Minimal Sorts for Interpreting Pictures

Henk Zeevat

Department of Computational Linguists,
University of Amsterdam
Spuistraat 134, 1012 VB Amsterdam,
The Netherlands
henk@mars.let.uva.nl

Dejuan Wang[*]

CWI
P.O.Box 94079, 1090 GB Amsterdam,
The Netherlands
dejuan@cwi.nl

Abstract

Computational approaches to simulate human beings interpreting pictures are important for understanding perceptual Gestalt and for building computer systems that support visual communication.

Based on the minimum principle, we present a new approach to the interpretation of pictures. Our contribution is that we developed a novel way in which the geometrical information is calculated.

In our approach, geometrical shapes are divided into various sorts and the sorts are organised into a hierarchical structure. A sort together with a number of points determines an actual graphical object. So, the objects themselves can be represented by the combination of their sort with certain points that code their position in the field and whatever other attributes they may possess as members of the sort. The geometrical information load is calculated as the number of points which are needed in the representation. Pictures are represented as a set of graphical objects. There are no other requirements on the input pictures. As long as the objects in a list are well-formed terms, interpretation can start. An inference mechanism reduces the terms in the list into terms which have the lowest information load. The deduced list of objects is the interpretation of the picture.

AMS Subject Classification (1991): 68N15
CR Subject Classification (1991): D.1.5,D.2.2,H.5.1,I.3.2
Keywords & Phrases: Gestalt rules, minimum principle, information load.

[*]I did this work under the auspices of the ERCIM Computer Graphics Network funded under the CEC HCM Programme.

1 Introduction

Computational approaches to simulate human beings interpreting pictures are important for understanding perceptual Gestalt and for building computer systems that support visual communication. Current parsers for visual languages [6] do not accept more general user input and do not recognise emergent objects. This is because there are no proper computational rules to guide the recognition of such input: allowing more general input would result in unmanageable ambiguity. According to [1], Gestalt rules have been used by graphic designers for centuries [4]. They were codified by the Gestalt psychologist Wertheimer [10] for all forms, using principles like similarity, proximity, closure and good continuation. These rules are however neither formal nor computational. A well-known approach that overcomes the latter problem is Leeuwenberg's [3]. Here the interpretation of a graphical representation is equated with finding the minimum code for the information presented. The main problem with his approach is however that it applies an in essence textual analysis to the geometric objects that we find in a graphical representation. Reductions are defined over sequences of characters (repetitions, alternation, symmetry) which makes it necessary to first reduce the graphical representation to a sequence of characters. That is why his approach has to rely on turtle graphics, which results in a series of problems for which the original Leeuwenberg approach has nothing to offer.

In this paper, we present a new approach to the interpretation of pictures which like Leeuwenberg uses the minimum principle [2]. We however develop a novel way in which the geometrical information is calculated. In our approach, geometrical shapes are divided into various sorts and the sorts are organised into a hierarchical structure (see [7] [9] [8]). A sort together with a number of points determines an actual graphical object. So, the objects themselves can be represented by the combination of their sort with certain points (parameter values) that code their position in the field and whatever other attributes they may possess as members of the sort.

The geometrical information load is calculated as the number of points which are needed in the representation, given the specification of the sort. Pictures are represented as a set of graphical objects. There are no other requirements on the input pictures. As long as the objects in a list are well-formed terms, interpretation can start. An inference mechanism, guided by the hierarchical structure of the sorts and using graphical inference, reduces the terms in the list into terms which have the lowest information load. The deduced list of objects is the interpretation of the picture.

A contribution of our approach is using the relationship between geometrical shapes and their point-representations. E.g. we can represent a square by its centre and one of its corner points, a rectangle by its centre and two corner points, a circle by its centre and one point of its circumference etc. This gives a simple way to measure the complexity of a geometrical shape. Furthermore, in our approach, the sorts of geometrical shapes which have been recognised and conceptualised by a human being play roles in the procedure of interpreting pictures. This makes it possible to naturally interpret many geometrical shapes, which cannot be properly interpreted by means of other approaches. Compare e.g. the example of the a square and a five-pointed star with one line missing, which in Leeuwenberg's representation cannot be distinguished (see figure 3).

The approach also admits of parametrisation by the adoption of different classes of sorts when only certain sorts are important or when special sorts need to be considered. It is also possible to provide certain sorts with a bonus value, thus directing the system to prefer certain options rather than others. An once interrupted line is not inherently less complex than the two line fragments of which it is made up (both require four points), but we may choose to favour broken lines over a set of two lines.

Another area where the current approach can be useful is avoiding misunderstandings in visual presentation systems. Under the current approach, we would reanalyse the output of a presentation system and compare the result with the data to be presented, rather than invoking a Gricean system [5]. In the latter case, a misunderstanding would be understood as the result of the violation of a Gricean principle, such as brevity or order and the misunderstanding would be a false implicature generated by an accidental feature of the (automatically generated) visual output. We would claim

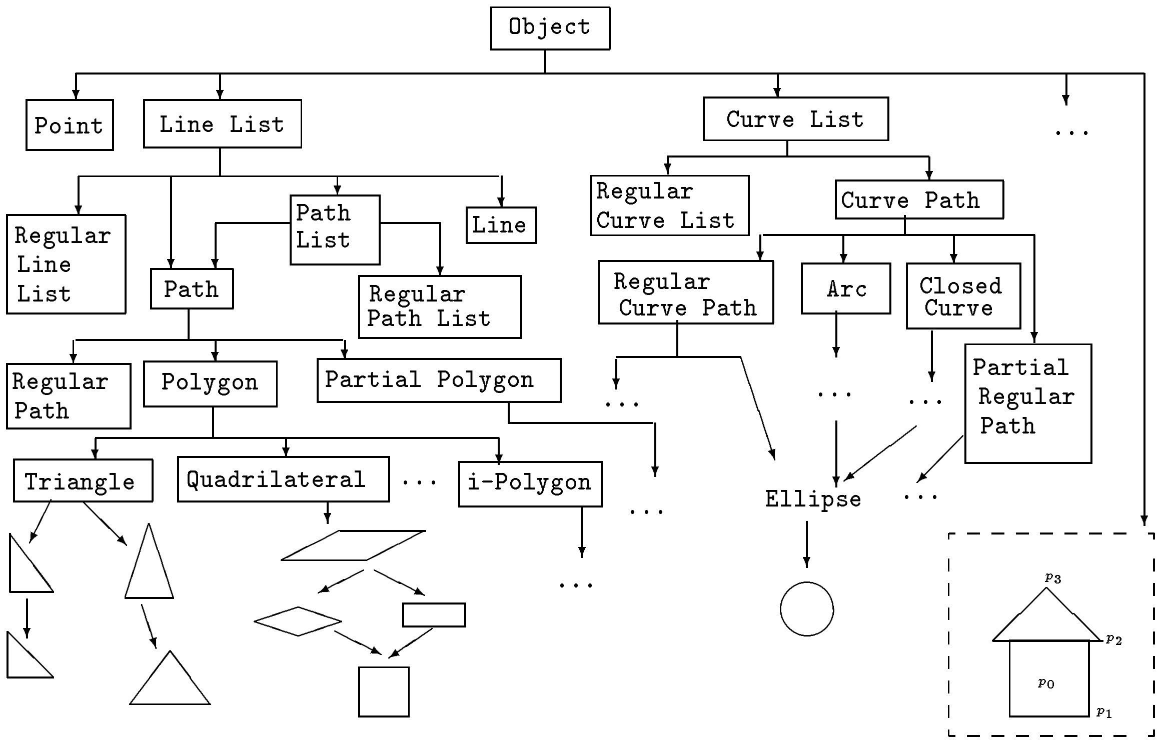


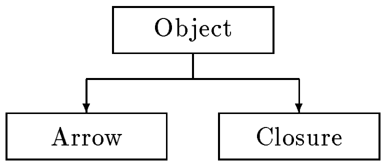
Figure 1: An example of the hierarchical structure of sorts.

that the misunderstanding is the result from the interpretation of an unintended structural description found by parsing the visual output with the accidental feature.

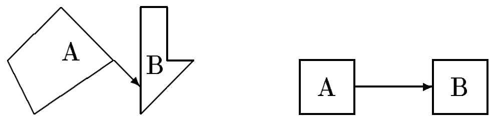
2 Sorts

Graphical objects are divided into *sorts* such as *Circle*, *Line*, *Rectangle* etc. The sorts are organised into a hierarchical structure by the subsort relation (see Figure 1 as an example). For instance, *Square* is a subsort of *Rectangle* in the sense that all the properties satisfied by rectangles are also satisfied by squares (For the use of sorts in externally interpreted graphics, see [7][9][8], for their use in visual languages [6]). There is a largest sort *Object* which is a supersort of all other sorts. The introduction of a sort *Object* allows the accommodation of those graphical objects that have no proper sort. For example, a drawing like which is difficult to recognise as anything, may be taken as an object of sort *Object*.

Figure 1 is just an example hierarchical structure of sorts. When one designs a particular system, the hierarchical structure of sorts should be organised according to the specific requirements. For example, consider a system which accepts user-drawn automata through a visual parser. In such a system, only *Closure* which represents automata states and *Arrow* which represents state transitions are relevant. A hierarchical structure of sorts for this system may simply consist of three parts:



In this example, the hierarchical structure between various closures is not important. It is enough to recognise an object as a closure or an arrow. Whether a closure is a quadrilateral or a square does not make difference for the system in interpreting the user's intention for this particular application. For instance, the following two pictures will be recognised as the same automaton.



However, if the system itself presents the recognised automaton, it should look like the one in the right side. In another words, pretty printing should be guided by the Gestalt principle. The representation on the right side uses the low complexity sort *Square* among closures and low complexity sort *Horizontal Arrow* among arrows. In this case, a more complicated hierarchical structure of sorts which can be used to guide pretty printing should be defined.

3 Representations of Graphical Objects

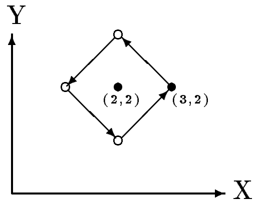
A graphical object is represented as:

$$SortName(PointInformation)$$

where $SortName$ is the name of the sort which the object belongs to and $PointInformation$ is used to locate and size the object. For example, suppose a square whose center is $point(2,2)$ and one of the corner is $point(3,2)$, then the square can be represented as:

$$square(point(2,2), point(3,2)).$$

Giving this representation, we can rotate $point(3,2)$ 90^o around $point(2,2)$ in one direction (either clock-wise or anti-clock wise) three times to obtain another three points. The three new points plus $point(3,2)$ are the four corners of the square, so the square is uniquely determined.



Of course, it can also be represented as two corner points, or a center point and a middle point on one of its edge.

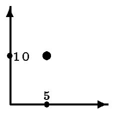
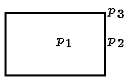
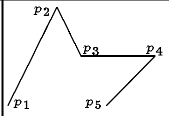
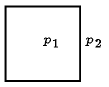
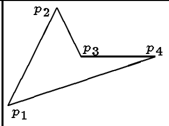
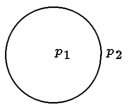
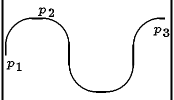
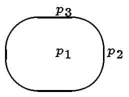
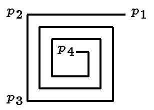
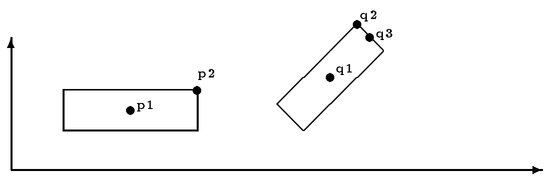
Sort	Example picture	representation	*	Sort	Example picture	representation	*
Point		$point(5,10)$	1	Rectangle		$rectangle(p_1,p_2,p_3)$	3
Line		$line(p_1,p_2)$	2	Path		$path(p_1,p_2,p_3,p_4,p_5)$	5
Square		$square(p_1,p_2)$	2	Polygon		$polygon(p_1,p_2,p_3,p_4)$	4
Circle		$circle(p_1,p_2)$	2	Sinusoid		$sin(p_1,p_2,p_3)$	3
Ellipse		$ellipse(p_1,p_2,p_3)$	3	R-Path		$rPath(p_1,p_2,p_3,p_4)$	4

Figure 2: Some sorts and their example representations, the column $*$ gives the complexities of the representations measured by the number of the points in the representations.

4 Complexities of Geometrical Shapes

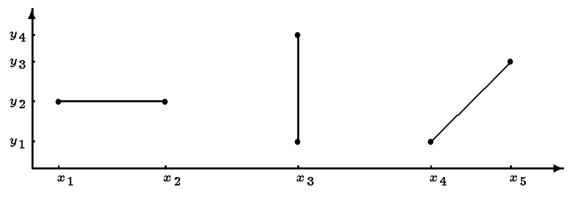
The interesting point is to see that most regular geometrical shapes can be uniquely identified by several geometrical points and the number of the points, which are necessary to locate a geometrical shape, relates to the complexity of the geometrical shape. Suppose C_1 and C_2 are two sorts. If C_1 is a subsort of C_2 then the objects in C_2 need more points to be located than the objects in C_1. For example, two points are needed to fix a square (i.e. one as the center and the other one as a corner), but to fix a rectangle three points are needed. This relates naturally to the fact that rectangles are more complicated than squares.

Figure 2 gives some example representations. Our intention is to illustrate the idea of such representations. In practice, many details need to be considered. For instance, should we distinguish rectangles which lay horizontally from those which do not?



A horizontal rectangle can be determined by only two points (p1 and p2) and a non-horizontal one needs three points (q1, q2 and q3). Does that mean horizontal rectangles are simpler than non-horizontal rectangles?

We think this depends on whether or not the coordinate system itself is counted in the measurement of the complexities. If it is counted, the basic unit for measuring the complexity should be the unit which constitutes a point instead of point itself. For instance, the complexity of a point $point(x, y)$ will be 2, i.e. one for the x and one for the y. Lines are still represented as two points, but there complexities are either 3 if they are horizontal or vertical, or 4 if they are not. See the following three lines;



The horizontal line is represented as:

$$line(point(x_1, y_2), point(x_2, y_2))$$

which has three different units x_1, x_2 and y_2 so the complexity is 3. The vertical one is represented as:

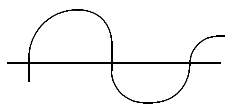
$$line(point(x_3, y_1), point(x_3, y_4))$$

whose complexity is also 3 measured from x_3, y_1 and y_4. The last one is represented as:

$$line(point(x_4, y_1), point(x_5, y_3))$$

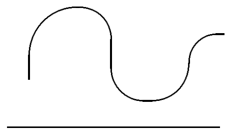
which has four units, so has complexity 4. In a similar way, the complexities between horizontally placed regular objects can be distinguished from the slanted ones.

Objects, which have no standard representations, can be represented as a set of points in a bitmap. This still allows them to have subparts which are organised as a bit of line, an ellipse fragment etc. The difficult part here is to decide that they are one object rather than many. Given that we are primarily interested in continuous shapes, a useful approach seems to be treat maximal continuous objects as objects. Maximality can be made into a factor which reduces complexity (the Gestalt rule *good continuation*). For example, the following picture:

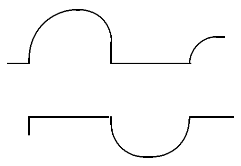


can be interpreted in different ways, such as:

1. a curve and a straight line:



2. two curves



Though both 1 and 2 consist of two continuous objects, Good continuation favours 2, because the differential of 1 is still continuous but the differential of 2 is not. The interesting thing is how to combine the minimal principle with such irregular curves? Can those irregular curves be represented so that there is a way in which their complexity can be measured according to their representations? It seems that we can adopt the representations for curves used in computer graphics, such as Nurbs curves. The *order* of a curve gives its continuation, i.e. higher order has good continuation, and the set of *control points* gives the approximate positions to defined the shape. Intuitively, if a shape needs more control points to be fixed, its complicity is higher than the one needs less if they have same order. So we speculate that their complexity can be measured by:

$$(the\ number\ of\ the\ control\ points)/order.$$

5 Graphical inference

There is a set of reduction rules corresponding to the sorts in the hierarchical structure. A reduction rule is represented as:

$$\Gamma \bullet [Condition(\Gamma)] \Rightarrow \Delta.$$

Γ is a list of objects, Δ is one object and $Condition(\Gamma)$ is a boolean function. If $Condition(\Gamma)$ is true, the list of objects Γ can be re-writted to Δ. $Condition(\Gamma)$ can be represented as a conditional expression if it is simple, e.g. $point(X,Y) = point(Z,W)$, otherwise it can be represented as a name of a grogram.

The purpose of applying reduction rules to a picture representation is to obtain a new representation which has lower complexity than the old representation. Therefore, if a rule

$$\Gamma \bullet [Condition(\Gamma)] \Rightarrow \Delta$$

is helpful, the following condition must hold.

$$Complexity(\Gamma) \geq Complexity(\Delta).$$

For example, applying the following rule

$$line(X,Y), line(X,Y) \bullet [slope(line(X,Y)) = slope(line(Y,Z))] \Rightarrow line(X,Z)$$

to the representation of two collinear lines (complexity is 4), we get a new representation which is one line (complexity is 2).

According to the discussion in the previous sections, if s is a subsort of s', the complexity of objects of s is smaller than those of s'. Therefore, the reduction rules should go downwards in the hierarchical structure of sorts. Therefore, if a rule has the following form:

$$s_1(b_1), ...s_n(b_n) \bullet [Condition(...)] \Rightarrow s(b)$$

then s should not be a super sort of s_1, s_2, ... s_n.

6 Add new sorts

When we create a new sort, the position of the new sort in the hierarchical structure of the existing sorts and the point representation of the new sort must be specified. For example, if we want a new sort *House* whose pictures consist of an isosceles triangle as the roof and a square as the room, it must be pointed out that *House* is a subsort of *Object* in Figure 1 and its representation is $house(p_0, p_1, p_2, p_3)$ (see the picture inside the dash box in Figure 1).

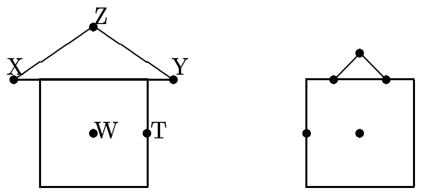
Moreover, the reduction rules, which actually give meanings to the new sort, should also be added into the set of existing rules. For example, when we add the sort *House*, we also add the following rule to the system.

$$isoscelesTriangle(X, Y, Z), square(W, T) \bullet [isAHouse(X, Y, Z, W, T)] \Rightarrow house(W, T, Y, Z)$$

$isAHouse(X, Y, Z, W, T)$ defines what an isosceles triangle and a square make a house. For instance, if $isAHouse(X, Y, Z, W, T)$ is defined as:

$$distance(Z, W) = distance(W, T) \ + \ \sqrt{(distance(Z, X)^2 - (distance(X, Y))^2/4}$$

Both the following pictures can be recognised as *house*.



However, if $isAHouse(X, Y, Z, W, T)$ is defined as:

$$distance(X, Y) > 2 * distance(W, T) \ \wedge$$

$$distance(Z, W) = distance(W, T) + \sqrt{(distance(Z, X)^2 - (distance(X, Y))^2/4}$$

only the first picture can be recognised as a *house*.

7 Pictures representations and interpretations

A picture is represented by a list of objects. For example the picture in Figure 3-I can be represented in various ways (see Figure 3-II).

The complexity of a picture is calculated by the number of the points in its representation. Different representations give a picture different complexity. In Figure 3, (1) is 16, (2) is 11, (3) is 7, (4) is 6 and (5) is 5.

Each of the representations corresponds to an interpretation of the picture. Here, following the minimum principle, the one which gives the lowest complexity to the picture is the preferred interpretation. In Figure 3, (5) is the preferred interpretation. I.e. the picture in Figure 3-I is interpreted as: a square and a partial star.

8 An interpretation system

A system which gives the preferred interpretations of pictures can be designed based on the above principle.

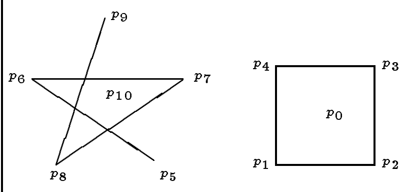
I	II	III
 p_9 p_6 p_7 p_{10} p_8 p_5 p_4 p_3 p_0 p_1 p_2	(1) $(line(p_1,p_2), line(p_2,p_3), line(p_3,p_4), line(p_4,p_1),$ $line(p_5,p_6), line(p_6,p_7), line(p_7,p_8), line(p_8,p_9))$	16
	(2) $(rectangle(p_0,p_2,p_3),$ $line(p_5,p_6), line(p_6,p_7), line(p_7,p_8), line(p_8,p_9))$	11
	(3) $(square(p_0,p_2), path(p_5,p_6,p_7,p_8,p_9))$	7
	(4) $(square(p_0,p_3), rPath(p_5,p_6,p_7,p_9))$	6
	(5) $(square(p_0,p_2), pStar(p_10, p_5, p_9))$	5
	....	....

Figure 3: I is an example picture, II are the possible representations (interpretations) of the picture (where *rPath* means a regular path and *pStar* means a partial star) and III is the complexities of the representations. Representation (5) has the smallest complexity, so it is the preferred interpretation of the picture.

The input of the system is a representation of a picture. As we saw in the last section that a picture can be represented in many different ways, each of the representations can be an input of the system. Suppose l is an input, the system will follow the algorithm:

1. let R be the set of reduction rules which can be used to rewrite l
2. if $R = \emptyset$ then stop (l is the interpretation)
3. $l' := l$
4. if $R = \emptyset$ then $l := l'$ and goto 1
5. let $r \in R$ be a rule according to r rewrite l to l''
6. $R := R - \{r\}$
7. if $Complexity(l'') < Complexity(l')$ then $l' := l''$
8. go to 4

The above algorithm guarantees that the reductions terminate, but whether or not a reduction gives the desired result depends on the system of reduction rules. For instance, if the reduction rules for the sorts in Figure 1 do not include any rules which can reduce a *square* with an *isosceles triangle* on the top to a *house*, and when an input has been reduced to such a triangle and a square, the algorithm stops though the result is not the one with the lowest complexity yet. However, this issue is about the properties of rewriting systems and we do not discuss it here.

9 An example

Suppose an input is the representation (2) in Figure 3-II:

$$l = (rectangle(p_0,p_2,p_3), line(p_5,p_6), line(p_7,p_6), line(p_7,p_8), line(p_8,p_9))$$

By locating objects $rectangle(p_0,p_2,p_3)$ into the node *Rectangle* and the following line list into the *Line List*, we can first apply the following rule attached to the sort *Rectangle* to the object $rectangle(p_0,p_2,p_3)$.

$$rectangle(X,Y,Z) \bullet [Distance(Y,Z) = \sqrt{2} * Distance(X,Y)] \Rightarrow square(X,Y)$$

We obtain a new representation l'.

$$l' = (square(p_0, p_2), line(p_5, p_6), line(p_7, p_6), line(p_7, p_8), line(p_8, p_9))$$

The complexity of the new representation l' is 10 which is smaller than the complexity of the old representation l (11), so the representation is changed to l'. By applying the following rules attached to the sort

$$Line\ List : line(X, Y), line(Y, Z) \bullet [] \Rightarrow path(X, Y, Z), and$$

$$path(L \mid X), line(X, Y) \bullet [] \Rightarrow path(L \mid X, Y)$$

the representation is changed to l''

$$l'' = (square(p_0, p_2), path(p_5, p_6, p_7, p_8, p_9))$$

Then applying rules, the *path* will be replaced by regular path which will be replaced by partial star. When the representation is reduced to the following representation

$$l^n = (square(p_0, p_2), pStar(p_10, p_5, p_9))$$

there are no rules which can be used to reduce the complexity of the representation, so it is the interpretation of the picture in Figure 3-I.

10 Other Issues

An important issue about picture interpretation is that sometimes people interpret a partially drawn object by its complete version. For instance, a partial square, whose two end-points connect with the edges of another object, is likely to be interpreted as a complete square. Our approach simulates this kind of interpretation by applying special reduction rules (conditional reduction rules) to extend an object whose sort is an extendable sort (e.g. *Partial Star*) to another object so that the complexity of the representation is smaller than before. Such a rule is an instance of a general scheme: X can be reduced to $Y - Z$ iff $X + Z$ can be reduced to Y. here $Y - Z$ is best understood as a general way of deriving sorts from other sorts. A minus type Y exists if we have a type X and there is a way of representing X as a set of objects such that a subset of these objects is a representation of Y. Y can then be seen as $X - Z$, where Z is any concept including those objects in the representation of X that are not objects in the representation of Y. The complexity is the sum of the complexities of the composing parts. In implementation, one can rely on prior recognition of Z for inferring minus-types, on special concepts for minus types (e.g. dotted lines, apartment layouts with doors) and on general operations on types (e.g. rectangle with a hole). It seems all three techniques are necessary.

Higher level interpretation concepts such as an apartment lay-out do not reduce the graphical complexity as such (the number of points for representation of the graphical object remains the same. Yet it seems, that such concepts will often be the preferred interpretation. If there are such concepts in the system, it is necessary to bias the system towards them, by making their complexity lower than the corresponding graphical object. This is trivial in a concept-based system. To see that such parametrisation is necessary, consider the following example. We have a system that represents people by squares of a fixed size. Now if four squares form a regular shape (e.g. the larger square), we would not want our system to to remark the large square in preference over their meaning of 4 people. This can again be achieved by giving a special lower value to the representing squares.

In comparison with the minimum code approach for the information developed by Leeuwenberg [1], our approach does not only calculate the geometrical information code which gives the preferred interpretation of a picture, but also determines the sort of the objects in the picture. For example, the picture in Figure 3-I will be interpreted by strings like $4ab$ and $4cd$ by means of the minimum code approach (where a is the length of an edge, b is the angle of the partial star and c, d of the square).

But it is interpreted as a *square* and a *partial star* by means of our approach. Furthermore, a square is quite different from a partial star and the former is simpler than the latter. This is reflected in our interpretation, but neither the conceptual difference nor the different complexities of the two objects are distinguished by the minimum code approach. Furthermore, our approach differs from certain other approaches (pattern recognition by means of neural network, for instance) in that our approach is based on the understanding of the relationship between pictures and their interpretations, while such understanding is generally not the concern of these practical approaches.

Acknowledgement

We would like to thank John Lee for his valuable comments.

References

[1] Foley, van Dam, Feiner, and Hughes. *Computer Graphics: Principles and Practice.* Addison Wesley, 1990.

[2] J. Hochberg and E. McAlister. A quantitative approach to figural "goodness.". *Journal of Experimental Psychology*, 46:361–364, 1953.

[3] E. Leeuwenberg. A perceptual coding language for visual and auditory patterns. *American Journal of Psychology*, 83:307–349, 1971.

[4] A. Marcus. Computer-assisted chart making from the graphic designer's perspective. *SIGGRAPH*, pages 247–253, 1980.

[5] J. Marks and E. Reiter. Avoiding unwanted conversational implicatures in text and graphics. In *Proceedings AAAI*, pages 450–456. Menlo Park, CA, 1990.

[6] J. Rekers and A. Schuerr. A parsing algorithm for context-sensitive graph grammars- short version. In *IEEE Symposium on Visual Languages*, 1995.

[7] Dejuan Wang. *Studies on the formal semantics of pictures.* PhD thesis, University of Amsterdam, 1995. ILLC Dissertation Series 1995-4.

[8] Dejuan Wang and John Lee. Visual reasoning: its formal semantics and applications. *Journal of Visual Language and Computing*, 4:327–356, 1993.

[9] Dejuan Wang, John Lee, and Henk Zeevat. Reasoning with diagrammatical representations. In N. Hari. Narayanan Janice Glasgow and B. Chandrasekaran, editors, *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, pages 339–393. AAAI press/The MIT press, 1995.

[10] M. Wertheimer. Laws of organization in perceptual forms. In W.D. Ellis, editor, *A Source Book of Gestalt Psychology.* Harcourt Brace, New York, 1939.