Modeling Analogical Projection based on Pattern Perception*

Mehdi Dastani
Vrije Universiteit Amsterdam, The Netherlands
mehdi@cs.vu.nl

Bipin Indurkhya
Tokyo University of Agriculture and Technology, Japan
bipin@cc.tuat.ac.jp

Remko Scha
University of Amsterdam, The Netherlands
scha@hun.uva.nl

Abstract

One of the major unsolved problems in perception is to explain why a sensory pattern, which may in principle be perceived as having different constituent structures (gestalts), is usually perceived as having one particular gestalt. Structural Information Theory (SIT) treats Gestalt perception as disambiguation and describe it by a perceptually motivated preference relation defined on possible gestalts of patterns. The existing models of SIT ignore the context effects which is essential in determining the gestalt of patterns. We introduce a SIT model and embed it in a framework where two types of context effects are integrated: 1) the effect of simultaneous presence of other patterns, and 2) the effect of the task to be accomplished on the basis of the patterns. In order to illustrate the interaction of these types of context effects with perception, we consider proportional analogy problems defined on strings (alphabetic patterns). Such a proportional analogy problem consists of three simultaneously present strings and the task is to find a fourth string by means of analogical projections on the given strings.

To this end, we propose an extended algebraic model of SIT for strings. In this model, strings are considered as domain elements of an algebra and different gestalts of a string are represented as different algebraic terms corresponding to that string. A perceptually motivated preference relation orders different algebraic terms (gestalts) of one string. Analogical relations are defined as mappings between algebras. To integrate the context effects, the model is embedded in a framework that is characterized by two constraints. One concerns the simplicity of algebras generating the terms as well as the simplicity of the mapping between algebras (the first context effect), and the second concerns the possibility of a mapping between algebras (the second context effect).

Keywords: Algebra, analogy, context effect, grouping, information load, perception, projection, proportional analogy, structural information theory.

1 Introduction

A fundamental activity underlying cognition (by a natural or artificial agent) is that of integrating perceptual stimuli into a system of concepts. This, in turn, requires structuring and representing the stimuli. Gestalt psychologists since Wertheimer have noted that any sensory stimulus allows many structural descriptions and have sought to articulate principles which explain why certain

*We are grateful to Cees van Leeuwen and Peter van Emde Boas for many valuable discussions and suggestions during the research presented here.

structures (rather than others) are usually perceived. For example, the Gestalt research demonstrates that the structure that our cognitive system assigns to a particular stimulus is largely dependent on what context it takes into consideration. In fact, our cognitive system analyzes (either deliberately or unconsciously) the components of the stimulus as structurally analogous to components of the context. In our view, this capability of 'analogical projection' plays a crucial role in perception. We also believe that people are essentially perception machines, and their other cognitive prowess is parasitic on their perceptual capabilities. Therefore, studying and modeling the perceptual processes will also provide us an insight into the higher cognitive processes such as metaphor and creativity. (See also Hofstadter (1995) and Indurkhya (1992)). This is the prime motivating factor underlying our research program.

A well-defined class of problems which involve analogical projection in interaction with gestalt perception is constituted by proportional analogy problems with perceptual patterns. 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 Hofstadter (1984), as in Figure 1; or line-drawings, as in Figure 2.

	A		B		C		D
1.	abba	:	abab	::	pqrrqp	:	pqrpqr
2.	abba	:	abbbbba	::	pqrrpq	:	pqrrrrrpq

Figure 1: Examples of proportional analogies based on letter strings

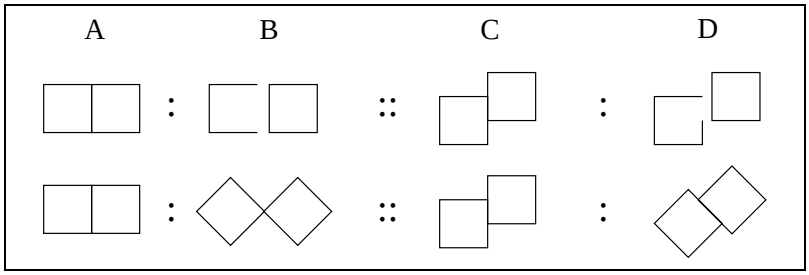

Figure 2: Examples of proportional analogies based on line-drawings

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 a similar way as B is related to A. Solving a proportional analogy problem requires 1) a mutual contextualization of the terms A and B, and 2) construction of a mapping between A and C which generalizes to a broader domain that includes B.

Proportional analogies between perceptual patterns demonstrate the working of gestalt disambiguation: namely how the perceived structure of one pattern influences the perceived structure of another pattern by analogical projection. In particular, the structures of A and C are to be construed as analogous to each other; and the structure of B 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 complex interaction between gestalt perception and analogy computation can be evidenced even in very simple domains, as was demonstrated by Hofstadter (1984). For example, the first term in the two proportional analogy relations shown

in Figure 1 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 this paper, we propose a method for solving proportional analogy problem by integrating the context effect in gestalt perception, analogical projection, and their interaction. To this end, we employ *Structural Information Theory* [SIT] initiated by Leeuwenberg (1971) as a starting point. This theory explains how perceptually relevant constituent structures of isolated patterns can be determined and how they can be ordered according to their perceptual preferences. The claim of SIT is that the actually perceived gestalt of a pattern in isolation is represented by its most preferred constituent structure. We then introduce an algebraic model of SIT that generates the actual perceived gestalt of one-dimensional string patterns in isolation. This algebraic model is extended to cover the mutual contextualization effect, namely how gestalts of two patterns can influence each other. Finally, we demonstrate all these features of our approach in Hofstadter's Copycat domain, and show how proportional analogy problems such as shown in Figure 1 can be solved in it.

This article is organized as follows. In the next section we briefly review the structural information theory that aims to explain the gestalt of perceptual patterns. Then, in Section 3, we develop an algebraic version of SIT. Following that, an algebraic model for proportional analogy is introduced in Section 4. In Section 5, we discuss various factors that constrain the search for appropriate gestalts in proportional analogies. In Section 6, we present an algorithm for solving proportional analogy problems — it computes the fourth term of a proportional analogy relation given the other three terms — and compare it with other computational approaches. Finally, in Section 7, we summarize the major points of our paper and briefly discuss some future research issues.

2 Gestalt Perception and SIT

The idea that our perceptual system, on encountering a stimulus, imposes a certain structure on it, rather than an arbitrary one, can be traced back to the German psychologist Wertheimer (1923). He argued that sensations, caused by stimulus patterns, are experienced as organized structures for which he used the term *Gestalt*. An important objective of the Gestalt research was to determine a set of principles which explain why certain types of patterns are experienced as having a certain organization. In particular, Wertheimer proposed five principles — known as *proximity, similarity, continuity, closure* and *habit or past experience* — to constrain the perceptual organization of patterns, and he claimed that these constitute the laws underlying the mechanism of human perception.

Relatively recently, Leeuwenberg (1971) argued, following the notion of Prägnanz introduced by Koffka (1935), that the Gestalt principles can be explained by one more basic principle of human perception. This principle is related to the structural regularity and the simplicity of stimulus patterns, which forms the fundamentals of Leeuwenberg's theory of pattern perception called Structural Information Theory (SIT). 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. Note that these identities are in a hierarchical structure such that the identities of the substrings ab at a higher hierarchical level implies the identities of the a's and the b's at a lower hierarchical level.

Although there are various kinds of regularities in terms of which patterns may be described, only a small subset of these regularities is claimed to be perceptually motivated Van der Helm and Leeuwenberg (1991). These perceptually relevant structural regularities 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. Each of these operators specifies a different kind of pattern regularities to which the perceptual system is sensitive. They are defined as follows:

Iteration :	$kkk\cdots kk\ (N \geq 2\ times\ k)\ \rightarrow\ N*(k)$
e.g.	$ababab\ \rightarrow\ 3*(ab)$
Symmetry :	$k_1k_2\cdots k_npk_n\cdots k_2k_1\ \rightarrow\ S[\ (k_1)(k_2)\cdots(k_n)\ ,\ (p)\ ]$
e.g.	$abccba\ \rightarrow\ S[\ abc\ ,\ ()\ ]$
	$abcab\ \rightarrow\ S[\ (ab)\ ,\ (c)\ ]$
Alternation :	$kx_1kx_2\cdots kx_n\ \rightarrow <(k)>/<(x_1)(x_2)\cdots(x_n)>$
	$x_1kx_2k\cdots x_nk\ \rightarrow <(x_1)(x_2)\cdots(x_n)>/<(k)>$
e.g.	$abkabtabw\ \rightarrow <(ab)>/<ktw>$
	$kabtabwab\ \rightarrow <ktw>/<(ab)>$

Notice the use of parentheses to indicate grouping. For instance, in the second example of symmetry, '(ab)' acts as an indivisible unit, so that its internal structure is not changed by the symmetry operator. Actually, in the SIT notation, parantheses are used around single elements too, so the first symmetry example above would be written as 'S[((a)(b)(c)), ()]', and the second as 'S[(((a)(b))), (c)]'. However, here we omit the parantheses around single elements to make the notation less cumbersome and easier to read.

Based on these operators, a pattern can be parsed into its perceptual descriptions. A perceptual description of a pattern represents a specific Gestalt of that pattern. A pattern may be parsed according to different sequences of ISA operators such that different perceptual descriptions, representing different Gestalts of the pattern, result. For example, $3*(\ (ab)\ )$ represents one Gestalt for *ababab*, which sees it as composed of repeating (ab) thrice. However, $<a>/<bbb>$ represents a different Gestalt for the same string which sees it as a being alternated by b's. The process of finding the Gestalt of a stimulus, then, becomes essentially an exercise in disambiguation.

In order to disambiguate the set of alternative Gestalts and decide the preferred perceptual description of a pattern, a measure is introduced which assigns a complexity value to each description of that pattern. The complexity value, called *information load* [IL], is designed to measure the amount of perceptually motivated regularities within a pattern. As SIT has evolved over the years, the precise manner of computing IL has also been subject to revisions. We describe the general idea intuitively here. Later, in Section 5, we will present a precise manner of computing IL based on the 'new' IL definition proposed in Van der Helm and Leeuwenberg (1991).

The intuitive idea behind IL is that it should reflect the structural complexity of a gestalt. In one version, the IL of a Gestalt description is computed by adding the number of occurrences of individual elements in that description (not including the structural operators) to the number of units larger than one element. For example, the information load of $3*(\ (ab)\ )$ is 3 because it has two elements, namely a and b, and there is one unit, namely (ab), consisting of more than one element. On the other hand, the information load of $<a>/<bbb>$ is 4 because it has four elements (note that each occurrence of b is counted separately). A description of a pattern which has the lowest complexity value is thought to employ the highest amount of perceptually motivated regularities of that pattern. Therefore, the pattern description with the lowest complexity value is claimed to represent the pattern in the most simple and cognitively economical way. Thus, in the above example, the Gestalt description $3*(ab)$ is preferred for *ababab* because its complexity value is lower.

3 An Algebraic Version of SIT

There are three motivating factors in our wanting to develop an extended version of SIT. First of all, notice that the perceptual structures covered in SIT are based only on identities between constituents of the structure. For example, when *aaaa* is written in SIT as $4*(a)$ it is recognized that the four elements appearing in the pattern are identical. But it is also possible to consider other relations that may be specific to the domain. For instance, in the geometric figures domain, one may consider the operation *rotate-left.* Or in a domain with ordered elements, such as Copycat, we may consider successor and predecessor functions, generalized to also apply to sequences of elements. In the Copycat domain, this mean we define functions *succ* and *pred* such that $succ(b) = c$, $succ(abc) = bcd$, and so on. Using these functions, we may notice a regularity in *abc*, namely that it is a successor sequence: the next element of the sequence is obtained by taking the successor

of the previous element. [1] Another factor is that though SIT works well in choosing the preferred gestalt for isolated stimuli, it cannot model the context effect, or how different gestalts of the same stimulus are preferred in different contexts. This phenomenon is demonstrated by the proportional analogies of Figs. 1 and 2. Finally, for modeling analogies and metaphors, we would like to be able to consider mappings between gestalts.

The algebraic approach developed here is aimed at addressing these three issues. An algebra is essentially a domain of objects, and a number of operators (or functions) defined on these objects. An n-ary operator takes as input n objects, and results in another object in the domain. The operators of the algebra essentially endow the objects of its domain with structure in that they allow some objects to be combined into another object; or, in another way of looking at it, allow an object to be decomposed into its component objects. In order to develop an algebraic version of SIT, we define the structural operators of SIT as algebraic operators. We now demonstrate how to do this in Hofstadter's Copycat domain. First, we must define the domain of objects that are of interest to us. For the Copycat domain, this is simply the set of all finite non-null strings composed from letters $\{a, b, \cdots, z\}$. We call this the set of *simple* string patterns.

Definition 1 *The domain D_s of simple one-dimensional string patterns is defined as the set of all finite strings of length one or more composed from $\{a, b, \ldots, z\}$.*

Notice that SIT uses an implicit grouping operator to make a string pattern into an indivisible unit. This is indicated in SIT by nested levels of parentheses. These indivisible units are needed to describe non-mirror symmetrical patterns. For example, the mirror-symmetrical string *abccba* can be described as $S[abc, ()]$, but the non-mirror-symmetrical string *abcbca* can be described as $S[a(bc), ()]$. In the second case, the string *bc* is treated as a single unit and is written as (bc) to distinguish it from *bc*. We follow the same convention. In order to define an algebraic symmetry operator, we now extend the domain of simple one-dimensional strings to include 'grouped' strings.

Definition 2 *The domain of grouped one-dimensional string patterns D is defined as follows:*
1) If $x \in \{a, \ldots, z\}$ then $x \in D$,
2) If $x_1, \ldots, x_n \in D$ then $x_1 \cdots x_n \in D$,
3) If $x_1, \ldots, x_n \in D$ and $n > 1$ then $(x_1 \cdots x_n) \in D$

Note that $x_1 \cdots x_n$ in this definition is one string. According to this definition (ab), $(ab)(cd)$, $ab(cd)$, $(ab)(c(de))fg$, etc. are all in D. Notice that we do not allow single element groups like (a) to be generated because it makes the notation needlessly cumbersome by generating groups like $(((a)))$ or $(((abc)))$ which are essentially equivalent to a and (abc), respectively. It is this domain D that we use for defining the SIT operators algebraically. We also define the notion of *unit* object, which is either a simple one-element string like a or x, or a single unit enclosed by a pair of parentheses like (ab), $(ab(cd))$, $((abc)(de))$, etc.

Definition 3 *An object x of D is called a unit object if either:*
1) $x \in \{a, \ldots, z\}$, or
2) x is of the form $(x_1 \cdots x_n)$

Thus, according to this definition, grouped strings like abc, $(ab)(cd)$, $ab(cd)$ are ruled out as unit objects.

The SIT operators can now be easily defined over D. As mentioned earlier, we would also like to allow *domain-dependent* regularities and cognitive operators to influence the preferred structural gestalt. Consequently, we let F_D be the set of all domain-dependent or cognitive operators that might play a role in the construction of gestalts. For the time being we limit ourselves to one-place

[1] This issue is explicitly addressed in Van der Helm and Leeuwenberg (1991); however they argue that such operators are cognitive and not perceptual, and therefore do not need to be incorporated in the perceptual coding principles. But we feel that while operators requiring addition or otherwise complex operations may be left out, simple operators like 'successor', 'rotate-left' do end up playing a critical role in perception, and ought to be included in the gestalts. In any case, we are interested in representing cognitive operators as well, and being able to model how they affect the structural representation of perceptual stimuli.

operators. We then augment the iteration and alternation operators so that they take an extra argument that is an element of F_D. Moreover, we will represent the concatenation operator, which is tacitly assumed in SIT, explicitly in our algebra by introducing an operator called *Con*. Finally, in order to express the unit structure of string patterns explicitly, a new operator, called *Unit* operator, is added to the set of algebraic operators. 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. We are now ready to define the *SIT algebra* over the domain D as follows:

Definition 4 *A SIT-algebra over the domain D is a quadruple $< D, \mathcal{N}, \mathcal{F}, \mathcal{S} >$, where D is the domain of objects, $\mathcal{N}$ is the set of natural numbers; F is a set of domain-dependent operators (so that each $f \in F$ is a function from D^n to D for some n); and $\mathcal{S}$, the set of SIT-operators, contains the following operators. In the following, $X = x_1 \cdots x_k$ is a string pattern from D, where each of x_i is an object; $Y, Y_1, \ldots, Y_m$, with $m > 1$, are arbitrary string patterns from D; f is a one-place domain-dependent operator ($f \in \{id, succ, pred, \ldots\}$); and $n \in \mathcal{N}$. We write $f^n(X)$ to indicate that the one-place function f is applied n times to X.*

$$
\begin{array}{ll}
Iter(Y, f, n) & \rightarrow Y\ f(Y) \cdots f^{n-1}(Y) \\
Sym_e(X) & \rightarrow x_1 \cdots x_k x_k \cdots x_1 \\
Sym_o(X, Y) & \rightarrow x_1 \cdots x_k Y x_k \cdots x_1 \\
Alt_r(Y, f, X) & \rightarrow Y\ x_1\ f(Y)\ x_2 \cdots f^{k-1}(Y)\ x_k \\
Alt_l(Y, f, X) & \rightarrow x_1\ Y\ x_2\ f(Y) \cdots x_k\ f^{k-1}(Y) \\
Con(Y_1, \ldots, Y_m) & \rightarrow Y_1 \cdots Y_m \\
Unit(Y_1 \cdots Y_m) & \rightarrow (Y_1 \cdots Y_m)
\end{array}
$$

Note that in the definitions above, the particular variable used in each argument position conveys its sort. For example, the first argument of *Iter* must come from D, the second argument from F, and so on. The output of each operator belongs to D. Note also that we need to introduce the identity operator *id* explicitly, whenever we want the iteration and alternation operators to use identity of elements and units in constructing gestalts.

Below are some examples of gestalt of string patterns as provided by the SIT-algebra.

$$
\begin{array}{ll}
Iter(a, succ, 3) & \rightarrow abc \\
Iter(Iter(c, id, 2), pred, 3) & \rightarrow ccbbaa \\
Sym_e(Con(a, Unit(Con(b, c)), d)) & \rightarrow a(bc)dd(bc)a \\
Alt_r(a, succ, Con(f, k, t)) & \rightarrow afbkct \\
Con(k, f, u) & \rightarrow kfu
\end{array}
$$

Now we can consider the gestalts of patterns as terms of the *SIT-algebra*. Thus, in the *SIT-algebra*, the string *abc* has an iteration structure according to which it is considered a single chunk. Also note that the structural operators of the SIT-algebra do not specify any regularity in the pattern kfu; its perceptual chunking therefore consists of three subchunks, k, f, and u.

The structural description of an object shows how the object is built out of other elements. This corresponds to what is called a *term* or an Ω-*word* of an algebra Cohn (1981)(page 116). Thus, the set of all terms of the *SIT-algebra* corresponds to the set of all possible gestalts for the universe D. This is defined as follows:

Definition 5 *The class of gestalts over the SIT-algebra, denoted by $\mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, is recursively defined as follows:*

- *For all $X \in D$, $X \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$*
- *If $f \in F$, $n \in \mathcal{N}$, and $X \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, then $Iter[X, f, n] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$*
- *If $X \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, then $Sym_e[X] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$*
- *If $X_1, X_2 \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, then $Sym_o[X_1, X_2] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$,*

- *If* $f \in F_D$, *and* $X_1, X_2 \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, *then*
  $Alt_r[X_1, f, X_2] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$ *and*
  $Alt_l[X_1, f, X_2] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$,
- *If* $X_1, \ldots, X_n \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, *with* $n > 1$, *then* $Unit[X_1 \cdots X_n] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, *and*
- *If* $X_1, \ldots, X_n \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$, *with* $n > 1$, *then* $Con[X_1, \ldots, X_n] \in \mathcal{G}_{<\mathcal{D},\mathcal{N},\mathcal{F},\mathcal{S}>}$,

Notice that we use square brackets instead of round ones to emphasize that these gestalts are not evaluated. Then, an *evaluation* function can be defined in the standard way which evaluates each gestalt into a string (grouped or simple). For lack of space, we will omit its definition here.

Based on this evaluation function, the extensional identity (which is not the structural identity) of gestalts can be defined. Two gestalts are extensionally identical if and only if they both get evaluated to the same element. Two extensionally identical terms may thus constitute different gestalts of the same pattern.

Though Defs. 4 and 5 make a clear distinction between the domain-dependent operators F and SIT-operators $\mathcal{S}$, which is necessary because one-place domain-dependent operators can appear as arguments to some of the SIT-operators, for defining analogical mappings and describing algorithms for computing them, we can lump these two classes of operators together in a set, say $\mathcal{F}$. Then, by also dropping the explicit mention of $\mathcal{N}$, we can simplify our notation of SIT algebra to be simply $< D, \mathcal{F} >$, where D is the set of objects and $\mathcal{F}$ is the set of operators.

4 Formalizing Proportional Analogy in SIT

Given that gestalts are formalized as algebraic terms, we can now turn to the problem of how to characterize analogies between different gestalts — in particular, how to characterize proportional analogy relations. 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* Siekmann (1989) between tree-like structural descriptions (gestalts). There are two ways to do it. One is to define a notion of *structural template* which is essentially a gestalt containing variables, and then say that two gestalts g_1 and g_2 in $\mathcal{G}_{<\mathcal{U},\mathcal{F}>}$ are analogous if there exists some 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 exists substitutions θ_1 and θ_2 such that $\theta_1^{-1} \circ g_1 = \theta_2^{-1} \circ g_2$.

This approach, however, seems too rigid because trivial changes in the structural description tree destroy the analogy. For example, the operator Con is associative, so that agk could be written as $Con(a, Con(g, k))$ or $Con(Con(a, g), k)$. However, the former would be considered analogous to $Con(b, Con(i, j))$ but the latter would not. To make the characterization of analogy invariant with respect to such trite changes in structural descriptions, we can allow a *unification theory* to be specified and used in the unification algorithm. The unification theory is essentially a set of axioms, where each axiom specifies a difference that may exist (or is allowed) between the terms that have to be unified. These differences may be concerned with, for instance, arity, recursion depth, order of arguments, etc. For example, to reflect the associativity of Con, we could include the axiom $Con(X, Con(Y, Z)) = Con(Con(X, Y), Z))$ in the unification theory. Now both the above mentioned gestalts of agk would unify with the above gestalt for bij. It should be pointed out, however, that adding axioms makes the unification problem in most cases computationally intractable Baxter (1977).

Another approach to this problem is to follow the framework presented by Indurkhya (1991) and Indurkhya (1992). Here the source and the target domains are formalized as algebras – in particular, as sub-algebras of the SIT-algebra – and analogical relations between these domains are defined as local homomorphisms between these subalgebras, which are referred to as *representation algebras*. This is the approach we will employ in this paper. We explain it in more detail in the rest

of this section. To do that, we successively introduce the notions of correspondence, sub-algebra, and (local) homomorphism.

Definition 6 *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* $< D_2, F_2 >$*, 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, where n refers to the arity of a function, and*
3. *if* $[a_1, b_1], \ldots, [a_n, b_n] \in \Delta$ *and* $[\omega, \sigma] \in \Omega(n)$ *then* $[\omega(a_1 \ldots a_n), \sigma(b_1 \ldots b_n)] \in \Delta$

In particular, 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.

The algebraic model can be applied directly to the *SIT-algebra*. However, in any given context, we may want to restrict our attention to only a small subset of the universe and a small class of operators; in other words, to a representation algebra. To define this formally, we need the notion of a subalgebra.

Definition 7 *A pair* $< E, G >$ *is a subalgebra of an algebra* $< D, F >$ *when* $E \subseteq D$, $G(n) \subseteq F(n)$ *for all n, and if* $e_1, \ldots, e_n \in E$ *and* $g \in G$, *then* $g(e_1, \ldots, e_n) \in E$.

A representation algebra is essentially a subalgebra that can be finitely generated. Finite generativity means that the set of operators is finite; and all the objects in its domain can be generated by a finite subset of it (by repeated application of operators).

Using these definitions, we model proportional analogies of the form "A is to B as C is to D" as consisting of two representation algebras of the SIT algebra, one generating the terms A and B, and the other generating the terms C and D, and a correspondence between the two representation algebras. For example, consider the proportional analogy "$abc : abd :: ijk : ijl$". The representation algebra $< \{ab, c\}, \{Con, succ\} >$ generates some descriptions for both "abc" and "abd". These descriptions might be "$Con(ab, c)$" and "$Con(ab, succ(c))$", respectively. Similarly, the representation algebra $< \{ij, k\}, \{Con, succ\} >$ 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 correspondence between the two representation algebras: $< \{[ab, ij], [c, k]\}, \{[Con, Con], [succ, succ]\} >$.
This characterization seems quite resilient because for any given pair of gestalts deemed analogous by this definition, any change in one of them resulting from the semantic properties of the domain-dependent operators can be reflected in an analogous change in the other gestalt.

There are two major consequences of defining the analogy relation as a correspondence. The first one is that many-to-many analogy relations are allowed, so there is no directionality to an analogy. Secondly, there is the constraint due to Definition 6, which says that though partial relations are allowed, there must be a closure. Incidentally, this closure property ensures that the domain and the co-domain of an analogy relation are algebras themselves.

For computational modeling of proportional analogy problems we would like to tighten the first constraint and loosen the second constraint. First, we introduce the functionality requirement on the analogy relations, so that one-to-many relations are not allowed. Such correspondences are known as *homomorphisms* and are formally defined as follows:

Definition 8 *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 a homomorphism if and only if* Δ *and* Ω *are such that whenever* $[x, y]$ *and* $[x, y']$ *are both in* Δ *or in* Ω *then* $y = y'$.

Imposing the homomorphism condition on analogy relations gives them directionality: it results in a mapping *from* $< D_1, F_1 >$ *to* $< D_2, F_2 >$, rather than between them. Reversing the direction of the mapping does not always yield a homomorphism (though it will always be a correspondence.) The homomorphism requirement is necessary because in solving a proportional analogy problem, there is the implicit direction from the known elements to the unknown.

Now with respect to the closure characteristic mentioned above, we find that it is computationally expensive, because to establish whether a mapping is structure preserving, a large amount of computation is necessary to check for all the ways of applying various operators to all possible tuples of objects. To address this issue, we introduce the notion of *local homomorphism.*

Definition 9 *Let* $< D_1, F_1 >$ *and* $< D_2, F_2 >$ *be two algebras. A relation* $< \Delta, \Omega >$ *between* $< D_1, F_1 >$ *and* $< D_2, F_2 >$ *is a* local homomorphism *if and only if,*

- *whenever* $[x, y]$ *and* $[x, y']$ *are both in* Δ *or in* Ω *then* $y = y'$*; and*
- *whenever* $[a_1, b_1], \ldots, [a_n, b_n] \in \Delta$, $[f, g] \in \Omega(n)$, *and there is* y *such that* $[f(a_1, \ldots, a_n), y] \in \Delta$ *then* $y = g(b_1, \ldots, b_n)$

Thus, a local homomorphism is concerned with only those objects that are included in its domain. Notice that it is a weaker condition in that every homomorphism is a local homomorphism, but not vice versa. In developing a computational model of proportional analogy in Section 6, we will use the notion of local homomorphism.

5 Introducing Constraints on Gestalts

Given a *SIT-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 correspondences between them. In this section, we present various factors that constrain the generation of gestalts for proportional analogy problems.

As mentioned before in Section 2, SIT assigns a complexity value called information load (IL) to each gestalt description. We also described there the IL definition proposed by Van der Helm and Leeuwenberg (1991), according to which the information load of a Gestalt description is computed by adding the number of occurrences of individual elements to the number of objects (See def. 3) consisting of more than one element (like (ab)). For adapting this to the algebraic version of SIT, we introduce one small modification: we count any embedded domain-dependent operators (except the identity operator) as 'elements'; but, as in Van der Helm and Leeuwenberg (1991), SIT operators are not viewed as contributing to the information load. So, for instance, given the pattern *abccba*, its gestalt $g_1 = Sym_e(Iter(a, succ, 3))$ has an IL of 2, because it has two elements a and $succ$, and neither is an object consisting of more than one element. But the gestalt $g_2 = Sym_o((ab), Iter(c, id, 2))$ has an IL of 4, because it has three elements a, b and c, and one object consisting of more than one element (ab). 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 collective 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 it has a higher information load. The reason is that g_2 shares common substructures with the preferred gestalt $g_3 = S_o(Iter(c, id, 2), Sym_e(ab))$ of *ccabbacc*. The fact that the preferred gestalts of two simultaneously present patterns share most substructures and result therefore in a lower collective information load will be called the *simplicity constraint.*

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 mentioned 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.

For proportional analogy, there is an additional constraint, which we refer to as the *projectability constraint*, 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* : *abccccba* :: *ppqrpp* : *ppqrstupp*. The first term, the same as in the example above, has a preferred gestalt of $Sym_e(Iter(a, succ, 3))$ $(IL = 2)$, but this does not form any appropriate analogical mapping with the preferred gestalt for the third term (*ppqrpp*), namely, $Sym_o(pp, (qr))$ $(IL = 5; p, p, q, r, (qr))$. To discover the mapping underlying this analogy, we may consider a higher information load gestalt for the first term, namely $Sym_o(ab, (cc))$ $(IL = 5; a, b, c, c, (cc))$. Thus, the preferred gestalts for patterns occurring in a proportional analogy relation ought to have a low complexity, ought to be generated by a simple representation algebra, and ought to satisfy the projectability 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 combine these constraints.

6 Computational Modeling of Proportional Analogy

In this section we propose a heuristic algorithm based on our formalism for solving proportional analogy problems. We first discuss an additional constraint imposed on the formalism due to computational requirements, and then formally define the notion of 'projectability' that plays a key role in our algorithm. Then, in Section 6.2, we describe Van der Helm & Leeuwenberg's algorithm for computing the preferred gestalt of a pattern, on which the top-level structure of our algorithm is based. In Section 6.3, we present an algorithm in our algebraic framework to solve proportional analogies, which can be seen as an extension of Van der Helm & Leeuwenberg's algorithm to incorporate the mutual contextualization effect. This algorithm uses a "test for projectability condition" module to generate mapping between algebras, and the algorithm for this module is explained in Section 6.4. We will present two simple examples in Section 6.5 to illustrate the working of our algorithm. Finally, in Section 6.6, we discuss a limitation of our algorithm.

6.1 Formal Preliminaries: Projectability

Earlier in Section 4, we defined proportional analogy "A is to B as C is to D" as a correspondence between two algebras, one of which generates gestalts for the terms A and B, and the other for C and D. In solving the proportional analogy problems, however, the task is to find an appropriate fourth term D, given the other three terms A, B and C. In this case, it is more useful and computationally efficient to require a stricter condition so that the analogy is a local homomorphism from the algebra that generates A and B to the algebra that generates C. Recall that the homomorphism condition rules out one-to-many relations, and the locality constraint requires a check for structure preserving property only in a restricted domain. Given these conditions, we can apply the local homomorphism to the gestalt of B to generate a gestalt for the fourth term D, thereby coming up with a solution of the analogy problem.

Next we define the concept of *projectability* that was introduced at the end of Section 5. Intuitively, a triple of gestalts (t_1, t_2, t_3) is said to be projectable if and only if there exists a representation algebra that generates gestalts t_1 and t_2, a representation algebra that generates

gestalts t_3, and a local homomorphism from the first algebra to the second which, when applied to t_1, yields t_3.[2] Formally, it is defined as follows:

Definition 10 *A triple of gestalts (t_1, t_2, t_3) is said to be projectable if and only if there exists a representation algebra $< D_1, F_1 >$ such that t_1 and t_2 are in $\mathcal{G}_{<D_1,F_1>}$, a representation algebra $< D_2, F_2 >$ such that t_3 is in $\mathcal{G}_{<D_2,F_2>}$, and a local homomorphism $< \Delta, \Omega >$ from $< D_1, F_1 >$ to $< D_2, F_2 >$ such that $< \Delta, \Omega > (t_1) = t_3$.*

Later in Section 6.4 we will present a heuristic algorithm to test for projectability.

6.2 Computing the Context-Independent Preferred Gestalt

Van der Helm and Leeuwenberg (1986) present an algorithm for computing the preferred perceptual gestalt of a pattern. This algorithm works in three steps. In the first step, a directed acyclic graph is constructed for the given pattern. If we place an index after each element in the pattern, starting from the leftmost element, then each node in the graph would correspond to an index, and each link in the graph from node i to j corresponds to a gestalt for the subpattern starting at position i and ending at position j. For the pattern *abccba*, this is illustrated in Figure 3(A). In the second step, all those links that represent subpatterns of length more than one, and are not covered by a SIT structural operator are removed. This is shown in Figure 3(B). Now in this reduced graph, each path from the start node to the end node corresponds to a gestalt for the whole pattern. So in the third step, the shortest path from the start node to the end node is computed.

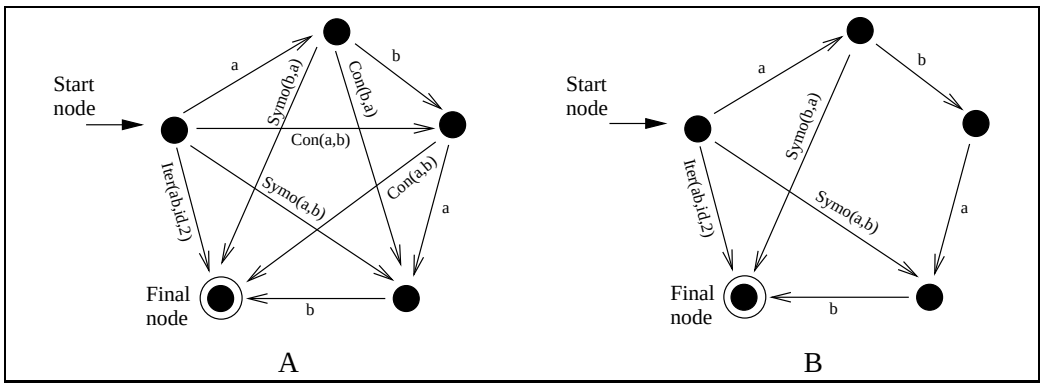

Figure 3: The complete top-level graph (A) and its reduced graph (B) of the pattern "abab"

Notice that this approach envisages a recursive structure between graphs: applied once, the algorithm gives the structural organization at the highest level only. For example, if the pattern is *aabbccaabbcc*, then the shortest path in its graph will be a single link corresponding to the gestalt $Iter(aabbcc, id, 2)$. But on a recursive application of the algorithm to *aabbcc*, we can find a preferable gestalt for it, namely $Iter(aa, succ, 3)$; and on a further application on *aa* we will find $Iter(a, id, 2)$; resulting in the final gestalt $Iter(Iter(Iter(a, id, 2), succ, 3), id, 2)$.

6.3 An Algorithm to Solve Proportional Analogies

We now present an algorithm to solve proportional analogies in our algebraic framework. This algorithm can be seen as an extension of Van der Helm and Leeuwenberg's algorithm to incorporate the mutual contextualization effect and the projectability constraint (as explained in Section 5).

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 simplicity and

[2] In applying a local homomorphism to a term, all the elements of the term that are mapped by the local homomorphism are replaced by the mapped elements.

projectability criteria 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 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. This yields a list of sets of triples where elements of each set have the same collective information load.

3. Test for the Simplicity Condition: Iterate through the list of sets of triples in the order of increasing information load until the end of the list is reached; and for each set do:

   (a) Pick an arbitrary triple from the selected set.

   (b) Test the triple for Projectability Condition: Check if there are two minimal representation algebras such that:

      i. One representation algebra $\mathcal{S}$ that can generate the gestalts for A and B.

      ii. One representation algebra $\mathcal{T}$ that can generate the gestalt for C.

      iii. A local homomorphism M between $\mathcal{S}$ and $\mathcal{T}$ exists which maps A to C.

   (c) If the end of the list is reached and no correspondence is found, return Fail.

4. Apply the local homomorphism M to the gestalt of B (in the triple that satisfied the projectability condition in the previous step) to get a gestalt for X.

A flow chart for the algorithm is shown in Figure 4.

In the first step of this algorithm, three sets S_1, S_2, and S_3 that contain algebraic terms for perceptual patterns A, B, and C, respectively, are generated. Then, from these three sets, the triples having the lowest collective information load are computed and put in the set *Set-of-triples*. Note that *Set-of-triples* is a subset of the Cartesian product of the sets S_1, S_2, and S_3. We compute *Set-of-triples* by keeping a variable *Curr-min-load* that is initialized to be the sum of the lowest information load gestalts in S_1, S_2, and S_3. All triples that have their collective information load equal to *Curr-min-load* are put into *Set-of-triples*. Then we increment *Curr-min-load* so that the set of triples with the next higher information load can be computed (if necessary.)

In the next step, a triple is randomly selected from *Set-of-triples*, and the projectability criterion is applied to it. If the triple is not projectable, then this module returns 'Fail'. Otherwise, it returns a representation algebra $\mathcal{S}$ that can generate the gestalts for the terms A and B (the first two gestalt in the triple), a representation algebra $\mathcal{T}$ that can generate the gestalt for the term C (the third gestalt in the triple), and a local homomorphism F from $\mathcal{S}$ to $\mathcal{T}$. The details of this algorithm for testing projectability are presented in Section 6.4.

If the projectability condition is satisfied for the selected triple, the fourth term of the analogy is generated by applying the local homomorphism F to the gestalt of the second term B. But if the projectability condition fails, then we select another triple from *Set-of-triples*, and repeat the procedure. If *Set-of-triples* becomes empty without yielding any projectable triple, we compute a new *Set-of-triples* using the incremented *Curr-min-load*. This procedure is repeated until a projectable triple is found.

Notice that we select the possible triples first according to the simplicity criterion and then according to the projectability criterion. Obviously, the inverse order of applying these criteria is also possible but would be very inefficient. Moreover, our algorithm incorporates the assumption 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.

Note that if the context-free lowest information load descriptions of the patterns are not appropriate for the proportional analogy, then our algorithm would choose different descriptions. Therefore, our model can construct creative as well as non-creative analogies Indurkhya (1992).

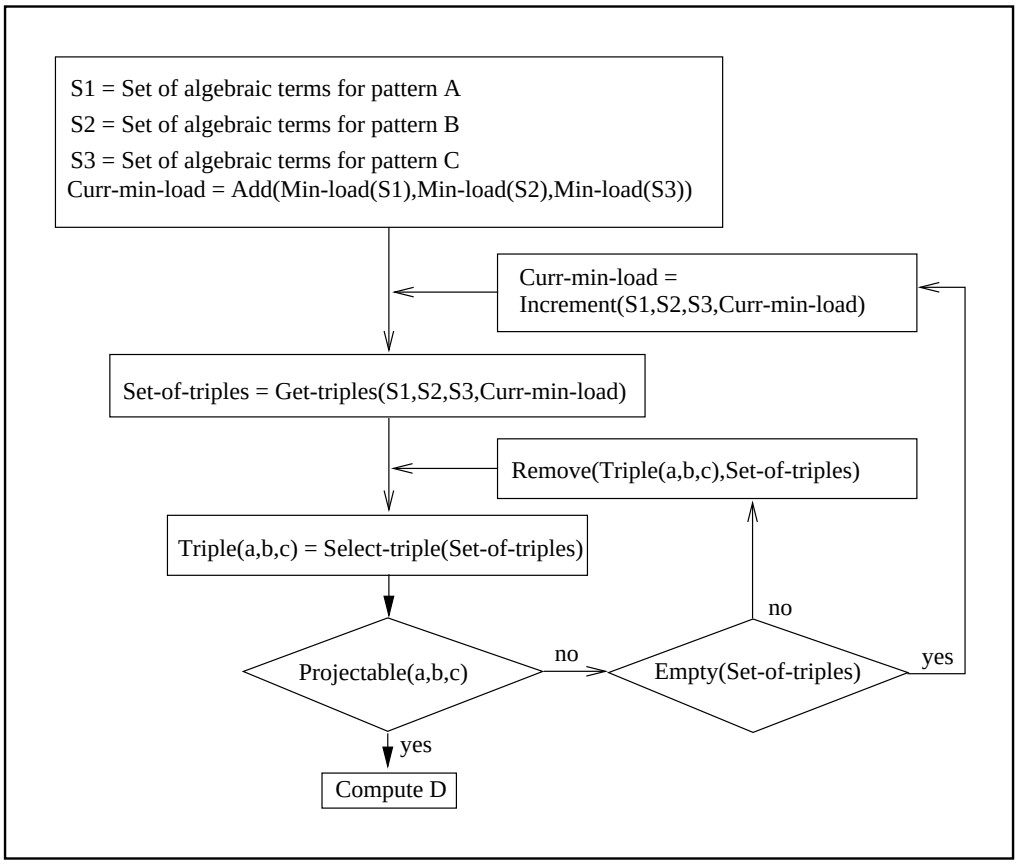

Figure 4: *The top-level structure of the algorithm for solving proportional analogy problems.*

6.4 An Algorithm for Testing the Projectability Condition

As mentioned above, a triple of gestalts (t_1, t_2, t_3) is considered to be projectable if we can find a representation algebra that generates t_1 and t_2, a representation algebra that generates t_3, and a local homomorphism from the first algebra to the other such that it maps t_1 into t_3. Moreover, in keeping with the constraints introduced in Section 5, we would like to keep the complexity of the two algebras and of the mapping to a minimum. In this section we present a heuristic algorithm for generating the two algebras and the mapping. The algorithm returns 'Fail' if the triple is not projectable.

As both representation algebras are subalgebras of the SIT algebra $< \mathcal{U}, \mathcal{F} >$, our algorithm focuses on finding a local homomorphism $M =< \Delta, \Omega >$ that maps t_1 to t_3. The two representation algebras are then simply obtained by taking the closure of the domain and the co-domain of $< \Delta, \Omega >$. The algorithm, a function called 'projection', starts by initializing Δ and Ω both to be empty sets. It then recursively examines the hierarchical structure of the three gestalts t_1, t_2 and t_3, starting from the outermost operator, and gradually adds the necessary elements to Δ and Ω.

In examining the outer structure of the gestalts t_1, t_2 and t_3, we distinguish between four cases, each of which is shown in Figure 5. In the cases shown in Figures 5(a) and 5(b), all the three gestalts are primitive terms, meaning that they all are either strings (i.e. belong to $D = \{a, abd, a(bc), \ldots\}$) or domain-dependent operators (i.e. belong to $F_D = \{succ, pred, \ldots\}$ or natural numbers (i.e. belong to $\mathcal{N} = \{1, 2, \ldots\}$). In the case shown in Figure 5(a), t_1 and t_2 are identical. In this case we increment Δ by adding the mapping $[t_1, t_3]$ to it. In the case shown in Figure 5(b), t_1 and t_3 are identical, and we add the two identity elements $[t_1, t_1]$ and $[t_2, t_2]$ to Δ.

In the cases shown in Figures 5(c) and 5(d), all three gestalts have some SIT operator at the outermost level. In the case shown in Figure 5(c), t_1 and t_2 have the same outermost operator,

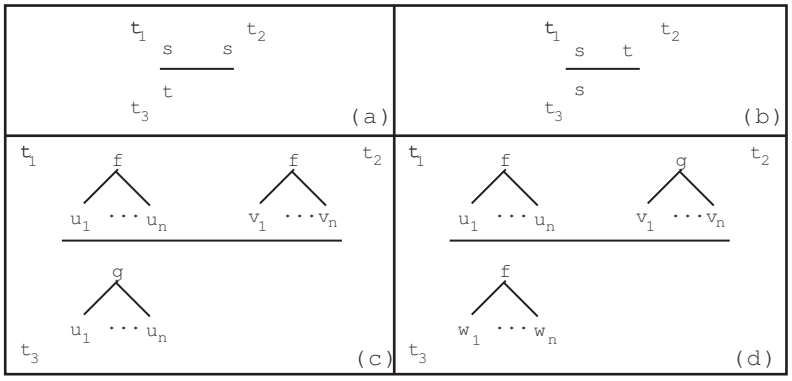

Figure 5: *Four cases for the projection algorithm.*

say f, and t_3 has a different outermost operator, say g, but of the same arity as f. Moreover, t_1 and t_3 have the same arguments to the outermost operator, say $u_1, \ldots, u_n$, assuming the arity of f and g to be n. In this case, we increment Ω by adding the mapping $[f, g]$ to it, and increment Δ by adding $[u_i, u_i]$, $[v_i, v_i]$, for $i = 1, \ldots, n$, assuming that $v_1, \ldots, v_n$ are the arguments to the outermost operator of t_2.

The fourth case shown in Figures 5(d) is recursive. Here, t_1 and t_3 have the same outermost operator, say f, and t_2 has a different outermost operator, say g, but of the same arity as f (say n.) The arguments to the outermost operators are all different. In this case, we call the projection function recursively with the triples formed by picking the arguments at the same positions of the outermost operators of t_1, t_2 and t_3. In other words, if the first arguments to the outermost operators of t_1, t_2 and t_3 were u_1, v_1 and w_1, respectively, then we make a recursive call to the projection function with (u_1, v_1, w_1); and so on for the second arguments, the third arguments, etc. All these recursive calls, would return us some mappings, say $M_1 =< \Delta_1, \Omega_1 >; \ldots;\ M_n =< \Delta_n, \Omega_n >$. Then $\Delta_1, \ldots, \Delta_n$ are joined together to get Δ, and $\{[f, f], [g, g]\}$, $\Omega_1, \ldots, \Omega_n$ are joined together to get Ω, making sure that the condition of homomorphism (Definition 9) is not violated. In all other cases, the projection function ends in a failure and returns 'fail'. The specification of the Projection function and its implementation is shown in Figure 6. Note that in this algorithm we do not follow the algebraic closure property to generate all possible objects. In fact, we generate only those objects that are needed to construct a mapping between the given terms.

The repeat loop in the algorithm implements the recursive case shown in Figure 5(d). The algorithm calls itself recursively by passing the arguments in the i^{th} positions of each of three terms. The returned value is accumulated in the variable M using a function called *Join,* which joins the objects and operators of two given mappings into a single mapping, making sure that the partial local homomorphism condition is satisfied, or else returns 'Fail'. Whenever the Join function returns 'Fail', the loop is immediately terminated.

In combining two mappings with the Join function, because the mappings are added incrementally, we can assume that the first mapping is already a partial local homomorphism. Then what remains to be seen is that adding the second mapping (1) does not make it so that a single element (an object or an operator) is mapped to two different elements, and (2) does not violate the operational structure of the two algebras. Though the first of these condition is easily checked, the second one requires the operators in the combined mapping be applied to the objects in the combined mapping to see that the algebraic structure is preserved. However, here also the fact that the mappings are added incrementally can be used to develop a more efficient algorithm. Figure 7 shows the specification and implementation of the Join function. Here M_1 and M_2 are two mappings that are known to be partial local homomorphisms. If M_1 and M_2 can be combined into a partial local homomorphism, then their union is returned as M, otherwise 'Fail' is returned.

Note that in this algorithm Ω is applied to Δ and it is checked if the local homomorphism

/* t_1, t_2 and t_3 are terms of the gestalt algebra for A, B and C, respectively. M is a local homomorphism that is returned by this function. */

Function : Projection (t_1, t_1, t_3): return $M =< \Delta, \Omega >$

Projection(t_1, t_2, t_3)
begin

/* Check to see if it is the case shown in Figure 5(a) */

if $t_1, t_2, t_3 \in D$, or $t_1, t_2, t_3 \in F_D$, or $t_1, t_2, t_3 \in \mathcal{N}$, then,
if $t_1 = t_2$, then return $M =< \{[t_1, t_3]\}; \emptyset >$,

/* Check to see if it is the case shown in Figure 5(b) */

else if $t_1 = t_3$, then return $M =< \{[t_1, t_1], [t_2, t_2]\}; \emptyset >$,
else return 'Fail'

/* Check to see if it is the case shown in Figure 5(c) */

else if t_1 is of the form $f(u_1, \ldots, u_n)$,
t_2 is of the form $f(v_1, \ldots, v_n)$, and
t_3 is of the form $g(u_1, \ldots, u_n)$, then
return $M =< \{[u_1, u_1], \ldots, [u_n, u_n], [v_1, v_1], \ldots, [v_n, v_n]\}; \{[f, g]\} >$,

/* Check to see if it is the case shown in Figure 5(d) */

else if t_1 is of the form $f(u_1, \ldots, u_n)$,
t_2 is of the form $g(v_1, \ldots, v_n)$, and
t_3 is of the form $f(w_1, \ldots, w_n)$, then
$M =< \emptyset, \{[f, f], [g, g]\} >$
$i = 1$
repeat
$M =$ Join(M, Projection(u_i, v_i, w_i))
$i = i + 1$
until ($M =$'Fail') or ($i > n$)
return M
else return 'Fail'.
end;

Figure 6: The algorithm that computes the Projection function.

/* M_1 and M_2 are two partial local homomorphisms. M_1 and M_2 are combined into a partial local homomorphism M. */

Function: Join ($M_1 =< \Delta_1, \Omega_1 >$, $M_2 =< \Delta_2, \Omega_2 >$):
return $M =< \Delta, \Omega >$

Join($M_1 =< \Delta_1, \Omega_1 >, M_2 =< \Delta_2, \Omega_2 >$)
begin

/* Check that the combination of two mappings is still a function and remove repetitions. */

For each pair $[x, y] \in \Delta_2$, if there exists a pair $[x, y'] \in \Delta_1$ then,
If $y = y'$ then remove $[x, y]$ from Δ_2,
else return 'Fail'.
For each pair $[x, y] \in \Omega_2$, if there exists a pair $[x, y'] \in \Omega_1$ then,
If $y = y'$ then remove $[x, y]$ from Ω_2,
else return 'Fail'.
Set $\Delta = \Delta_1 \cup \Delta_2$,
Set $\Omega = \Omega_1 \cup \Omega_2$,

/* Check to see that the combination of two mappings is a partial local homomorphism. */

For each $[f, g] \in \Omega_1$, and for each $[x, y] \in \Delta_1$,
if there is some y' such that $[f(x), y'] \in \Delta_2$ but $y' \neq g(y)$
then return 'Fail'.
For each $[f, g] \in \Omega_2$, and for each $[x, y] \in \Delta_1$,
if there is some y' such that $[f(x), y'] \in \Delta$ but $y' \neq g(y)$
then return 'Fail'.
For each $[f, g] \in \Omega_1$, and for each $[x, y] \in \Delta_2$,
if there is some y' such that $[f(x), y'] \in \Delta$ but $y' \neq g(y)$
then return 'Fail'.
For each $[f, g] \in \Omega_2$, and for each $[x, y] \in \Delta_2$,
if there is some y' such that $[f(x), y'] \in \Delta_1$ but $y' \neq g(y)$
then return 'Fail'.
Return $M =< \Delta, \Omega >$.
end;

Figure 7: The algorithm that computes the Join function.

condition is violated. But because $< \Delta_1, \Omega_1 >$ is already known to be a local homomorphism, we can reduce some of the tests. So, when Ω_1 is applied to Δ_1, we don't need to check for it in Δ_1, but only in Δ_2 (which are new elements that are being added.) Similarly, when Ω_2 is applied to Δ_2, we don't need to check for it in Δ_2, but only in Δ_1. But when we apply Ω_1 to Δ_2, we need to check in both Δ_1 and Δ_2 (hence Δ), and when we apply Ω_2 to Δ_1, we need to check in both Δ_2 and Δ_1 (hence Δ).

Regarding the constraints mentioned in Section 5, namely that the complexity of the two algebras and the mapping should be kept at a minimum, we would like to note that our incremental approach starts with empty algebras and adds elements only as necessary to generate the given gestalts. Similarly, the mapping is also started an empty set, and pairs of elements are added only when necessary and as far as possible identically. This turns our approach into a kind of *greedy algorithm*. This is essentially a heuristic approach, which does not always guarantee an optimal solution.

6.5 Two Examples

In this section, we present two proportional analogy problems, and show their underlying homomorphisms as computed by our projection algorithm of the last section.

Consider the proportional analogy problem "$abc : abcd = zyx : X$". If we consider $Iter(a, succ, 3)$, $Iter(a, succ, 4)$, and $Iter(z, pred, 3)$ to be the gestalts for the first three terms, respectively, then the local homomorphism induced by them is obtained as shown in Figure 8.

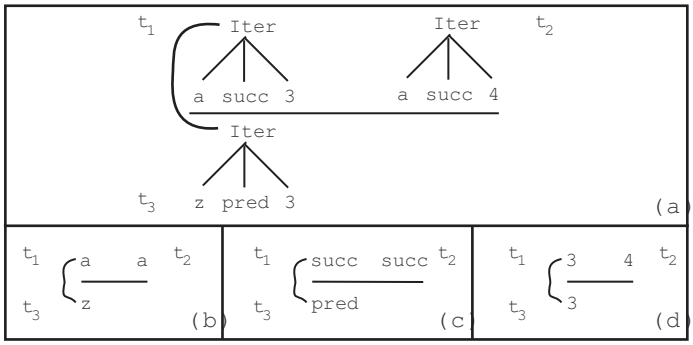

Figure 8: *An example of a mapping created by the projection algorithm.*

In the first step, the outer structures of the gestalts are compared as shown in Figure 8(a), to which the case of Figure 5(d) applies. As a result, the local homomorphism, initialized to be $[\emptyset, \emptyset]$, is set to $< \emptyset, \{[\text{iter}, \text{iter}]\} >$, and the projection function is recursively called on three triples formed by taking the arguments in the same position from each of the three terms, as shown in Figures 8(b), (c), and (d). The situations in Figures 8(b) and (c) correspond to the case shown in Figure 5(a), and the one in Figure 8(d) corresponds to Figure 5(b). Thus, the final value of the local homomorphism becomes $< \{[a, z], [3, 3], [4, 4]\}, \{[iter, iter], [succ, pred]\} >$. Note that this homomorphism maps the second element of the proportional analogy problem (i.e. $abcd$) to $zyxw$ which is considered as the solution (i.e. the fourth term) of the problem.

As a second example, consider the proportional analogy problem "$abc : abd = iijjkk : X$". Assume that the selected gestalts are $Con(ab, id(c))$, $Con(ab, succ(c))$, and $Con(iijj, id(kk))$, respectively. The local homomorphism induced by these gestalts for A, B and C is recursively obtained as shown in Figure 9.

As in the last example, the outer structure of the three terms, shown in Figure 9(a), corresponds to Figure 5(d), and so the local homomorphism is set to $< \emptyset, \{[\text{con}, \text{con} >\}]$, and the projection function is called with the two sets of triples formed by taking the first and the second

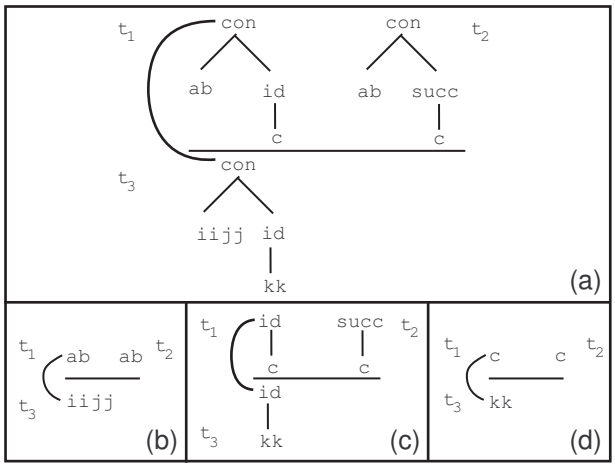

Figure 9: *Another example of a mapping created by the projection algorithm.*

arguments, respectively, from each of the three terms. The first of these calls, shown in Figure 9(b), corresponds to the case shown in Figure 5(a), and contributes the pair $[ab, iijj]$ to the local homomorphism. The second call, shown in Figure 9(c), corresponds again to Figure 5(d), and makes another recursive call to the projection function with the set of triples formed by the single argument of each of the three terms. At this point, the value of the local homomorphism is $< \{[ab, iijj]\}, \{[\text{con}, \text{con}], [\text{id}, \text{id}], [\text{succ}, \text{succ}]\} >$. In the last step, shown in Figure 9(d), which corresponds to the case shown in Figure 5(a), the element $[c, kk]$ is added to the local homomorphism, resulting in a final value of $< \{[ab, iijj], [c, kk]\}, \{[con, con], [id, id], [succ, succ]\} >$. Note that this homomorphism maps the second element of the proportional analogy problem (i.e. *abd*) to *iijjll* which is considered as the solution (i.e. the fourth term) of the problem.

6.6 A Limitation

We would like to point out here that there is an assumption built into our formalism: namely that the same object occurring at different places in an algebraic term is the same. But there are proportional analogies that do not satisfy this assumption. For example, consider the algebraic term $Con(Iter(Con(a, b), id, 2), a)$ which is a description of the first term of the proportional analogy $ababa : abbaa = cdcdg : cddcg$. In order to construct a reasonable analogical mapping for this example, one must distinguish between the two occurrences of the letter "a" in the mentioned algebraic term. In fact, the first "a" should be mapped to "c" and the second "a" should be mapped to "g".

This is a general phenomenon which exists for proportional analogies involving visual patterns as well. In order to cover such analogies, one needs a mechanism to distinguish different occurrences of identical elements. In our formalization, one solution is to assign indices to the elements that occur at different places in algebraic terms: such indices can be generated on the basis of the positions of the elements in the algebraic term. For example, we may rewrite the above algebraic term as $Con_{1,1}(Iter_{2,1}(Con_{3,1}(a_{4,1}, b_{4,2}), id, 2), a_{2,2})$. Notice that the two occurrences of the letter "a" can now be distinguished. The assignment of indices to algebraic terms and the construction of algebraic correspondences are further elaborated by Dastani (1998).

7 Conclusions and Future Research Issues

In this article, we have outlined an algebraic approach to modeling a process by which new gestalts of a perceptual stimulus can emerge in the context of proportional analogy. Our approach extends Leeuwenberg's structural information theory in two significant ways. One is to embed it in an

algebraic framework so that regularities based on domain-dependent relations are allowed to play a role in structural descriptions. The other is to modify the minimality principle so that mutual contextualization effect can be modeled. That is, when two (or more) patterns are presented together, we prefer those perceptual gestalts for the patterns that minimize the overall complexity; the gestalts of the patterns that minimize their individual complexity in isolation do not always result in the minimum overall complexity. This effect is modeled by introducing the concept of *representation algebras,* which generate a class of patterns, and defining a measure of complexity on them. Then mutual contextualization is modeled by examining representation algebras for all the patterns that are to be considered together, and choosing one that is minimal and also minimizes the complexity of all the patterns together.

Then we also formalize the notion of analogical mapping in our algebraic framework, and show how all these constraints can be integrated in an algorithm for solving proportional analogy problems. We claim that our algorithm is superior to other approaches to modeling proportional analogies in that we are able to incorporate the mutual contextualization effect, and the principles underlying our algorithm are clearly and formally explicated.

We would like to make some remarks now comparing our algorithm to the other existing approaches to solving proportional analogy problems. In the early days of artificial intelligence research, Evans (1968) implemented a system for solving proportional analogy problems in the geometric figures domain. However, in Evans' system, the representations of the figures (terms of the analogy relation) were determined first, and then the mappings were computed. Though Evans explicitly discusses the mutual contextualization effect, his system did not yet model it. Moreover, even the context-independent gestalts for individual figures were computed in a somewhat *ad hoc* fashion in Evans' system. 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.*

More recently, Mitchell (1993) implemented the Copycat system, which was expressly designed to model the creativity phenomenon in proportional analogies as we discussed in the introduction. 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, so it is not clear how its approach can be applied to other domains. 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 *SIT-algebra* 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 Section 6 mirrors in anyway how humans solve proportional analogy. By contrast, Copycat implicitly claims to model the human perceptual processes.

Needless to say, there are many other open issues that still need to be investigated, and we would like to mention a few of them here. The algorithm presented here provides one way to integrate the three constraints on analogy. We need to develop and study other possible computational models based on the theory.

Secondly, structural information theory, including the extension of it presented here, considers only one-dimensional patterns. While the coding system of Leeuwenberg allows some two-dimensional regularities to be captured in one-dimensional patterns, there are many other regularities that cannot be so captured. We are currently working on generalizing SIT in this respect Dastani (1998).

Finally, we would like to be able to model cognitive operations of grouping as well, which will allow us to explain how creative insights occur in cognitive domains. One cognitive domain where analogies play a key role, and where creative aspects of analogy can be glimpsed, is legal reasoning Indurkhya (1997). At this point, however, it remains an open issue whether and how the model of gestalt perception and disambiguation that we outlined in this paper would apply to creativity in cognitive domains such as legal reasoning. This, nevertheless, remains our long-term research goal.

References

Baxter, L. (1977). The complexity of unification. Technical Report Internal Report CS- 77-25, University of Waterloo, Faculty of Mathematics, Ontario, Canada.

Cohn, P. (1981). Universal algebra. Revised edition, D.Reidel, Dordrecht, The Netherlands.

Dastani, M. (1998). *Languages of Perception*. Ph.D. thesis, Institute of Logic, Language, and Computation (ILLC), The Netherlands.

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

Hofstadter, D. (1984). The copycat project: An experiment in nondeterminism and creative analogies. In *A.I. Memo 755, Artificial Intelligence Laboratory*. MIT, Cambridge, Mass.

Hofstadter, D. (1995). *The Fluid Analogies Research Group*. Basic Books, New York.

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

Indurkhya, B. (1992). *Metaphor and cognition: an interactionist approach*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Indurkhya, B. (1997). On modeling creativity in legal reasoning. In *Proceedings of the Sixth International Conference on AI and Law*, pages 180–189, Melbourne, Australia.

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

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

Mitchell, M. (1993). *Analogy-Making as Perception*. Bradford Books/MIT Press, Cambridge, Mass.

Siekmann, J. (1989). Unification theory. *Journal of Symbolic Computation*, 7:207–274.

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

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

Wertheimer, M. (1923). Untersuchungen zur lehre von der gestalt. *Psychologische Forschung*, 4:301–350.