, the iteration regularity which is defined on position values imposes a sequence order among primitive visual elements. Given this sequence order, the shape values can be considered as inducing an alternation structure on primitive visual elements.
Consequently, the reformulation of perceptual structures in terms of regularities of values of different attributes suggests that the alternation regularity is a special case of iteration regularity where the position values can be described by an abstract parameterized Euclidean transformation and the values of one or more other non-positional attributes can be described by *alternation transformations.* An alternation transformation is a parameterized transformation which uses the sequential order of parameter values and specifies the transformation values such that one subset of values of an attribute is alternated with other subsets of values of the same attribute. Following the left and the right alternation operators in the original version of SIT and given A as the subset of attribute values that are alternated with subsets of attribute values A_i, we define *even* and *odd* alternation transformations as follows:

$$\begin{array}{lll} Even-alternation & =\{(2k-1,A,A_{2k-1}),(2k,A,A) & \mid k=1,\ldots,i\} \\ Odd-alternation & =\{(2k,A,A_{2k}),(2k-1,A,A) & \mid k=1,\ldots,i\} \end{array}$$

Considering only the x-position, the y-position, and the shape attribute and based on the even and the odd alternation transformations, the visual pattern illustrated in Figure 3.11 can be represented in the same way as visual patterns with iteration structures are represented. The only difference is that the parameterized transformations for non-positional attributes can be alternation transformations. In the case of the visual pattern illustrated in Figure 3.11 the alternation transformation for the shape attribute $\omega^{(i)}_{shape}$ is defined as follows:

$$\omega^{(i)}_{shape}(S) =$$
$$\{(2,S,triangle),(4,S,circle),(6,S,oval),(2k-1,S,S) \mid k=1,\ldots,3\}.$$

3.4.5 Composition Structure

Primitive visual elements for which there is no regularity of the above kinds among their attribute values are assumed to have a composition structure. The composition structure corresponds with the concatenation structure of the original version of SIT. A visual pattern has a composition structure if it consists of unrelated primitive visual elements. For example, the visual pattern illustrated in Figure 3.12 has a composition structure. Note that every arbitrary set of primitive visual elements can be considered as having a composition structure. A visual pattern that has the composition structure is represented as a set of n-tuples of visual attribute values.

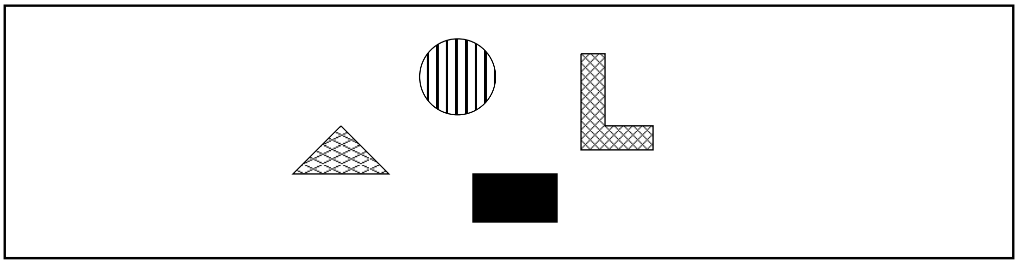


Figure 3.12: *An example of two-dimensional visual pattern which should be described as having the composition structure.*

It is important to note that this visual pattern can also be described as having an iteration structure where the transformations of all attributes, inclusive positional attributes, are concrete transformations. However, as we will explain later in this chapter (when we discuss the information complexity of expressions that represent gestalts of two-dimensional visual patterns) the description of such a visual pattern in terms of iteration structure with only concrete transformations results in higher information complexity than its alternative description in terms of composition structure. In general, we claim that the information complexity will disambiguate all possible descriptions of a visual pattern and select those which represent the preferred gestalts of that visual pattern.

3.4.6 VREG : A Language for Visual REGularities

In the previous sections, we explained that two-dimensional visual patterns can be represented as a set of n-tuples each of which represents a primitive visual element by means of n values of different visual attributes. It is argued that the regularities of values of identical attributes among n-tuples can be described by specific ISA operators. These regularities give rise to perceptually motivated structures and determine the gestalts of two-dimensional visual patterns.

In this section, the proposed view of regularities of two-dimensional visual patterns are used to define a formal coding language the expressions of which represent gestalts of two-dimensional visual patterns. This coding language is defined in terms of a set of n-tuples that represent primitive visual elements and a set of structural rules that specifies the gestalts of two-dimensional visual patterns. Let $Att_1, \ldots, Att_n$ be the domains of values of n different visual attributes. Then, the set E of representations of possible primitive visual elements is determined by the Cartesian product of these n domains of attribute values, i.e.

$$E = Att_1 \times, \ldots, \times Att_n = \\ \{e^j =< a_1^j, \ldots, a_n^j >| \ \forall a_1^j \in Att_1, \ldots, \forall a_n^j \in Att_n\}.$$

Since regularities of attribute values for iteration structures are characterized by admissible parameterized transformations, we define for each attribute a set of admissible parameterized transformations. In general, we assume $\Omega_{Att_1}, \ldots, \Omega_{Att_n}$ to be the sets of typed parameterized admissible transformations. The type Att_j indicates the domain of attribute values to

which parameterized transformations $\omega^{(i)}_{Att_j} \in \Omega_{Att_j}$ are applicable.
Since primitive visual elements are represented as n-tuples of values of mutually distinct visual attributes, the parameterized transformations that are applied to them are n-tuples of admissible parameterized transformations having the same mutually distinct types. Thus, the set Ω of possible n-tuples of admissible parameterized transformations is the Cartesian product of n mutually distinct typed sets of admissible parameterized transformations, i.e. $\Omega = \Omega_{Att_1} \times \ldots \times \Omega_{Att_n}$.
Finally, we define an additional set $\Psi = \Psi_{Att_1} \times \ldots \times \Psi_{Att_n}$ to be a set of possible admissible non-parameterized transformations in which the positional transformations are admissible reflection transformations. This set is specially defined for the symmetry structures.

In the following, we define a formal language called the *VREG language* (Visual REGularity language) which generates expressions that represent gestalts of two-dimensional visual patterns. We will call these expressions the *VREG expressions.*

10. DEFINITION. *The language VREG is defined over the primitives* $< \mathbf{E}, \Omega, \Psi, Iter, Sym, Comp, Unit >$, *where:*

- $\mathbf{E}$ *is the set of primitive visual elements,*
- Ω *is the set of admissible parameterized transformations, i.e.* $\Omega = \{\omega^{(i)} \mid \omega^{(i)} =< \omega^{(i)}_{pos}, \omega^{(i)}_{size}, , \ldots > \;\&\; \omega^{(i)}_{Att_j} \in \Omega_{Att_j}\}$, *where* $\omega^{(i)}_{pos}$ *is a parameterized 2D Euclidean transformation,*
- Ψ *is the set of admissible reflection transformations, i.e.* $\Psi = \{\psi \mid \psi =< \psi_{pos}, \psi_{size}, \ldots > \;\&\; \psi_{Att_j} \in \Psi_{Att_j}\}$, ψ_{pos} *is an 2D Euclidean reflection transformation,*
- $Iter, Sym, Comp, Unit$ *are four functor names, corresponding with iteration, symmetry, compose and unit of visual patterns, respectively,*

The VREG expressions are defined on the basis of the above primitives as follows:

1) If $e^j \in \mathbf{E}$, *then* e^j *is a VREG expression,*
2) If $A, A_1, \ldots, A_m$ *are VREG expressions,* $n \in \mathbf{N}$, $\omega^{(i)} \in \Omega$, $\psi \in \Psi$, *and* $t \in \Omega \cup \Psi$, *then the following expressions are VREG expressions:*

$Iter(A, \omega^{(i)}, n)$,
$Sym(A, \psi)$,
$Unit(A)$,
$Comp(A_1, \ldots, A_m)$,
$t(A)$.

Note that we did not specify a specific set of visual attributes. In this way, one may consider a set of VREG languages that differ from each other by their visual attributes and thus by their primitive visual elements. Each VREG language generates then the gestalt expressions of visual patterns characterized by a certain set of attributes.

The expressions of a VREG language refer to visual patterns in $\mathbf{R}^2$. We may define the semantics for the VREG language in a model theoretic sense with the set of possible visual patterns in $\mathbf{R}^2$ as the domain and the set of all admissible transformations from domain elements (visual patterns) to domain elements. A visual pattern in $\mathbf{R}^2$ is considered to be a set of primitive visual elements in $\mathbf{R}^2$. Therefore, the domain of visual patterns in $\mathbf{R}^2$ is considered to be the set of sets of primitive visual elements in $\mathbf{R}^2$. Moreover, since we want to allow unit-based symmetry structures, visual patterns may include unit-based visual elements as well. We assume that the domain of visual patterns in $\mathbf{R}^2$ is pre-structured according to the united primitive visual elements. This is done by assuming that the united primitive visual elements in visual patterns are indicated by being within $\langle\ \rangle$ brackets.

11. DEFINITION. *A model M for a VREG language consists of a domain $D = D_e \cup D_t$ (where D_e is a domain of visual patterns in $\mathbf{R}^2$ and D_t is a domain of admissible transformations from D_e to D_e) and an interpretation function I defined on n-tuples of attribute values (that represent primitive visual elements) and n-tuples of transformations. We assume a fixed coordinate system for the model M such that the positional values of constitutive primitive visual elements in D_e are defined in that coordinate system and all Euclidean transformations in D_t are defined with respect to the origin of that coordinate system. Given $t \in \Omega \cup \Psi$, the interpretation function has the following property:*

If $e^j \in \mathbf{E}$, then $I(e^j) \in D_e$,
If $t =< t_1, \ldots, t_n >$ is a tuple of admissible transformations, then $I(t) \in D_t$.

Given the interpretation function I, the semantics of the VREG expressions can be recursively defined as follows:

12. DEFINITION. *Let $M =< D, I >$ be a model for the VREG language, $A, A_1, \ldots, A_m$ be VREG expressions, and $\bullet$ indicate the application of transformations to domain elements. Then, given $e^j \in \mathbf{E}, \omega^{(i)} \in \Omega$, $\psi \in \Psi$, $t \in \Omega \cup \Psi$, and $n \in \mathbf{N}$, the semantic interpretation $|[\ ]|_M$ for VREG expressions is defined as follows:*

$$|[t]|_M \;=\; I(t),$$

$$|[e^j]|_M \;=\; I(e^j),$$

$$|[Iter(A, \omega^{(i)}, n)]|_M = \;\cup_{i=1}^{n}\; |[\omega^{(i)}]|_M \bullet (\; |[A]|_M \;),$$

$$|[Sym(A, \psi)]|_M = \; |[A]|_M \;\cup\; |[\psi]|_M \bullet (\; |[A]|_M \;),$$

$$|[Comp(A_1, \ldots, A_m)]|_M = \;\cup_{j=1}^{m} |[A_j]|_M,$$

$$|[Unit(A)]|_M \;=\; \langle\; |[A]|_M \;\rangle,$$

$$|[t(A)]|_M = |[t]|_M \bullet (|[A]|_M).$$

Note that the semantics of VREG expressions indicates how a visual pattern in model M is constituted by other visual patterns, i.e. it indicates the constituent structures of visual patterns. The constituent structure of visual patterns is expressed by the union relation between visual patterns.

As we have noticed, the application of a reflection transformation to a unit of primitive visual elements is not trivial. In fact, the spatial relations between united primitive visual elements are invariant under the reflection transformation. In order to preserve the spatial relations between united primitive visual elements under the reflection transformation, we define the application of the reflection transformation to united elements as a double reflection: the united elements should first be reflected according to the actual reflection axis and, subsequently, the resulting united elements should be reflected according to a reflection axis which is parallel to the actual reflection axis and goes through the center of united elements. This double reflection guarantees that the spatial relations between the united primitive visual elements will be preserved. The double reflection process is illustrated in Figure 3.13.

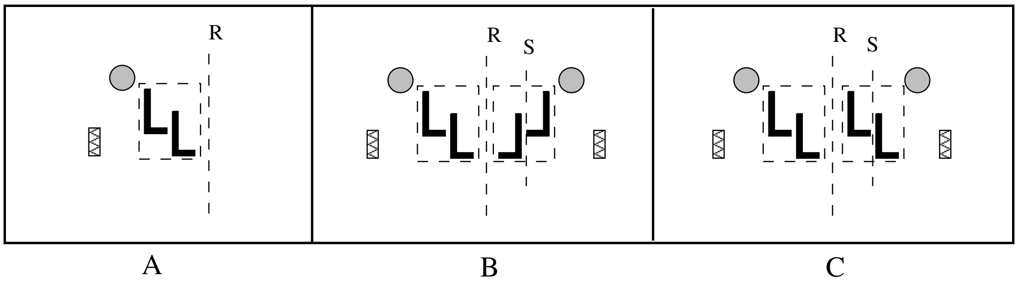


Figure 3.13: *The process of reflecting patterns containing united elements.*

In Figure 3.13-A, the pattern consists of two united L-shape elements. This pattern is reflected in Figure 3.13-B according to the actual reflection axis R. Finally, in Figure 3.13-C the reflected united elements are reflected again according to the reflection axis S which is parallel to the actual reflection axis R and goes through the center of united elements. In general, the application of transformations to united elements can be defined as follows:

13. DEFINITION. *Let Mid-reflect(ψ_{pos},A) generates a reflection transformation that reflects the visual pattern "A" according to a reflection axis which is parallel to the axis used by the reflection transformation ψ_{pos} and goes through the center of "A". Let also ψ_{pos} and $\omega_{pos}^{(j)}$ be respectively a positional reflection transformation and a positional parameterized transformation (translation, rotation, but not a reflection). Then, the application of the transformations to united elements, indicated by the $\bullet$ operator, is defined as follows:*

- $|[< \psi_{pos}, \psi_{size}, \ldots >]|_M \bullet (\langle |[A]|_M \rangle) \ =$
$\langle \ |[< \psi_{pos}, \psi_{size}, \ldots >]|_M \ \bullet (\ \textit{Mid-reflect} \ (\psi_{pos}, |[A]|_M) \ \bullet \ (\ |[A]|_M \ ) \ ) \rangle$

- $|[< \omega_{pos}^{(i)}, \omega_{size}^{(i)}, \ldots >]|_M \bullet (\ \langle |[A]|_M \rangle \ ) \ =$
$\langle |[< \omega_{pos}^{(i)}, \omega_{size}^{(i)}, \ldots >]|_M \bullet (\ |[A]|_M \ ) \rangle.$

Note in the first clause of this definition that the visual pattern $\langle |[A]|_M \rangle$ which is considered as a unit is reflected twice. This results in a visual pattern that is similar to a translation of $\langle|[A]|_M\rangle$. This is exactly what we wished to achieve with the reflection of the units of primitive visual elements.

Finally, it is important to note that for a transformation t the expression $|[t]|_M \bullet (|[A]|_M \cup |[B]|_M)$ obeys the distribution law, i.e.

$$|[t]|_M \bullet (|[A]|_M \cup |[B]|_M) = (|[t]|_M \bullet |[A]|_M) \cup (|[t]|_M \bullet |[B]|_M).$$

3.4.7 An Example of VREG Expressions

As an example of a gestalt and its corresponding VREG expression consider the two-dimensional visual pattern in Figure 3.14 which consists of arrows as primitive visual elements. In this visual pattern, the letters $e_1, \ldots, e_9$ that are attached to the individual arrows are considered to be the representations of these arrows, i.e. $e_1, \ldots, e_9$ are n-tuples of attribute values. These letters are thus not visual elements, but they are included in the visual pattern in order to indicate the primitive visual elements that they represent. These letters will be used in the VREG expressions that describe a gestalt of this visual pattern.

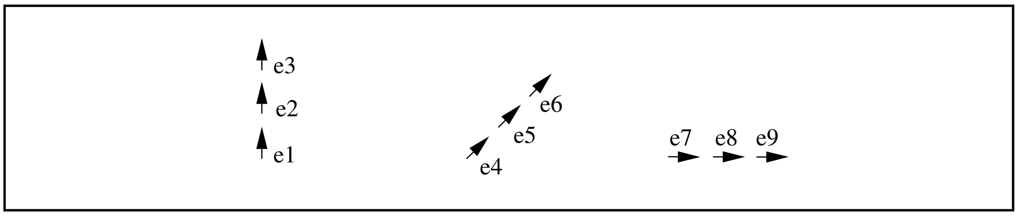


Figure 3.14: *A visual pattern having iteration structure.*

The visual pattern illustrated in Figure 3.14 can be described as consisting of three chunks. Each of these chunks contains three arrows. The second chunk of arrows represented by e_4, e_5, and e_6 results from applying a transformation (consisting of a translation in the direction of x-axis, $translate_x^{(i)}$, and a 45-degree s-rotation, $s - rotate_{45}^{(i)}$) to the first chunk of arrows represented by e_1, e_2, and e_3. The third chunk of arrows represented by e_7, e_8, and e_9 is then resulted by applying the same transformation twice to the first chunk. This gestalt (which is the preferred perceptual structure) can be represented by the following VREG expression:

$$Iter(Iter(e_1, translate_y^{(j)}, 3), translate_x^{(i)} \;\circ\; s - rotate_{45}^{(i)}, 3)$$

In this expression, $\circ$ is the transformation composition operator. Following the semantics of the VREG language, the semantics of this expression can be determined as follows:

$$|[Iter(Iter(e_1, translate_y^{(j)}, 3), translate_x^{(i)} \circ s - rotate_{45}^{(i)}, 3)]|_M =$$

$$\bigcup_{i=1}^{3} \; |[translate_x^{(i)} \circ s - rotate_{45}^{(i)}]|_M \bullet (|[Iter(e_1, translate_y^{(j)}, 3)]|_M) =$$

$$\bigcup_{i=1}^{3} |[translate_x^{(i)} \circ s - rotate_{45}^{(i)}]|_M \bullet (\bigcup_{j=1}^{3} |[translate_y^{(j)}]|_M \bullet (|[e_1]|_M)) =$$

$$\begin{aligned}
&\bigcup_{i=1}^{3} \; |[translate_x^{(i)} \circ s - rotate_{45}^{(i)}]|_M \bullet (\\
&\qquad\qquad |[translate_y^{(1)}]|_M \bullet (|[e_1]|_M) \cup \\
&\qquad\qquad |[translate_y^{(2)}]|_M \bullet (|[e_1]|_M) \cup \\
&\qquad\qquad |[translate_y^{(3)}]|_M \bullet (|[e_1]|_M)) =
\end{aligned}$$

$$\begin{aligned}
&|[translate_x^{(1)} \circ s - rotate_{45}^{(1)}]|_M \bullet (|[translate_y^{(1)}]|_M \bullet (|[e_1]|_M)) \cup \\
&|[translate_x^{(1)} \circ s - rotate_{45}^{(1)}]|_M \bullet (|[translate_y^{(2)}]|_M \bullet (|[e_1]|_M)) \cup \\
&|[translate_x^{(1)} \circ s - rotate_{45}^{(1)}]|_M \bullet (|[translate_y^{(3)}]|_M \bullet (|[e_1]|_M)) \cup \\
&|[translate_x^{(2)} \circ s - rotate_{45}^{(2)}]|_M \bullet (|[translate_y^{(1)}]|_M \bullet (|[e_1]|_M)) \cup \\
&|[translate_x^{(2)} \circ s - rotate_{45}^{(2)}]|_M \bullet (|[translate_y^{(2)}]|_M \bullet (|[e_1]|_M)) \cup \\
&|[translate_x^{(2)} \circ s - rotate_{45}^{(2)}]|_M \bullet (|[translate_y^{(3)}]|_M \bullet (|[e_1]|_M)) \cup \\
&|[translate_x^{(3)} \circ s - rotate_{45}^{(3)}]|_M \bullet (|[translate_y^{(1)}]|_M \bullet (|[e_1]|_M)) \cup \\
&|[translate_x^{(3)} \circ s - rotate_{45}^{(3)}]|_M \bullet (|[translate_y^{(2)}]|_M \bullet (|[e_1]|_M)) \cup \\
&|[translate_x^{(3)} \circ s - rotate_{45}^{(3)}]|_M \bullet (|[translate_y^{(3)}]|_M \bullet (|[e_1]|_M)).
\end{aligned}$$

In the above expression, all transformations are defined with respect to the origin of an assumed coordinate system. The above semantic description indicates how the whole visual pattern is regularly constructed by means of the arrow represented by e_1. All arrows involved in the visual pattern are determined by different compound transformations that are defined with respect to the origin of one and the same coordinate system.

3.5 An Extension of the Coding Language

The VREG language is supposed to generate expressions that represent perceptually motivated structures of two-dimensional visual patterns. However, there are perceptually motivated pattern structures that cannot be represented by the expressions of the VREG language. In this section, we consider a new class of perceptually motivated structures of two-dimensional visual patterns and propose an extension of the VREG language that can cover these pattern structures.

In the VREG language, pattern structures are represented by embedded expressions. In particular, the iteration expression is defined in terms of an embedded expression A, an admissible parameterized transformation $\omega^{(i)}$, and an index n. The semantics of the iteration expression is defined in terms of the application of the sequence of transformations $\omega^{(1)}, \ldots, \omega^{(n)}$ to the expression A, i.e. $|[Iter(A, \omega^{(i)}, n)]|_M = \bigcup_{i=1}^{n} |[\omega^{(i)}]|_M \bullet (|[A]|_M)$. Following the semantic definition of the iteration expression, the pattern represented by the expression A is considered as one single organized structure which will be successively transformed by the sequence of admissible transformations. In this way, the internal organization (structure) of the pattern represented by A is preserved under the applied transformations such that different occurrences of the transformed A have identical internal organizations. For example, in the above example the visual pattern represented by the embedded expression $Iter(e_1, translate_y^{(j)}, 3)$ consists of three arrows along one straight line. This pattern is transformed successively by the parameterized transformation $translate_x^{(i)} \circ s - rotate_{45}^{(i)}$ such that the resulting patterns under the successive transformations still consist of three arrows along one straight line, i.e. the internal organization of the pattern $Iter(e_1, translate_y^{(j)}, 3)$ is preserved.

However, there is a class of visual patterns for which the (regular) perceptual structure can be considered as a *general* case of iteration structure in contrast with the *special* case of iteration structure as it is proposed so far. Henceforth, we call patterns with the general case of iteration structure *G-structure* patterns and those with the special case of iteration structure *S-structure* patterns. Like the S-structure patterns, the G-structure patterns can be described in terms of a subpattern and a sequence of admissible parameterized transformations. But, unlike the S-structure patterns, the internal organization of the subpattern in the G-structure patterns will not

be preserved under the successive transformations.
For example, consider the visual pattern in Figure 3.15 consisting of arrows as primitive visual elements.

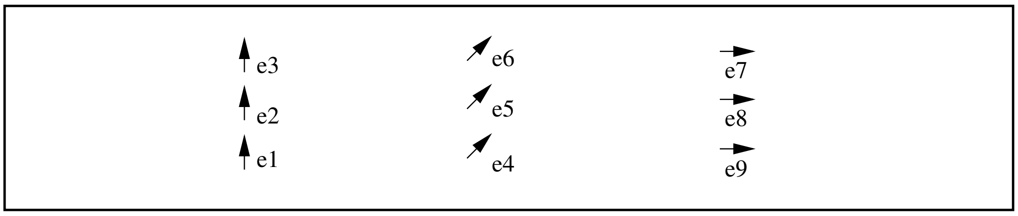


Figure 3.15: *An example of the general case of iteration structure.*

Like the visual pattern in Figure 3.14, in this visual pattern the most left vertical subpattern is translated in the direction of the x-axis, but unlike the pattern in Figure 3.14, the internal organization of the subpattern (three arrows along one straight line), is not preserved. In fact, the individual arrows in each vertical chunk are locally s-rotated. Thus, one cannot consider the left-most subpattern as one single object, which is translated in the direction of the x-axis.

The G-structure patterns can be considered as consisting of a subpattern which is successively transformed by two kinds of parameterized transformations: *local* and *global* transformations. The local transformations are successively applied to some (embedded) parts of the subpattern while the global transformation is successively applied to the whole subpattern. The local transformations are thus responsible for the internal changes among different occurrences of the subpattern.
For example, in the visual pattern illustrated in Figure 3.15 the left vertical subpattern (consisting of three arrows) is successively transformed by a global translation transformation and its individual arrows are successively transformed by a local s-rotation transformation. Another example of a G-structure pattern is illustrated in Figure 3.16.

In this pattern, one rectangle with one circle is considered as the subpattern that is transformed three times. The subpattern is translated in the horizontal direction by means of a global transformation, i.e. a translation transformation in the direction of the x-axis is applied to the circle and the rectangle. Moreover, beside the application of the global transformation, a local transformation is applied to the circle of the subpattern which trans-

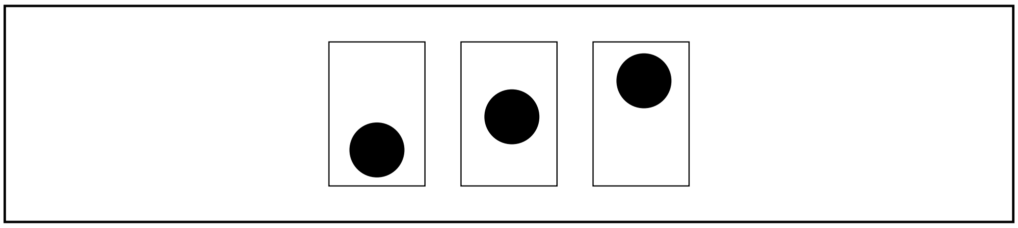


Figure 3.16: *A G-structure pattern.*

lates the circle in the direction of the y-axis. Note that this translation transformation is not applied to the rectangle of the subpattern.

The global transformation is responsible for the perception of the iterating subpattern. In fact, if the global transformation is an identity transformation such that the structure of patterns is merely determined in terms of the local transformations, then one cannot guarantee the perception of one subpattern which is successively transformed. For example, consider the visual pattern shown in Figure 3.17.

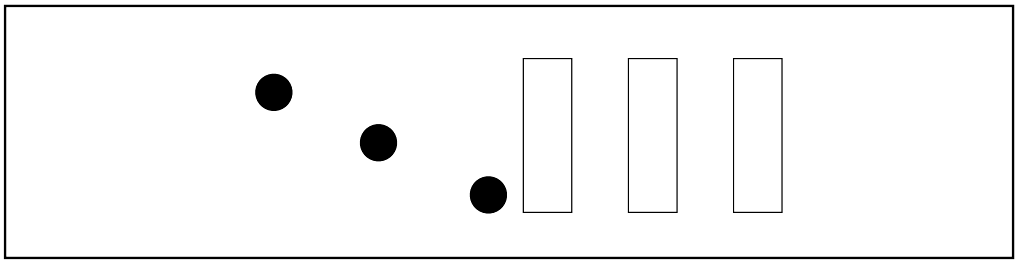


Figure 3.17: *The absence of a global transformation suppresses the perception of the subpattern that consists of one circle and one rectangle.*

In this visual pattern, one does not perceive a rectangle and a circle as forming one perceptual subpattern which is successively transformed. Nevertheless, this visual pattern can be described as having a G-structure which is based on a subpattern consisting of one rectangle and one circle. According to this G-structure description, the circle is transformed by a local diagonal translation, the rectangle is transformed by a local horizontal translation, but they are not transformed by any global transformation (or one may consider that the global transformation is the identity transformation). Therefore, in order to avoid these descriptions which do not reflect the perceptual structure of visual patterns, we exclude G-structure descriptions in which the global transformation is either absent or is the identity transformation.

The above examples of G-structure visual patterns are defined on one single local transformation. However, G-structure patterns can be defined on two or more local transformations as well. For example, consider the visual pattern illustrated in Figure 3.18.

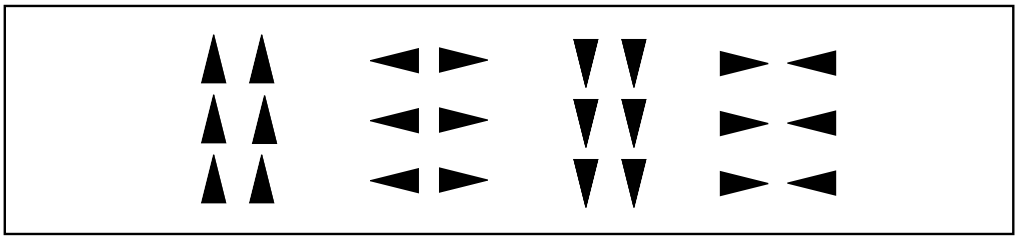


Figure 3.18: *A G-structure visual pattern defined on two local transformations.*

This visual pattern has a G-structure since it can be described in terms of the left subpattern, consisting of two vertical groups of three triangles, which is iteratively transformed by a global horizontal translation transformation and two local transformations. One local transformation is a 90° left s-rotate transformation applied to the individual triangles of the most left subpattern (the vertical group of three triangles) and the second local transformation is a 90° right s-rotate transformation applied to the individual triangles of the second left subpattern (the second vertical group of three triangles). In general, G-structure patterns are defined as those that can be described in terms of one or more local transformations. Although we believe that the G-structure of visual patterns defined on one or two local transformations are perceptually relevant (descriptive simplicity is related to the phenomenal simplicity), we do not claim the perceptual relevance of G-structure patterns defined on arbitrary number of local transformations. This issue should be the subject of empirical studies.

The expressions that represent G-structure visual patterns can be defined in terms of an iteration expression consisting of an embedded expression A, an index n, a global transformation and one or more local transformations. Since local transformations have to be applied to particular parts of the embedded expression A (at some level of embedding) we assume A to be an expression which contains transformation variables at lower embedding levels. These transformation variables stand for the local parameterized transformations. Moreover, since the parameter of local transformations are defined on the index n of the iteration expression, we use the lambda operator to change an iteration expression into a function that should be applied to parameterized local transformations. The parameter of these transformations will then be determined by the index of the embedding iteration expression and the resulting transformations will be assigned to the transformation variables that occur in the embedded expression A.

The VREG language can thus be extended to cover the general case of iteration regularities by adding a set of transformation variables VAR_t to its set of primitive elements. Subsequently, the definition of the VREG expressions is extended to cover the general case of iteration by using the lambda operator and the transformation variables. In particular, we may introduce a clause that employs the lambda operator to change an iteration expression into a functional expression, and a clause that introduces function application. Function application specifies the application of an functional expression to a parameterized local transformation. The extended version of the VREG language will be called E-VREG language (Extended VREG language). The E-VREG language can be defined by adding the following two additional clauses to the definition of the VREG language (Definition 10). The VREG language covers the general case of iteration structures when we introduce the following two clauses.

14. Definition. *Let $A[x_1, \ldots, x_n]$ be a VREG expression that contains transformation variables $x_1, \ldots, x_n$. Then, the following expressions are VREG expressions:*

3) If $Iter(A[x_1, \ldots, x_n], \omega^{(i)}, n)$ is a VREG expression and $x_1, \ldots, x_n \in VAR_t$ are transformation variables occurring in the expression A, then $\lambda x_1, \ldots, x_n \; Iter(A[x_1, \ldots, x_n], \omega^{(i)}, n)$ is a functional VREG expression,

4) If E is a functional VREG expression of the from $\lambda x_1, \ldots, x_n \; Iter(A[x_1, \ldots, x_n], \omega^{(i)}, n)$ and $\tau_1^{(i)}, \ldots, \tau_n^{(i)}$ are parameterized transformations, then $E(\tau_1^{(i)}, \ldots, \tau_n^{(i)})$ is a VREG expression.

Let M be a model for the VREG language consisting of a domain $D = D_e \cup D_t$, where D_e is the domain of visual patterns in $\mathbf{R}^2$ and D_t is the domain of all transformations from D_e to D_e. Then, the semantics of the clauses 3) and 4) are defined in the standard way using an assignment function, g, which assigns transformations to transformation variables, i.e.

$g : VAR_t \rightarrow D_t$.

15. DEFINITION. *Given the model M of the VREG language and using the assignment function, g, the semantics of the above two VREG expressions are defined by the following two rules:*

- $|[\lambda x_1, \ldots, x_n \; Iter(A[x_1, \ldots, x_n], \omega^{(i)}, n)]|_{M,g}$ is that function $h : D_t, \ldots, D_t \longrightarrow D_e$ which maps parameterized transformations from D_t to an element from D_e such that for all $\tau_1^{(i)}, \ldots, \tau_n^{(i)} \in D_t$:

$h(\tau_1^{(i)}, \ldots, \tau_n^{(i)}) = \bigcup_{i=1}^{n} |[w^{(i)}]|_{M,g} \bullet (|[A[x_1, \ldots, x_n] \;]|_{M,g[x_1/\tau_1^{(i)}, \ldots, x_n/\tau_n^{(i)}]})$,

- If E is a functional expression of the from $\lambda x_1, \ldots, x_n \; Iter(A[x_1, \ldots, x_n], \omega^{(i)}, n)$ and $\tau_1^{(i)}, \ldots, \tau_n^{(i)}$ are parameterized transformations, then

$|[E(\tau_1^{(i)}, \ldots, \tau_n^{(i)})]|_{M,g} = |[E]|_{M,g}(|[\tau_1^{(i)}]|_{M,g}, \ldots, |[\tau_1^{(i)}]|_{M,g})$.

Following the expressions for the general case of iteration, the visual pattern in Figure 3.15 can be represented as the following function application expression.

$\lambda x \; Iter(Iter(e_1, translate_y^{(j)} \circ x, 3), translate_x^{(i)}, 3) \; (s - rotate_{45})$.
The semantics of this lambda expression is a function h which maps parameterized transformations to visual patterns. This function can be specified as follows:

$|[\lambda x \; Iter(Iter(e_1, translate_y^{(j)} \circ x, 3), translate_x^{(i)}, 3)]|_{M,g} = h(\tau^{(i)}) =$

$\bigcup_{i=1}^{3} |[translate_x^{(i)}]|_{M,g} \bullet (|[Iter(e_1, translate_y^{(j)} \circ x, 3)]|_{M,g[x/\tau^{(i)}]}) =$

$\bigcup_{i=1}^{3} |[translate_x^{(i)}]|_{M,g} \bullet (\bigcup_{j=1}^{3} |[translate_y^{(j)} \circ x]|_{M,g[x/\tau^{(i)}]} \bullet (|[e_1]|_{M,g})) =$

$$\bigcup_{i=1}^{3} |[translate_x^{(i)}]|_{M,g} \bullet ($$
$$|[translate_y^{(1)} \circ x]|_{M,g[x/\tau^{(i)}]} \bullet (|[e_1]|_{M,g}) \ \cup$$
$$|[translate_y^{(2)} \circ x]|_{M,g[x/\tau^{(i)}]} \bullet (|[e_1]|_{M,g}) \ \cup$$
$$|[translate_y^{(3)} \circ x]|_{M,g[x/\tau^{(i)}]} \bullet (|[e_1]|_{M,g}) \ )$$

$$|[translate_x^{(1)}]|_{M,g} \bullet (|[translate_y^{(1)} \circ x]|_{M,g[x/\tau^{(1)}]} \bullet (|[e_1]|_{M,g})) \ \cup$$
$$|[translate_x^{(1)}]|_{M,g} \bullet (|[translate_y^{(2)} \circ x]|_{M,g[x/\tau^{(1)}]} \bullet (|[e_1]|_{M,g})) \ \cup$$
$$|[translate_x^{(1)}]|_{M,g} \bullet (|[translate_y^{(3)} \circ x]|_{M,g[x/\tau^{(1)}]} \bullet (|[e_1]|_{M,g})) \ \cup$$
$$|[translate_x^{(2)}]|_{M,g} \bullet (|[translate_y^{(1)} \circ x]|_{M,g[x/\tau^{(2)}]} \bullet (|[e_1]|_{M,g})) \ \cup$$
$$|[translate_x^{(2)}]|_{M,g} \bullet (|[translate_y^{(2)} \circ x]|_{M,g[x/\tau^{(2)}]} \bullet (|[e_1]|_{M,g})) \ \cup$$
$$|[translate_x^{(2)}]|_{M,g} \bullet (|[translate_y^{(3)} \circ x]|_{M,g[x/\tau^{(2)}]} \bullet (|[e_1]|_{M,g})) \ \cup$$
$$|[translate_x^{(3)}]|_{M,g} \bullet (|[translate_y^{(1)} \circ x]|_{M,g[x/\tau^{(3)}]} \bullet (|[e_1]|_{M,g})) \ \cup$$
$$|[translate_x^{(3)}]|_{M,g} \bullet (|[translate_y^{(2)} \circ x]|_{M,g[x/\tau^{(3)}]} \bullet (|[e_1]|_{M,g})) \ \cup$$
$$|[translate_x^{(3)}]|_{M,g} \bullet (|[translate_y^{(3)} \circ x]|_{M,g[x/\tau^{(3)}]} \bullet (|[e_1]|_{M,g}))$$

This function can now be applied to the parameterized local transformation $s - rotate_{45}$. The semantic of the function application is then defined by assigning the transformation $s - rotate_{45}$ to the variable x in the above expression.

3.6 The Information Complexity of E-VREG Expressions

In this chapter, we have introduced a coding language, called E-VREG language, for the two-dimensional visual patterns. The expressions of this coding language, called E-VREG expressions, are defined on n-tuples of attribute values (representing primitive visual elements) and n-tuples of transformations that are applicable to them.

The E-VREG expressions represent gestalts of two-dimensional visual patterns. A visual pattern may have several gestalts each of which is represented by an E-VREG expression. In order to disambiguate the gestalts of a visual pattern and select the preferred gestalts of that visual pattern, we follow the gestalt disambiguation idea of SIT by considering the information complexity of E-VREG expressions. The reformulation of gestalts in terms of E-VREG expressions suggests that the information complexity should be defined in terms of the information complexities of n-tuples of attribute values and n-tuples of transformations.

The information complexity of n-tuples of attribute values depends on perceptual *dominance ordering* of visual attribute values. The perceptual dominance ordering among different attributes is studied in [Kub81]. According to this empirical study, the position and orientation attributes have a more important perceptual status than other attributes like the color and the shape attributes. In the same way, the color attribute has a more important perceptual status than the shape or the size attributes, etc. To illustrate the perceptual dominance ordering consider the patterns in Figures 3.19 and 3.20. In Figure 3.19 the perceptual grouping which is induced by the position attribute dominates the grouping which is induced by the shape, the size, and the texture attributes. Similarly, in Figure 3.20 the grouping which is based on the color attribute dominates the grouping which is based on the shape attribute.

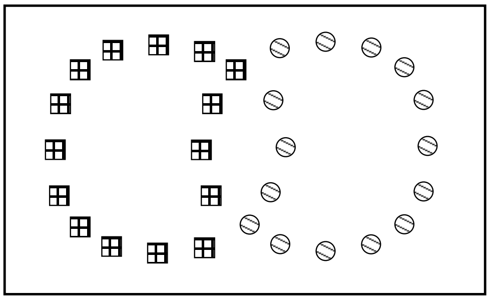


Figure 3.19: *The position attribute dominates the grouping structure.*

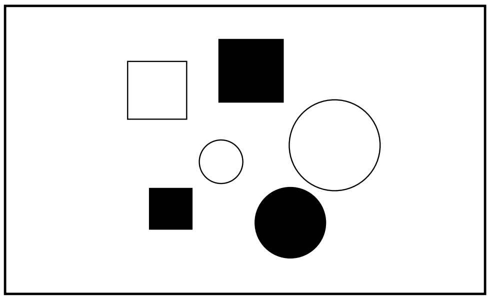


Figure 3.20: *The color attribute dominates the grouping structure.*

In order to integrate the effect of the dominance ordering of attributes into the definition of information complexity, we follow the original definition of information complexity: the number of occurrences of primitive elements (and units) in the gestalt description of patterns. Since in our approach the primitive elements are represented in terms of attribute values, the information complexity should be based on the number of attribute values in the gestalt description of visual patterns. In the original definition of the information complexity in SIT, each primitive element is assumed to have the same information complexity. However, attribute values do not have the same information complexity since different attributes have different dominance effect. In this way, it is assumed that a reduction in the number of position values decreases the information complexity more than the same reduction of shape values. The information complexity of values of each attribute should then be determined relative to each other and through empirical studies.

The information complexity of n-tuples of transformations depends on the information complexities of individual transformations. In order to define the information complexity of individual transformations, we consider abstract and concrete transformations separately. In the case of concrete transformations, it is reasonable to assume that the information complexity is determined by the number of attribute values that constitute a transformation. For example, suppose that the perceptual dominance ordering of visual attributes implies that the information complexity of a shape attribute value is 1. Then, the information complexity of the concrete non-parameterized shape transformation
$\{(rectangle, circle), (square, triangle), (triangle, line)\}$
would be 6 and the information complexity of the concrete parameterized shape transformation
$\{(1, rectangle, rectangle), (2, rectangle, circle), (3, rectangle, square)\}$
would be 4 (the second argument is a constant value and thus counted only once). Of course, this way of measuring information complexity is based on our intuition and should be verified by empirical studies.
However, it is not trivial to determine the information complexity of abstract transformations. Although for some classes of abstract transformations, like those for which the relation between attribute values is expressed by polynomials, it may be intuitive to define the information complexity on the basis of the lengths and the degrees of the polynomials, we believe that a perceptually motivated information complexity for abstract transformations should be established through empirical studies.

Following the original definition of the information complexity and considering the additional complexities of attribute values and transformations for the E-VREG expressions, the information complexity of E-VREG expressions can be defined as follows.

16. DEFINITION. *Let v_{att_j} be a value of attribute att_j, t_{att_j} be a transformation defined on values of attribute att_j, A and A_j be E-VREG expressions, $A[x_1, \ldots, x_m]$ be an E-VREG expression that contains transformation variables $x_1, \ldots, x_m$, $\omega^{(i)}, \tau_1^{(i)}, \ldots, \tau_m^{(i)} \in \Omega, \psi \in \Psi,$, and $t \in \Omega \cup \Psi$, then the information complexity C of the E-VREG expressions can recursively be defined as follows:*

$C(\ < v_{att_1}, \ldots, v_{att_m} >\ ) = \sum_{j=1}^{m}\ C(v_{att_j})$
$C(\ < t_{att_1}, \ldots, t_{att_m} >\ ) = \sum_{j=1}^{m}\ C(t_{att_j})$
$C(\ t(A)\ ) = C(A) + C(t)$
$C(\ Unit(A)\ ) = C(A) + 1$
$C(\ Comp(A_1, \ldots, A_m)\ ) = \sum_{j=1}^{m} C(A_j)$
$C(\ Sym(A, \psi)\ ) = C(A) + C(\psi)$
$C(\ Iter(A, \omega^{(i)}, n)\ ) = C(A) + C(\omega^{(i)})$
$C(\ \lambda x_1, \ldots, x_m\ Iter(A[x_1, \ldots, x_m], \omega^{(i)}, n)\ (\tau_1^{(i)}, \ldots, \tau_m^{(i)})\ ) =$
$\quad C(A) + C(\omega^{(i)}) + \sum_{j=1}^{m} C(\tau_j^{(i)})$

Finally, it is often claimed that there is a tendency to prefer a perceptual grouping which consists of a small number of perceptual groups [Ima66]. We claim that this aspect of perceptual preference will be automatically achieved by applying the simplicity principle to all possible perceptual groupings, i.e. the preferred perceptual grouping is the one that has the lowest information complexity. To argue our claim note that a lower number of perceptual groups means placing more elements into each perceptual group and vice versa. Since placing more elements in one perceptual group implies more reduction of information complexity, the lowest number of perceptual groups implies the lowest information complexity and vice versa.

In this chapter, we have introduced formal languages the expressions of which represent gestalts of two-dimensional visual patterns. In the next chapter, we will discuss how the preferred gestalts of two-dimensional visual patterns can be computed.

Chapter 4

Computing Gestalts of Visual Patterns

In the previous chapter, we have introduced various languages: some of these languages consist of expressions that represent the gestalts of one-dimensional string patterns while other languages consist of expressions that represent the gestalts of two-dimensional visual patterns. For each language, we have defined the denoting relation between its expressions and the perceptual patterns they refer to, i.e. the denoting relation between perceptual patterns and their gestalts. We explained how the gestalts of string patterns can be computed by using the computational model introduced by Van der Helm and Leeuwenberg [VdHL86].

In this chapter, we investigate how the gestalts of a two-dimensional visual pattern can be computed. In order to compute the gestalts of a two-dimensional visual pattern, we start with the unstructured (initial) representation of that visual pattern and proceed with analyzing and structuring that representation in successive steps. The unstructured representation of a visual pattern is taken to be the set of representations of primitive visual elements that constitute that visual pattern. At each successive step, an unstructured part (a subset) of the pattern representation is coded in terms of one (E-)VREG (i.e. VREG or E-VERG) expression. The resulting (E-)VREG expression is a perceptually motivated structured representation of the coded pattern part. A gestalt of a visual pattern is then generated when the representation of the whole visual pattern is coded by one single (E-)VREG expression. The process of analyzing an unstructured representation of a visual pattern as a perceptually motivated structured representation is a parsing process. Thus, the parsing process consists of successive steps each of which replaces the unstructured representation of a pattern part by a structured representation of it. A partially structured representation of a visual pattern is called a *parse state*. A parse state is transformed into the next parse state by means of a *coding rule*. When a parse state contains only one (E-)VREG expression the parsing process terminates. The resulting (E-)VREG expression represents a gestalt of the visual pattern that was parsed.

In the next section, we define a parse state as a set of (E-)VREG expressions. Then, we introduce coding rules that are motivated by the (E-)VREG expressions and explain their application to parse states. The computational complexity of the parsing process is briefly discussed.
In section 2, hierarchical sets are introduced to represent parse states such that the hierarchical unit structure of visual patterns is reflected by the hierarchical structure of their parse states. An additional coding rule is proposed that can be applied to the hierarchical parse states. The application of this rule to hierarchical parse states generates the unit expression which is necessary to cover non-mirror symmetrical patterns.
In previous chapters, we explained that the (E-)VREG expressions ignore the proximity factor. In section 3 of this chapter, the notion of a hierarchical set is extended to integrate the proximity structure of visual patterns into their initial parse states. To do this, we first compute the proximity structure of unstructured visual patterns. The proximity structure of a visual pattern is a hierarchical clustering of its constitutive visual elements. Based on the hierarchical clustering of a visual pattern, its initial hierarchical parse state is constructed such that the hierarchical proximity structure of the visual pattern is reflected by the hierarchical structure of its initial parse state. Then, an additional coding rule is introduced to cover the effects of the hierarchical proximity structure of parse states. This and other coding rules can then be applied to hierarchical parse states in a bottom-up fashion. The resulting parse states provide (E-)VREG expressions that represent gestalts of visual patterns such that the influence of simplicity as well as proximity is taken into account.

4.1 Coding Rules for (E-)VREG Expressions

A two-dimensional visual pattern is represented as a set of tuples of visual attribute values. Each tuple of visual attribute values represents a primitive visual element [1].

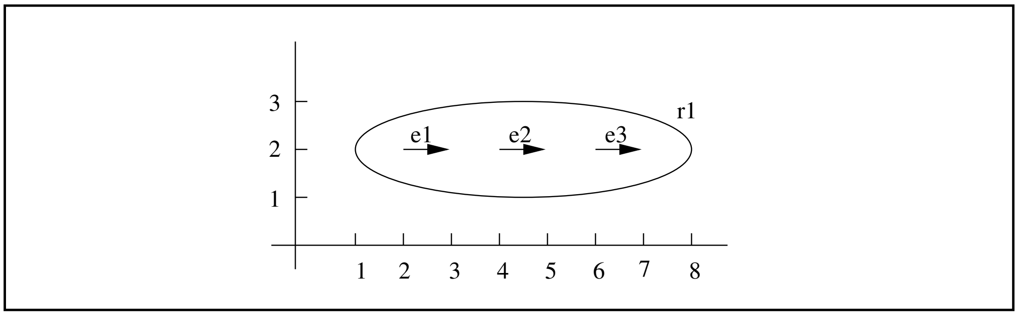


Figure 4.1: *A visual pattern consisting of three arrows and an oval.*

Thus, given $< a^i_{pos}, a^i_{shape}, \ldots >$ as the representation of a primitive visual element e_i, a visual pattern can be represented as the set $\{e_i \mid e_i =< a^i_{pos}, a^i_{shape}, \ldots >\}$. The set of representations of primitive visual elements that constitute a visual pattern is the unstructured representation of that visual pattern. It is an unstructured representation since the constituent relation between constitutive visual elements (gestalt structure) are not represented.

For example, consider the two-dimensional visual pattern illustrated in Figure 4.1. This pattern consists of four primitive visual elements, i.e. three arrows and one oval. In this visual pattern, the labels attached to primitive visual elements (i.e. e_1, e_2, e_3, r_1) are just used to refer to them and they are not parts or elements of the visual pattern. Assuming a canonical way to represent primitive visual elements uniquely (i.e. an arrow by start and end points, an oval by left-middle and lower-middle points, etc.) and considering only the position and the shape attributes, the unstructured representation of this visual pattern is the following set:

$$
\begin{array}{llllll}
\{ & < & < <2,2>, <3,2> & > & , \ arrow & >, \\
 & < & < <4,2>, <5,2> & > & , \ arrow & >, \\
 & < & < <6,2>, <7,2> & > & , \ arrow & >, \\
 & < & < <1,2>, <4.5,1> & > & , \ oval & > \}.
\end{array}
$$

In order to parse a visual pattern and compute its gestalts, we consider

[1]There is no pre-defined set of primitive visual elements assumed. In fact, primitive visual elements can be defined with respect to application domains. For example, one may define line segments as primitive elements or one may define points as primitive elements.

the representation of that visual pattern as its initial parse state. In general, we consider a parse state as a set of (E-)VREG expressions. Note that an initial parse state is a set of primitive (E-)VREG expressions, each of which represents a primitive visual element. An initial parse state is a state that does not represent any structure of the pattern that is to be parsed, i.e. a set of tuples of visual attribute values. In contrast, the final parse state is a state that represents a gestalt of the visual pattern that was parsed, i.e. a set that contains one (E-)VREG expression. The intermediate parse states are then partially structured representations of visual patterns, i.e. a set of (E-)VREG expressions each of which represents a pattern part.

A parse state is transformed into a subsequent parse state by coding (representing) a subset of (E-)VREG expressions from the first parse state by one single (E-)VREG expression. The transformation of one parse state into another is accomplished by a coding rule. Thus, each coding rule is designed to replace a subset of (E-)VREG expressions of a parse state, which represents a non-structured subset of visual elements, with a structured representation of that subset which provides a structured description of that pattern part.
Different coding rules may be applied to each parse state. The application of different coding rules to parse states results in different final parse states which consequently assign different gestalts to the visual pattern. The preferred gestalt is selected by applying the minimum principle to the set of final states.

There are constraints associated with each coding rule. These constraints determine the applicability of the coding rule to a particular subset of the (E-)VREG expressions of a parse state. A coding rule can be applied to a subset of (E-)VREG expressions if the constraints associated with that rule are satisfied by the (E-)VREG expressions contained in that subset. The application of a coding rule to a subset of (E-)VREG expressions generates one (E-)VREG expression which represents the same pattern part as the original subset of (E-)VREG expressions does. Note that the application of coding rules to successive parse states results in a hierarchical structuring of (E-)VREG expressions.

4.1.1 Coding Rules for VREG Expressions

We introduce three coding rules that construct the coding of the structures covered by the VREG language, i.e. three coding rules that generate the expressions of the VREG language. These expressions represent the special case of iteration, the symmetry, and the composition structures. Then, in the next subsection we discuss the coding rules for the E-VREG (Extended VREG) language. Since the E-VREG language is the same as the VREG language except that it covers the additional general case of iteration structure, we will introduce only one additional coding rule that constructs the coding of the general case of iteration structure. In these two subsections, we do not account for the unit structure such that the symmetry rule can cover only the mirror symmetry structure. In section 2 of this chapter, we will discuss how the non-mirror symmetry structures can be covered by integrating the unit structure of visual patterns into their representations.

17. DEFINITION. *Let A be a parse state, $a_1, \ldots, a_n$ be VREG expressions, M be a model for the VREG expressions, Ψ be the set of reflection transformations, Ω be the set of parameterized transformations, and $n \in \mathbf{N}$. The coding rules are defined as follows:*

1) Iteration Rule:

For any $a_1, \ldots, a_n \in A \;\&\; \omega \in \Omega$
IF
$$\forall_{i=1,\ldots,n} \; |[a_i]|_M = |[\omega^{(i)}(a_1)]|_M$$
THEN
$$A \longrightarrow A \setminus \{a_1, \ldots, a_n\} \;\cup\; \{Iter(a_1, \omega, n)\}$$

2) Symmetry Rule:

For any $a_1, a_2 \in A \;\&\; \psi \in \Psi$
IF
$$|[a_1]|_M = |[\psi(a_2)]|_M$$
THEN
$$A \longrightarrow A \setminus \{a_1, a_2\} \;\cup\; \{Sym(a_1, \psi)\}$$

3) Composition Rule:

For any $a_1, a_2 \in A$
$A \longrightarrow A \setminus \{a_1, a_2\} \ \cup \ \{Comp(a_1, a_2)\}$

Each of these coding rules transforms the parse state A, which contains a certain subset of VREG expressions, into another state in which that subset is replaced by one VREG expression. Therefore, the application of these coding rules to a parse state reduces the number of VREG expressions such that successive parse states contain fewer VREG expressions.

4.1.2 Coding Rule for E-VREG Expressions

In the above definition, the first coding rule constructs the coding of iteration structures. This iteration rule is applicable to a sequence (ordered subset) of VREG expressions (i.e. $< a_1, \ldots, a_n >$) if the visual patterns represented by these VREG expressions (i.e. $|[a_1]|_M, \ldots, |[a_n]|_M$) can be described in terms of the visual pattern which is represented by a_1 and a parameterized transformation ω (i.e. $|[\omega^{(i)}(a_1)]|_M$). The transformation ω is used to transform a visual pattern as a whole such that its internal structure is not changed at iterative steps. Therefore, this iteration rule constructs only the coding of the special case of iteration structures (S-structure patterns), as they are introduced in the previous chapter.

In this section, we consider the general case of iteration structure (G-structures), and introduce a coding rule for constructing them. The constructed coding rule for the general case of iteration structure together with the coding rules for the VREG expressions, as introduced in previous subsection, can be considered as the coding rules for the expressions of the E-VREG language. Accordingly, we will use the term E-VREG expressions to mean VREG expressions plus the expression that represents the G-structure visual patterns.
A visual pattern with the G-structure can be described as a subpattern which is successively transformed by local and global parameterized transformations. The local transformations are then applied only to some parts of the subpattern such that the internal structure of the successive transformed subpatterns changes. A visual pattern that has the general case of iteration structure is illustrated in Figure 4.2.

Consequently, a sequence of visual patterns $p_1, \ldots, p_n$ forms a G-structure pattern when they can be described in terms of p_1, a global transformation, and one or more local transformations. The global transformation should be applied to p_1 as it was the case with the S-structure patterns, i.e. the transformation should be applied to p_1 as a whole. Unlike the global transformation, the local transformations should be applied to some parts of p_1. Since the representation of a visual pattern is a hierarchical E-VREG expression, the local transformations are applied to some of its embedded sub-expressions.

For example, consider a visual pattern with the G-structure as illustrated in Figure 4.2. The initial parse state of this visual pattern is a set that contains twelve primitive E-VREG expressions each of which represents either an arrow or an oval. The initial parse state can then be parsed by the proposed coding rules through which successive parse states result. One possible parse results in a parse state that contains three E-VREG expressions. Each of these E-VREG expressions represents a vertical chunk of three arrows and an oval.

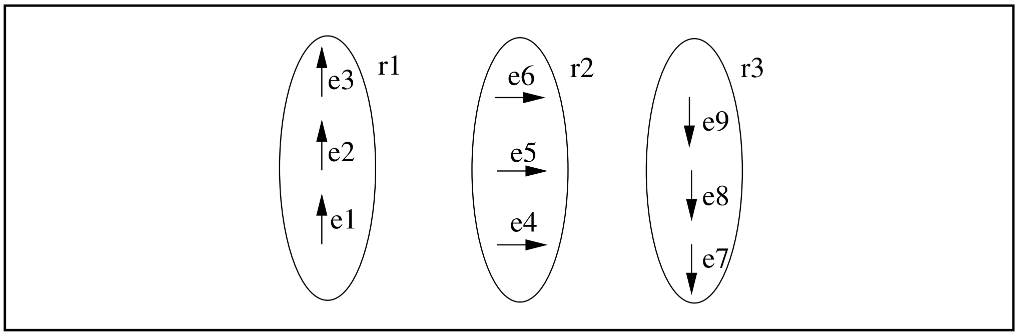


Figure 4.2: *A G-structure pattern.*

This parse state, called D, may be represented as the following set:

$$\begin{aligned} D = \{ \; & Comp(r_1, Iter(e_1, translate_y, 3)), \\ & Comp(r_2, Iter(e_4, translate_y, 3)), \\ & Comp(r_3, Iter(e_7, translate_y, 3)) \; \} \end{aligned}$$

Note that r_i and e_i are n-tuples of attribute values that represent primitive visual elements. These three E-VREG expressions represent three visual patterns that together form a G-structure pattern since they can be represented in terms of the first E-VREG expression, a global parameterized translation in the direction of the x-axis (ω transformation), and a local parameterized 90° s-rotation (τ transformation). The local transformation τ should be applied to an embedded E-VREG expression within the first E-VREG expression. The three successive applications of the local and the global transformations to the first E-VREG expression are as follows:

$\omega^{(1)}(Comp(r_1, Iter(\tau^{(1)}(e_1), translate_y, 3)))$,
$\omega^{(2)}(Comp(r_1, Iter(\tau^{(2)}(e_1), translate_y, 3)))$,
$\omega^{(3)}(Comp(r_1, Iter(\tau^{(3)}(e_1), translate_y, 3)))$.

These E-VREG expressions represent visual patterns that are identical to those represented by the three E-VREG expressions in the parse state D.

In order to determine whether a sequence of E-VREG expressions $a_1, \ldots, a_n$ represents a G-structure visual pattern, the E-VREG expression a_1 should be modified by inserting one or more transformation variables in it. These transformation variables are applied to some embedded expressions of a_1 and stand for the local parameterized transformations. A modification of an E-VREG expression by inserting transformation variables in it results in an abstract E-VREG expression. We define the *Insert* function which maps a E-VREG expression to a set of abstract E-VREG expressions.

18. DEFINITION. *Given a hierarchical E-VREG expression "a" consisting of k embedded expressions, the insert function generates a set of abstract E-VREG expressions. Each generated abstract E-VREG expression is identical to the E-VREG expression "a" except that l $(1 \leq l \leq k)$ embedded E-VREG expressions $e_1, \ldots, e_l$ in "a" are replaced by $x_1(e_1), \ldots, x_l(e_l)$, where $x_1, \ldots, x_l$ are transformation variables that are applied to the arguments $e_1, \ldots, e_l$. An embedded E-VREG expression can be replaced only once and become the argument of only one transformation variable.*

Note that an embedded E-VREG expression e_i can be embedded in another embedded E-VREG expression e_j.
Another way to look at the insert function is by considering the tree structure of the E-VREG expressions. Given the tree structure of a E-VREG expression "a", the insert function may add a node to the root of a subtree of the tree structure of "a". A node represents a transformation variable that is applied to the subtree below it.

The above definition of the insert function implies that the maximum number of embedded E-VREG expressions in "a" determines the maximum

number of transformation variables that can be inserted in the E-VREG expression "a". Therefore, for a E-VREG expression "a", the number of non-trivially distinct abstract E-VREG expressions that are generated by the insert function (i.e. the cardinality of the output set of the *Insert* function) depends on the number of embedded E-VREG expressions that can be identified in "a". In particular, for a set of k embedded E-VREG expressions the number of non-trivially distinct abstract E-VREG expressions that are resulted by applying l ($l \leq k$) transformation variables to l different embedded expressions is the same as the number of different sets that contain l embedded E-VREG expressions. Thus, given k embedded expressions, the insert function generates $\sum_{l=1}^{k} \binom{k}{l} = \sum_{l=1}^{k} \frac{k!}{l!(k-l)!}$ non-trivially distinct abstract E-VREG expressions. Each component of this sum determines the number of distinct abstract E-VREG expressions that are resulted by inserting l transformation variables.
For instance, given the E-VREG expression
$a = Comp(r_1, Iter(e_1, translate_y, 3))$, consisting of three embedded E-VREG expressions (i.e. r_1, $Iter(e_1, translate_y, 3)$, and e_1), the insert function generates $\sum_{l=1}^{3} \binom{3}{l} = \sum_{l=1}^{3} \frac{3!}{l!(3-l)!} = 7$ abstract E-VREG expressions. The following are possible abstract E-VREG expressions that contain one transformation variable x.

$$\begin{aligned}
a'_1[x] &= Comp(x(r_1), Iter(e_1, translate_y, 3)) \\
a'_2[x] &= Comp(r_1, x(Iter(e_1, translate_y, 3))) \\
a'_3[x] &= Comp(r_1, Iter(x(e_1), translate_y, 3))
\end{aligned}$$

Following the coding rule for the special case of iteration structures as defined above, we introduce a rule that constructs the coding for the general case of iteration structures.

19. DEFINITION. *Let A be a parse state, $a_1, \ldots, a_n$ be E-VREG expressions each consists of k embedded expressions, M be a model for the E-VREG expressions, $g : VAR_t \to D_t$ be an assignment function that assigns transformations from D_t to transformation variables from VAR_t, Ω be the set of parameterized transformations, $n \in \mathbf{N}$, and $Insert(a_1) = \{a'[x_1, \ldots, x_l] \mid 1 \leq l \leq k\}$ as defined above. The rule that constructs the coding for the general case of iteration structures is defined as follows:*

G-iteration Rule:

For any $a_1, \ldots, a_n \in A$ & $a'[x_1, \ldots, x_l] \in Insert(a_1)$ & $\omega, \tau_1, \ldots, \tau_l \in \Omega$
IF
$\quad \forall_{i=1,\ldots,n}\ |[a_i]|_M = |[\omega^{(i)}(a'[x_1, \ldots, x_l])]|_{M,g[x_1/\tau_1^{(i)},\ldots,x_l/\tau_l^{(i)}]}$
THEN
$\quad A \longrightarrow A \backslash \{a_1, \ldots, a_n\} \cup \{(\lambda x_1, \ldots, x_l\ Iter(a'[x_1, \ldots, x_l], \omega, n))\ (\tau_1, \ldots, \tau_l)\ \}$

In summary, the representation of a visual pattern, considered as its initial parse state, may be parsed according to the coding rules proposed. During each step of the parsing process, a single E-VREG expression will replace a subset of E-VREG expressions. The single E-VREG expression specifies a perceptual structure for the pattern part that it covers. When a parse state is achieved which contains only one E-VREG expression, a perceptual structure of the whole pattern has been generated. Of course, given a pattern, there may be several parses for that pattern. The parse that provides the preferred perceptual structure of the original pattern is selected by applying the simplicity principle to all parses of that pattern, i.e. the parse with the minimum information load provides the preferred perceptual structure of the pattern.

4.1.3 The Complexity of the Parsing Process

Unlike most linguistic grammars, the proposed coding rules do not involve a specific non-terminal, like the Start symbol, that determines the success of a parse process. A successful parse process is one that ends up with a parse state that contains one single E-VREG expression. The application of the coding rules to an initial parse state provides always a parse since it is always possible to apply repeatedly the compose rule and construct a parse.
A gestalt of a visual pattern is represented by the E-VREG expression that is constructed out of its initial parse state by applying the coding rules to that parse state. The preferred gestalt of a visual pattern is determined by first constructing all proper [2] parses from its initial parse state and then applying the minimum principle to these parses.

[2] As we will see in a short while, we will not construct all possible parses, but only those that result in reasonable information complexities.

The computational complexity of the process which determines the preferred gestalt of a visual pattern is therefore the computational complexity of constructing all proper parses from the initial parse state of that visual pattern plus the computational complexity of selecting the preferred one. The computational complexity of selecting the preferred parse is linear in the number of all proper parses since we have to check the information complexities of proper parses and select the one which has the lowest information complexity. However, the number of proper parses for a parse state depends on the number of proper codings at successive parse steps. Subsequently, the number of proper codings for a parse state is determined by the number of possible subsets of E-VREG expressions of that parse state and the number of coding rules that can be applied to them. In the following, we will consider the computational complexity for each coding rule separately.

S-Iteration Rule

The iteration rule for the special case of iteration structure can be applied to all sequences (ordered sets) that consist of 2 or more E-VREG expressions. Given a parse state that contains n E-VREG expressions, there exists $\sum_{m=2}^{n} \frac{n!}{(n-m)!}$ possible sequences of E-VREG expressions to which the iteration rule can be applied.

In order to apply the iteration rule to a sequence of m E-VREG expressions, a parameterized transformation must be chosen that transforms the first E-VREG expression of the sequence to all other $m-1$ E-VREG expressions of that sequence. Given t pre-defined parameterized transformations, there is in principle an infinite number of compound parameterized transformations possible. However, not every compound parameterized transformation is a proper candidate since we are not interested in those transformations for which the information complexity is higher than a concrete transformation, i.e. a set of pairs that assign attribute values to parameter values.
For example, consider the visual pattern illustrated in Figure 4.3. Let the primitive visual elements be represented by the position (middle point position with a fixed orientation) and the shape values.

A possible parse state for this visual pattern contains three E-VREG expressions:

$\{\ Comp(e_1, e_2), Comp(e_3, e_4), Comp(e_5, e_6)\ \}$.

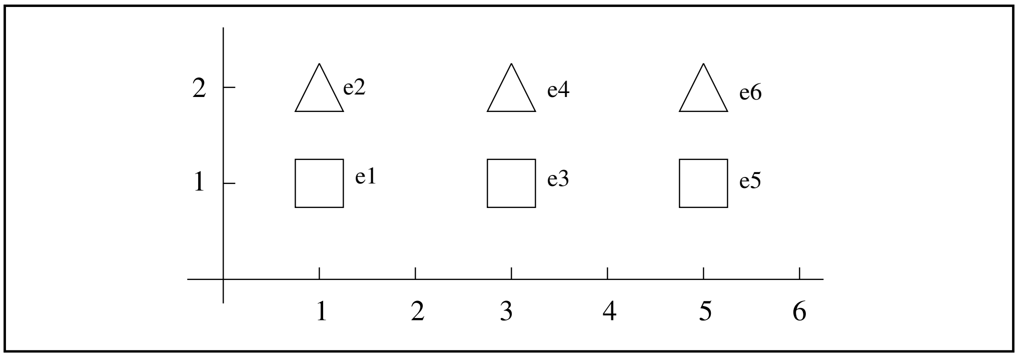


Figure 4.3: *A visual pattern with the iteration structure.*

Suppose that we have the sequence $< comp(e_1, e_2), comp(e_3, e_4), comp(e_5, e_6) >$ to which the iteration rule should be applied. Then, we should decide on a parameterized transformation that transforms the visual pattern represented by the first E-VREG expression to the visual patterns represented by the second and the third E-VREG expressions. The following concrete parameterized transformation is a possible transformation for the iteration rule:

$$\begin{aligned}\{ \quad & (1, <1,1>, <1,1>), (1, <1,2>, <1,2>), \\ & (2, <1,1>, <3,1>), (2, <1,2>, <3,2>), \\ & (3, <1,1>, <5,1>), (3, <1,2>, <5,2>) \ \}\end{aligned}$$

Since the iteration rule can always be applied to every sequence of E-VREG expressions by means of a concrete transformation, the information complexity of the concrete transformation for the iteration rule determines the maximum (upper-bound) information complexity that an alternative transformation may have. Given a sequence $< a_1, \ldots, a_n >$ to which the iteration rule should be applied, the information complexity S of such a concrete transformation depends on the information complexities of $a_1, \ldots, a_n$, i.e. the upper-bound information complexity S for an alternative transformation would be defined as:

$S \leq \sum_{i=1}^{n} C(a_i)$, where $C(a_i)$ is the information complexity of a_i.

Moreover, we will use the fact that the information complexity of a compound transformation is greater than the information complexity of the constitutive transformations, i.e. $C(t_1 \circ t_2) \geq C(t_1) + C(t_2)$ for any trans-

formations t_1 and t_2.
Finally, we assume that each pre-defined transformation except the identity transformation should have an information complexity greater than zero. The identity transformation does not play a role in the construction of compound transformations since for any transformation $t\ :\ id \circ t = t \circ id = t$.

Now, let $t_1, \ldots, t_p$ be the pre-defined transformations where $C(t_i) \geq 0$ for any $t_i \neq id$ and S be the upper-bound information complexity for a sequence of E-VREG expressions. Then, given ϵ as the lowest information complexity that a pre-defined transformation may have, any composition of $r = S/\epsilon$ transformations have an information complexity greater than S (the maximum information complexity that a compound transformation may have). This implies that for p pre-defined transformations a compound transformation can be constructed from a combination of at most r transformations. Therefore, given p pre-defined transformations $t_1, \ldots, t_p$, there exists at most $T = \sum_{i=1}^{r} p^i$ possible transformation compositions that provide compound transformations with information complexities lower than S. This implies that for a finite number of pre-defined parameterized transformations the application of the iteration rule to a sequence of E-VREG expressions can be decided in a deterministic way.

Let $t_1, \ldots, t_p$ be pre-defined transformations, Then, given a parse state that contains n E-VREG expressions $a_1, \ldots, a_n$ and given T number of possible (pre-defined and compound) parameterized transformations as determined above, the computational complexity of the iteration rule for the special case of iteration structure is determined as follows:

$O(\sum_{m=2}^{n} \frac{n!}{(n-m)!} \times T)$

Symmetry Rule

Given a parse state that contains n E-VREG expressions, the symmetry rule can be applied to all its subsets that have cardinality 2. This results in $\binom{n}{2}$ possible pairs of E-VREG expressions to which the symmetry rule can be applied.

In order to apply the symmetry rule to a pair of E-VREG expressions, a reflection transformation should be chosen. The reflection transformation for the position attribute can be decided by computing a reflection axis. A reflection axis can be easily computed by taking the positions of mirroring

primitive visual elements that are represented by the pair of E-VREG expressions and decide the middle line between them. The mirroring visual elements can easily be found by assuming that two E-VREG expressions form a symmetry structure if they are structurally similar expressions. The structural similarity of E-VREG expressions determines which primitive visual elements should be mirror images of each other such that it is easy to compute the reflection axis for the given two E-VREG expressions.
The transformation for non-positional attributes may be either from a pre-defined set of transformations or a composition of them. However, for a finite set of pre-defined transformations there is an infinite number of compound transformations possible. But, not every transformation is a proper candidate since we are not interested in those transformations for which the information complexity is higher than a concrete transformation, i.e. a transformation that assigns attribute values of visual elements from one symmetry half to values of the same attributes of visual elements from the second symmetry half. Following the same argumentation as for the S-Iteration coding rule, the number of proper transformations depends on the pair of E-VREG expressions and will be a finite number.

Let T be the maximum number of proper transformations that may exists among possible pairs of E-VREG expressions taken from $a_1, \ldots, a_n$ and p pre-defined transformations as determined for the S-Iteration coding rule. The computational complexity of the symmetry rule is determined as follows:

$$O\left(\binom{n}{2} \times T\right)$$

Composition Rule

Given a parse state that contains n E-VREG expressions, the compose rule can be applied to all its subsets that have cardinality 2 to n. Therefore, the computational complexity of the composition rule is determined as follows:

$$O\left(\sum_{m=2}^{n} \binom{n}{k}\right)$$

G-iteration Rule

Like the iteration rule for the special case of iteration structure, the iteration rule for the general case of iteration structure can be applied to all sequences that consist of 2 or more E-VREG expressions. Thus, given a parse state that contains n E-VREG expressions, there are $\sum_{m=2}^{n} \frac{n!}{(n-m)!}$ possible sequences of E-VREG expressions to which the iteration rule for the general case of iteration structures can be applied. In order to apply the general case of iteration rule to a sequence E-VREG expressions, one global parameterized transformation and one or more local parameterized transformations should be decided on.

The computational complexity of deciding on the global transformation depends, like for the special case of iteration rule, on the maximum number of proper transformations T which is determined by the E-VREG expressions $a_1, \ldots, a_n$ of a given parse state and p pre-defined parameterized transformations $t_1, \ldots, t_p$.

Before we consider the computational complexity of deciding on local transformations, note that the visual patterns represented by the sequence of E-VREG expressions $a_1, \ldots, a_n$ differ from each other in terms of their internal organizations. The successive insertion of local transformations in certain embedded expressions in a_1 is supposed to generate E-VREG expressions that represent visual patterns which are identical to those represented by $a_1, \ldots, a_n$.
The computational complexity of deciding on local transformations depends on two factors. Given a sequence $< a_1, \ldots, a_n >$ of E-VREG expressions, the first factor depends on the number of local transformations and the determination of the embedded expressions in a_1 to which the local transformations must be applied. The second factor depends on the number of local transformations that can be decided.
As noted in the previous section, the insert function generates for a E-VREG expression a_i all possible abstract E-VREG expressions each of which applies a number of local transformations to the same number of embedded expressions in a_i. Thus, given a sequence $< a_1, \ldots, a_n >$ the first factor of computational complexity for deciding local transformations is determined by the number of abstract E-VREG expressions that the insert function generates for the E-VREG expression a_1. Given k as the maximum number of embedded expressions in a_1, we argued in the previous section that the number of possible non-trivially distinct abstract E-VREG expressions for a_1 is determined as follows:

$$\sum_{l=1}^{k} \binom{k}{l} = \sum_{l=1}^{k} \frac{k!}{l!(k-l)!}$$

Thus, for a given sequence $< a_1, \ldots, a_n >$ the first factor of computational complexity for deciding local transformations is determined by this formula.
The second factor of computational complexity for deciding local transformations is determined by the number of local transformations that can be decided. Given an abstract E-VREG expression containing v transformation variables, we need to decide v local parameterized transformations each of which should be assigned to one transformation variable. Note that all variables are transformation variables (and not variables standing for elements) such that the assignment of any arbitrary local transformations result in E-VREG expressions that represent visual patterns with the same number of constitutive primitive visual elements. This implies that there is always a concrete transformation possible which transforms one E-VREG expression into another one by assigning attribute values of the involved visual elements to parameter values.
Like the transformations for the special case of iteration rule, we consider the information complexity of concrete local transformations for n E-VREG expressions as the maximum information complexity that alternative local transformations may have. Since the information complexities of compound parameterized transformations depends on the information complexities of the composed pre-defined parameterized transformations and for t pre-defined parameterized transformations, there are a finite number of compound transformations for which the information complexity is lower than the maximum information complexity.

Let GT be the maximum number of local parameterized transformations (pre-defined or compound) for n E-VREG expressions $a_1, \ldots, a_n$ and p pre-defined parameterized transformations. Then, given a parse state that contains n E-VREG expressions $a_1, \ldots, a_n$ and given k as the maximum number of embedded expressions in one of the E-VREG expressions $a_1, \ldots, a_n$, the computational complexity of the iteration rule for the general case of iteration structure is determined as follows:

$$O\left(\sum_{m=2}^{n} \frac{n!}{(n-m)!} \times T \times \sum_{l=1}^{k} \left(\begin{smallmatrix} k \\ l \end{smallmatrix} \right) \times GT\right)$$

Note that for a finite number of pre-defined transformations the application of all coding rules can be decided in a deterministic way.

4.2 Computing Unit-based Gestalts

The proposed coding rule for symmetry structure can only be applied to two E-VREG expressions which represent visual patterns that are mirror images of each other. In other words, the proposed approach does not account for non-mirror symmetrical patterns. An example of non-mirror symmetrical pattern is illustrated in Figure 4.4.
In order to cover non-mirror symmetrical patterns we introduce the notion of *units* of visual elements into this approach. A unit of visual elements is either a primitive visual element or a group of primitive visual elements for which the internal position structure should be preserved under the reflection transformation. Recall that the unit structure of visual patterns is a hierarchical structure.

In order to construct the coding of the unit structure of visual patterns we modify the notion of parse states such that units of visual elements can be explicitly represented. (This modification may be compared to the preprocessing where parenthesis are introduced to indicate the units of pattern constituents, as defined in section 1 of chapter 3).
The modified parse states are *hierarchical sets*. In hierarchical sets, each element may be either a E-VREG expression or a hierarchical set. An element which is a E-VREG expression will be called a *terminal element*. An element which is a hierarchical set is then called a *non-terminal element*. In this way, a parse state, which is a hierarchical set, is itself considered as a non-terminal element.
Thus, the unit-based initial parse state of a visual pattern is a hierarchical set in which the hierarchical unit structure of the visual pattern is reflected by the hierarchical structure of the set. In the initial hierarchical parse state, a non-terminal element at a certain hierarchical level corresponds with a unit of visual elements at the corresponding level of the hierarchical unit structure of the visual pattern. It is important to note that the unit structure of visual patterns is assumed to be given, i.e. the parsing process starts with pre-structured initial parse states in which the hierarchical unit structure is reflected by the hierarchical structure of the initial parse state.
For example, consider the visual pattern illustrated in Figure 4.4. In this visual pattern the dashed boxes indicate the units of visual elements. The initial parse state for this visual pattern is therefore the hierarchical set $\{a, b, \{c, d\}, \{e, f\}, g, h\}$. Note that the embedded sets like $\{c, d\}$ and $\{e, f\}$ contain elements which represent united visual elements.

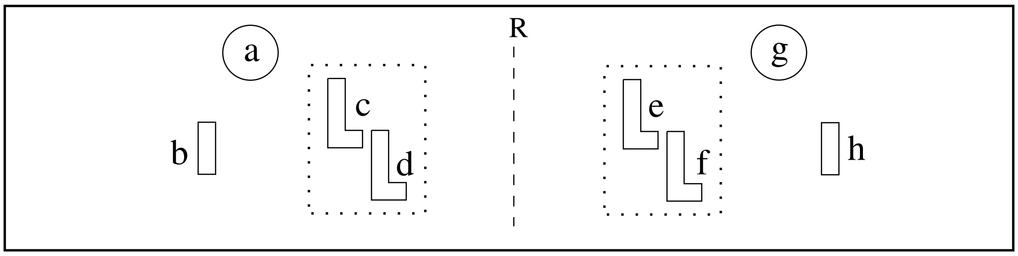


Figure 4.4: *A unit-based visual pattern.*

4.2.1 Parsing Unit-based hierarchical Representations

In order to compute the unit-based gestalts of a two-dimensional visual pattern its initial hierarchical parse state should be parsed. Since non-terminal elements are sets (not E-VREG expressions), the proposed coding rules should be applied to the lowest level embedded sets that contain merely terminal elements (E-VREG expressions). This suggests a bottom-up parsing process such that the embedded sets that contain E-VREG expressions are parsed before higher level sets that contain non-terminal elements are parsed.

The result of parsing the E-VREG expressions of an embedded set which represent the visual elements that belong to one unit is a single E-VREG expression. This single E-VREG expression represents a gestalt of the visual elements that belong to that unit. The embedded set containing one E-VREG expression is a singleton set which represents a unit structure. In order to construct the coding for the unit structure, we introduce a coding rule which replaces the singleton set by a unit expression the argument of which is the E-VREG expression that is contained in the singleton set.

20. Definition. *Let A be an embedded set (at a certain hierarchical level of a parse state) and "a" be a E-VREG expression. The coding rule which constructs the coding of the unit structure is then defined as follows:*

Unit Rule:

For any $\{a\} \in A$
$$A \longrightarrow A \setminus \{\{a\}\} \cup \{Unit(a)\}$$

The unit rule together with the proposed coding rules for E-VREG expressions can be used to parse a hierarchical unit-based representation in a bottom-up fashion.
For example, consider again the visual pattern illustrated in Figure 4.4 for which the initial (hierarchical) parse state is $\{a, b, \{c, d\}, \{e, f\}, g, h\}$. One possible gestalt of this visual pattern is provided by first applying the compose rule to the lowest level embedded sets, i.e. $\{c, d\}$ and $\{e, f\}$. This results in two singleton sets $\{Comp(c, d)\}$ and $\{Comp(e, f)\}$. Then, the unit rule can be applied to the resulting singleton sets which generates the E-VREG expressions $Unit(Comp(c, d))$ and $Unit(Comp(e, f))$. Subsequently, the compose rule can be applied twice which generates two E-VREG expressions $Comp(a, b, Unit(Comp(c, d)))$ and $Comp(g, h, Unit(Comp(e, f)))$. Finally, the symmetry rule can be applied to these E-VREG expressions which generates a gestalt of the visual pattern, i.e.

$Sym(Comp(a, b, Unit(Comp(c, d))), \psi_R)$.

Note that this E-VREG expression represents a gestalt of the visual pattern that has non-mirror symmetry structure.

Since the unit structure of visual elements are explicitly represented by embedded structure of the representing sets, the parser can check at each level whether an element is a singleton set. If so, the unit rule can be applied. Therefore, given the embedded representation of a visual pattern consisting of n primitive visual elements and m embedded sets, the computational complexity of the unit rule is linear in the number of primitive visual elements and the number of embedded sets, i.e. $O(\ n + m\ )$.

4.3 Computing Proximity-based Gestalts

The proposed coding rules are designed to compute the gestalts of two-dimensional visual patterns. These gestalts are constituted by perceptually motivated structural regularities that are involved in visual patterns. In fact, each coding rule detects a certain kind of structural regularities in visual patterns. Each gestalt has an information complexity which indicates the amount of its structural irregularity: the more regularity the less information complexity. Different gestalts of a visual pattern compete with each other according to the amount of structural regularities that they cover and thus according to their information complexities.

In chapter 2, however, we introduced the proximity structure of visual patterns, and explained that it cannot compete with structural regularity by means of information complexity. Recall that the proximity structure of a visual pattern is a cluster hierarchy of its constitutive visual elements. In order to integrate structural regularities and proximity factors, we assumed that structural gestalts are always defined within one proximity cluster, i.e. structural gestalts cannot be defined in terms of proper subsets of elements from different proximity clusters. We showed that a violation of this assumption results in gestalts with a low degree of perceptual goodness (see Figure 2.17 in chapter 1). The assumption that structural gestalts may only be defined within a proximity cluster excludes gestalts with a low degree of goodness.

The influence of the proximity factor on the gestalts of visual patterns can be integrated in our approach by introducing the *proximity-based parse state* of visual patterns. The proximity-based parse state of a visual pattern is a hierarchical set such that the hierarchical structure of the set reflects the hierarchical proximity structure of the visual pattern. The bottom-up application of the coding rules to the proximity-based parse state of visual patterns will then generate structural gestalts within one proximity cluster. In the following, we first introduce a method to compute the hierarchical proximity structure of visual patterns. Then, we define proximity-based parse state and explain how the initial parse state of a visual pattern can be represented based on the proximity structure of that pattern. Finally, we explain the application of the coding rules to the proximity-based parse states and show that the generated gestalts cover structural regularities as well as proximity structures. In these gestalts, the structural regularities occur within one proximity cluster.

4.3.1 Methods for Computing Proximity Structures

There are various methods to compute the proximity structure of visual patterns [JD88, Dub93]. Examples of these methods are partitioning (k-means), hierarchical clustering, (divisive and agglomerative), clique optimization, additive similarity trees, etc. In general, these methods are designed to compute clusters of elements (not necessarily visual elements) on the basis of some proximity measure. These methods can be used to compute the proximity structure of visual patterns by considering the spatial

distances between visual elements in the Euclidean space as the proximity measure.
In this section, we briefly explain the agglomerative hierarchical clustering method. The reason to elaborate on this method is its simplicity and clear mathematical definition. It should be noted that all mentioned clustering methods can be used to compute the proximity structure of visual patterns and that there is no evidence for preferring one above the other as a psychological model.

The agglomerative clustering method starts by assuming each single primitive element of a visual pattern as one cluster, and iteratively generates compound clusters (i.e. clusters that contain more than one primitive element) by merging two existing clusters. The process of clustering terminates when only one compound cluster remains. In the agglomerative clustering method, clusters are merged according to various strategies that use distance matrices. The columns and the rows of a distance matrix indicate clusters and the value of each matrix entry indicates the distance between clusters that specify that entry.
For example, consider the visual pattern illustrated in Figure 4.5. This visual pattern consists of six primitive visual elements that form three clusters.

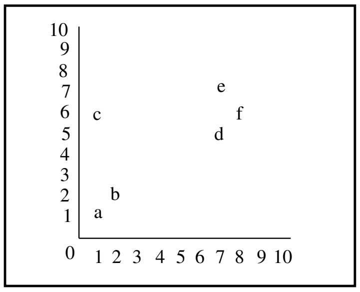


Figure 4.5: *Visual elements provide three proximity clusters.*

	a	b	c	d	e	f
a	0	1.4	5	7.2	8.4	8.6
b	1.4	0	4.1	5.8	7	7.2
c	5	4.1	0	6	6	7
d	7.2	5.8	6	0	2	1.4
e	8.4	7	6	2	0	1.4
f	8.6	7.2	7	1.4	1.4	0

Figure 4.6: *The distance matrix for the visual pattern in Figure 4.5.*

The distance matrix for this visual pattern is then illustrated in Figure 4.6. Note the symmetrical structure of the values relative to the diagonal axis which consists of zero distances. This symmetrical structure is due to the fact that the distance between two clusters x and y is the same as the distance between y and x.
Given the distance matrix of a visual pattern, the agglomerative clustering

method merges at each step two clusters that have the smallest distance from each other. For instance, given the distance matrix illustrated in Figure 4.6, this clustering method merges the primitive elements a and b and generates a new compound cluster (ab). Note that in this matrix the smallest distance is not unique, i.e. the distance between primitive elements e and f and the distance between primitive elements d and f are also the smallest distance. When more than one pair of clusters have the smallest distance, one of them is chosen randomly.
Based on the existing distance matrix, the agglomerative clustering method proceeds with generating a new distance matrix in which the distances between the newly constructed cluster (in this case (ab)) and all other existing clusters are computed. There are various strategies to compute the distance between the newly constructed cluster and all other existing clusters. Examples of these strategies are single-linkage, complete-linkage, and average-linkage strategies.

Let x and y be the clusters that are merged and z be an existing cluster. Then, the three mentioned strategies compute the distance between clusters (xy) and z on the basis of the distance between x and z and the distance between y and z. The single-linkage strategy takes the minimum of these two distances, the complete-linkage strategy takes the maximum of these two distances, and the average-linkage strategy takes the average of these two distances, i.e.

$$\begin{aligned}
&single-linkage: \\
&\quad distance((xy),z) = minimum(distance(x,z), distance(y,z)) \\
&complete-linkage: \\
&\quad distance((xy),z) = maximum(distance(x,z), distance(y,z)) \\
&average-linkage: \\
&\quad distance((xy),z) = \frac{distance(x,z)+distance(y,z)}{2}
\end{aligned}$$

For instance, based on the distance matrix illustrated in Figure 4.6 and given the newly constructed cluster (ab) the single-linkage, the complete-linkage, and the average-linkage strategies result in three new distance matrices which are illustrated in Figures 4.7, 4.8, and 4.9, respectively. Because of the symmetrical structure of values in the distance matrix, we only mention the values of one symmetrical halves. In these distance matrices, the distance between cluster (ab) and all other existing clusters are represented. The distances between existing clusters remain the same.

	(ab)	c	d	e	f
(ab)	0	-	-	-	-
c	4.1	0	-	-	-
d	5.8	6	0	-	-
e	7	6	2	0	-
f	7.2	7	1.4	1.4	0

Figure 4.7: *The single-linkage strategy.*

	(ab)	c	d	e	f
(ab)	0	-	-	-	-
c	5	0	-	-	-
d	7.2	6	0	-	-
e	8.4	6	2	0	-
f	8.6	7	1.4	1.4	0

Figure 4.8: *The complete-linkage strategy.*

	(ab)	c	d	e	f
(ab)	0	-	-	-	-
c	4.5	0	-	-	-
d	6.5	6	0	-	-
e	7.7	6	2	0	-
f	7.9	7	1.4	1.4	0

Figure 4.9: *The average-linkage strategy.*

Note that the average-linkage strategy is not associative, i.e. given three elements a, b, and c, the distance between clusters (ab) and c is not necessarily the same as the distance between clusters a and (bc). This implies that the order of clustering of elements may result in different cluster structures.

Given a newly generated distance matrix, the agglomerative clustering method reiterates the previous steps, i.e. it proceeds with merging the clusters with the smallest distance. At the next step in the example, the smallest distance is between clusters d and f (but also between e and f). Assuming the single-linkage strategy, the merge of d and f results in a new distance matrix illustrated in Figure 4.10. The smallest distance is now between clusters (df) and e. The merge of these clusters results in the distance matrix illustrated in Figure 4.11. Subsequently, the distance between (ab) and c is the smallest distance. The merge of (ab) and c results in the distance matrix illustrated in Figure 4.12. Finally, the remaining two clusters $((ab)c)$ and $((df)e)$ are merged which results in one single cluster

$(((ab)c)((df)e))$.

	(ab)	c	(df)	e
(ab)	0	-	-	-
c	4.1	0	-	-
(df)	5.8	6	0	-
e	7	6	1.4	0

Figure 4.10: *Distances with respect to the cluster* (df)*.*

	(ab)	c	((df)e)
(ab)	0	-	-
c	5	0	-
((df)e)	5.8	6	0

Figure 4.11: *Distances with respect to the cluster* $((df)e)$*.*

	((ab)c)	((df)e)
((ab)c)	0	-
((df)e)	5.8	0

Figure 4.12: *Distances with respect to the cluster* $((ab)c)$*.*

It is important to note that at each clustering step two clusters are merged between which the distance was the smallest distance at that step. The hierarchical structure of clusters and the distances on the basis of which the clusters are merged can be represented by a dendrogram. For example, the dendrogram illustrated in Figure 4.13 represents the cluster hierarchy $(((ab)c)((df)e))$ where the length of each subhierarchy indicates the distance on the basis of which the two clusters of that subhierarchy are merged.

The hierarchical agglomerative clustering method results always in one single cluster. In order to decide on perceptually motivated proximity clusters, one needs to specify a criterion to cut the dendrogram at a certain hierarchical level. In this way, a cluster contains all elements that belong to one of the resulting subhierarchies.
There are various measures to determine the level that provides perceptually motivated proximity clusters. The basic idea of such a measure is to determine the significant increase of distances on the basis of which clusters are merged. For instance, in the above example, the clusters are merged on the basis of distances $1.4, 1.4, 1.4, 5$, and 5.8. The significant increase of the distances in this example is from 1.4 to 5. This significant increase determines the level that provides three clusters, i.e. (ab), c, and $((df)e)$.
Note that such a measure can be applied to subclusters to determine their

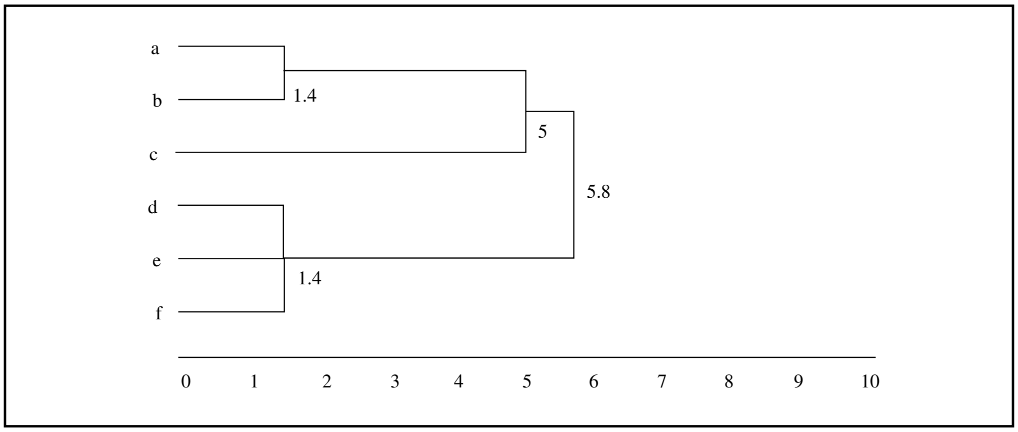


Figure 4.13: *The dendrogram representing the cluster hierarchy of the visual pattern illustrated in Figure 4.5. The assigned numbers indicate distances on the basis of which clusters are merged.*

perceptually motivated subclusters. In the rest of this chapter, we assume such a perceptually motivated measure that determines the perceptually motivated hierarchical proximity structure of visual patterns.

4.3.2 Proximity-based Representation of Visual Patterns

In order to integrate the proximity factor in our approach, we use an existing cluster method, for example the agglomerative method with the single-method strategy, to generate the hierarchical cluster structure of two-dimensional visual patterns. Given the hierarchical cluster structure of a visual pattern, we consider its initial proximity-based parse state as a hierarchical set where the hierarchical structure of the set reflects the hierarchical cluster structure of that visual pattern. The construction of the initial proximity-based parse state for a visual pattern is accomplished by processing the hierarchical cluster structure of that visual pattern in a bottom-up fashion. At each level of the cluster hierarchy, the direct children of each non-terminal node form the elements of a set. This process results in an hierarchical set. In this way, non-terminal elements at different hierarchical levels of a parse state represent the parse state of clusters at different hierarchical levels of proximity structure.

However, non-terminal elements are already used to represent *units* of visual elements. In order to represent the unit structure as well as the proximity structure of visual elements by the hierarchical structure of parse states, we consider units of visual elements as special clusters. In fact, the common idea behind units and proximity clusters is that proper subsets of elements from different units or from different proximity clusters cannot be grouped together to constitute a structural gestalt.

Non-terminal elements can be used to represent unit clusters as well as proximity clusters by adding a label to non-terminal elements. In this way, the label of a non-terminal element indicates whether the non-terminal element represents a unit cluster or a proximity cluster. For this reason, we introduce two labels U and P to indicate unit and proximity clusters, respectively. Moreover, we write $\{a_1, \ldots, a_n\}_U$ and $\{a_1, \ldots, a_n\}_P$ to represent a unit cluster and a proximity cluster each of which contains n elements.

Given the hierarchical unit structure [3] and the hierarchical cluster structure of a visual pattern, we determine the initial unit- and proximity-based parse state of that visual pattern by the following method.
First, the initial proximity-based parse state is computed as a hierarchical set such that the hierarchical proximity structure is represented by the hierarchical structure of the set. Then, each embedded set (non-terminal element) is labeled by assigning the P label to it. Once the hierarchical proximity structure is generated, we determine the units of elements at each hierarchical level. This can be done by selecting at each hierarchical level those elements that correspond to united visual elements. Subsequently, a non-terminal element is introduced at that level to contain the selected elements. The introduced non-terminal elements are labeled by assigning the U label to them. In this way, the proximity-based hierarchical set is extended with the unit structure. Note that this method correctly implies that the united elements belong to one proximity cluster.
As an example of generating the initial unit- and proximity-based parse state of a visual pattern consider the visual pattern shown in Figure 4.14-A.

For this visual pattern the clustering method will provide the cluster hierarchy shown in Figure 4.14-B. Finally, based on this cluster hierarchy and given the unit structure of visual elements (indicated by dashed boxes) the

[3] We explained that the hierarchical unit structure of two-dimensional visual patterns are given beforehand.

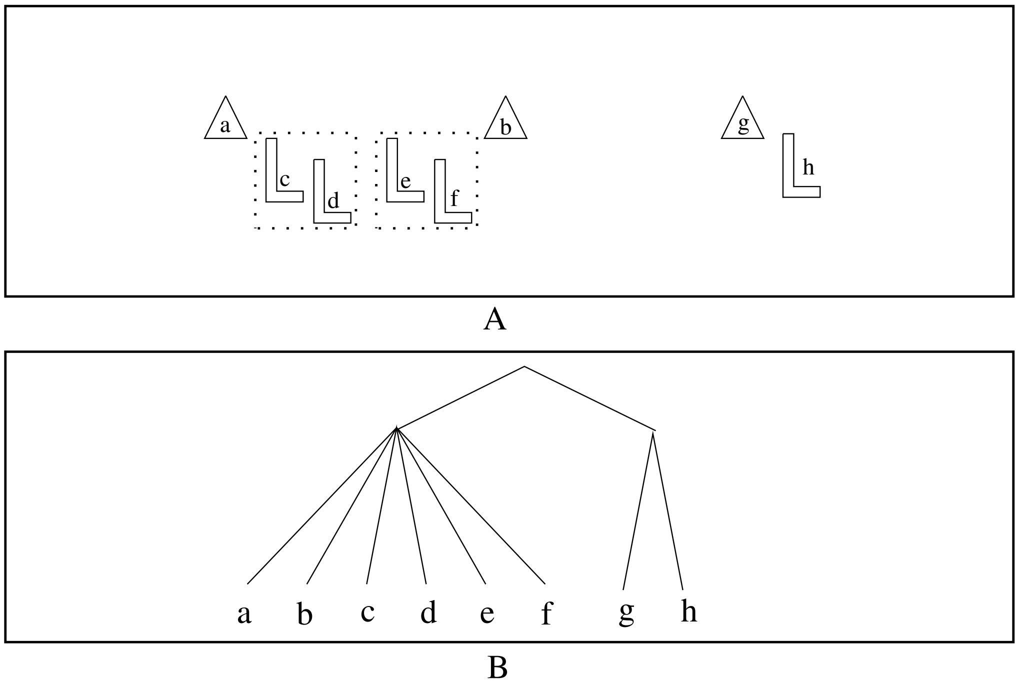


Figure 4.14: *A) A visual pattern, B) its cluster hierarchy.*

initial parse state is generated as the following set:

$\{\{a, b, \{c, d\}_U, \{e, f\}_U\}_P, \{g, h\}_P\}.$

4.3.3 Parsing Unit- and Proximity-based Representations

The initial parse state of a visual pattern, which is constructed on the basis of its unit and proximity structures, is a hierarchical set in which terminal elements are E-VREG expressions. Like for the hierarchical unit-based parse states, we propose a bottom-up parsing method for the unit- and proximity-based parse states such that the lowest level embedded sets are parsed before higher level sets are parsed.

In unit- and proximity-based parse states, each non-terminal element has a label that determines whether it represents a unit or a proximity cluster. Therefore, we modify the proposed unit coding rule and introduce an additional coding rule to construct the coding of the proximity structure.

Since the parsing process is a bottom-up process, a non-terminal element that represents a unit will contain only one E-VREG expression at a certain stage of the parsing process. In this case, the non-terminal element should be replaced by a unit expression for which the argument is the E-VREG expression that is contained in that non-terminal element. Similarly, a non-terminal element that represents a proximity cluster will contain only one E-VREG expression at a certain stage of the bottom-up parsing process. This non-terminal element should then be replaced by the E-VREG expression that is contained in it. In section 2.1 of this chapter, we introduced a coding rule for the unit structure. This coding rule should be replaced by the unit coding rule that is proposed in the following definition.

21. Definition. *Let A be an embedded set (at a certain hierarchical level of a parse state), "a" be a E-VREG expression, $\{\}_U$ and $\{\}_P$ indicate the non-terminal elements that represent a unit and a proximity cluster, respectively. Then, the unit and the proximity rules which construct the codings of the unit and the proximity structures of visual patterns are defined as follows:*

Unit Rule:

For any $\{a\}_U \in A$
$$A \longrightarrow A \setminus \{\{a\}_U\} \cup \{Unit(a)\}$$

Proximity Rule:

For any $\{a\}_P \in A$
$$A \longrightarrow A \setminus \{\{a\}_P\} \cup \{a\}$$

Recall that the function of non-terminal elements that represent proximity clusters is to avoid the construction of gestalts between proper subsets of elements from different proximity clusters. When a non-terminal element contains only one E-VREG expression, its function is fulfilled such that it can be replaced by the terminal element that is contained in it.
Note also that a structural regularity of the elements of one cluster (i.e. a structural regularity within one cluster) can constitute a gestalt since in the parse states the elements of one cluster are contained in one set. Similarly, a structural regularity among clusters can also constitute a gestalt since

in the parse states non-terminal elements that represent the parse state of (higher-order) clusters are contained in one set as well.

In order to illustrate the bottom-up application of the coding rules to a unit- and proximity-based parse state consider the initial parse state of the visual pattern illustrated in Figure 4.14-A, i.e.

$\{\{a, b, \{c, d\}_U, \{e, f\}_U\}_P, \{g, h\}_P\}$.

At the first parse step the embedded set that contains terminal elements (E-VREG expressions) c and d is parsed. This parse results the E-VREG expression $Comp(c, d)$. At the second parse step, the set that contains terminal elements e and f is parsed which results the E-VREG expression $Comp(e, f)$. Then, the unit rule is applied to the non-terminal elements that contain the derived E-VREG expressions. The application of the unit rule to these non-terminals results two E-VREG expressions $Unit(Comp(c, d))$ and $Unit(Comp(e, f))$. Subsequently, the compose and the symmetry rules are successively applied to the set that contains the above derived E-VREG expressions together with the terminal elements a and b. The result is the single E-VREG expression $Sym(Comp(a, Unit(Comp(c, d))), \psi)$. Similarly, the non-terminal element that contains the E-VREG expressions g and h is parsed by applying the compose rule. The E-VREG expression $Comp(g, h)$ results. The resulting E-VREG expressions are contained in two non-terminals that have the proximity label. At the next parse step, the applications of the proximity rule to these non-terminals generate E-VREG expressions that have been contained in these non-terminals. Finally, the two remained E-VREG expressions $Sym(Comp(a, Unit(Comp(c, d))), \psi)$ and $Comp(g, h)$ are parsed by applying the compose rule to them. This generates a E-VREG expression that provides a gestalt of the parsed visual pattern, i.e. $Comp(Sym(Comp(a, Unit(Comp(c, d))), \psi), Comp(g, h))$.

The above mentioned parse steps can be illustrated as follows:
$\{\{a, b, \{c, d\}_U, \{e, f\}_U\}_P, \{g, h\}_P\} \longrightarrow$
$\{\{a, b, \{Comp(c, d)\}_U, \{e, f\}_U\}_P, \{g, h\}_P\} \longrightarrow$
$\{\{a, b, \{Comp(c, d)\}_U, \{Comp(e, f)\}_U\}_P, \{g, h\}_P\} \longrightarrow$
$\{\{a, b, Unit(Comp(c, d)), Unit(Comp(e, f))\}_P, \{g, h\}_P\} \longrightarrow$
$\{\{Sym(Comp(a, Unit(Comp(c, d))), \psi)\}_P, \{g, h\}_P\} \longrightarrow$
$\{\{Sym(Comp(a, Unit(Comp(c, d))), \psi)\}_P, \{Comp(g, h)\}_P\} \longrightarrow$

$\{Sym(Comp(a, Unit(Comp(c,d))), \psi), Comp(g,h)\} \longrightarrow$
$\{Comp(Sym(Comp(a, Unit(Comp(c,d))), \psi), Comp(g,h))\}.$

Note that this E-VREG expression describes the gestalt of the parsed visual pattern such that structural regularities as well as proximity factor is taken into account, i.e. it covers structural gestalts within proximity clusters.

Since the unit and the proximity structure of visual elements are explicitly represented by embedded structure of the representing sets, the parser can check at each level whether an element is a singleton set. If so, the unit or the proximity rule can be applied based on the label of that singleton set. Therefore, given the embedded representation of a visual pattern consisting of n primitive visual elements and m embedded sets, the computational complexities of the unit and the proximity rules are linear in the number of primitive visual elements and the number of embedded sets, i.e. for each coding rule the computational complexity is $O(\ n + m\ )$.

Representing visual patterns as hierarchical sets implies that the computational complexity of the coding rules for E-VREG expressions are not defined in the number of all primitive elements that constitute a visual pattern. The reason is that the coding rules can now be applied only to the elements that are included in one embedded set and not to the elements of different embedded sets. This means that the computational complexities of the coding rules remain the same, but they should be defined in the maximum number of visual elements (for the worst case) that are included in one of the embedded sets.

Until now, we have concentrated on gestalts of perceptual patterns without considering any context effect. In fact, we have assumed that a visual pattern is perceived in isolation. In the next chapter, we will consider the influence of context factors on the gestalts of perceptual patterns.

Chapter 5

The Context-Effect in Perception

Gestalt psychologists since Wertheimer have noted that any sensory stimulus allows many gestalts, and have attempted to explain why certain gestalts are preferred over others. In the previous chapters various languages were introduced that provide possible gestalts of various classes of perceptual patterns. We also showed how the preferred gestalt of a perceptual pattern, among all its possible gestalts, can be selected by means of a perceptually motivated complexity measure which is called information load: the preferred gestalt of a perceptual pattern is the one that has the lowest information load. In this approach, one can determine the preferred gestalt of isolated perceptual patterns.

However, when our perceptual processes encounter a new stimulus pattern, they integrate it into a larger whole, which involves other simultaneously present input patterns, internal representations of past experiences, and what might be called the intentional setting of the perceiver [VLBVdV88, SVL93]. The factor of past experiences was first emphasized by Helmholtz [VH62] and later formulated as a perceptual principle by Wertheimer. The intentional setting of a perceiver is a conscious intention of the perceiver to perceive an input pattern in a certain way. For instance, the intentional setting may be the use of a perceptual pattern for a certain purpose.
These three factors that influence the perception of input patterns are called *context factors.* Thus, what structure our cognitive system in fact assigns to a particular input pattern is largely dependent on what context it takes into consideration: an input pattern in a certain context may be perceived

differently in a different context. In integrating the new input pattern, our cognitive system may (either deliberately or unconsciously) analyze the components of the stimulus as structurally analogous to components of the context. This capability of *analogical projection* plays a crucial role in perception. In particular, metaphor and cognitive analogy exercise this capability at a more abstract level.

A well-defined class of problems that involves analogical projection in interaction with perception is constituted by proportional analogy problems about perceptual patterns. We will focus on these problems in this chapter to show how we model this particular context effect in our approach. This chapter is organized as follows.
In section 1, we introduce proportional analogy problems and show that the gestalts of patterns change when they are perceived in the context of proportional analogy problems. In fact, solving proportional analogy problems about perceptual patterns consists in modeling the interaction between analogical projection and perception.
In section 2, we use the algebraic systems that were introduced in chapters 3 and 4 to model proportional analogy problems about perceptual patterns. In fact, the proportional analogy problem is modeled by considering the gestalts of patterns as terms of two algebras and define the analogical projection between those gestalts by means of a correspondence between the two algebras.
For a given proportional analogy problem, there are many correspondences possible. Different correspondences may define different analogical relations and therefore may relate different gestalts to each other. In order to define a proper correspondence for a proportional analogy problem, we introduce in section 3 various constraints on possible correspondences. These constraints are motivated by the perceptual preference ordering among related gestalts and the complexity of analogical relations.
In section 4, we use the algebraic framework and introduce a computational model for solving proportional analogy problems about perceptual patterns. For a given proportional analogy problem, we use the perceptual preference ordering among gestalts of the involved patterns and attempt to compute a proper correspondence which specifies an analogical projection between those gestalts. If a proper correspondence is computed, then the given proportional analogy problem is solved. If no proper correspondence can be computed, then it is attempted to compute a proper correspondence for the less preferred gestalts of the involved perceptual patterns, etc.

Finally, in section 5 we consider two alternative computational systems that are designed to solve proportional analogy problems about perceptual patterns. These systems are ANALOGY [Eva68] and Copycat [HM88]. We compare these systems with our approach and discuss their shortcomings.

5.1 Interaction between Perception and Analogy

Analogies between perceptual patterns demonstrate a complexity in gestalt disambiguation: namely how the perceived structure of one pattern influences the perceived structure of another pattern by making an analogy between them. In particular, the requirement to make an analogy between two perceptual patterns implies that the structure of the patterns have to be analogous to each other. For example, consider the two visual patterns illustrated in Figure 5.1.

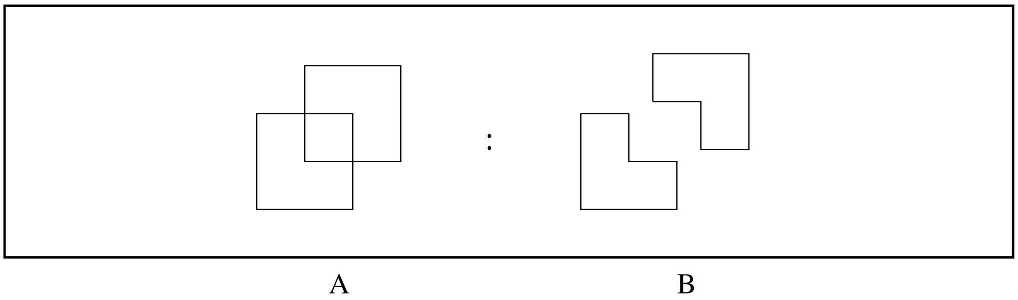


Figure 5.1: *The analogy context influences gestalts of line patterns.*

In this example, making analogy between line patterns A and B implies that the preferred gestalts of these line patterns consist of two concave shapes (L-shapes), while without such a requirement (context effect) the preferred gestalt of the line pattern A consists of two convex shapes (squares). It is important to note that the convex/concave property of these shapes expresses the perceptual grouping of the constitutive line segments and thereby the gestalts of these line patterns. The analogical relation that characterizes this analogy is that the line pattern A consists of two connected concave shapes which are disconnected in the line pattern B.

In this chapter, we will study and model the interaction between analogy making, as a context factor, and perception of patterns in proportional analogies. Proportional analogies follow a scheme that can be represented as "A is to B as C is to D", abbreviated as $A : B = C : D$. The elements A, B, C and D can be verbal descriptions of concepts, as in "*gills* are to *fish* as *lungs* are to *humans*"; but they can also be perceptual patterns, such as the letter strings of Hofstadter's Copycat domain [Hof84], as illustrated in Figure 5.2; or line-drawings, as illustrated in Figure 5.3. In fact, a proportional analogy consists of two identical analogies: one between A and B and the second between C and D. For example, consider the first proportional analogy illustrated in Figure 5.3. The characterizing analogical relation between patterns A and B is that the two connected concave shapes in A are disconnected in B. The same characterizing analogical relation exists between patterns C and D as well. The identity sign "=" in the proportional analogy scheme expresses that the two analogies are identical.

abba : abab = pqrrqp : pqrpqr

abba : abbbbba = pqrrpq : pqrrrrpq

Figure 5.2: *Examples of proportional analogies defined on letter strings.*

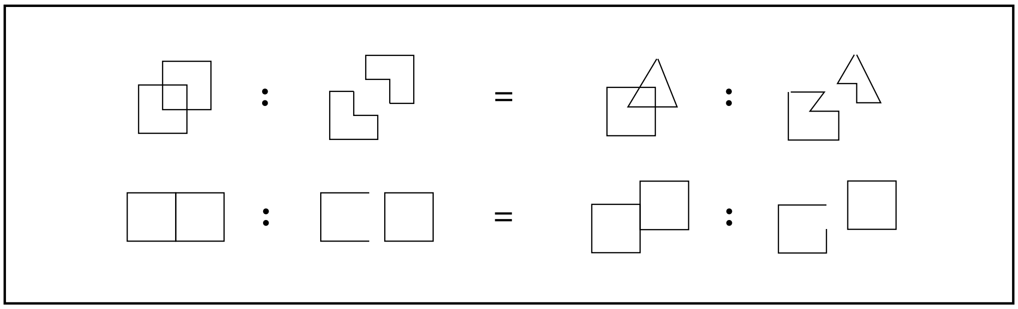


Figure 5.3: *Examples of proportional analogies defined on line-drawings.*

It is worth noticing that the arithmetical metaphor suggested by this notation can be taken almost literally. For instance, it is indeed the case that the elements B and D in this scheme can be interchanged with the elements A and C respectively, without changing the validity of the analogy.

Similarly, one may interchange the elements B and C. For example, given the proportional analogies "*abc* is to *abd* as ijk is to ijl", the proportional analogies "*abd* is to *abc* as ijl is to ijk" and "*abc* is to ijk as *abd* is to ijl" are also valid analogies.

A proportional analogy problem is constructed by omitting one element in a proportional analogy relation. We write $A : B = C : X$; the task is to find an X which is related to C in the same way as B is related to A. Solving a proportional analogy problem requires 1) perceptually preferred gestalts of A, B, and C, 2) construction of a mapping between the gestalts of A and B which generalizes to a broader domain that includes C, and 3) applying the mapping to the gestalt of C. In particular, the structures of A and B are to be construed as analogous to each other; and the structure of C must be construed in such a way that it is in the domain of the function that articulates that analogy. Proportional analogy problems are thus fairly complex: there are *three* patterns whose perceived structures mutually influence each other through analogical mappings.
This interaction between perception and analogy making in proportional analogies occur even in very simple domains, as has been demonstrated by Hofstadter [Hof84]. For example, the first term in the two proportional analogies shown in Figure 5.2 is the same. Yet the context influence caused by the terms B and C forces different gestalt decompositions of the term A in each case.
In our view, similar interaction between analogy making and possible descriptions of elements, as exemplified by the proportional analogies illustrated in Figure 5.2 and 5.3, is also responsible for conceptual proportional analogies as illustrated in Figure 5.4, and the metaphors derived from it.

life	:	progress	=	death	:	stagnation
life	:	day	=	death	:	night

Figure 5.4: *Examples of proportional analogies defined on concepts.*

5.2 Formalizing Proportional Analogy in SIT

In chapter 3 and 4, we proposed an algebraic framework for the Structural Information Theory and defined the gestalts of patterns as algebraic terms. Given that gestalts are formalized as algebraic terms, we can now turn to the problem of how to characterize analogies between different gestalts.

One obvious approach is to use the notion of structural identity between gestalts. For example, $Iter(a, succ, 3)$ has the same structure as $Iter(p, pred, 3)$. This can be formalized using *term unification* [Sie89] between algebraic terms (gestalts). There are two ways to do this. One is to define a notion of *structural template* which is essentially a term containing variables (abstract algebraic terms), and then say that two gestalts g_1 and g_2 (i.e. two algebraic terms) are analogous if there exists a structural template g_{st}, and substitutions θ_1 and θ_2 such that $\theta_1 \circ g_{st} = g_1$ and $\theta_2 \circ g_{st} = g_2$. An equivalent definition is to allow *inverse substitutions* — so that if θ is a substitution replacing certain variables with terms, θ^{-1} would replace the terms with the corresponding variables — and say that two gestalts g_1 and g_2 are analogous if there exist substitutions θ_1 and θ_2 such that $\theta_1^{-1} \circ g_1 = \theta_2^{-1} \circ g_2$.

A different approach to characterize analogies between gestalts is to use the algebraic framework presented in [Ind91, Ind92]. If the source domain and the target domain are formalized as algebras, analogical relations between terms may be characterized as correspondences over algebras. Since gestalts of patterns are generated by algebraic systems, we can easily adopt this algebraic framework to characterize proportional analogy relations and model the interaction between gestalts and proportional analogy relations. In the rest of this chapter, we work out this approach in more detail.

22. DEFINITION. *A correspondence over two algebras is a relation between them that preserves the algebraic structures. In other words, given two algebras* $< D_1\ ;\ F_1 >$ *and* $< D_2\ ;\ F_2 >$ *and given* $F_1(n)$ *and* $F_2(n)$ *as sets of n-ary operators from* F_1 *and* F_2 *respectively, a* correspondence *between them is a pair* $< \Delta; \Omega >$ *where:*
1) $\Delta \subseteq D_1 \times D_2$,
2) $\Omega(n) \subseteq (F_1(n) \times F_2(n))$ *for all n, and*
3) if $< a_1, b_1 >, \ldots, < a_n, b_n > \in \Delta$ *and* $< \omega, \sigma > \in \Omega(n)$ *then*
$< \omega(a_1 \ldots a_n), \sigma(b_1 \ldots b_n) > \in \Delta$

A correspondence is itself an algebra. The elements of this algebra are pairs of elements, one from each initiating algebra, and the operators of this algebra are pairs of operators, also one from each initiating algebra.

This algebraic framework can be applied directly to the algebras that generate the gestalts of perceptual patterns. In chapter 3, we introduced several algebras that generate the gestalts of string patterns (String-algebra and Extended-string-algebra) and the gestalts of two-dimensional visual patterns (VREG-algebra and E-VREG-algebra). In the rest of this chapter, we abstract over a specific algebra and use the term *gestalt-algebra* to indicate any algebra that generates the gestalts of a certain class of perceptual patterns.

A gestalt-algebra generates a very large class of structural descriptions or gestalts, many of which are too detailed and complex. In any given context, we may want to restrict ourselves to only a small subset of the universe, and to a small class of operators. For this, we need the notion of a subalgebra.

23. DEFINITION. *An algebra* $< E; G >$ *is a subalgebra of an algebra* $< D; F >$ *if* $E \subseteq D$, $G(n) \subseteq F(n)$ *for all* $n \in \mathbf{N}$.

We also require that all subalgebras are finitely generated. This means that the set of operators is finite, and all the objects in the domain can be generated from a finite subset of the domain (by repeated application of operators). We call such a finitely generated subalgebra a *representation algebra*.
We need one more concept, namely that of an isomorphic correspondence, before we can show how to model proportional analogies.

24. DEFINITION. *Let* $< D_1; F_1 >$ *and* $< D_2; F_2 >$ *be two algebras. A correspondence* $< \Delta; \Omega >$ *over* $< D_1; F_1 >$ *and* $< D_2; F_2 >$ *is isomorphic if and only if* Ω *is a one-to-one function on* F_1*, and* Δ *is a one-to-one and onto function on* D_1.

Using these definitions, we model proportional analogies of the form "A is to B as C is to D" as consisting of two subalgebras of a gestalt-algebra (i.e. two representation algebras), one generating the terms A and C, and the other generating the terms B and D, and an isomorphic correspondence between the two subalgebras.
For example, consider the proportional analogy "$abc : ijk = abd : ijl$". The subalgebra $< \{ab, c\}; \{Con, succ\} >$ generates some descriptions for

both "*abc*" and "*abd*". These descriptions might be "$Con(ab, c)$" and "$Con(ab, succ(c))$", respectively. Similarly, the subalgebra $< \{ij, k\}; \{Con, succ\} >$ generates (among others) the descriptions "$Con(ij, k)$" and "$Con(ij, succ(k))$", respectively, for "ijk" and "ijl". The proportional analogy is then represented by the following isomorphic correspondence between the two subalgebras:

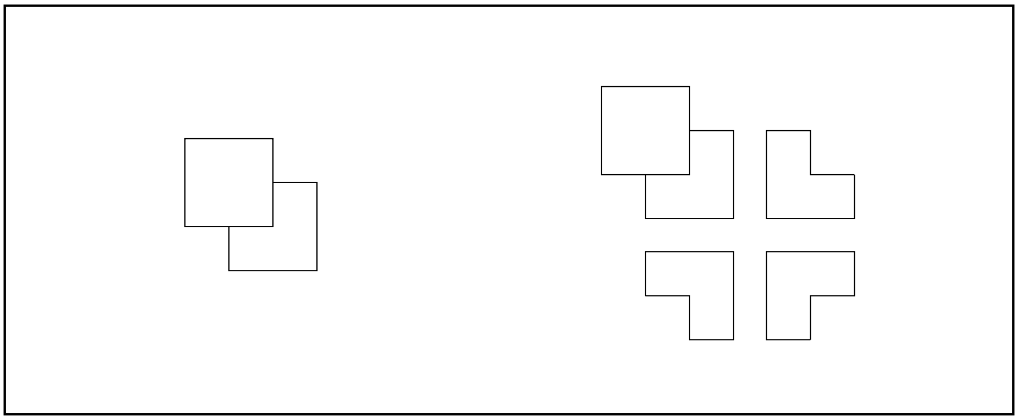
$< \{(ab, ij), (c, k)\}\ ;\ \{(Con, Con), (succ, succ)\} >.$

5.3 Introducing Constraints on Correspondences

Given a gestalt-algebra, there are many possible representation algebras (subalgebras); given any representation algebra, there are many possible structural descriptions; and given any two representation algebras, there may be many possible isomorphic correspondences between them. In this section, we present context factors that constrain the generation of gestalts for proportional analogy problems.

In Structural Information Theory, a complexity measure called *information load* (IL, henceforth) is assigned to each gestalt description. In chapter 3 and 4, we discussed how the information load of the gestalt descriptions for different classes of perceptual patterns can be computed. For example, for the gestalt descriptions of string patterns that are generated by the Extended-string-algebra we do not count the ISA operators, but do count any embedded domain-specific operators, except the identity operator. Hence, given the string pattern *abccba*, its gestalt $g_1 = Sym_e(Iter(a, succ, 3))$ has an IL of 2, and gestalt $g_2 = Sym_o(ab, Iter(c, id, 2))$ has an IL of 3. Applying the minimality principle — which states that the preferable gestalt for a given pattern P, among all extensionally equivalent terms evaluating to P, is the one with the lowest complexity value — predicts that g_1 would be chosen over g_2.
However, the minimality principle ignores the mutual contextualization effect in proportional analogies. This is because the lowest complexity gestalts, taken in isolation, of two individual patterns may not always result in the lowest complexity when the two patterns are presented together. This effect can be seen in the analogy $abccba : ccabbacc = pqrrqp : rrpqqprr$. We just saw above that the first pattern, namely *abccba*, considered out of

context has the preferred gestalt g_1 of an even symmetry structure, but within the context of the analogy, it would be preferable to see it as an odd symmetry structure g_2, even though g_2 has a higher information load. The reason is that g_2 shares common substructures with the preferred gestalt $g_3 = Sym_o(Iter(c, id, 2), Sym_e(ab))$ of *ccabbacc*.
For the domain of geometric patterns, the contextualization effect is illustrated in Figure 5.5.



A B

Figure 5.5: *The context influence for geometric patterns.*

The gestalt of Figure 5.5-A consists of two squares, one in front of the other providing the foreground/background interpretation, while the gestalt of Figure 5.5-B consists of a square with four L-shaped visual objects providing the mosaic interpretation. Although Figure 5.5-A is a part of Figure 5.5-B, the gestalt of the first is not a part of the gestalt of the second.

In order to incorporate this feature, what we would like is that when two patterns are present together, any common substructure between them adds to the IL only once. This effect can also be achieved by defining a complexity ordering on representation algebras, which is determined by the number of elements in it: the more elements in it, the higher the complexity. The underlying idea here is that when two patterns have gestalts that overlap, the representation algebra that generates these two gestalts will have a lower complexity.
For instance, the minimal representation algebra that generates the gestalts g_1 and g_3 for the first two patterns of the proportional analogy relation men-

tioned above is $< \{a, ab, c\}; \{succ\} >$, which has four elements. (Notice that we do not mention or count the ISA operators). But if we consider the gestalt g_2 for *abccba*, then g_2 and g_3 can be generated by the representation algebra $< \{ab, c\}; \emptyset\} >$ with only two elements.

Moreover, a perceptually preferred representation algebra may not be the proper one since the length of all generated gestalt descriptions are too long. Note that the information load as the complexity measure does not depend on the length of descriptions but on the number of elements in the descriptions. However, a description constituted by a small number of elements can still be a long description because it has a rich structure, i.e. it uses a large number of operators. Thus, perceptual models for proportional analogies should also consider this constraint by imposing an (empirically established) maximum length for descriptions. Obviously, such a maximum length implies a sharp distinction between appropriate and inappropriate pattern descriptions; we do not claim the existence of such a sharp distinction, however, and consider the maximum length only as an artificial and gross approximation. These perceptually motivated constraints will be called *simplicity criteria*.

For proportional analogy, there is an additional constraint, which we refer to as the *projectibility criterion*, namely that it must be possible to construct an analogical mapping between the corresponding terms of the analogy relation. The lowest complexity gestalts of two patterns may not be appropriate for constructing analogical relations between them.
For example, consider the analogy relation *abccba* : *ppqrpp* = *abccccba* : *ppqrqrpp*. The first term, the same as in the example above, has a preferred gestalt $Sym_e(Iter(a, succ, 3))$ $(IL = 2)$, but this does not form any analogical mapping with the preferred gestalt for the second term (*ppqrpp*), namely, $Sym_o(pp, qr)$ $(IL = 4)$. To discover the mapping underlying this analogy, we must consider a higher information load gestalt for the first term, namely $Sym_o(ab, cc)$ $(IL = 4)$.

Moreover, the existence of an analogical mapping between the preferred gestalts of patterns of proportional analogy problems is not enough. In fact, a mapping between two preferred gestalts may be more complex than another mapping, or even more complex than a mapping between two less preferred gestalts. Recall that an analogical mapping between two gestalts is represented as an algebraic correspondence over initiating representation algebras that generate those gestalts. An algebraic correspondence consists of a set of domain elements and a set of operators: the set of domain elements relates (maps) domain elements of the initiating representation algebras and the set of operators relates (maps) operators of the initiating algebras. Therefore, an analogical mapping consists of two mappings: a mapping which relates perceptual components (domain elements of the correspondence) and a mapping which relates operators (operators of the correspondence).

We define the complexity of an analogical relation as the complexity of the algebraic correspondence that represents it. A correspondence is then assumed to be simpler to the extent that more of the mappings that constitute it are identity mappings. Although the number of mappings (identity and non-identity mappings) that constitute a correspondence may also influence the complexity of the correspondence, we assume that identical mappings do not increase the complexity of the correspondence. In this way, we may define the complexity of a correspondence as the number of non-identical mappings that constitute it. Note that this complexity measure is based on intuition and need to be tested, or perhaps adjusted, by empirical research. Based on this suggested complexity measure, the correspondence $< \{(a,a),(b,b)\};\{(succ,succ),(pred,pred)\} >$ is considered to be simpler than the correspondence $< \{(a,b),(b,a)\};\{(succ,pred),(pred,succ)\} >$. This complexity measure can then be used to order all possible correspondences over two given representation algebras.

It may be the case that all possible correspondences over two initiating representation algebras are too complex. Thus, perceptual models for proportional analogy should also take this constraint into the consideration. In order to account for this complexity issue, one may introduce an empirically established maximum complexity value which should indicate the plausibility of a correspondence. Again, although such a maximum complexity value implies a sharp distinction between plausible and implausible correspondences, we do not claim the existence of such a sharp distinction and consider the maximum complexity value as indicating an artificial and gross approximation for the plausibility of correspondences.

In summary, the preferred gestalts for patterns occurring in a proportional analogy relation ought to have a low complexity, ought to be generated by a simpler representation algebra, and ought to satisfy the projectibility condition. Notice that we are saying 'ought to' because these three constraints interfere with each other. We will now see one heuristic approach that tries to find an optimal balance between them.

5.4 A Computational Model of Proportional Analogy

In the previous section, we defined preference orderings on gestalts, on the representation algebras that generate them, and on the analogical relations between them. In this section, we define a preference ordering on combinations of gestalts and analogical relations in the context of proportional analogy problems. Subsequently, we define a computational model which finds the preferred gestalt of the fourth pattern of proportional analogy problems in an efficient way.

The first step in solving proportional analogy problems about perceptual patterns is the computation of gestalts of the involved patterns. In chapter 3 and 4, we introduced computational models for generating the gestalts of string patterns and the gestalts for two-dimensional visual patterns. Therefore, we assume here three sets of algebraic terms provided by those computational models. Each set contains possible gestalts of one of the three patterns involved in proportional analogy problem.
Once we have computed the three sets of algebraic terms we turn to the actual task, i.e. to select one term for each pattern in order to define the preferable representation algebras and their correspondence. The second step in solving proportional analogy problems can therefore be understood as determining two representation algebras and a correspondence over them on the basis of perceptually ordered descriptions of the patterns involved in proportional analogy problems. As noted, there are two criteria for deciding on appropriate representation algebras: the projectibility and the simplicity criterion.
The projectibility criterion states that for the two selected representation algebras there must be a correspondence over them which generates pairs of patterns, among which, one consists of the first and the third pattern (or the first and the second pattern) and one consists of the second and the fourth pattern (or the third and the fourth pattern).

As noticed in the previous section, it may be the case that all possible correspondences over two selected representation algebras are complex correspondences. These correspondences indicate complex analogical relations that may not be appropriate. Therefore, it should be reasonable to introduce an experimentally established upper bound complexity value for analogical relations. Thus, two representation algebras for which all possi-

ble correspondences are complex are considered as representation algebras for which there is no proper correspondence. This means that these representation algebras do not satisfy the projectibility criterion.
The second criterion follows Leeuwenberg's perceptual preference principle. The preferred representation algebras are those which in addition to satisfying the projectibility criterion, result in the lowest collective complexity. This second criterion is the reformulation of the gestalt interaction.

Let us assume that the perceptual patterns A, B, and C of the proportional analogy problem $A : B = C : X$ are given, and the goal is to find the pattern X. We will utilize the above two criteria in order to direct the search process among possible gestalts of the given patterns, determine the appropriate representation algebras, and finally decide the fourth pattern X. The top-level structure of our algorithm for solving proportional analogy problems is as follows:

1. For each of the perceptual patterns A , B, and C separately, generate the set of possible gestalts (algebraic terms).

2. Order triples of gestalts for A, B, and C according to their collective information load.

3. Iterate through the sequence of triples in the order of increasing information load until two minimal representation algebras are founded such that:

   (a) One representation algebra $\mathcal{S}$ can generate the gestalts for A and C.

   (b) One representation algebra $\mathcal{T}$ can generate the gestalt for B.

   (c) An isomorphic correspondence between $\mathcal{S}$ and $\mathcal{T}$ exists which maps A to B.

4. Choose the simplest representation of C in the representation algebra $\mathcal{S}$.

5. Compute X: apply the correspondence to the representation of C.

In particular, three sets S_1, S_2, and S_3 that contain possible algebraic terms for perceptual patterns A, B, and C are generated. Then, from these three sets the subset of triples, *Set-of-triples*, that have the same and the lowest collective information load, *Curr-min-load*, is computed. Note that the

subset of triples is a subset of the Cartesian product of the sets S_1, S_2, and S_3. A triple is then selected randomly from the computed subset. The projectability criterion is checked for the selected triple. This is done by constructing two minimal representation algebras on the basis of the selected triple of gestalts such that an appropriate isomorphic correspondence between them is specified. The appropriateness of the isomorphic correspondences is related to their complexities as it is discussed above. The fourth term can then be generated on the basis of this specified correspondence. In fact, the correspondence generates a term that consists of the preferred gestalt of pattern C and a second gestalt. This second gestalt is assumed to be the preferred gestalt of the fourth pattern X. The fourth pattern can directly be generated on the basis of its gestalt. However, If the selected triple of gestalts does not satisfy the projectability criterion, another triple of gestalts from *Set-of-triples* is checked. When the *Set-of-triple* is empty, a new subset of triples is computed. The triples of the new subset have the next lowest collective information load.
In this way, we examine the triples of gestalts in the order of increasing collective information load. Thus, we order the search space according to the perceptual preference of the involved gestalts. This approach is represented by the flow chart illustrated in Figure 5.6. How a correspondence is specified on the basis of three gestalts will be discussed in the next subsection.

Notice that we select the possible triples first according to the simplicity criterion and then according to the projectibility criterion. Obviously, the inverse order of applying these criteria is also possible. However, we take the first ordering of applying these criteria because we assume that those descriptions that represent the most salient features of patterns have the best chance of being the intended descriptions within the context of the proportional analogy. This is a heuristic that enhances the efficiency of the process.
Note that if the descriptions of the patterns outside the proportional analogy context are not the appropriate descriptions, the descriptions will be changed in order to create a similarity which will constitute the analogical relation. Therefore, our model constructs creative as well as non-creative analogies, as Indurkhya [Ind92] calls them.

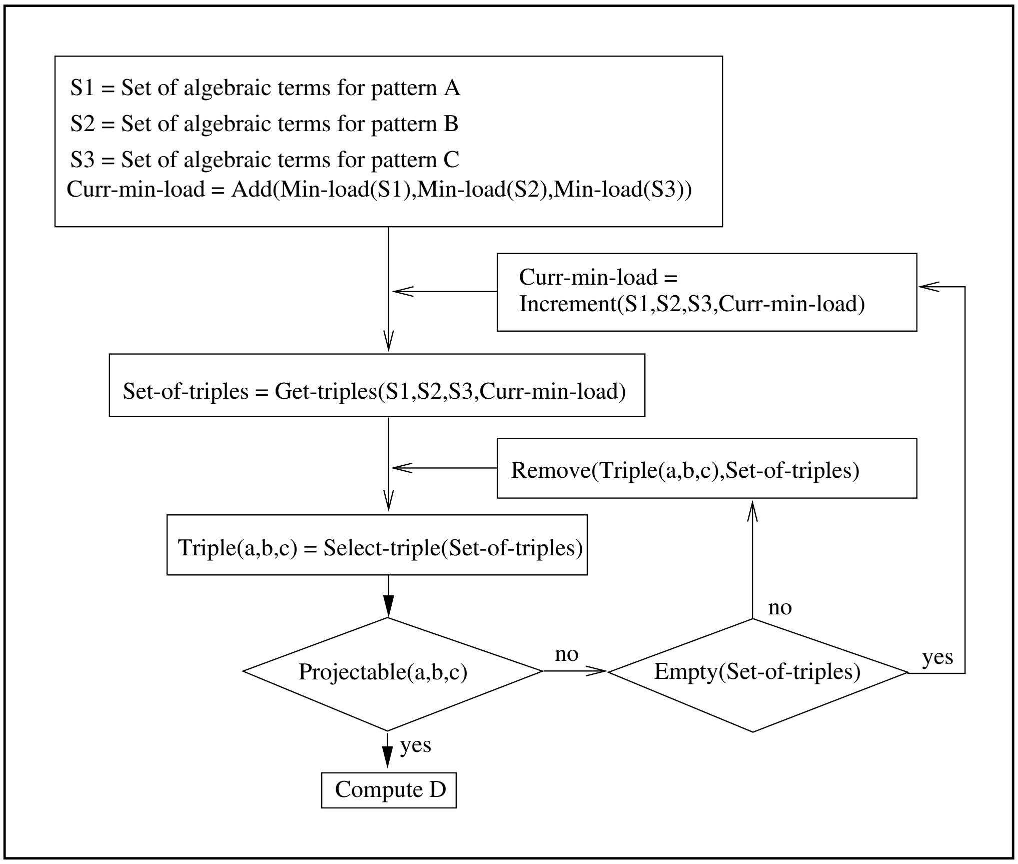


Figure 5.6: *The main process of solving proportional analogy problems.*

5.4.1 The Computation of the Perceptually Motivated Correspondence

In this section, we explain how the perceptual descriptions of patterns involved in a proportional analogy can be used in order to define the correspondence over their generating representation algebras. As noted, the perceptual description of patterns is a hierarchical description. We assume that analogical relations relate perceptual entities (elements and operators) at similar positions within the perceptual hierarchies with each other. This assumption implies that two identical elements (or identical operators) at two different positions in a hierarchy will be considered as different elements (or operators). This means that in proportional analogies the occurrence of elements in the hierarchy determines the identity of them.
In order to distinguish perceptual entities at different positions within the

perceptual hierarchies, we assign to each perceptual entity an index. Each index consists of two integers. The first integer indicates the depth level of the entity and corresponds with the level of embeddedness of that entity. The second integer indicates the latitude position of that entity at a certain level of the hierarchy and corresponds with its sequential order. Assigning indices to perceptual entities, the description $Con(Iter(Con(a,b),id,2),a)$ will be represented as $Con^{1,1}(Iter^{2,1}(Con^{3,1}(a^{4,1},b^{4,2}),id,2),a^{2,2})$. In this way, the elements $a^{4,1}$ and $a^{2,2}$ will be considered as two different elements. The indexing of hierarchical descriptions can be illustrated by drawing them as trees and assigning to each node its horizontal and vertical position. For example, the indexing of the above hierarchical description is illustrated in Figure 5.7.

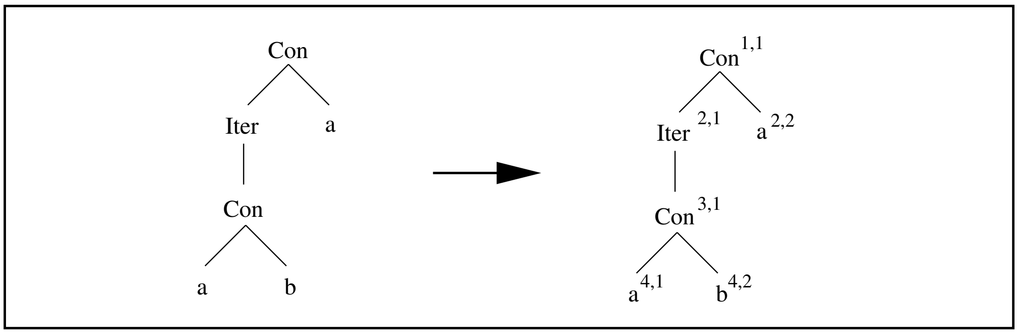


Figure 5.7: *An example of indexing hierarchical descriptions.*

Notice that the index of entities can be used as a criterion for the well-formedness of the descriptions generated by the correspondence. For example, the generated description $Con^{2,1}(a^{4,2},a^{3,1})$ may be considered as an ill-formed description while $Con^{1,1}(a^{2,1},a^{2,2})$ may be considered as a well-formed description.

The indices encode the position of elements in sequential patterns. The position of elements may be needed to identify them. The identification of elements is subsequently necessary for the specification and the application of analogical relations. However, in some proportional analogies the indices are not needed, or even, have to be avoided in order to solve proportional analogy problems. In these cases, the position of elements in the sequential patterns does not play any role in analogical relations.
For example, in the proportional analogy $abc : ijk = bbacbaacc : X$ the use of indices in the description of the involved string patterns makes the

analogical relation between *abc* and *ijk* not applicable to *bbacbaacc*. This is because the indexed element $a^{1,1}$ will be mapped to the indexed element $i^{1,1}$ (the specification of the analogical relation), but there is no indexed element $a^{1,1}$ in the third element to which the analogical relation can be applied. The indices can thus be used in the cases where the positions have to disambiguate different occurrences of one element in a sequential pattern.

In order to cover both cases of proportional analogies, we should specify criteria that decide on the assignment of indices to the descriptions of the involved patterns. In general, analogical relations are functions which map entities of one pattern description to entities of a second pattern description. Given two pattern descriptions, if the analogical relation between them can be specified as a function, then we need not to assign indices to the descriptions. Otherwise, when analogical relation can only be specified as a relation rather than a function, then different occurrences of similar elements must be distinguished by assigning indices to them. Then, the indices make it possible to specify an analogical relation as a function.

In order to specify a correspondence $< \Delta; \Omega >$, which defines the analogical relation between hierarchical descriptions, we specify $\Delta \subseteq D_1 \times D_2$ and $\Omega \subseteq F_1(n) \times F_2(n)$ recursively on the basis of three given hierarchical descriptions which express the gestalts of the three patterns (A, B, C) involved in the proportional analogy problem. Initially, Δ and Ω are empty sets. In specifying the correspondence, we distinguish four cases: three basic cases and one recursive case.
The first case applies when the description of the pattern A and the description of the pattern B are identical. In this case, the induced analogical relation is the identity relation which relates every description to itself.
The second case applies when the description of the pattern A and the description of the pattern B are distinct while the description of the pattern A and the description of the pattern C are identical. In this case, the induced analogical relation is this particular relation, i.e. a particular description A is related to another particular description B.
The third case applies when the description of the pattern A and the description of the pattern B have distinct outermost operators, but the description of the pattern A and the description of the pattern C have the same outermost operator. In this case, the induced relation is the relation between outermost operators of the description of patterns A and B. The arguments of three descriptions are then assumed to be identical.

Finally, the fourth case applies when the description of the pattern A and the description of the pattern B have identical outermost operators. In this case, the induced relation consists of an identity relation between operators and a relation between the argument of the description of the pattern A and the argument of the description of the pattern B. The relation between these arguments can be recursively specified by applying the above cases to them.

25. DEFINITION. *Let $\mathcal{G}_{<\mathcal{U};\mathcal{F}>}$ be the class of algebraic terms generated by a gestalt algebra $<\mathcal{U};\mathcal{F}>$ (String-algebra or VREG-algebra).*
Let $r, r_i, s, s_i, t, t_i \in \mathcal{G}_{<\mathcal{U};\mathcal{F}>}$ $(i \in \mathbf{N})$, and $f, g \in \mathcal{F}$. We write $[r, s, t] \mapsto \Delta$ to state that the three algebraic descriptions r, s and t induce the correspondence Δ. The correspondence induced by a given triple of descriptions can be specified as follows:

- $[s^{i,j}, s^{i,j}, t^{i,j}] \;\mapsto\; < \{(p^{i,j}, p^{i,j}) \mid \forall p^{i,j} \in \mathcal{G}_{<\mathcal{U};\mathcal{F}>}\} \;;\; \emptyset > .$

- $[s^{i,j}, t^{i,j}, s^{i,j}] \;\mapsto\; < \{(s^{i,j}, t^{i,j})\} \;;\; \emptyset > .$

- $[f^{i,x}(t_1^{i+1,j}, \ldots, t_n^{i+1,j+n-1}), g^{i,x}(t_1^{i+1,j}, \ldots, t_n^{i+1,j+n-1}), f^{i,x}(s_1^{i+1,j}, \ldots, s_n^{i+1,j+n-1})] \mapsto$
  $< \;\; \{(p_k^{i+1,j+k-1}, p_k^{i+1,j+k-1}) \mid \forall p_k^{i+1,j+k-1} \in \mathcal{G}_{<\mathcal{U};\mathcal{F}>} \;\&\; k = 1, \ldots, n\} \;;$
  $\{(f^{i,x}, g^{i,x})\} > .$

- $[f^{i,x}(r_1^{i+1,j}, \ldots, r_n^{i+1,j+n-1}), f^{i,x}(s_1^{i+1,j}, \ldots, s_n^{i+1,j+n-1}), g^{i,x}(t_1^{i+1,j}, \ldots, t_n^{i+1,j+n-1})] \mapsto$
  $< \bigcup_{k=1}^{n} \; \Delta_{i+1,j+k-1} \;;\; \{(q^{i,x}, q^{i,x}) \mid \forall q^{i,x} \in \mathcal{F}\} >$ *where,*
  $[r_k^{i+1,j+k-1}, s_k^{i+1,j+k-1}, t_k^{i+1,j+k-1}] \;\mapsto\; \Delta_{i+1,j+k-1}.$

When the descriptions do not involve indices, the above definition without indices can be used to specify the correspondence between those descriptions.

5.4.2 Examples of Solving Proportional Analogy Problems about String Patterns

In this section, we work out two examples of proportional analogy problems by specifying an analogical relation based on the algebraic descriptions that denote the involved patterns. Consider the proportional analogy problem "$abc : abd = iijjkk : X$". Let the selected preferred descriptions be "$Con^{1,1}(ab^{2,1}, id^{2,2}(c^{3,1}))$", "$Con^{1,1}(ab^{2,1}, succ^{2,2}(c^{3,1}))$", and "$Con^{1,1}(iijj^{2,1}, id^{2,2}(kk^{3,1}))$", respectively. The correspondence that is induced by the descriptions for A, B and C is recursively specified as follows:

1- $[Con^{1,1}(ab^{2,1}, Id^{2,2}(c^{3,1}))\ ,\ Con^{1,1}(ab^{2,1}, succ^{2,2}(c^{3,1}))\ ,$
$Con^{1,1}(iijj^{2,1}, Id^{2,2}(kk^{3,1}))] \mapsto$
$\Delta =< \Delta_1 \cup \Delta_2\ ;\ \{(Con^{1,1}, Con^{1,1})\} >$
2- $[ab^{2,1}, ab^{2,1}, iijj^{2,1}] \mapsto \Delta_1 =< \{(iijj^{2,1}, iijj^{2,1})\}\ ;\ \emptyset >$
3- $[Id^{2,2}(c^{3,1}), succ^{2,2}(c^{3,1}), Id^{2,2}(kk^{3,1})] \mapsto \Delta_2 =< \Delta_3\ ;\ \{(Id^{2,2}, succ^{2,2})\} >$
4- $[c^{3,1}, c^{3,1}, kk^{3,1}] \mapsto \Delta_3 =< \{(kk^{3,1}, kk^{3,1})\}\ ;\ \emptyset >$
$\Delta_2 =< \{(kk^{3,1}, kk^{3,1})\}\ ;\ \{(Id^{2,2}, succ^{2,2})\} >$
$\Delta =< \{(kk^{3,1}, kk^{3,1}), (iijj^{2,1}, iijj^{2,1})\}\ ;$
$\{(Con^{1,1}, Con^{1,1}), (Id^{2,2}, Succ^{2,2})\} >$

Note that the application of the rules (the first, the third, and the fourth rules) from definition 4 suggests that the induced correspondences include all possible pairs of identical terms (the first and the third terms) or all possible pairs of identical operators (the fourth term). This is expressed by the universal quantifier in those definitions. However, in the above example (and also in the next example) we did not include all identical pairs of terms or operators for simplicity reason. In fact, we use the information from the given three expressions and specify the correspondences in so far that they can generate the fourth term.
The correspondence Δ generates, among others, a pair consisting of the third and the fourth descriptions as follows:
"$(Con^{1,1}(iijj^{2,1}, id^{2,2}(kk^{3,1}))$" , "$Con^{1,1}(iijj^{2,1}, succ^{2,2}(kk^{3,1})))$".
This pair of descriptions denotes the pair $(iijjkk, iijjll)$ consisting of the third and the fourth patterns, respectively.

As a second example, consider the proportional analogy problem "$ababab : cdcdab = ababef : X$". The preferred triple of descriptions will be as follows:

"$Iter^{1,1}(ab^{2,1}, id, 3)$",
"$Con^{1,1}(Iter^{2,1}(cd^{3,1}, id, 2), ab^{2,2})$", and
"$Con^{1,1}(Iter^{2,1}(ab^{3,1}, id, 2), ef^{2,2})$".

Given these three descriptions and following the recursive procedure introduced in the previous section, there is no correspondence inducible. So, we should examine a less preferred triple of perceptual descriptions, for example, the following triple:

$Con^{1,1}(Iter^{2,1}(ab^{3,1}, id, 2), ab^{2,2}),$
$Con^{1,1}(Iter^{2,1}(cd^{3,1}, id, 2), ab^{2,2}),$
$Con^{1,1}(Iter^{2,1}(ab^{3,1}, id, 2), ef^{2,2}).$

The correspondence that is induced by these descriptions can be recursively specified as follows:

1- $[Con^{1,1}(Iter^{2,1}(ab^{3,1}, id, 2), ab^{2,2})\ ,\ Con^{1,1}(Iter^{2,1}(cd^{3,1}, id, 2), ab^{2,2})\ ,$
$Con^{1,1}(Iter^{2,1}(ab^{3,1}, id, 2), ef^{2,2})] \mapsto$
$\Delta =< \Delta_1 \cup \Delta_2\ ;\ \{(Con^{1,1}, Con^{1,1})\} >$
2- $[Iter^{2,1}(ab^{3,1}, id, 2), Iter^{2,1}(cd^{3,1}, id, 2), Iter^{2,1}(ab^{3,1}, id, 2)] \mapsto$
$\Delta_1 =< \{(Iter^{2,1}(ab^{3,1}, id, 2), Iter^{2,1}(cd^{3,1}, id, 2))\}\ ;\ \emptyset >$
3- $[ab^{2,2}, ab^{2,2}, ef^{2,2}] \mapsto \Delta_2 =< \{(ef^{2,2}, ef^{2,2})\}\ ;\ \emptyset >$
$\Delta =< \{(Iter^{2,1}(ab^{3,1}, id, 2), Iter^{2,1}(cd^{3,1}, id, 2)), (ef^{2,2}, ef^{2,2})\}\ ;$
$\{(Con^{1,1}, Con^{1,1})\} >$

The correspondence Δ generates, among others, a pair consisting of the third and the fourth descriptions as follows:
"$(Con^{1,1}(Iter^{2,1}(ab^{3,1}, id, 2), ef^{2,2})\ ,\ Con^{1,1}(Iter^{2,1}(cd^{3,1}, id, 2), ef^{2,2}))$".
This pair of descriptions denotes the pair $(ababef, cdcdef)$ consisting of the third and the fourth patterns, respectively.

5.5 A Comparison with other Approaches

We would like to make some remarks comparing our approach to other existing approaches for solving proportional analogy problems. Two noteworthy computational systems that solve proportional analogy problems about perceptual patterns are ANALOGY [Eva68] and Copycat [HM88]. We briefly introduce these systems and then discuss their shortcomings.

5.5.1 ANALOGY

ANALOGY is designed to solve so-called "geometric analogy" problems: proportional analogy problems that concern geometric figures. The system works with a "multiple choice" version of the problem that employs eight figures, A,B,C,1,2,3,4 and 5. The problem is formulated as: *find the rule by which figure A has been changed to make figure B; apply the rule to figure C, and indicate which of the figures 1 through 5 results from this process.*

The class of geometric figures that ANALOGY deals with is limited to line drawings that can be generated by applying two operators to a set of primitive visual elements consisting of dots, straight-line segments, and circle segments. These two operators are connect and compose operators: connect operator will literally connect visual elements to each other by letting them share one or more points; the compose operator collects the visual elements in one list.
In ANALOGY, two kinds of figures are distinguished: connected and non-connected figures. Connected figures are either primitive visual elements or generated by applying the connect operator to connected figures. Non-connected figures are generated by applying the compose operator to other (connected or non-connected) figures.
Furthermore, three classes of connected figures are distinguished: dots (DOT), simple closed curves (SCC), and all other connected figures (REGular). The distinction between SCC and REG is based on the connectivity of their line segments. In order to make this distinction, the notion of *vertex* is introduced. A vertex is either an endpoint of a curve or a point where three or more line segments meet. Now, a connected figure is a SCC if and only if it does not contain any vertex; it is a REG if it does contain at least one vertex. Furthermore, the figures may have properties like small, shaded, thin, etc. and they may be in some spatial relation like in, left, above, etc.

Note that the expressions of the VREG language, introduced in chapter 3 and 4, can describe the figures that are covered by ANALOGY. In fact, the VREG language includes the compose operator such that the composition structure of figures can be described and represented by the compose expressions of the VREG language. However, although the VREG language does not include a connect operator, the connect structure of figures can be described and represented by using the single-linkage strategy to compute the proximity structure (introduced in chapter 4 section 3) and demand that two visual elements are in one proximity cluster if and only if their closest distance is zero, i.e. two visual elements are in one proximity group if they share at least one point. In this way, a proximity group is considered as a connected figure.

ANALOGY consists of two parts. The input of part 1 is the primitive descriptions of the eight given figures. For example, a triangle and a point within it (i.e. a non-connected figure) is represented as :

((DOT (0.4 , 0.8)) ((SCC ((0.3 , 0.2) (0.7 , 0.2) (0.5 , 0.7) (0.3 , 0.2)))))

The primitive description of figures is then the subject of a decomposition process. The decomposition process will divide a connected figure into its connected parts; non-connected figures are already represented as divided elements. Note that the decomposition process can only be applied meaningfully to REG figures; the DOT and SCC figures are not decomposable. The decomposition process is able to find all occurrences of an arbitrary simple closed figure x in an arbitrary connected figure y and will divide figure y into occurrences of x and the rest of y. At the next step, a set of properties and relations of the objects that are resulted from the decomposition process is computed. For example, the inside relation between a dot and a polygon is expressed as INSIDE(DOT(x, y),SCC((x_1, y_1),...,(x_n, y_n))).
The important step of part 1 is a set of "similarity" calculations. Given a certain class of transformations like change-size, change-texture, rotate, etc. the similarity calculations determine for each pair of objects those transformations that map one object of the pair to the second object of the pair.

The decomposition of figures into objects together with the computed properties, relations and the similarity transformations form the input of part 2 of ANALOGY. The first step of part 2 generates a transformation rule which transforms Figure A onto Figure B. A transformation rule, which is based on similarity transformations from part 1, specifies how the objects of Figure A are removed, added to, or altered in their properties and relations to other objects to generate Figure B. Subsequently, the transformation rules are generalized such that in addition to transforming figure A onto figure B, they will transform figure C onto exactly one of the solution figures. The rules must be *general* enough to choose at least one of the answer figures and *specific* enough to choose not more than one. The generalized rule must have the property such that "The numbers of parts added, removed and matched in taking A onto B and in taking C onto the answer figure are the same".
If there is more than one possibility, the best rule is selected according to a so-called *strength* function. This function assigns a value to each rule and thereby it sorts all possible transformation rules. The "best" or "strongest" rule is the one that is achieved by the least alteration in the original $A \to B$ rule that still maps C onto exactly one answer figure.

The notion of "strongest" rule is then understood as how little alteration is needed to extend the theory of $A \to B$ to a new situation, i.e. $C \to X$.

5.5.2 Copycat

Copycat solves proportional analogy problems about strings of letters. A string of letters is either an individual letter from the western alphabet or generated by concatenating two or more strings of letters. In Copycat, string patterns may be related to each other by means of a number of predefined relation (called concepts) like left-of, right-of, successor, predecessor, etc. Note that the terms of the Extended-String-algebra, introduced in chapter 3, are descriptions of these strings.

In dealing with a proportional analogy problem of the form "$A : B = C : X$", Copycat conducts a non-deterministic parallel search for structuring of the letter strings and finds mappings between them in order to solve the analogy. The structures that Copycat builds are descriptions of bonds between substrings within a string (only between neighboring substrings), groups of substrings within a string, correspondences between substrings in different strings, a rule specifying the transformation between the string A and the string B, and a translated rule which transforms the string C into an answer string.

The structural description of strings and the analogical mapping between them are constructed on the basis of concept-related system parameters. These parameters are for instance the strength of various relations ("conceptual depth") or the strength of transformations between relations ("slippage"). The values of these parameters determine the appropriateness of descriptions and analogical mappings. For instance, an activation value that is assigned to a relation indicates the importance of that relation for a generated structural description of a string. The system parameters have some pre-assigned values at the beginning of the process of solving proportional analogy problems. These values may change during the process.

The Copycat descriptions of strings are built out of the highly activated relations. When the (partially) constructed descriptions do not provide a proper analogical relation, the activation values may change in order to create different descriptions of strings. The appropriateness of analogical relations is based on a notion called the *slippage* of concepts.

In Copycat, predefined relations are conceptually close or far from each other. The conceptual distance between pairs of relations, called *conceptual proximity*, are determined beforehand. The idea is that the closer relations are more likely to be slipped into each other than the farther relations. For example, the “successor” relation may slip more likely into the “predecessor” relation than into the “reflect” relation. Since analogical relations are defined in terms of transformations between relations and because transformations between conceptually close relations are more likely than transformations between conceptually far relations, those analogical relations that are based on more likely transformations are considered as being more appropriate.

In general, the system attempts to integrate the most important relations in order to build the proper description of the strings such that an appropriate analogical relation between them can be constructed. At any point, the computed description of strings, mappings and other structures may change again in order to create a more stable state of the system. The stability of the system is measured by a global parameter called “temperature” which in turn is computed by the values of other system parameters.
Note that the temperature parameter in Copycat is not used as an independent parameter on the basis of which the change of the activation values of other system parameters and thus the amount of change in the system can be determined. In fact, the temperature parameter in Copycat is used the other way around, i.e. the activation values of system parameters (except the temperature value) are used to determine the activation value of the temperature parameter and thus the stability of the system state.
One state of the system is more stable than another when the temperature value of the first is lower than the temperature value of the second. The system stops by a probabilistic decision using the value of the temperature parameter. The state in which the system stops provides the description of strings and the mapping between them.

In Copycat, some activation values are assigned at the beginning of a run and will change during the run. Some other activation values are fixed. The values that are pre-assigned at the beginning of a run are chosen by a combination of intuition, trial and error, and some arbitrariness. These activation values will be changed by the mutual influence in a probabilistic way causing a change in description of strings or in mapping between them.

5.5.3 Limitations of ANALOGY and Copycat

In both ANALOGY and Copycat, there are some shortcomings that we have tried to overcome. These shortcomings are related to the methods that are used to decompose perceptual patterns into subpatterns. As noted in this chapter, a proper decomposition is crucial since the description of the involved perceptual patterns and the analogical relation between them are defined in terms of the resulting subpatterns. In particular, we assume that the decomposition of perceptual patterns involved in proportional analogy problem is appropriate when the interaction between perception and context effects are modeled properly.

ANALOGY computes the decomposition of figures (and thereby their descriptions) in a separate initial module. Once these descriptions are computed, ANALOGY generates an analogical relation between them. Although Evans was aware of both context-sensitivity and gestalt phenomena in generating the description of the involved figures and an analogical relation between them, his system dealt with them in a very limited way [Ind92] (pp. 378-384). The only gestalt phenomenon he deals with is connectivity: a REG figure must be decomposed into simple-closed-curves as much as possible. Other gestalt phenomena like good continuity, foreground/background, proximity, embeddedness, symmetry, etc. are ignored. In decomposing REG figures, ANALOGY deals with the context-effect also in a limited way. The modeling of the context-effect is related to the fact that ANALOGY is able to find all occurrences of an arbitrary simple closed-curve (SCC) in an arbitrary REG figure and can divide the REG figure into occurrences of that simple closed-curve and the rest. This ability is used to decompose one REG figure based on other figures involved in proportional analogy problem. Although it is not explicitly mentioned, it seems that the context-effect is modeled by assuming only one REG figure in the proportional analogy problem; other involved figures should unambiguously consist of simple closed-curves. This implies that the mutual contextualization effects is not modeled in ANALOGY properly, i.e. figure x can be used to decompose figure y, but figure y cannot be used to decompose figure x. Thus, in ANALOGY only one of the involved figures may have gestalt ambiguity. In order to demonstrate the mutual contextualization effect, consider the proportional analogy problem illustrated in Figure 5.8.

In this proportional analogy problem, the three figures have gestalt ambiguity such that the decomposition method used in ANALOGY cannot use

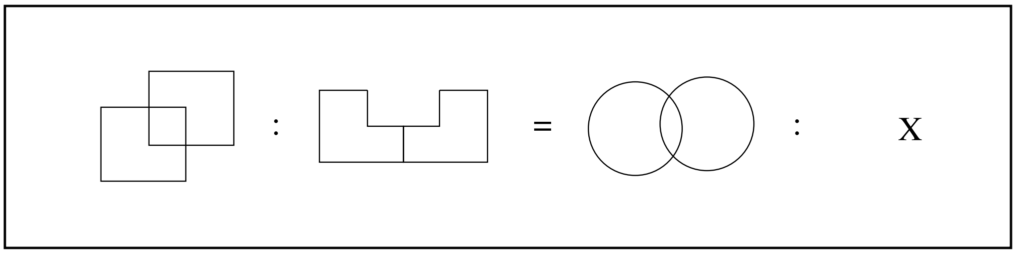


Figure 5.8: *Mutual contextualization effect between involved line patterns.*

the unambiguous gestalt of one figure to decompose the other figures. In fact, to solve the proportional analogy problem illustrated in Figure 5.8, there is no way to determine the proper decomposition of figures at once and in an initial module. The possible decompositions (and thus the possible descriptions) of figures should be disambiguated by trying to construct an analogical relation between them.
The fact that ANALOGY computes the description of figures in an initial module makes ANALOGY a system which is not capable of solving *creative analogies* [Ind92]. In non-creative analogies, the descriptions of the source and the target elements are given or computed once, and it is assumed that these descriptions already contain the similarity that will constitute the basis of the analogical relation. However, in creative analogies, the (conventional) descriptions of the source and the target may not provide the needed similarities. In fact, the similarity has to be created by building these descriptions.

In our system, on the other hand, our main goal has been to model the contextualization effect, and also because our system is based on the structural information theory, for which a considerable empirical support has been found, we feel that it is much less *ad hoc.*

The Copycat system was expressly designed to model the creativity phenomenon in proportional analogies. Consequently, in Copycat, representations of the terms are constructed hand in hand with the mappings, and thus the mutual contextualization effect is fully taken into account. However, many of the features of Copycat are not clearly, or formally, specified. Neither it is clear how the Copycat approach can be applied to other domains like the domain of visual patterns.
For example, consider the concept of *temperature,* that plays a key role in the Copycat architecture. Intuitively, the idea is that the 'deeper' or more

cognitively appealing an analogy, the lower its temperature (which is based on an analogy with thermodynamics). However, nowhere in the Copycat architecture, or in its discussion, one finds any principles or rules or any explicit description for computing the temperature of an analogy relation. In fact, as the concept of information load in the Structural Information Theory can be considered analogous to Copycat's temperature, this reveals starkly the contrast between our two approaches, for the focus of our research has been on explicating the principles that constrain the gestalts of a pattern, both in isolation and in context.

Another point to emphasize is that our algebraic model is aimed at modeling only the input-output functionality of the human perceptual process. That is, we do not claim that people carry algebraic descriptions in their heads, or that the algorithm presented in the previous section mirrors in anyway how humans solve proportional analogy. By contrast, Copycat implicitly claims to model the human perceptual processes. However, no scientific reason or evidence to support this claim is offered besides the authors' intuitions that it must be so.

In this and all previous chapters, we have concentrated on modeling gestalt perception to determine the gestalts of perceptual patterns. However, as we noted in the first chapter, we may use a model of the human visual perception to generate visual patterns as well. In this way, one can guarantee that the generated visual patterns will have certain perceptual structures. In the next chapter, we will discuss the role of such a model in data visualization where visual patterns with certain perceptual structures have to be generated.

Chapter 6

The Role of Perception in Data Visualization

In this chapter, we discuss an application domain in which perception plays an important role. This application domain is often called *data visualization* [Twy79, Ber81, Tuf83, Tuf90]. The aim of data visualization is to generate visual patterns that represent data. These visual representations denote data relations by means of perceivable relations that exist among their constitutive visual elements. For example, the number of cars produced by different factories can be visualized by a bar-chart. Each bar represents a car factory and its length represents the number of produced cars by that factory. The perceivable length relation between bars denotes then the relation between numbers of cars produced by different factories.

Data can of course also be represented in many other ways, for instance by means of natural language expressions or first-order predicate sentences; the advantage of representing data by visual patterns is often claimed to be the *effectiveness* of this mode of representation. However, it has not yet been attempted to articulate the effectiveness property in a formally precise way. We argue that the effectiveness of visual representations is related to the perceptual structures that represent the structure of data. In fact, we argue that data is effectively represented if the intended structure of the represented data and the perceived structure of the representing visual pattern coincide.

In the previous chapters, we have focused on the gestalts of visual patterns, i.e. the perceived constituent structure of visual patterns which determines how compound visual elements are built out of primitive visual elements. We argued that a gestalt of a visual pattern is constructed based on either the regularity of perceivable relations among its constitutive visual elements or the relative closeness of those elements (proximity). It is explained that the perceivable relations on visual elements are induced by the involved visual attributes.

In this chapter, we consider a broader class of perceptual structures of visual patterns which includes, besides the perceived constituent structure of visual patterns (gestalts), some additional perceived structures. These additional perceived structures determine how the involved visual elements are related (e.g. qualitatively, quantitatively, topologically, etc.) to each other. In particular, we study, besides the part-whole relation, some other perceivable relations (e.g. qualitative, quantitative, topological, etc.) that are induced by visual attributes of *primitive visual elements*. We emphasize primitive visual elements since we will ignore the perceivable relations that are induced by the characteristic visual attributes of the compound visual elements. A characteristic attribute of a compound visual element indicates a property of that compound visual element which is not a property of the constitutive primitive visual elements. In fact, we admit, but ignore, the characteristic visual attributes for compound visual elements.
In this chapter, we will use the term *domain structures* for all perceptual structures of visual patterns except the perceived constituent structure for which we use the term gestalt. Thus, while a gestalt is defined as a set of visual elements that are related to each other by the part-whole relation, a domain structure is defined as a set of visual elements that are related to each other by a set of relations except the part-whole relation.

The purpose of this chapter is twofold. On the one hand, we study some important domain structures by analyzing different classes of perceivable relations that can be induced on visual elements by visual attributes. On the other hand, we examine a practical application for which a model of perception is indispensable. This application is data visualization. We show that the effectiveness of visual representations can be guaranteed if the perceptual structure of visual representations is modeled appropriately.
In section 1, we concentrate on data visualization by defining the class of data structures that we want to visualize, the projection of data structures to perceptual structures, the layout of visual patterns based on perceptual structures, and finally the effectiveness criterion.

This approach to data visualization will be formalized in sections 2 and 3 of this chapter. In section 2, we formally define (data and visual) attributes and the relations that an attribute may induce on (data and visual) elements. Some general types of attributes that are important in data visualization are introduced. The type of a visual attribute indicates the class of perceivable relations that it induces on visual elements.
In section 3, we introduce a formal framework for effective data visualization. In this framework, data and visual patterns are defined as relational structures. The relation between these structures is then defined by means of a formal mapping between data and visual relational structures. In effective data visualization, the mapping between represented data and representing visual patterns is a structure-preserving mapping. The structure-preserving condition guarantees that the structural properties of data relations and the structural properties of perceivable relations coincide. This condition guarantees also that whenever there exists regularity among relations on the data side, there will exist an identical regularity among corresponding perceivable relations on the visual side and vice versa. This property is important when visualizations are used for data mining. Finally, we elaborate on data visualization and consider various visualization systems as visual languages. In this way, we consider various types of maps, diagrams, graphs, flow-charts, etc. as different visual languages. A visual pattern such as a map or a diagram is then considered as an expression of a visual language.

6.1 Automatic Data Visualization

In this section, we focus on an application in which domain structures as well as gestalts of visual patterns play essential roles. In this application, called *automatic data visualization*, visual patterns are generated that *effectively* represent data [Mac86, Kam97, RKG95, Mit93, PG93]. We define the effectiveness of visual representation as follows: a visual pattern represents data effectively if the *intended* structure of the represented data and the *perceptual* structure of the representing visual pattern coincide.
In particular, we consider the process of data visualization as the inverse of the interpretation process, i.e. given the data to be represented, the visualization process generates a visual pattern the interpretation of which results in the original data. Based on this view, we assume a process model for data visualization which consists of a number of components and the mapping between them. This model is schematically illustrated in Figure

6.1. In the rest of this section, we will briefly explain these components and the mappings between them.

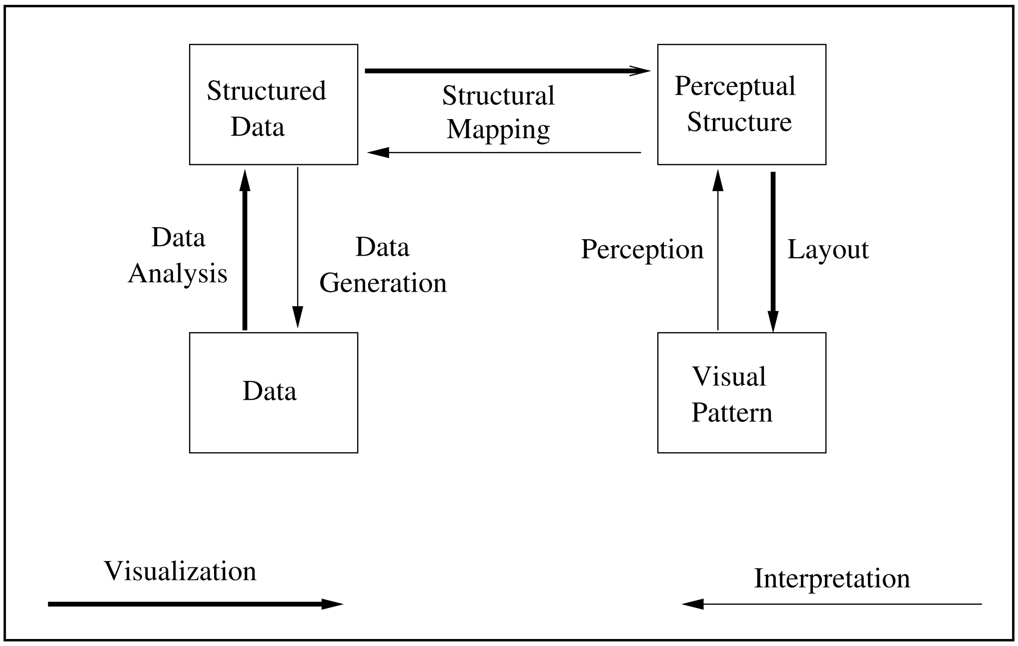


Figure 6.1: *A model for data visualization.*

In this model, the input data is a nested relational database. We will present a nested relational database as a table of attribute values. According to this model, the table of attribute values should be analyzed to determine its (intended) structure. The process of *data analysis* results in structured data. In this way, data is considered in terms of data elements (i.e. n-tuples of attribute values) that are related to each other by means of a number of relations. Thus, the process of data analysis determines how the n-tuples of attribute values from the data table should be related to each other by means of explicitly represented relations. In most visualization systems, this process is avoided by assuming these relations, i.e. the starting point is an annotated table of attribute values rather than a plain table. An annotated table is a table of attribute values in which the structure of the domain of values of each attribute is explicitly given by specifying the relations applicable to that domain.

Given the structured data presented as an annotated table, the visualization process proceeds with specifying a visual attribute for each data attribute

and specifying a one-to-one mapping between data elements and abstract visual elements. An abstract visual element is an n-tuple of variables each of which stands for a value of one of the specified visual attributes. In fact, the one-to-one mapping between data elements and abstract visual elements consists of a number of one-to-one mappings each of which maps values of one data attribute to a set of variables that stands for values of one specific visual attribute. We assume that the specification of visual attributes together with the specification of the one-to-one mapping results in a set of related abstract visual elements. The relations between abstract visual elements are determined by selecting visual attributes since each selected visual attribute will induce some relations on visual elements.
Thus, at this stage of the visualization process the perceptual structure of the to-be-generated visual pattern is known, while the actual values of visual attributes that are needed to draw the visual pattern are not known. In the case of effective data visualization, this perceptual structure must coincide with the data structure. For this reason, we claim that this stage of effective data visualization process results in a *structure preserving mapping* between the structured data and the perceptual structure.

In order to generate a visual pattern that has the specified perceptual structure, the variables that stand for visual attribute values should be instantiated with concrete attribute values. This process is therefore a constraint satisfaction process since the generating visual attribute values, and thus the generating visual elements, should be in accordance with the perceptual structure which is imposed on visual elements at the previous step. It is important to note that a visual element can be drawn if and only if the values of certain visual attributes are specified. We explained that each data attribute is mapped to one visual attribute. But, it may be the case that some visual attributes, the values of which are necessary to draw visual elements, are not used in the constructed mapping. For instance, if the position attribute is not used in the mapping then the visual elements should receive position values in order to become drawable. These unused visual attributes will be called *undecided attributes* in contrast to the *decided attributes* to which data attributes are mapped.
The determination of values for both decided and undecided visual attributes will be called the *layout process*. It should be noted that the instantiation of values for the decided visual attributes is usually not considered as a part of the layout process since these values are determined by the data. However, we have two reasons for considering the determination

of decided and undecided visual attributes values as two functions of the same (layout) process.
The first reason is based on the fact that the instantiation of the actual values for both kinds of (decided and undecided) visual attributes are constrained by the specified perceptual structure. In particular, the values for the undecided visual attributes cannot be chosen randomly since the random choice for these visual attribute values may induce perceptual structures that do not represent any data structure. This phenomenon is sometimes called *unwanted visual implicatures* [MR90, BS93].
The second reason is that the specified perceptual structure leaves still a spectrum of possible values for both decided and undecided visual attributes. In particular, the constraints that are imposed by the perceptual structure do not determine the values for the decided visual attributes uniquely. For example, when the color hue attribute is decided on for a data attribute, the choice for the actual color hue values for visual elements is still an open choice. Thus, given four visual elements such that two of them should have identical color hue values and the other two should have different color hue values (the constraint imposed by the specified perceptual structure), one may decide on red, green, and yellow color hue values, while someone else may decide on brown, orange, and black color hue values. Based on these two reasons, we will consider the determination of both decided and undecided attribute values as two functions of the same (layout) process.
The decisions that determine a visual attribute for each data attribute and a one-to-one mapping between data elements and visual elements from the previous step together with the instantiation of values for variables at this step result in what is called the *interpretation function* or the *legende* of visual representations.

As noted, we consider the process of effective data visualization as the inverse of the interpretation process. In particular, the interpretation process starts with a given visual pattern and an interpretation function (legende). Since the object of the interpretation process is a visual pattern and because the visual pattern must be perceived before it can be understood as denoting data, the first step in interpreting a visual pattern is its *perception*. The process of perception determines the perceivable relations among visual elements on the basis of their visual attribute values. Since visual attribute values are assigned to visual elements by the layout process, one may conclude that perception is the inverse of the layout process. Subsequently, visual elements that are related to each other by perceivable relations are mapped to related data elements. This mapping is accomplished by means of the given interpretation function (legende) which determines the correspondence between visual and data attributes and the correspondence between visual and data values. In the case of effective data visualization, the relation between data elements are structurally identical with the perceivable relations between their representing visual elements. Finally, given the structured data, the table of attribute values can be trivially generated. This process is called *data generation*. In the rest of this section, we elaborate on different stages of the data visualization process.

6.1.1 Input Data

In this section, we focus on one important class of input data: (nested) relational data bases. Such input data can be presented by a (nested) table which is constructed by means of n data attributes. A column of such a table consists of values of one data attribute and a row of it consists of n values each of which belongs to one data attribute (i.e. a row forms a n-tuple of attribute values). In nested tables, the values of an attribute may be a row of another nested table which is constructed by means of m data attributes. Thus, the rows of a nested table may consist of n data attribute values each of which may consists of m data attribute values, etc. An attribute value which is a single value rather than consisting of m attribute values is called an *atomic value*.
A row of a table (i.e. a n-tuple of attribute values) at a certain nesting level is called a *data entry*. An attribute value is assigned to a data entry by means of one of the involved data attributes. Consequently, in nested tables the values that are assigned to data entries by an attribute may be data entries from a nested table. For example, consider the *Car-state* data table which is illustrated in Figure 6.2.

This data table describes the co-operation degree between pairs of car companies. In this data table, *Co-operating, Co-operated,* and *Co-operation-degree* are three data attributes. The values that the *Co-operating* and the *Co-operated* attribute assign to each data entry are rows of one nested data table. These nested rows are indicated by the key values of the nested data table. This nested data table is constructed by three data attributes and is called the *Company-state* data table. The *Company-state* data table is illustrated in Figure 6.3. This table describes the number of sold-cars of

	Co-operating	*Co-operation-degree*	*Co-operated*
d_1	d'_1	Good	d'_2
d_2	d'_1	Best	d'_3
d_3	d'_2	Good	d'_1
d_4	d'_3	Best	d'_1
d_5	d'_3	Normal	d'_4
d_6	d'_4	Normal	d'_3

Figure 6.2: *The Car-state data table.*

	Car-type	*Sold-number*	*Production-country*
d'_1	VW	40000	Germany
d'_2	Toyota	50000	Japan
d'_3	Opel	30000	Germany
d'_4	Mazda	20000	Japan

Figure 6.3: *The nested Company-state data table.*

a certain type that are produced in a certain country. In this nested data table, *Car-type, Sold-numbers,* and *Production-country* are three data attributes. The values that these attributes assign to each nested data entry are atomic values.

We distinguish two kinds of data structures that are important in data visualization. The structures of the first kind are constituted by relations that are defined on values of individual data attributes, i.e. relations that are defined on the set of values from one column of a data table. The structures of the second kind are constituted by binary relations that are defined on values of two different attributes with the same range, i.e. relations that are defined by two columns of a data table that contain values from one and the same set.
In order to cover the first kind of data structures, we assume that the set of possible values of each data attribute is a *structured set.* Moreover, we assume that the structure of a set of attribute values is given by assigning the constitutive relations of that structure to that attribute. For instance, values of the *Sold-number* attribute are from the domain of integers where integers are related to each other by arithmetical relations. These arithmetical relations are assigned to the *Sold-number* attribute.
The set of relations that exist among values of an attribute is called the *type* of that attribute. A data attribute together with its type is called a *typed attribute.* Finally, a data table that is defined on typed attributes is called a *typed data table.*

In a data table, relations that are defined on values of individual attributes (the type of attributes) are induced on data entries. In this way, two data entries are related since the values of their individual attributes are re-

lated. These induced relations on data entries constitute structures of the first kind. Relations on data entries may be induced by different data attributes. For instance, data entries in the *Company-state* data table may be considered as equivalent or non-equivalent depending on the value of the *Car-type* attribute, while they may be quantitatively related to each other according to the *Sold-number* attribute (e.g. the number of sold VW's is twice the number of sold $Mazda$'s, or the number of sold $Toyota$'s is equal to the number of sold $Mazda$'s and *Opel*'s together).
In a data table, binary relations that are defined on values of two attributes with the same range relate data entries as well. The resulting relations on data entries constitute structures of the second kind. For instance, the binary relation defined on the values of the *Co-operating* and the *Co-operated* attributes (which have the same range) can be represented as:

$$\{(d'_1, d'_2), (d'_1, d'_3), (d'_2, d'_1), (d'_3, d'_1), (d'_3, d'_4), (d'_4, d'_3)\}.$$

This binary relation relates data entries of the *Car-state* data table.
Binary relations can be characterized by their structural properties like functional (injective, surjective, bijective), transitive, symmetric, reflexive, etc. For example, in the *Car-state* data table the values of the *Co-operating* and the *Co-operated* attribute define such a binary relation which is non-functional, symmetric, non-reflexive, and non-transitive.

6.1.2 Projecting Data to Visual Patterns

The main step in visualizing data is the projection of the structured data (i.e. presented by typed nested tables), derived from the *data analysis* step, into a *perceptual structure*. A perceptual structure is presented by a typed nested table which is constituted by means of typed visual attributes. The type of a visual attribute is determined by the relations that can be perceived among values of that attribute. A typed table which is constructed by typed visual attributes is called *typed visual table*. The rows of a visual table will then be called *visual entries*.
The projection of the structured data into a perceptual structure consists in defining a map from the typed data attributes to typed visual attributes and another map from rows of the data table (i.e. data entries) to the rows of the visual table (i.e. visual entries). In this way, the number of visual attributes and the number of visual entries are the same as the number of data attributes and the number of data entries, respectively.

We assume that this projection does not specify the actual attribute values of visual entries, but it only specifies visual entries by means of variables that stand for visual attribute values. Thus, this projection specifies abstract visual entries that are related to each other by perceptual relations. In fact, a resulting typed visual table represents an abstract perceptual structure that can be drawn by deciding on the actual values for the variables involved.

For example, given the *Car-state* data table, one may decide on the *Connecting* visual attribute for the *Co-operating* data attribute, *Connected* for *Co-operated*, *Size* for *Co-operation-degree*, *X-pos* for *Car-type*, *Y-pos* for *Sold-number* , and the *Color-hue* visual attribute for the *Production-country* data attribute. Moreover, six abstract visual entries are specified for the six data entries. This projection of typed data attributes and data entries to typed visual attributes and abstract visual entries results in the typed visual table which is illustrated in Figure 6.4 and 6.5.

	Connecting	*Size*	*Connected*
v_1	v'_1	$Size_2$	v'_2
v_2	v'_1	$Size_3$	v'_3
v_3	v'_2	$Size_2$	v'_1
v_4	v'_3	$Size_3$	v'_1
v_5	v'_3	$Size_1$	v'_4
v_6	v'_4	$Size_1$	v'_3

Figure 6.4: *The visual table for Car-state data table.*

	X-pos	*Y-pos*	*Color-hue*
v'_1	seg_1	pos_4	hue_1
v'_2	seg_2	pos_5	hue_2
v'_3	seg_3	pos_3	hue_1
v'_4	seg_4	pos_2	hue_2

Figure 6.5: *The visual table for the nested Company-state data table.*

Different typed visual attributes can be decided for a typed data attribute. Similarly, different visual entries can be specified for each data entry. These different ways of deciding visual attributes and specifying visual entries result in different visualizations of data. The set of decisions on the basis of which visual attributes (entries) are related to data attributes (entries) is often called the *interpretation function*. The interpretation functions are usually expressed by either legends which are attached to visual patterns or they are integrated in visual patterns by inserting textual information.

6.1.3 The Layout of Visual Patterns

In the previous section, we explained that the typed data table is projected to a typed abstract visual table in which visual entries are constituted by variables. The generation of the actual attribute values for visual entries is accomplished by the layout process. Moreover, since the number of visual attributes is determined by the number of data attributes, the layout process should not only generate values for the *decided* visual attributes, but it should also generate values for the *undecided* visual attributes. In fact, undecided visual attributes do not represent any data, but their values should be specified in order to draw visual patterns. For example, consider visual patterns illustrated in Figure 6.6.

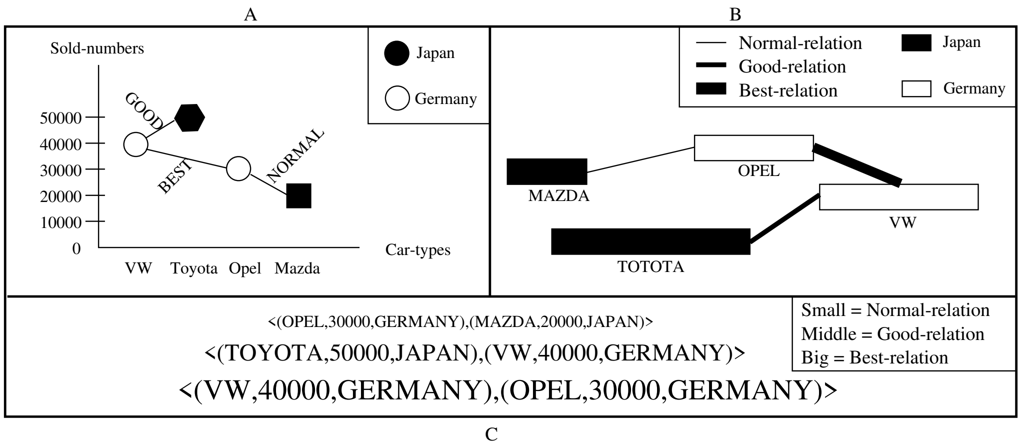


Figure 6.6: *Visualizations of the Car-state data table.*

In the visualization illustrated in Figure 6.6-A, the shape and the size of visual elements are undecided attributes, i.e. the shape and the size of visual elements do not represent any data attributes. Similarly, in the visualization illustrated in Figure 6.6-B the position and the shape attributes are undecided visual attributes. Finally, in the visualization in Figure 6.6-C the position attribute is a undecided visual attribute. In this visualization, the label attribute is decided for the *Car-type, Sold-number,* and the *production-country* attributes.

Note that the textual information that is inserted in these visualizations may either be values of a visual label attribute or it may be a part of the interpretation function. In order to distinguish these two kinds of textual information, we use only capital letters to indicate that a textual information is a value of the visual label attribute rather than a part of the

interpretation function.
For example, in the visualization illustrated in Figure 6.6-A the interpretation function is partially expressed by the legend that is attached to the upper-right corner of the visualization, but it is also partially expressed by inserting textual information into it. In this way, the textual information like $VW, Toyota, Opel$, and $Mazda$ are the interpretation values of different segment values. Similarly, the textual information like *Car-type, Sold-number,* and the numbers attached to the vertical axis are the interpretations of the horizontal axis, the vertical axis, and the positions of the vertical axis, respectively.

6.1.4 Effective Data Visualization

A nested data table can be visualized in many different ways by deciding different visual attributes for each data attribute and by specifying different visual entries for each data entry. In Figure 6.7, two different visualizations for the *Car-state* data table are illustrated.

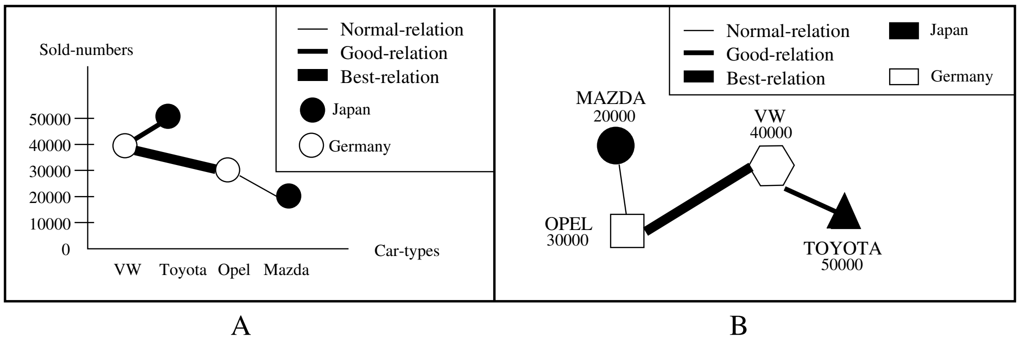


Figure 6.7: *Alternative visualizations of the Car-state data.*

In the visualization illustrated in Figure 6.7-A, the horizontal axis visualizes the *Car-type* attribute, the vertical axis visualizes the *Sold-number* attribute, and the *Color-hue* attribute visualizes the *Production-country* attribute. Moreover, the links between visual elements defined by the *Connecting* and the *connected* attribute visualize the *Car-company* attributes. Finally, the thickness of the links visualizes the degree of co-operations.
In the visualization illustrated in Figure 6.7-B, the *shape* attribute visualizes the *Sold-number* attribute, the *Color-hue* attribute visualizes the *Production-country* attribute, and the *label* attribute visualizes the *Car-type* attribute. Like the visualization illustrated in Figure 6.7-A, the links

and their thickness visualize the co-operations and their degrees between car companies, respectively. Note that in this visualization the inserted numbers are the interpretation values.

Although these two visualizations illustrated in Figure 6.7 represent the same data, the visualization illustrated in Figure 6.7-B is not an *effective visualization*. The notion of effective visualization is studied in [Ber81, Mac86]. In these studies, a visual pattern is claimed to represent an information state effectively if human perceiver can easily and without much cognitive effort extract the represented information by means of its interpretation function.
We consider the effectiveness of visualizations of data as being related to the intended relations of the data that should be visualized. The pragmatic issues of visualizations [Cas91] such as the use or the purpose of visualizations are then considered as the specification and the importance ordering of the intended data relations that should be visualized. Accordingly, a visualization of a data will be considered effective if the *intended* data relations are visualized by structurally identical *perceivable* relations. For instance, if one intends to visualize the number of sold-cars such that not only the exact number of sold-cars, but also their proportional relations or absolute difference relations can be perceived, then one should decide for the *Sold-number* attribute a visual attribute which induces structurally similar perceivable relations. In the visualization illustrated in Figure 6.7-B, the shape attribute is decided for the *Sold-number* attribute. However, the perceivable relation induced by the shape attribute is an identity relation. This implies that one may perceive that the numbers of sold-cars are equal or not. Since the intention is to visualize the quantitative relations between data entries, the visualization in Figure 6.7-B is considered as a non-effective visualization, i.e. the intended quantitative relations between data entries are not visualized by structurally similar perceivable relations between corresponding visual elements.

It is important to note that one may deduce the intended quantitative relations indirectly by means of the interpretation values that are inserted into the visualization in Figure 6.7-B. For example, one may use the interpretation values 20000 and 40000 that are attached to visual elements and deduce that the second is twice the first. However, this reasoning is based on the knowledge of the perceiver about the structure of real numbers rather than being based on perceivable relations. The effectiveness of visualization is the ability to perceive data relations *directly* by structurally similar perceivable relations. In order to illustrate the direct perception of relations, consider the visualizations in Figure 6.8.

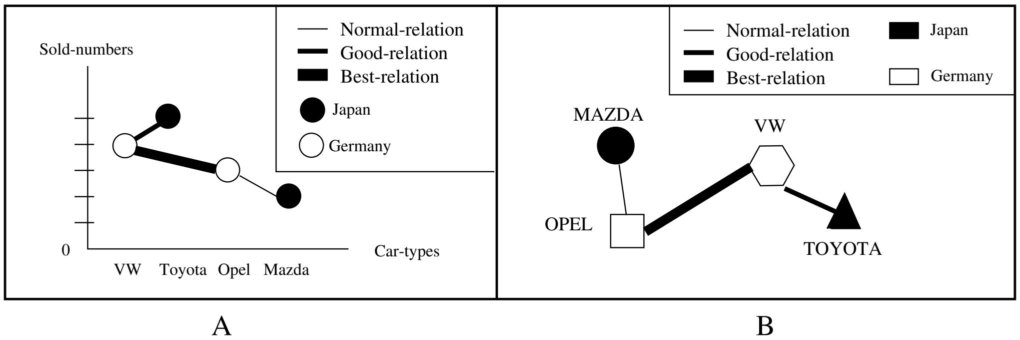


Figure 6.8: *Visualizations without any interpretation values.*

These visualizations are the same as visualizations in Figure 6.7 except that in these visualizations the interpretation values are left out. Knowing that the shape attribute visualizes the *Sold-number* attribute in Figure 6.8-B, it is impossible to perceive any quantitative relation between visual elements and therefore it is impossible to conclude any quantitative data relation. However, knowing that the vertical axis visualizes the *Sold-number* attribute in Figure 6.8-A, it is easy to perceive quantitative relations between visual elements and therefore easy to conclude quantitative data relations. In the rest of this chapter, whenever we refer to the perceptual structure of visualizations we mean the structure of visualizations without using any interpretation information.

In the above examples, we discussed how the effectiveness of data visualizations could be influenced by deciding different visual attributes for data attributes. However, the specifications of visual entries by the layout process may influence the effectiveness of visualizations as well. As noted, the layout process should specify values for both decided and undecided visual attributes.

In the context of effective data visualization, the generation of values for the decided attributes of visual elements is constrained by the perceivable relations. In fact, the values for the decided attributes of visual elements should induce perceptual relations that are structurally similar to the intended data relations.

For example, in the visualization illustrated in the Figure 6.7-A the *Color-hue* attribute is decided for the *Production-country* attribute and four circle shaped visual elements are specified for the *Company-state* data entries. The values of the *Production-country* attribute induce a classification relation on the data entries of the *Company-state* relation. Therefore, the layout process has generated the values of the *Color-hue* attribute of circle shaped visual elements such that they have identical *Color-hue* values whenever their corresponding data entries have identical *Production-country* values.
Like the values of the decided visual attributes, the values of the undecided attributes for visual elements can not be chosen arbitrarily since otherwise the effectiveness of visualization is not guaranteed anymore. For example, consider the visualization in Figure 6.6-A. Although this visualization represents the same information as the visualization in Figures 6.7-A it can not be claimed that the diagram in Figure 6.6 represents the same information effectively. The reason is that a human perceiver may perceive a classification relation between visual elements that is induced by the undecided *Shape* attribute and thereby conclude that this classification relation may represent a data relation. As noted, this phenomena is called unwanted visual implicatures.
Since in drawing visual patterns the values of decided and undecided visual attributes must be specified and in order to avoid unwanted visual implicatures, the values of undecided visual attributes should be specified in such a way that they induce an identity relation on visual elements. In other words, one may avoid unwanted visual implicatures by grouping the involved visual elements according to the undecided visual attributes into one perceptual group. In this way, visual elements will be classified by the undecided visual attributes as belonging to the same perceptual group.
In chapter 3, we argued that the gestalts of visual patterns determine the perceptual groupings of visual patterns. Accordingly, the layout process should generate the values of the undecided visual attributes such that the gestalts of visual patterns according to the undecided visual attributes reflect only one perceptual group. Note that the gestalts of the diagram illustrated in Figure 6.7-A reflects one perceptual group according to the *shape* attribute (i.e. a undecided attribute).

6.2 A Formal Classification of Domain Structures

In this section, we study perceivable relations that are induced on primitive visual elements by means of their visual attribute values. As noted in the introduction of this chapter, we admit but ignore the perceivable relations that are induced by means of values of the characteristic visual attributes of the compound visual elements. The values of the characteristic visual attributes of compound visual elements are not the values of their constitutive primitive visual elements. For example, different groups of visual elements (i.e. a compound visual element) may each have a label. These labels induce an equivalence relation on compound visual elements such that compound visual elements with identical labels are considered as equivalent while those with different labels are considered as non-equivalent.

The perceivable relations that are induced by means of visual attributes of primitive visual elements constitute domain structures of visual patterns. For example, color saturation values (of one color hue value) induce an ordering relation on primitive visual elements such that a primitive visual element with a certain saturated color value is perceived as less/equal/more saturated than another primitive visual element.
Different visual attributes may induce different perceivable relations on primitive visual elements. This implies that different visual attributes may result in different domain structures.
Domain structures can be subdivided into different types. In order to study these different types of domain structures, we define attributes (visual or non-visual) formally and introduce four general types of attributes. The type of an attribute is defined in terms of relations that may be induced on the involved primitive elements by that attribute.
Then, visual attributes are classified according to the introduced attribute types. In this way, the relations that define the type of a visual attribute characterize the perceptual relations that the visual attribute may induce on primitive visual elements.

In order to define attributes in a formal way, we use some notions from *measurement theory*. The theory of measurement provides both a classification of attributes and their mathematical definitions by introducing different types of *measurement scales* [Ste46, Pfa68]. In fact, a measurement scale is a map from a structured set of elements to a structured set of values like the set of real numbers, the set of integers, a set of strings, etc. In measurement theory, relational systems are used to represent structured sets.

26. Definition. *A relational system is a pair* $< A\ ;\ R_1, \ldots, R_n >$ *where* A *is a set of elements, and* $R_1, \ldots, R_n$ *are relations defined on* A.

In this chapter, we consider attributes as measurement scales; i.e. an attribute maps a structured set of primitive elements into a structured set of values, called the set of attribute values.

27. Definition. *An attribute is a homomorphism* m *from a relational system* $< A\ ;\ R_1, \ldots, R_n >$ *into a relational system* $< B\ ;\ S_1, \ldots, S_n >$. *The set* A *is the set of primitive elements and the set* B *is the set of attribute values which may be the set of real numbers, the set of integers, a set of strings, etc.*

In the case of a visual attribute, A is a set of primitive visual elements, $R_1, \ldots, R_n$ are perceptual relations that are induced on primitive visual elements in A by that visual attribute, B is a set of values, and $S_1, \ldots, S_n$ are characterizing relations defined on B. These characterizing relations are abstract mathematical relations like $=, \leq$, etc.

The homomorphism guarantees that the perceivable relations that an attribute induces on primitive visual elements have identical structural properties as the characterizing relations that exist among its attribute values. Therefore, the characterizing relations abstract over particular perceivable relations like darker-than, smaller-than, left-of, etc. and define the types of relations and thereby the type of perceptual structures that visual attributes induce on primitive visual elements.

In the rest of this chapter, two types of attributes are distinguished: qualitative and quantitative. The qualitative attributes are then subdivided into nominal and ordinal attributes. The quantitative attributes are defined by considering different sets of arithmetical relations. We introduce two types of quantitative attributes that are important in data visualization: interval and ratio.

Nominal Attributes A nominal attribute is a homomorphic map m from a relational system $< A\ ;\ \approx>$ into a relational system $< B\ ;\ =>$. Note that the homomorphic map assigns a real number, a string, etc. to elements of A.

Ordinal Attributes An ordinal attribute is a homomorphic map m from a relational system $< A \; ; \; \preceq >$ into the relational system $< B \; ; \; \leq >$. The homomorphic map matches the ordinal relation among attribute values with the ordinal relation among elements of A.

Interval Attributes An interval attribute is a homomorphic map m from a relational system $< A \; ; \; \preceq, \circ >$ into the relational system $< \mathbf{R}^k \; ; \; \leq \; , \; \oplus >$, where $\oplus$ is a quaternary metrical relation defined on real numbers. Note that the interval attribute is a quantitative attribute such that the set of attribute values are from $\mathbf{R}^k$. In the context of visual representations, we may use the quaternary metrical relation $\oplus$ defined by: $ab \oplus cd \;\Leftrightarrow\; |a-b| \leq |c-d|$. This attribute is often called the absolute interval attribute because we are interested in the absolute difference between pairs of elements.

Ratio Attributes A ratio attribute is a homomorphic map m from a relational system $< A \; ; \; \preceq, \circ \; , \; Null >$ into the relational system $< (\mathbf{R}_{\geq 0})^k \; ; \; \leq \; , \; \oslash \; , \; 0 >$, where $\oslash$ is a binary metric operator defined on real numbers and 0 is a zero-place operator identifying the zero element from the real numbers. Note that the zero element has the following property: $\forall e \in (\mathbf{R}_{>0})^k \quad 0 \oslash e = 0$.
This zero-place operator identifies an absolute origin element. The ratio attribute is a quantitative attribute such that the set of attribute values are from $(\mathbf{R}_{\geq 0})^k$. In the context of visual representations, we may use the binary metrical operator $\oslash$ defined by: $a \oslash b \;\Leftrightarrow\; a/b$ where $a \in (\mathbf{R}_{\geq 0})^k$ and $b \in (\mathbf{R}_{>0})^k$, i.e. b is not the absolute origin element 0 ($b \neq 0$).

Although only interval and ratio attributes are introduced here, there may be many quantitative attributes, each of which uses different sets of arithmetical relations and properties of real numbers.

6.2.1 Perceptual Classification of Visual Attributes

We define a visual attribute as an attribute which maps a relational system $< A \; ; \; R_1, \ldots, R_n >$, where A is a set of primitive visual elements, into a relational system $< B \; ; \; S_1, \ldots, S_n >$, where B is a set of values. For example, given the color saturation attribute, B is the set of color saturation values for one color hue (e.g. the color hue red) and the set of relations consists of the ordering relation "$\leq$".

The elements of the set B are objects of the human visual system. Because we are interested in the perceptual characteristics of visual attributes, the relations $S_1, \ldots, S_n$ should reflect these characteristics. For example, the human visual system can identify the equality of shape values and perceives that a red square and a green square have the same shapes, while it cannot identify an ordering among a red circle and a red square. In contrast, the human visual system can identify the order among different color saturations, or among sizes. In this way, one can perceive that a red square is bigger than a green square. These perceptual characteristics constitute the basis on which the type of visual attributes is defined. In the rest of this section, we study some important visual attributes from the perceptual point of view and classify them by the types of attributes that we have introduced. We claim that other visual attributes can be analyzed in the same way.

We will not study all visual attributes in one subsection and will divide them in four successive subsections: non-spatial, spatial, topological, and nested segmentation. These subsections are not meant to classify visual attributes in a systematic or meaningful way. Although it may not be intuitive, we will study the size and the shape attributes as non-spatial attributes.

A) Non-spatial Visual Attributes

The non-spatial visual attributes are studied by Bertin [1] [Ber81]. These attributes include hue, saturation, brightness, size, shape, label, texture, etc.

The color hue attribute can be defined as a homomorphic map from a relational system, consisting of a set of primitive visual elements and an equivalence relation, into a relational system consisting of a set of color hue values and the equality relation. In this way, only the equality of color hue values matters such that the primitive visual elements that have the same color hue value are considered to belong to the same equivalence class.
Note that this definition of the color hue attribute assumes that the human visual system can only identify the equality of the color hue values, but

[1] In his book, *Graphics and Graphic Information Processing*, he uses the term "visual variables" rather than "visual attributes".

cannot order them. Of course, this is a narrow characterization of the color hue attribute. In fact, the color hue values can in some cases be perceived as being ordered. For example, the rainbow colors (a certain sequence of the color hue values) are perceived as being in an ordering structure. This ordering structure can be characterized as a ring structure that is constituted by an abstract mathematical relation like *modulo* relation.
The characterization of the color hue attribute becomes even more difficult when we consider the fact that people (especially visual artists) can know that the purple hue value is between blue and red, orange between red and yellow, but have no awareness of what may be between orange and blue. Nevertheless, we assume that the properties of the color hue attribute can be specified by mathematical relations, so that we can define this attribute as a homomorphic map between relational systems. In the rest of this section, we define various visual attributes by considering minimal characterizations which may ignore some of their more complex properties.

The color saturation attribute is a homomorphic map from a relational system, consisting of a set of primitive visual elements and an ordered equivalence relation, into a relational system consisting of the set of color saturation values and the ordering relation $\leq$. The set of color saturation values is assumed to be the set of color saturation values of one single color hue value such that different color saturation values are in *less/equal/more-saturated* relations with each other. It is then assumed that both the identity and the ordering among color saturation values matters.

The color brightness attribute is a homomorphic map from a relational system, consisting of a set of primitive visual elements and an ordered equivalence relation, into a relational system consisting of the set of color brightness values and the ordering relation $\leq$. The set of color brightness values is assumed to be the set of color brightness values of one single color hue value such that different color brightness values are in *lighter-than/equal/darker-than* relations with each other. It is then assumed that both the identity and the ordering among color brightness values matters.

The shape attribute can be defined in a similar way as the color hue attribute. The range of the shape attribute is then a pair consisting of a set of shape values and an equality relation defined on it. It is then assumed that one can perceive the equality of shapes but can not perceive how one shape is related to another one. Note that if we consider only a specific set of shapes, sometimes we can perceive orderings. For instance, in the case of polygons, the shapes can be ordered according to the number of edges (nodes) involved.

In the same way, the range of the size attribute can be defined as a pair consisting of a set of possible size values and an order relation defined on it. However, the equality of size values can be perceived by the human visual system only in special cases where the set of size values is a small finite set. One can also argue that the size attribute is a quantitative attribute such that one can perceive that one square is twice as big as a second square. In this case, the domain and the range of the size attribute should contain some metric relation that defines a quantitative structure on the set of primitive visual elements and the set of attribute values, respectively.

The label and the texture attribute are defined as homomorphic maps where their ranges consist of a set of label and texture values, respectively. The relation for these relational systems is the equality relation.

In this section, we have studied the type of perceivable relations that each non-spatial attribute may induce on primitive visual elements separately. However, as we explained in section 3, some non-spatial attributes (e.g. color hue, color saturation, and color brightness attributes) are integral (i.e. non-separable) attributes. This implies that these attributes together may constitute a complex attribute (e.g. the color attribute) which may induce additional perceptual structures on primitive visual elements. We will not work out these complex attributes and leave their specifications and the structures that they induce for the future research.

B) Spatially-Based Visual Attributes

A rather different kind of visual attributes concerns various uses of the properties of space [EBJS96]. These properties depend on the dimensions of the space. The spatial attributes which employ the properties of multidimensional spaces are integral attributes. This implies that spatial attributes for multidimensional spaces may employ complex properties of those spaces which cannot be defined as properties of individual one-dimensional spaces. We will consider only those spatial attributes of multidimensional spaces which employ the properties of individual one-dimensional spaces and leave the specification of complex attributes and the perceivable relations that they may induce for the future research.

In this way, we consider multidimensional spaces as being defined in terms of one-dimensional spaces by considering only those properties of multidimensional spaces that can be defined as properties of (identical) one-dimensional spaces. Therefore, we study different spatial attributes for one-dimensional space, henceforth 1D-space, and consider only those spatial attributes of the 2D-space or the 3D-space that can be defined by two or three spatial attributes of the 1D-space.

The use of 1D-space by dividing it into subspaces will be called segmentation. In this way, the total space is divided into a finite number of segments. The *segmentation attribute* can then be defined as a homomorphic map from a relational system, consisting of a set of primitive visual elements and the equivalence relation, into a relational system consisting of a set of real numbers or names, each of which identifies a certain segment, and the equality relation defined on it. It is then assumed that the human visual system can perceive the space as consisting of a number of perceptually distinguishable segments.

One can also use the ordinal property of 1D-space and consider the space as an ordered set of subspaces. This consideration results in the *ordered segmentation attribute* which can be defined as a homomorphic map from a relational system, consisting of a set of primitive visual elements and an ordered equivalence relation, into a relational system consisting of a set of real numbers or names, each of which identifies a certain segment, and an order relation defined on this set.

The space can also be viewed in terms of its quantitative properties. The quantitative properties of the 1D-space can be used in two different ways: The first use of the quantitative properties of the 1D-space is by considering the space as a relative 1D-space consisting of positions in which no origin is defined. In this case, the space between two positions (interval) is a quantity that can be perceived and meaningfully used. The resulting visual attribute will be called *relative space attribute*. This attribute can be defined as a homomorphic map from a relational system consisting of a set of primitive visual elements, which does not necessarily include an element at the origin position, and a set of relations consisting of an equivalence relation, an ordering relation, and a metrical relation (for example the absolute distances between primitive visual elements). The range of the map is a relational system consisting of the set of real numbers, each of which identifies a position, on which the equality relation, the ordering relation and the absolute difference relation between pairs of real numbers are defined.

The second use of the quantitative properties of the space is by considering the space as an absolute space in which one position is (explicitly or implicitly) defined to be the origin of the space. In this case, each position in the space is a quantity such that the proportionality between positions can be perceived and meaningfully used. The resulting visual attribute will be called *absolute space attribute*. This visual attribute can be defined as a homomorphic map from a relational system consisting of a set of primitive visual elements, one of which may be placed at the origin of the space, and a set of relations consisting of an equivalence relation, an ordering relation, and a metrical relation (for example the proportion of distances among primitive visual elements). The range of the map is a relational system consisting of the set of real numbers, each of which identifies a position, on which the equality relation, the ordering relation and the proportionality relation are defined on real numbers.

The separate analysis of one-dimensional spatial attributes implies that some spatial attributes for two- and three-dimensional spaces can be defined as consisting of two and three one-dimensional spatial attributes. Given the above four spatial attributes as they are defined for 1D-space, there are sixteen possibilities for two-dimensional spatial attributes. But of course, some of those combinations may yield identical results. For example, the two-dimensional spatial attribute consisting of the absolute space attribute and the ordered segmentation attribute may be the same as the two-dimensional spatial attribute consisting of the ordered segmentation attribute and the absolute space attribute. However, in some cases the order of combination of 1D spatial attributes may result in ergonomical differences such that the two orders of combination should be considered as different.
The perceptual structure of most visual patterns is induced by a combination of visual attributes. An example of the combination of visual attributes is the two-dimensional bar-chart as illustrated in Figure 6.9.

In this bar-chart, one 1D-space (the x-axis) is structured by dividing it into three segments while the second 1D-space (the y-axis) is structured by its metrical quantities. The color hue of primitive visual elements (bars) within this two-dimensional space reflects a nominal structuring.

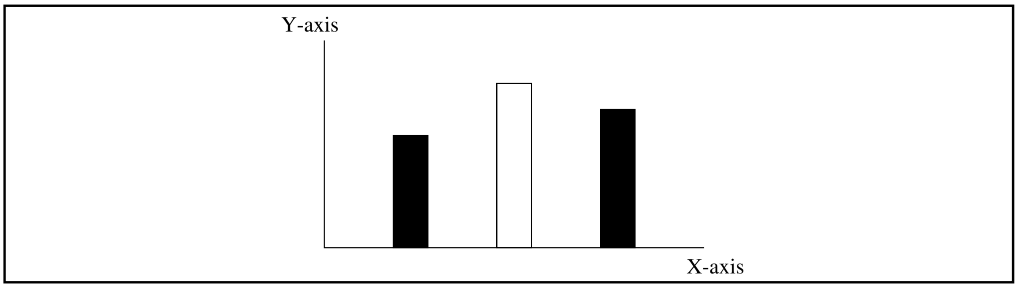


Figure 6.9: *The perceptual structure of the visual pattern is induced by a combination of visual attributes.*

C) Topological Attributes

A special type of domain structures concerns various uses of topological properties of the space. These domain structures are constituted by perceivable topological relations like *inside, outside, overlap, connectedness, etc.* Topological relations are often used in visual patterns like family trees, flow charts, network diagrams, path diagrams, subway maps, and Venn diagrams. In these visual patterns, the perceivable topological relations are either explicitly represented by primitive visual elements like line- and arrow-connections, shaded areas, etc. or implicitly represented by placing primitive visual elements in relations such as inclusion, exclusion, overlap, etc.
For example, in the graph illustrated in Figure 6.10-A the arrow connections represent explicitly a topological relation between circles, in the diagram illustrated in Figure 6.10-B the shaded areas represent explicitly a topological relation between closed-curves, and in the Euler diagram illustrated in Figure 6.10-C the overlap between disks represents implicitly a topological relation.

In the rest of this section, we consider a topological relation as a binary relation which is defined as a subset of the Cartesian product between a set of primitive visual elements with itself. An n-ary topological relation can then be described in terms of binary topological relations.
Domain structures that are constituted by topological relations are called *topological structures.* Unlike domain structures (proposed in previous subsections) where the constitutive perceivable relations between primitive visual elements are defined by means of structurally similar relations between their attribute values, topological structures are constituted by topological

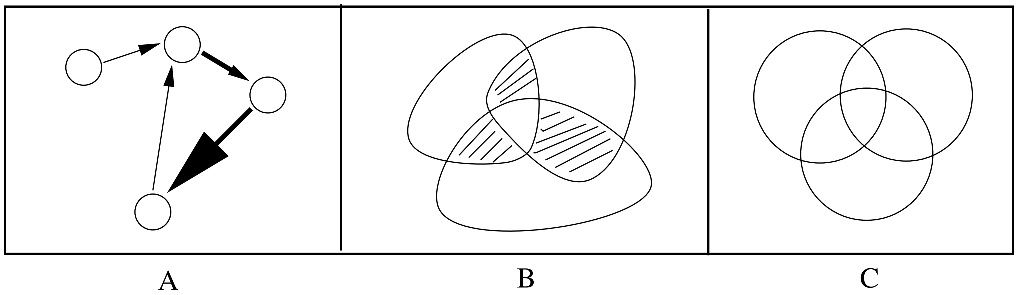


Figure 6.10: *Examples of explicitly (A & B) and implicitly (C) represented topological relations.*

relations between primitive visual elements.

Binary topological relations are assumed to be induced on primitive visual elements by means of two *topological attributes* the values of which are from one set of primitive visual elements. When topological relations are explicitly represented by primitive visual elements, like line- and arrow-connections or shaded areas, the topological attributes are considered to be attributes of these primitive visual elements. These primitive visual elements may have other attributes like size, color, texture, etc. In the case of implicitly represented topological relations, the two attributes generate a set of binary tuples, each of which relates a primitive visual element to another one. These binary tuples do not represent any visual elements.
For example, in the graph illustrated in Figure 6.10-A the arrow connections (primitive visual elements) are defined on the basis of two topological attributes $Start - link$ and $End - link$ which assign two circles (primitive visual elements) to each arrow connection and therefore relate two circles to each other. Similarly, in the diagram illustrated in Figure 6.10-B the shaded areas are defined on the basis of two topological attributes *included-in-1st-element* and *included-in-2th-element*. In this way, a shaded area relates two closed-curves to each other. Finally, in the Euler diagram illustrated in Figure 6.10-C, two attributes *first-overlap-element* and *second-overlap-element* generate a set of binary tuples relating one disk with another one. This set of binary tuples represents the overlap relation between disks.

Primitive visual elements that explicitly represent topological relations are related to each other in terms of relations that are defined on the primitive visual elements which are the values of the topological attributes involved. For example, the arrows in the visual pattern illustrated in Figure 6.10-A are also related to each other by means of relations that exist between the circles they connect. Considering the positional relation between connected circles in this visual pattern, one arrow is related to another arrow by connecting those circles that are on the left side of the circles connected by the second arrow. Also, in the visual pattern illustrated in Figure 6.10-B, one shaded area is related to another one by being included in those closed-curves that are higher than the closed-curves in which the second shaded area is included.

Consequently, we define a topological attribute as a homomorphic map from a relational system, consisting of a set of primitive visual elements (i.e. those that explicitly represent topological relations) and a set of relations $R'_1, \ldots, R'_n$, into a relational system consisting of a set of primitive visual elements (i.e. those that are values of topological attributes) and a set of relations $R_1, \ldots, R_n$. The relation R'_i is defined in terms of the relation R_i. Thus, primitive visual elements that explicitly represent topological relations are related to each other by means of $R'_1, \ldots, R'_n$ which are defined in terms of $R_1, \ldots, R_n$ that are defined on primitive visual elements that are values of the topological attributes involved.

Note that explicitly represented relations by lines, arrows, or shaded areas are themselves primitive visual elements that are also defined by means of non-topological visual attribute values like color, shape, size, texture, etc. These attribute values may induce additional perceivable relations on explicitly represented relations. For example, the arrow connections between circles in the graph illustrated in Figure 6.10-A are primitive visual elements that have different sizes (thickness). The size values of these arrows induce an ordinal relation on them such that one perceives one arrow as thicker or thinner than another arrow.

The situation is quite different with topological relations that are not explicitly represented by primitive visual elements. We explained that these relations are sets of binary tuples each of which consists of two primitive visual elements. It is argued that this set is generated by two topological attributes. We will call topological attributes which induce implicitly represented topological relations *implicit attributes* in contrast to *explicit attributes* which induce topological relations that are explicitly represented by primitive visual elements. In order to deal with implicit attributes in the same way as with explicit attributes, we consider the set of binary tu-

ples as the domain of a relational system on which an equivalence relation is defined. In other words, we consider each binary tuple as an invisible element defined in terms of values of two topological attributes. In the case of implicit attributes, we assume that primitive visual elements as values of implicit topological attributes do not induce any relations, except the equivalence relation, on the invisible elements, i.e. invisible elements are related to each other by an equivalence relation.
Thus, we define an implicit topological attribute as a homomorphic map from a relational system, consisting of a set of invisible elements (tuples of primitive visual elements) and an equivalence relation, into a relational system consisting of a set of primitive visual elements and an equivalence relation. Although this treatment of implicit topological attributes is not intuitive, it has the advantage of modeling all visual attributes uniformly, i.e. visual attributes induce perceivable relations that are defined on their domain values on primitive visual elements.

Topological relations may have various structural properties like being functional (injective, surjective, bijective), transitive, reflexive, symmetry, etc. Topological structures can be further subdivided into tree-structures, cyclic and acyclic graph-structures, etc. A detailed classification of topological relations and structures is presented in [Kam97].

D) Nested Segmentation Attribute

A special use of the space is the recursive division of the space into nested subspaces. The nested subspaces will be called *nested segments*. In fact, the recursive division of the space provides a nesting structure of segments. An example of a nested segmentation of the space is illustrated in Figure 6.11. In this visual pattern, the space is divided into two segments, each of which is again subdivided into two nested segments. Subsequently, in these nested segments the space is used to represent two-dimensional bar-charts.

The nesting structure of segments is assumed to be induced by *nested segmentation attribute*. The nested segmentation attribute can then be defined as a homomorphic map from a relational system, consisting of a set of perceptually distinguishable subspaces (segments) and the equivalence relation, into a relational system consisting of a set of nested elements on which the equality relation is defined. Note that the nested elements in the range of this attribute may be again segments on which another nested segmentation attribute is defined.

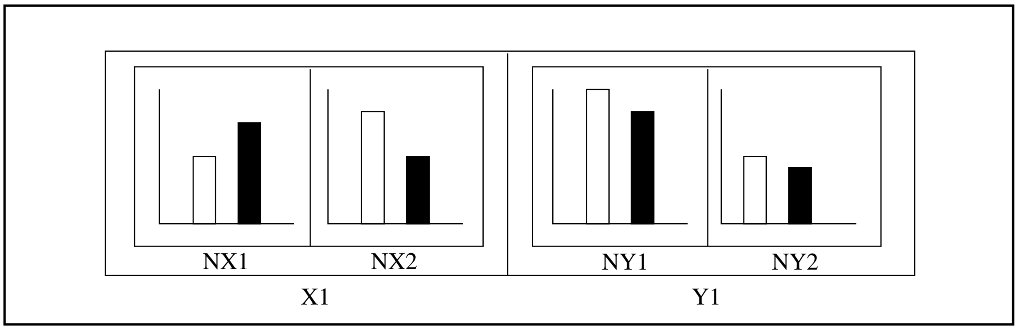


Figure 6.11: *An example of nested segmentation of the two-dimensional space.*

6.2.2 A Hierarchy of Domain Structures

In the previous subsections, different types of domain structures are introduced. Each type of domain structure is subdivided into more specific types. The subdivision of domain structures provides a hierarchical scheme as illustrated in Figure 6.12.

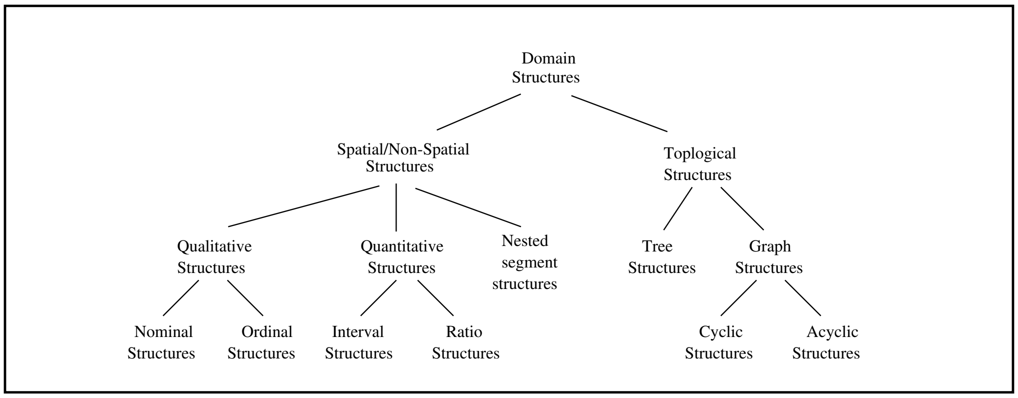


Figure 6.12: *A type hierarchy of domain structures.*

In this hierarchical scheme, the links between structure types are *ISA* links which indicate that a lower level structure type is a specific type of the higher level and connected structure type.

This hierarchical scheme can be extended by including more specific types of domain structures. For example, more specific spatial structures can be specified by considering different quantitative or arithmetic relations. Similarly, one may extend this hierarchy by adding more specific topological structures. For example, topological structures can be classified further according to the structural properties of their constituting relations. A more elaborate classification of relations according to their structural properties is discussed in [Kam97].

6.3 A Formal Framework For Data Visualization

In section 1 of this chapter, a process model for effective data visualization was described. It was argued that in effective data visualization the intended structure of data and the perceptual structure of the representing visual pattern should structurally coincide. The intended structure of data and the perceptual structure of visual patterns are constituted by data relations and perceivable relations, respectively. The data relations and the perceivable relations are induced on data and primitive visual elements by means of the involved data and visual attributes, respectively.
In section 2 of this chapter, we formally defined (data and visual) attributes and introduced a classification of them. The classification of attributes is based on the type of relations (and thereby structures) that they induce on (data and visual) elements.
In this section, we build on the definitions of section 2 to provide a formal description of the model of effective data visualization which was outlined in section 1. In particular, we define two relational systems that are constructed based on the data attributes and visual attributes, respectively. One relational system specifies the structure of data and the second relational system specifies the perceptual structure of visual patterns. The effectiveness condition of data visualization is then modeled by demanding a structure preserving mapping between these two relational systems. We will explain the role of gestalt in data visualization and conclude this section by defining the notion of a visual language.

6.3.1 The Structures of Data and Visual Patterns

In section 1, a nested table of data attribute values (which presents a nested relational database) was considered as a set of data entries each of which is a n-tuple of data attribute values. A data attribute value can be either an atomic data value or a nested data entry. Similarly, a visual pattern

is represented as a nested visual table. The nested visual table is a set of visual entries each of which is a n-tuple of visual attribute values. A visual attribute value can be either an atomic visual attribute value or a nested visual entry.
For example, in the *Car-state* data table the tuple $< d_1', Good, d_3' >$ is a data entry, where d_1' and d_3' are the nested data entries $< VW, 40000, Germany >$ and $< Toyota, 50000, Japan >$, respectively. Thus, there are three data entries involved where the first is defined in terms of the second and the third. Similarly, in visual data table the tuple $< v_1', Size_2, v_3' >$ is a visual entry (representing a link) where v_1' and v_3' are the nested visual entries (representing circles) $< seg_1, pos_4, hue_1 >$ and $< seg_2, pos_5, hue_2 >$, respectively.

Data entries or visual entries are related to each other by means of their attribute values. Since data (or visual) entries at different nesting levels are defined in terms of different tuples of attribute values, the relations between data (or visual) entries at different nesting levels may be disjoint, i.e. a relation which is defined on data (or visual) entries at a certain nesting level need not to be defined on data (or visual) entries at a different nesting level.
In the previous sections, we argued that a visualization of a nested data table is effective if the intended structure of the represented data and the perceptual structure of the representing visual pattern coincide. In order to define this structural correspondence, we define annotated data tables and annotated visual tables as relational systems consisting of a set of entries and a set of relations defined on the set of entries. As noted, these relations are defined on entries that are from one certain nesting level. Therefore, these relations are *partial* relations which means that they are defined on entries from one certain nesting level and undefined on entries from other nesting levels. Defining data and visual tables as relational systems expresses the relations that data and visual attributes induce on data and visual entries explicitly.

28. Definition. *Let DE be a set of data entries (n-tuples of data attribute values), $DR_1, \ldots, DR_m$ be partial relations that are defined on DE by means of the data attributes involved. Then, an annotated data table is defined as a relational system $< DE \ ; \ DR_1, \ldots, DR_m >$. Such a relational system will be called data system.*
Similarly, let VE be a set of visual entries (n-tuples of visual attribute values), $VR_1, \ldots, VR_m$ be partial perceivable relations that are defined on VE by means of the visual attributes involved. Then, an annotated visual

table is defined as a relational system $< VE \; ; \; VR_1, \ldots, VR_m >$. Such a relational system will be called visual system.

For example, consider data tables illustrated in Figures 6.13 and 6.14.

	Year	*Student*
d_1	1970	d'_1
d_2	1970	d'_2
d_3	1970	d'_3
d_4	1980	d'_4
d_5	1980	d'_5
d_6	1980	d'_6
d_7	1990	d'_7
d_8	1990	d'_8
d_9	1990	d'_9

Figure 6.13: *The Study-state data table.*

	Subject	*NR-of-Stud*
d'_1	Mathematics	200
d'_2	Physics	100
d'_3	Computer	300
d'_4	Mathematics	250
d'_5	Physics	150
d'_6	Computer	250
d'_7	Mathematics	150
d'_8	Physics	200
d'_9	Computer	300

Figure 6.14: *The nested Student-state data table.*

These data tables indicate the number of students (*Nr-of-stud*) that study a certain subject (*Subject*) in three different years (*Year*). In particular, the *Study-state* data table is defined by two attributes: *Year* and *Student*. The values of the *Student* attribute are nested tuples the set of which defines the nested data table as illustrated in Figure 6.14. The nested data table, which is called *Student-state*, is defined by two attributes: *Subject* and *Nr-of-stud*.

Let DE be the set of data entries consisting of values of the *Year* and the *Student* attribute, NDE be the set of nested data entries consisting of values of the *Subject* and the *Nr-of-stud* attribute, S_{NDE} be the same-subject relation defined on NDE, LOW_{NDE} be the lower-number relation defined on NDE, $PROP_{NDE}$ be a quantitative relation defined on NDE ($PROP_{NDE}$ expresses the proportional relation between data entities), $NULL_{NDE}$ be a zero-place operator which identifies a data entry from NDE which represents an absolute origin element, E_{DE} be the earlier-than relation defined on DE, and SS_{DE} be the same-state relation defined on DE. Then, the attributes of the *Study-state* data table and the attributes of the nested *Student-state* data table can be defined as the following homomorphic maps:

$$
\begin{array}{ll}
Subject & :< \ NDE \ ; \ S_{NDE} \ > \ \rightarrow \\
 & \ \ < \ \{Mathematics, Physics, Computer, \ldots\} \ ; \ = > \\[1em]
Nr - of - stud & :< \ NDE \ ; \ LOW_{NDE}, PROP_{NDE}, NULL_{NDE} \ > \ \rightarrow \\
 & \ \ < \ \mathbf{R}^+ \ ; \ \leq, \oslash, 0 \ > \\
 & \ \ where \ \forall a, b \in \mathbf{R}^+ \ if \ b \neq 0 \ then \ \ a \oslash b = a/b. \\[1em]
Year & :< \ DE \ ; \ E_{DE} \ > \ \rightarrow \\
 & \ \ < \ \{1900, \ldots, 2000\} \ ; \ \leq > \\[1em]
Student - state & :< \ DE \ ; \ SS_{DE} \ > \ \rightarrow \\
 & \ \ < \ NDE \ ; \ = >
\end{array}
$$

These data attributes constitute the following data system:

$$< DE \cup NDE \ ; \ S_{NDE}, LOW_{NDE}, PROP_{NDE}, NULL_{NDE}, E_{DE}, SS_{DE} >$$

This relational system expresses the structure of the data represented by the *Study-state* data table explicitly.
Given this relational system the above data attributes may also assign different attribute values to the elements (data entries) of this relational system. Different assignments of attribute values to the elements of a relational system are considered as different *instantiations* of that relational system. Note that different instantiations of a relational system result in structurally identical data tables.

In a similar way, we may define a relational system on the basis of a visual data table. For example, consider visual data tables illustrated in Figures 6.15 and 6.16.

Let VE be the set of visual entries consisting of visual values of the horizontal segmentation attribute *X-pos* and the topological attribute *Include*, NVE be the set of nested visual entries consisting of visual values of the nested horizontal segmentation attribute *NX-pos* and the nested vertical segmentation attribute *NY-pos*. Let also SP_{NVE} be the same-position relation defined on NVE, SH_{NVE} be the shorter-than relation defined on NVE, LEN_{NVE} be a quantitative length relation defined on NVE (one can perceive that the length of one visual element is two times shorter than the length of a second visual element), BP_{NVE} be a zero-place operator which identifies a visual element the length of which can not be perceived,

	X-pos	Include
v_1	seg_1	v'_1
v_2	seg_1	v'_2
v_3	seg_1	v'_3
v_4	seg_2	v'_4
v_5	seg_2	v'_5
v_6	seg_2	v'_6
v_7	seg_3	v'_7
v_8	seg_3	v'_8
v_9	seg_3	v'_9

Figure 6.15: *A visual data table.*

	NX-pos	NY-pos
v'_1	$n-seg_1$	pos_3
v'_2	$n-seg_2$	pos_1
v'_3	$n-seg_3$	pos_5
v'_4	$n-seg_1$	pos_4
v'_5	$n-seg_2$	pos_2
v'_6	$n-seg_3$	pos_4
v'_7	$n-seg_1$	pos_2
v'_8	$n-seg_2$	pos_3
v'_9	$n-seg_3$	pos_5

Figure 6.16: *A nested visual table.*

L_{VE} be the left-of relation defined on VE, and SI_{VE} be the same-inclusion relation defined on VE. Then, the involved visual attributes of the visual data illustrated in Figures 6.15 and 6.16 can be defined as the following homomorphic maps:

$$\begin{array}{ll} NX-pos & :<\ NVE\ ;\ SP_{NVE}\ >\ \rightarrow \\ & <\ \{x \mid x\ is\ a\ segment\ of\ the\ x-axis\ \}\ ;\ =\ > \end{array}$$

$$\begin{array}{ll} NY-pos & :<\ NVE\ ;\ SH_{NVE}, LEN_{NVE}, BP_{NVE}\ >\ \rightarrow \\ & <\ \{y \mid y \in \mathbf{R}^+\ \&\ y\ is\ a\ position\ on\ the\ y-axis\ \}\ ;\ \leq, \oslash, 0\ > \\ & where\ \forall a, b \in \mathbf{R}^+\ if\ b \neq 0\ then\ a \oslash b = a/b. \end{array}$$

$$\begin{array}{ll} X-pos & :<\ VE\ ;\ L_{VE}\ >\ \rightarrow \\ & <\ \{x \mid x \in \mathbf{R}\ \&\ x\ is\ a\ position\ on\ x-axis\ \}\ ;\ \leq\ > \end{array}$$

$$\begin{array}{ll} Include & :<\ VE\ ;\ SI_{VE}\ >\ \rightarrow \\ & <\ NVE\ ;\ =\ > \end{array}$$

These visual attributes constitute the following visual system:

$$< VE \cup NVE\ ;\ SP_{NVE}, SH_{NVE}, LEN_{NVE}, BP_{NVE}, L_{VE}, SI_{VE} >$$

Given this relational system, the above visual attributes may assign different attribute values to the elements (visual entries) of this relational system. This results in different instantiations of the relational system. In the case of visual data, different instantiations of a relational system provide different visual patterns. Two different instantiations of the above relational system are illustrated in Figure 6.17.

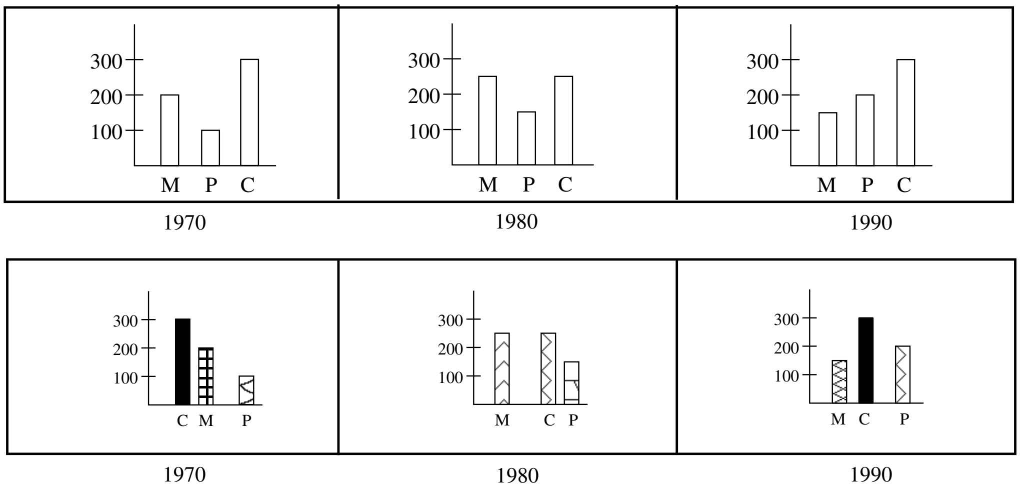


Figure 6.17: *Two different instantiations of a relational system.*

Note that the second visualization is generated in a smaller format. Moreover, as the nested x-positions are determined by a (non-ordered) segmentation attribute (i.e. *NX-pos*), the order of bars in these two visualizations are different.

In the second visualization two design errors has been made. These design errors result in unwanted visual implicatures. The first design error is related to the instantiation of the nested x-position values which are determined by the decided *NX-pos* attribute. Although the nested x-positions of the bars identify different segments, their random choices cause different distances between bars. These different distances between bars may suggest that the x-positions are not determined by a nominal attribute (in this case the segmentation attribute), but by an ordinal, an interval, or even a ratio type attribute.
The second design error is related to the instantiation of the texture values which are determined by the undecided texture attribute. Like the random choices for the nested x-positions, the random choices for the texture values may suggest that differences in texture values of different bars express semantic differences.
As noted in section 1 of this chapter, the instantiation of values for decided and undecided visual attributes are considered to be the function of the layout process. In subsection 3.3 of this chapter, we will use the idea of

perceptual grouping (gestalt) to formulate conditions for the instantiation of visual values by the layout process such that above mentioned design errors can be avoided. But first, we will formulate the structure preserving condition between data system and visual system.

6.3.2 Structure Preserving Mapping

It has been explained that visual patterns and data can be defined as visual and data systems, respectively. The interpretation or the visualization relation between visual patterns and data can then be defined as a mapping between their relational systems. A mapping between two relational systems consists of a mapping between the domain elements of the relational systems and a one-to-one mapping between their relations.

Although any mapping between a visual system and a data system will provide a visualization, only a subset of them can be claimed to guarantee effective visualizations. The effectiveness of visualizations will be achieved by demanding a structure preserving mapping between the intended structure of data and the perceptual structure of the representing visual patterns.
In order to define an effective mapping between the structure of a data and the perceptual structure of a visual pattern such that the structure of the data coincides with the perceptual structure of the visual pattern, we require an isomorphism (structure preserving map) between their corresponding data and visual systems, where the domain of elements of the visual system is constituted by the values of the *decided* visual attributes.

29. Definition. *An isomorphism between two relational systems* $< \Delta_1; R_1, \ldots, R_n >$ *and* $< \Delta_2; S_1, \ldots, S_n >$ *is a one-to-one and onto mapping between* Δ_1 *and* Δ_2 *and a one-to-one mapping between relations* R_i *and* S_i *(for* $i = 1, \ldots, n$*) which satisfies the following condition:*
If a relation R_i *holds between two elements of* Δ_1*, the corresponding relation* S_i *holds between the corresponding elements of* Δ_2*, and if* R_i *does not hold between two elements of* Δ_1*,* S_i *does not hold between the corresponding elements of* Δ_2*.*

The isomorphic mapping guarantees that each data entry corresponds (is represented by) a visual entry, and whenever there exists an (intended) relation between two data entries their corresponding visual entries are related to each other by a structurally identical relation.

The necessary (but not sufficient [2]) isomorphism requires a correspondence between data relations and visual relations that have the same types. This condition is a partial reformulation of the effectiveness criteria, i.e. *in an effective visualization data attributes are visualized by visual attributes that have the same attribute types.*

6.3.3 The Role of Gestalts in Data Visualization

Until now, we have mainly discussed the domain structures of visual patterns that can be used to represent information structures. In this section, we explain the role of gestalts of visual patterns in visualizing information. The gestalts of visual patterns are either the proximity structure of the involved visual elements or defined as the regularity of perceivable relations. These gestalts represent the perceptual grouping of visual elements.

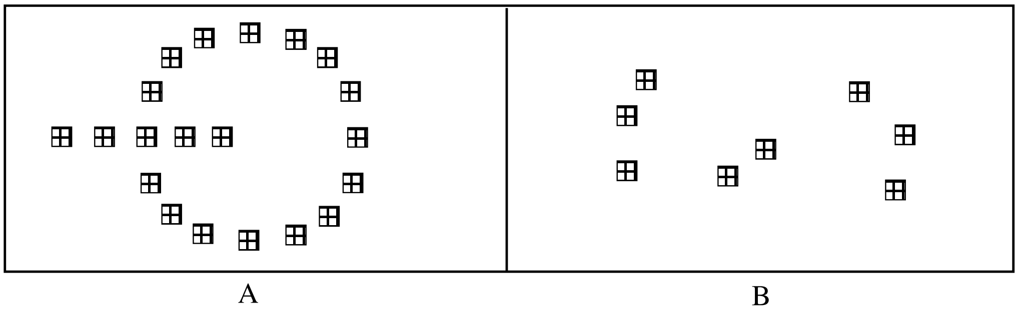


Figure 6.18: *Visual elements are perceptually grouped based on structural regularity in A Visual elements are perceptually grouped based on proximity in B.*

For example, as it is illustrated in Figure 6.18-A, a number of visual elements that are positioned at the circumference of a circle at equidistant positions are perceived as belonging to one perceptual group. Similarly, in Figure 6.18-B visual elements that are positioned relatively close to each other are perceived as belonging to one perceptual group (proximity effect). Thus, a perceptual grouping of visual elements can be considered as a classification structure providing perceptually motivated equivalence classes of visual elements. In this way, two visual elements that belong to one perceptual group are considered as perceptually equivalent and two visual ele-

[2] The isomorphic mapping is not a sufficient condition since other design factors such as cultural conventions play also a role in the effectiveness of visualizations.

ments that belong to different perceptual groups are considered as perceptually non-equivalent. This implies that the perceptual grouping of visual elements imposes a nominal structure on visual elements. In data visualization, these gestalt-based nominal structures can be used to represent nominal data structures.

As explained, different decided visual attributes like color, size, segmentation, etc. may induce a nominal type structure on visual elements. For instance, the segmentation attribute is considered as a nominal type attribute by defining the equality relation on the perceptually equivalent positions of visual elements. However, we did not explain yet under which conditions the actual attribute values of visual elements are perceptually equivalent. In order to determine perceptually equivalent attribute values, we use the gestalt idea of regularity and proximity of attribute values, as explained in chapter 3 and 4. In these chapters, we explained that regularity and proximity of attribute values result in visual elements that belong to one perceptual group. We may use this idea to define perceptually equivalent attribute values. In particular, we consider a subset of values of one visual attribute as perceptually equivalent if based on the instantiation of these attribute values a subset of visual elements are resulted that belong to one perceptual group.

30. Definition. *Let B be the set of values of the visual attribute α and $=_{SIT}$ be an equivalence relation defined on B. The equivalence relation $=_{SIT}$ is defined on the basis of regularity and proximity (gestalt analysis) of attributes values from B such that $\forall b_1, b_2\ b_1 =_{SIT} b_2$ iff based on the instantiation of b_1 and b_2 two visual elements are resulted that belong to one perceptual group. Then, the visual attribute α is a nominal attribute which is defined as a homomorphic map from a relational system $< A\ ;\ \approx>$, where A is a set of visual elements, into a relational system $< B\ ;\ =_{SIT}>$.*

In this way, gestalts are used as nominal structures that can be induced on visual elements by different decided visual attributes. Note that this is essential for inducing nominal structure on the basis of position values. Since different visual elements have always different position values, the perceptual equivalence of visual elements based on position values should be defined on perceptually equivalent positions rather than actual equivalent positions. Similarly, the fact that different colors, shapes, sizes, etc. may be perceived as equivalent can also be treated in similar way.

It may be the case that one or more visual attributes are not used to represent data attributes, i.e. one or more undecided attributes may be involved. In previous sections, we explained that an arbitrary assignment of values of the undecided attributes to visual elements might result in unwanted visual implicatures. In order to avoid unwanted visual implicatures, undecided attributes should assign attribute values to visual elements such that the gestalt induced by undecided attributes represents only one perceptual group, i.e. a perceptual group that contains all visual elements involved. For example, in the node-link diagrams illustrated in Figure 6.19, the positions do not represent any information.

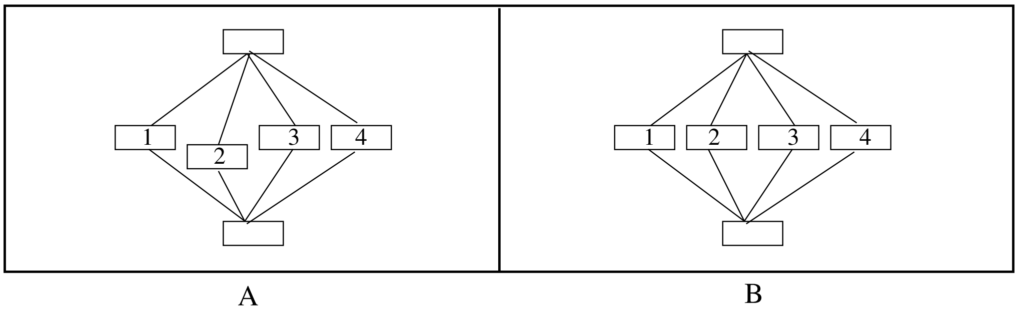


Figure 6.19: *The nodes in diagram A cause unwanted implicature while the nodes in diagram B do not.*

However, in the diagram illustrated in Figure 6.19-A one perceives the positional irregularity of nodes $1, \ldots, 4$ and may conclude that there should be a semantic difference between node 2 on the one hand and nodes 1,3, and 4 on the other hand. This unwanted visual implicature is avoided in diagram B by using the position-based gestalt of the nodes: the nodes $1, \ldots, 4$ are perceived as being in one perceptual group and therefore as being semantically equivalent with respect to their positions.

6.3.4 Visualization Systems as Visual Languages

A representation system consisting of two relational systems, a data system and a visual system, and a mapping between them specifies a visual language. The mapping between these relational systems represents the denoting relation between visual patterns and data. Different pairs of relational systems specify different visual languages. Moreover, for each pair of relational systems, there are many mappings possible, each of which results in a different visual language.

31. Definition. *Let* $D =< \Delta_1; R_1, \ldots, R_n >$ *and* $V =< \Delta_2; S_1, \ldots, S_n >$ *be a data system and a visual system, respectively, and let* Ψ *be a mapping between them. Then, the triple* $VL =< D, V, \Psi >$ *specifies a visual language. A visual language for which the mapping is isomorphic is called an effective visual language.*

Note that a visual language may generate different visual patterns (i.e. expressions of a visual language) by using a different number of elements from the visual systems involved or by using different range values (attribute values) for their visual attributes. This means that the visual expressions of a visual language employ identical relations and thus have the same structure since a change in the signature (the set of relations involved) implies a change in the identity of the language.

In this way, various kinds of maps, diagrams, graphs, flow-charts, etc. can be considered as different visual languages. Two different bar-charts that differ from each other in the number of bars or their visual attribute values are then considered as different expressions of one and the same visual language, namely the same bar-chart language.

This observation implies that there is a partial order among visual languages where the order is determined by the signatures of these languages. This partial order determines the expressive power of visual languages in terms of the richness of their signatures. Assuming a finite number of visual attributes, the partial ordering can be represented by a lattice that determines the expressiveness hierarchy of visual languages. Note that each visualization system characterizes a certain visual language such that formal analysis of the visualization systems can be applied to visual languages.

Chapter 7

Conclusion and Future Research

We are far from a precise all-encompassing model of human visual cognition. 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.
There is a certain complexity about the status of the model that we have proposed. The model is schematic. We discussed the plausibility of certain instantiations of it, but we hope that the structures that we have conjectured for these cases are much more generally applicable. To some extent, the model as specified is open-ended. We should now go on to investigate how various domains of perceptual patterns may be represented in terms of the input structures that the VREG-system presupposes, what the 'admissible functions' are that may be invoked, and how the complexity measure is defined.

In this thesis, we have developed various formal languages each of which represents a certain class of regularity-based gestalts for a certain class of perceptual patterns. Some of these languages are designed for one-dimensional string patterns while others are designed for two-dimensional visual patterns. We feel that other languages need to be developed to cover different classes of regularity-based gestalts for various other perceptual patterns.
Moreover, we have considered only those gestalts that are based on regularity of relations among subsets of attribute values of primitive visual

elements. We argued that gestalts of visual patterns could also be based on regularity of relations among subsets of attribute values of compound visual elements which are not the attribute values of their constitutive primitive visual elements. These kinds of regularity-based gestalts were not elaborated here and are left for future research.

The VREG-language deals with visual structures that can be characterized by partial mappings from the Euclidean plane into finite sets of non-recursive feature structures. It can be shown that this subsumes the Leeuwenberg languages that only analyze the structure of finite sequences of discrete symbols. We have emphasized that open and closed line patterns in the Euclidean plane are not adequately represented by finite sequences of discrete symbols, and for this reason their perceptual structure cannot always be correctly analyzed in terms of the existing Leeuwenberg-style coding languages. We must now admit, however, that it is also not completely clear whether the VREG-language as it stands allows an adequate treatment of two-dimensional line patterns. If we follow Leeuwenberg in focusing on the relatively simple and artificial case of line pattern consisting exclusively of straight line segments, it is relatively straightforward to represent line pattern as a set of length/direction pairs positioned on the plane, or as a set of pairs of positions. But, as in the case of the Leeuwenberg languages, the question is whether the observations of regularities in this representation allows us to derive a high-level description of the pattern that captures the gestalt structures that people actually perceive.
We have also not demonstrated that the algorithm which computes simplest codes on the basis of the regularities in this representation, takes the continuity of the lines into account in the proper way. Though we may not get 'good continuation' for free from the most general version of the VREG-encoding algorithm, we can certainly imagine a version of this algorithm which acknowledges the notion of a 'line segment' as a special kind of object, and which prefers gestalts without unnecessary line breaks and shifts of direction. But this remains to be worked out.
The next, more difficult issue, would be to design a proper finite representation for curved lines. In computer generation of visual forms this is commonly done by means of n-th order polynomials which are characterized, for instance, by certain points they pass through, and by the angles at which they do that. It is not clear whether this kind of representation has any value from a cognitive perspective.

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.
For example, we argued that the class of admissible transformations is a certain subclass of mathematical transformations that are perceptually relevant. However, this subclass of mathematical transformations was not identified. The identification of perceptually relevant admissible transformations is remained open for future experimentation.
One other aspect of our model which is based on our intuition and needs to be verified by future experimentation is the definition of information load. In fact, since we have developed new formal languages to express gestalts of perceptual patterns and because the information load is defined on expressions of these languages, we had to redefine the notion of information load for these languages. Although we have tried to stay as close as possible to the latest and experimentally verified definition of information load, we think that our proposed definition of information load for the newly developed languages should be verified by future 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 algorithm. These include not only the projection of 3D-perspective, but also the figure/ground ambiguity.
Moreover, we have described gestalts as being determined solely by proximity and 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. We allow complex patterns to be treated as 'units', but to get a lower complexity value than their full description anyway: 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.

Bibliography

[BDP$^+$93] R. Bod, M. Dastani, H. Prüst, R. Scha, and H. Zeevat. Hci from a discourse perspective. Technical Report ESPRIT Basic Research Action P6296, Human Communication Research Center, Edinburgh, August 1993.

[Ber81] J. Bertin. *Graphics and Graphic Information-Processing.* Walter de Gruyter, Berlin NewYork, 1981.

[BLR81] H. Buffart, E. Leeuwenberg, and F. Restle. Coding theory of visual pattern completion. *Journal of Experimental Psychology: Human Perception and Performance*, 7:241–274, 1981.

[Bos88] F. Boselie. Local versus global minima in visual pattern completion. *Perception & Psychophysics*, 43:431–445, 1988.

[BS93] R. Bod and Scha. Deriving optimal network diagrams by means of structural information theory. Technical Report ESPRIT Basic Research Action P6296, Human Communication Research Center, Edinburgh, August 1993.

[BW89] F. Boselie and D. Wouterlood. The minimum principle and visual pattern completion. *Psychological Research*, 51:93–101, 1989.

[Cas91] S. Casner. A task-analytic approach to the automated design of graphic presentations. In *ACM Transactions on Graphics*, volume 10(2), pages 111–151, 1991.

[Das96] M. Dastani. Computing the perceptual structure of visual expressions. In *Artificial Intelligence in Design (AID'96), in the Workshop on Visual Representation, Reasoning and Interaction in Design. 24-27 June, Stanford University, California*, 1996.

[Das97a] M. Dastani. An algebraic approach to data visualization. In *Accolade '97, Dutch Graduate School in Logic*, 1997.

[Das97b] M. Dastani. A formal framework for visual representations. In *The 2nd International Workshop on Theory of Visual Languages*, Capri, Italy, September 1997.

[DI97] M. Dastani and B. Indurkhia. An algebraic approach to similarity and categorization. In *An Interdisciplinary Workshop On Similarity And Categorisation*, Edinburgh, Scotland, November 1997.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DIS97] M. Dastani, B. Indurkhya, and R. Scha. An algebraic method for solving proportional analogy problems. In *Mind II - Computational Models of Creative Cognition*, Dublin, Ireland, September 1997.

[DOS93] M. Dastani, J. Oberlander, and K. Stenning. 'x' marks the spot: Drawing conclusions with euler's circles. In *Semantic Issues in Graphical Representation, ESPRIT report*, HCRC, Edinburgh University, 1993.

[DT98] M. Dastani and E. Tan. Representation systems for video annotations. In *Object Foundation Classes, ESPRIT report, VICAR*, Sentient Machine Research, Amsterdam, 1998.

[Dub93] R.C. Dubes. Cluster analysis and related issues. In C. Chen, L.F. Pan, and P.S.P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*. World Scientific, 1993.

[EBJS96] Y. Engelhardt, J. de Bruin, T. Janssen, and R. Scha. The visual grammar of information graphics. In *Artificial Intelligence in Design (AID'96), in the Workshop on Visual Repre-*

sentation, Reasoning and Interaction in Design. 24-27 June, Stanford University, California, 1996.

[Eva68] T.G. Evans. A program for the solution of a class of geometric-analogy intelligence-test questions. In M. Minsky, editor, *Semantic Information Processing*, pages 271–353, Cambridge, Mass., 1968. MIT Press.

[Gar74] W. R. Garner. *The processing of information and structure.* Lawrence Erlbaum Associates, Potomac, MD, 1974.

[HM88] D.R. Hofstadter and M. Mitchell. Conceptual slippage and analogy-making: A report on the copycat project. In *Proceedings, Tenth Annual Cognitive Science Society Conference*, Hillsdale, New Jersey, 1988. Lawrence Erlbaum Associates.

[Hof84] D.R. Hofstadter. The copycat project: An experiment in nondeterministic and creative analogies. In *A.I. Memo 755, Artificial Intelligence Laboratory*, Cambridge, Mass., 1984. MIT.

[ID97] B. Indurkhya and M. Dastani. An algebraic model of the context effect in analogy. In *European Conference on Cognitive Science*, pages 239–242, Manchester, UK, 1997.

[Ima66] S. Imai. Classification of sets of stimuli with differenet stimulus characteristics and numerical properties. *Perception & Psychophysics*, 1:48–54, 1966.

[Ind91] B. Indurkhya. On the role of interpretive analogy in learning. *New Generation Computing*, 8:385–402, 1991.

[Ind92] B. Indurkhya. *Metaphor and cognition: an interactionist approach.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

[JD88] A.K. Jain and R.C. Dubes. *Algorithms for clustering data.* Prentice-Hall, inc., New Jerseyy, 1988.

[Kam97] Thomas M. Kamps. *A Constructive Theory for Diagram Design and its Algorithmic Implementation.* Ph.D. Thesis, 1997.

[Kof35] K. Koffka. *Principles of Gestalt Psychology.* Harcourt, Brace and World, New York, 1935.

[Kub81] M. Kubovy. Concurrent pitch segregation and the theory of indispensable attributes. In M. Kubovy and J. Pomerantz, editors, *Perceptual Organization*, Hillsdale, New Jersey, 1981. Lawrence Erlbaum Associates.

[Lee71] E. Leeuwenberg. A perceptual coding language for visual and auditory patterns. *American Journal of Psychology*, 84:307–349, 1971.

[Mac86] J. Mackinlay. Automating the design of graphical presentations of relational information. In *ACM Transactions on Graphics*, volume 5, pages 110–141, 1986.

[Mit93] W. J. Mitchell. A computational view of design creativity. In J. S. Gero and M. L. Maher, editors, *Modeling Creativity and Knowledge-based Creative Design*, pages 25–42, Hillsdale, New Jersey, 1993. Lawrence Erlbaum Associates.

[MR90] J. Marks and E. Reiter. Avoiding unwanted conversational implicatures in text and graphics. In *Proceeding AAAI*, Menlo Park, CA., 1990.

[Pfa68] J. Pfanzagl. *Theory of Measurement.* Physica-Verlag. Würzburg-Wien, 1968.

[PG93] M. Petre and T.R.G. Green. Learning to read graphics: Some evidence that 'seeing' an information display is an acquired skill. *Journal of visual languages and Computing*, 4:55–70, 1993.

[RKG95] K. Reichenberger, T. Kamps, and G. Golovchinsky. Towards a generative theory of diagram design. In N. Gershon and S. Eick, editors, *IEEE Proceedings on Information Visualization*, pages 11–18, Atlanta, Georgia, October 1995. IEEE Computer Society Press.

[She91] R.N. Shepard. Integrality versus separability of stimulus dimensions: From an early convergence of evidence to a proposed theoretical basis. In G.R. Lockhead and J.R. Pomerantz, editors, *The Perception of Structure*, pages 53–71, Washington, DC, 1991. American Psychological Association.

[Sie89] J. Siekmann. Unification theory. *Journal of Symbolic Computation*, 7:207–274, 1989.

[Ste46] S.S. Stevens. On the theory of scales of measurement. *Science*, 103:667–680, 1946.

[SVL93] J.F. Stins and C. Van Leeuwen. Context influence on the perception of figures as condition upon perceptual organization strategies. *Perception & Psychophysics*, 53 (1):34–42, 1993.

[Tuf83] E.R. Tufte. *The visual display of quantitative information.* Graphics Press, Cheshire (CT), 1983.

[Tuf90] E.R. Tufte. *Envisioning Information.* Graphics Press, Cheshire (CT), 1990.

[Twy79] M. Twyman. A schema for the study of graphic language. In P.A. Kolers, M.E. Wrolstad, and H. Bouma, editors, *Processing of visible language: Volume 1*, New York, 1979. Plenum Press.

[VdH94] P. Van der Helm. The dynamics of prägnanz. *Psychological Research*, 56:224–236, 1994.

[VdHL86] P. Van der Helm and E. Leeuwenberg. Avoiding explosive search in automatic selection of simplest pattern codes. *Pattern Recognition*, 19:181–191, 1986.

[VdHL91] P. Van der Helm and E. Leeuwenberg. Accessibility: A criterion for regularity and hierarchy in visual pattern code. *Journal of Mathematical Psychology*, 35:151–213, 1991.

[VH62] H. Von Helmholtz. *Treatise on Psychological Optics, Vol.III (Translation from the 3rd German ed.).* Dover Publications (Original work published 1867), NY, 1962.

[VLB89] C. Van Leeuwen and H. Buffart. Facilitation of retrieval by perceptual structure. *Psychological Research*, 50:202–210, 1989.

[VLBVdV88] C. Van Leeuwen, H. Buffart, and J. Van der Vegt. Sequence influence on the organization of meaningless serial stimuli: economy after all. *Journal of Experimental Psychology: Human Perception and Performance*, 14:481–502, 1988.

[Wer23] M. Wertheimer. Untersuchungen zur lehre von der gestalt. *Psychologische Forschung*, 4:301–350, 1923.

Index

Samenvatting

Visuele informatie vormt een belangrijke bron van menselijke kennis. Hoewel mensen visuele informatie vrijwel moeiteloos tot kennis verwerken, is het niet eenvoudig te beschrijven hoe deze informatie in kennis omgezet wordt. Het begrijpen van visuele informatie veronderstelt een constituentenstructuur of wel het bepalen van elementen en hun samenstelling tot grote gehelen. Deze constituentenstructuur wordt ook wel Gestalt genoemd. Het moge duidelijk zijn dat vrijwel ieder visueel patroon, of het nu gaat om een stadsgezicht of een artificieel geconstrueerde landkaart, verschillende constituentenstructuren kan hebben. Toch nemen we in veel gevallen één specifieke constituentenstructuur van een visueel patroon waar: bepaalde structuren worden blijkbaar geprefereerd boven andere. Het bepalen van de geprefereerde en dus de feitelijk waargenomen constituentenstructuur van visuele patronen, kan gezien worden als een desambigueringsmechanisme. Wij nemen aan dat zo'n desambigueringsmechanisme ten grondslag ligt aan het menselijke visuele systeem.

In dit proefschrift staat Gestaltdesambiguering van visuele patronen centraal, waarbij de empirisch getoetste structurele informatie theorie (SIT) het uitgangspunt vormt. De kerngedachte van SIT is dat het menselijke visuele systeem de meest eenvoudige Gestalt van een visueel patroon prefereert boven andere mogelijke Gestalten van dat patroon. In SIT is de notie van eenvoud gedefinieerd in termen van specifieke en perceptueel relevante regelmatigheden van visuele patronen. Een Gestalt van een visueel patroon is eenvoudiger dan een andere Gestalt van datzelfde patroon wanneer de eerste meer perceptueel relevante regelmatigheden beschrijft dan de tweede.

Om de eenvoud van structuren te bepalen, is een empirisch gemotiveerde complexiteitsmaat geïntroduceerd. Een Gestalt is eenvoudiger dan een andere Gestalt wanneer de complexiteit van de eerste lager is dan die van de tweede.

We bespreken het bestaande model van SIT. In dit model wordt een visueel patroon middels een symbolenreeks gecodeerd. De fundamentele aanname is, dat de perceptueel gemotiveerde regelmatigheden van het visuele patroon in de symbolenreeks weerspiegeld worden. Deze symbolenreeks wordt vervolgens geanalyseerd en beschreven in termen van deze regelmatigheden. Een symbolenreeks kan op verschillende wijzen beschreven worden. Elke beschrijving geeft een mogelijke constituentenstructuur van de symbolenreeks en daarmee een Gestalt van het gecodeerde visuele patroon weer. De geprefereerde Gestalt van het visuele patroon wordt bepaald door de complexiteit van deze beschrijvingen te berekenen. De beschrijving die de laagste complexiteit heeft geeft de geprefereerde Gestalt van dat patroon weer.

We laten zien dat dit model van SIT niet toereikend is om de geprefereerde Gestalten van een brede klasse van tweedimensionale visuele patronen te bepalen. Daarom wordt een verfijnder model ontwikkeld waarin de visuele patronen van deze klasse gerepresenteerd (gecodeerd) kunnen worden. Binnen dit model worden visuele patronen gerepresenteerd in termen van visuele attributen zoals positie, kleur, vorm en textuur. De perceptueel gemotiveerde regelmatigheden van tweedimensionale visuele patronen worden dan gedefinieerd in termen van regelmatigheden van de visuele attribuutwaarden. Een verzameling van attribuutwaarden wordt als regelmatig beschouwd wanneer deze gegenereerd kan worden door het toepassen van een transformatie op een van de deelverzamelingen van die verzameling. De transformaties voor de tweedimensionale positiewaarden zijn Euclidische transformaties. Om de Gestalten van tweedimensionale visuele patronen te kunnen uitdrukken, is een taal ontwikkeld. Deze taal wordt in het Engels "Visual REGularity language (VREG)" genoemd. De syntaxis en de semantiek van VREG worden formeel gespecificeerd. Voor de uitdrukkingen van deze taal wordt een complexiteitsmaat voorgesteld. Deze complexiteitsmaat is gemotiveerd door de complexiteitsmaat van SIT.
Vervolgens wordt een rekenmodel besproken waarmee de geprefereerde Gestalt van tweedimensionale visuele patronen berekend kan worden. In dit rekenmodel wordt een visueel patroon gerepresenteerd door een verzamel-

ing van visuele attribuutwaarden. Deze representatie wordt dan geparseerd door een aantal codeerregels waaruit de uitdrukkingen van VREG gegenereerd kunnen worden. Voor een visueel patroon kunnen verschillende uitdrukkingen gegenereerd worden. De complexiteitsmaat wordt dan gebruikt om de geprefereerde uitdrukking, en daarmee de geprefereerde Gestalt van het geanalyseerde visuele patroon, te kunnen selecteren. We laten zien dat de rekencomplexiteit van dit model deterministisch is maar exponentieel in de grote van het invoerpatroon.

De waarneming van de constituentenstructuur van visuele patronen is context-afhankelijk. Eerdere ervaringen met visuele patronen, het gebruik van visuele patronen voor een specifiek doeleinde, of de aanwezigheid van andere visuele patronen, kunnen de waarneming van een gegeven patroon beïnvloeden. De rol van contextfactoren in visuele waarneming wordt bestudeerd aan de hand van een specifieke context, namelijk de context van proportionele analogieën. In deze context staan vier visuele patronen in de volgende relatie tot elkaar: het eerste patroon staat in een relatie tot het tweede patroon en het derde patroon staat in dezelfde relatie tot het vierde patroon. We laten zien dat de Gestalten van deze patronen binnen en buiten de proportionele analogie context verschillend kunnen zijn.
De analogie wordt gemodelleerd door een algebraïsche correspondentie tussen de Gestalten van de betrokken visuele patronen te eisen. Tussen twee Gestalten kunnen verschillende algebraïsche correspondenties bestaan. We bespreken de plausibiliteit van correspondenties door het eenvoudscriterium op deze correspondenties toe te passen. Binnen de context van de proportionele analogieën worden de geprefereerde Gestalten van visuele patronen gezien als de eenvoudigste Gestalten waartussen een eenvoudige algebraïsche correspondentie bestaat. De voorgestelde methode wordt aan hand van verschillende voorbeelden geïllustreerd.

Tenslotte wordt een toepassing besproken waarin visuele waarneming een essentiële rol speelt. In deze toepassing, vaak informatie-visualisatie genoemd, worden visuele patronen gegenereerd om relationele informatie visueel te presenteren. Het bestuderen van deze applicatie is interessant omdat, naast de constituentenstructuur die de Gestalt van visuele patronen bepaalt, andere visuele relaties die kwalitatief en kwantitatief van aard zijn, gebruikt worden om relaties uit het informatiedomein uit te drukken. Deze visuele relaties komen tot stand op basis van attribuutwaarden van visuele elementen. Bijvoorbeeld, een lichtrood vierkant en een donkerrood vierkant

zijn identiek met betrekking tot het attribuut vorm terwijl hun relatie met betrekking tot het illuminatie-attribuut ordinaal van aard is. De ordinale relatie tussen deze vierkanten kan gebruikt worden om een ordinale relatie uit het informatiedomein, zoals ouder-dan of warmer-dan, uit te drukken.
Hoewel relationele informatie op verschillende wijzen gevisualiseerd kan worden, zijn niet alle visualisaties even effectief. Wij beargumenteren dat de effectiviteit van visualisatie gerelateerd is aan de waarnemingseigenschappen van visuele relaties. Wij beschouwen een visualisatie effectief, wanneer relaties uit het informatiedomein uitgedrukt worden middels visuele relaties die dezelfde structurele eigenschappen hebben. Gebaseerd op dit effectiviteitscriterium, introduceren we een formeel model voor effectieve informatie-visualisatie. In dit model worden zowel relationele informatie als visuele patronen beschreven als relationele structuren. Het effectiviteitscriterium wordt gemodelleerd door een structurele correspondentie tussen deze relationele systemen te eisen.

Dit proefschrift wordt afgesloten met een discussie en evaluatie van het onderzoek. Tevens worden open problemen en toekomstige onderzoeksrichtingen geïdentificeerd en geformuleerd.

Titles in the ILLC Dissertation Series:

ILLC DS-1993-01: **Paul Dekker**
Transsentential Meditations; Ups and downs in dynamic semantics

ILLC DS-1993-02: **Harry Buhrman**
Resource Bounded Reductions

ILLC DS-1993-03: **Rineke Verbrugge**
Efficient Metamathematics

ILLC DS-1993-04: **Maarten de Rijke**
Extending Modal Logic

ILLC DS-1993-05: **Herman Hendriks**
Studied Flexibility

ILLC DS-1993-06: **John Tromp**
Aspects of Algorithms and Complexity

ILLC DS-1994-01: **Harold Schellinx**
The Noble Art of Linear Decorating

ILLC DS-1994-02: **Jan Willem Cornelis Koorn**
Generating Uniform User-Interfaces for Interactive Programming Environments

ILLC DS-1994-03: **Nicoline Johanna Drost**
Process Theory and Equation Solving

ILLC DS-1994-04: **Jan Jaspars**
Calculi for Constructive Communication, a Study of the Dynamics of Partial States

ILLC DS-1994-05: **Arie van Deursen**
Executable Language Definitions, Case Studies and Origin Tracking Techniques

ILLC DS-1994-06: **Domenico Zambella**
Chapters on Bounded Arithmetic & on Provability Logic

ILLC DS-1994-07: **V. Yu. Shavrukov**
Adventures in Diagonalizable Algebras

ILLC DS-1994-08: **Makoto Kanazawa**
Learnable Classes of Categorial Grammars

ILLC DS-1994-09: **Wan Fokkink**
Clocks, Trees and Stars in Process Theory

ILLC DS-1994-10: **Zhisheng Huang**
Logics for Agents with Bounded Rationality

ILLC DS-1995-01: **Jacob Brunekreef**
On Modular Algebraic Protocol Specification

ILLC DS-1995-02: **Andreja Prijatelj**
Investigating Bounded Contraction

ILLC DS-1995-03: **Maarten Marx**
Algebraic Relativization and Arrow Logic

ILLC DS-1995-04: **Dejuan Wang**
Study on the Formal Semantics of Pictures

ILLC DS-1995-05: **Frank Tip**
Generation of Program Analysis Tools

ILLC DS-1995-06: **Jos van Wamel**
Verification Techniques for Elementary Data Types and Retransmission Protocols

ILLC DS-1995-07: **Sandro Etalle**
Transformation and Analysis of (Constraint) Logic Programs

ILLC DS-1995-08: **Natasha Kurtonina**
Frames and Labels. A Modal Analysis of Categorial Inference

ILLC DS-1995-09: **G.J. Veltink**
Tools for PSF

ILLC DS-1995-10: **Giovanna Cepparello**
Studies in Dynamic Logic

ILLC DS-1995-11: **W.P.M. Meyer Viol**
Instantial Logic. An Investigation into Reasoning with Instances

ILLC DS-1995-12: **Szabolcs Mikulás**
Taming Logics

ILLC DS-1995-13: **Marianne Kalsbeek**
Meta-Logics for Logic Programming

ILLC DS-1995-14: **Rens Bod**
Enriching Linguistics with Statistics: Performance Models of Natural Language

ILLC DS-1995-15: **Marten Trautwein**
Computational Pitfalls in Tractable Grammatical Formalisms

ILLC DS-1995-16: **Sophie Fischer**
The Solution Sets of Local Search Problems

ILLC DS-1995-17: **Michiel Leezenberg**
Contexts of Metaphor

ILLC DS-1995-18: **Willem Groeneveld**
Logical Investigations into Dynamic Semantics

ILLC DS-1995-19: **Erik Aarts**
Investigations in Logic, Language and Computation

ILLC DS-1995-20: **Natasha Alechina**
Modal Quantifiers

ILLC DS-1996-01: **Lex Hendriks**
Computations in Propositional Logic

ILLC DS-1996-02: **Angelo Montanari**
Metric and Layered Temporal Logic for Time Granularity

ILLC DS-1996-03: **Martin H. van den Berg**
Some Aspects of the Internal Structure of Discourse: the Dynamics of Nominal Anaphora

ILLC DS-1996-04: **Jeroen Bruggeman**
Formalizing Organizational Ecology

ILLC DS-1997-01: **Ronald Cramer**
Modular Design of Secure yet Practical Cryptographic Protocols

ILLC DS-1997-02: **Nataša Rakić**
Common Sense Time and Special Relativity

ILLC DS-1997-03: **Arthur Nieuwendijk**
On Logic. Inquiries into the Justification of Deduction

ILLC DS-1997-04: **Atocha Aliseda-LLera**
Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence

ILLC DS-1997-05: **Harry Stein**
The Fiber and the Fabric: An Inquiry into Wittgenstein's Views on Rule-Following and Linguistic Normativity

ILLC DS-1997-06: **Leonie Bosveld - de Smet**
On Mass and Plural Quantification. The Case of French 'des'/'du'-NP's.

ILLC DS-1998-01: **Sebastiaan A. Terwijn**
Computability and Measure

ILLC DS-1998-02: **Sjoerd D. Zwart**
Approach to the Truth: Verisimilitude and Truthlikeness

ILLC DS-1998-03: **Peter Grunwald**
The Minimum Description Length Principle and Reasoning under Uncertainty

ILLC DS-1998-04: **Giovanna d'Agostino**
Modal Logic and Non-Well-Founded Set Theory: Translation, Bisimulation, Interpolation

ILLC DS-1998-05: **Mehdi Dastani**
Languages of Perception