Chapter 4 Spatial Relations

As discussed in Chapter 2, a *symbolic picture* or simply a *picture* is a grid where some of the slots are filled by picture objects. The symbolic projections originally were defined for a fixed-sized grid, where all objects are point-like objects. When we deal with objects of variable sizes and shapes, the 2D strings become insufficient to represent the complex relationships among them. This chapter first presents the concept of the generalized 2D string. In Section 4.1 we discuss why variable-sized grids are needed. The segmentation problem is introduced in Section 4.2. The formal definition of generalized 2D strings is given in Section 4.3, followed by the definition of empty space objects in Section 4.4. Section 4.5 introduces the different types of projections, which will allow the rotation of objects.

4.1. Variable-Sized Grids

The symbolic projection technique introduced in Chapter 2 assumes a fixed-sized grid. An immediate generalization is to allow for variable-sized grids. In other words, the objects can have variable size and shape, but we still reduce them to point-like objects, so that each object is represented by, for example, its centroid. The grid lines are drawn between the point-like objects, so that the "slots" are variable in size.

As an example, the picture in Figure 4.1 can be represented by the 2D string (u,v) = (a < b = c, a = b < c). The symbol '<' denotes the left-right spatial relation in string u, and the below-above spatial relation in string v. The symbol '=' denotes the spatial relation "at approximately the same spatial location as". Since the three objects are regarded as point-like objects, and the x-coordinates of the centroid of b and c are within a nearness threshold, the two objects are regarded as at approximately the same horizontal location. Therefore, the 2D string representation can be seen to be the symbolic projections of the point-like objects in a picture along the vertical and horizontal directions.

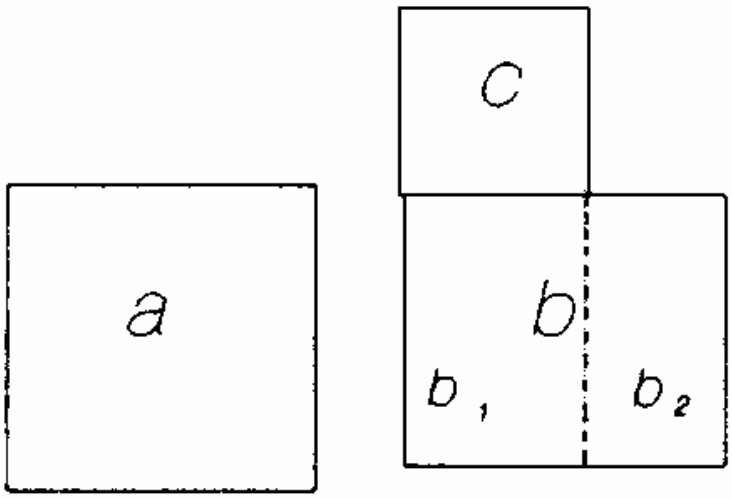


Figure 4.1. Symbolic projections of a picture consisting of objects with different size.

In the above representation, the spatial relational operator '=' can be omitted, so that the 2D string is more efficiently represented by (u,v) = (a < b c, a b < c). If the symbolic picture is given, we can take the symbolic projections to obtain the 2D string (u,v). Conversely, if the (u,v) is given, we can reconstruct a picture having symbolic projections (u,v), although the reconstruction may not be unique. Efficient algorithms for picture reconstruction and similarity retrieval have been developed.

The two basic spatial relational operators can be augmented by other operators. The edge-to-edge local operator, denoted by the symbol '|', can be used when two objects are in direct contact either in the left-right or in the below-above direction. Figure 4.2 illustrates the edge-to-edge relationship and the corresponding string representation. As to be discussed in Section 4.2 and illustrated in Figure 4.4, the edge-to-edge operator can be used advantageously to segment an object into connected sub-objects.

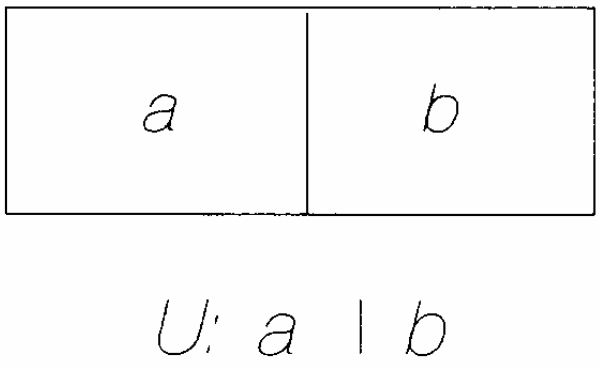


Figure 4.2. An example of the edge-to-edge operator.

4.2. Image Segmentation by Cutting Lines

The three spatial operators, '<', '=' and '|', are the most basic spatial operators. The edge-to-edge operator allows us to represent the

touching of two objects. For example, if we are to describe the relationships between the two objects b and c in Figure 4.1, we can use the following 2D string: (bc, b|c).

However, the above representation still does not describe fully the relationships between b and c. With the edge-to-edge operator, we can further segment an object into its constituent parts. This is accomplished by introducing cutting lines. Going back to Figure 4.1, the object b can b_2 segmented into b_1 and b_2, by drawing a cutting line as shown. Now the relationships between b and c are expressed by: (b_1 c | b_2, b | c).

In the above we use b_1 and b_2 to denote the two parts of b. If we use b to represent either the entire object b, or a part of it, then the 2D string becomes: (b c | b, b| c).

As another example, for objects that encompass other objects, such as the objects 'a' and 'b' shown in Figure 4.3(a), we can describe their relations by regarding them to be in the same "slot". Using a variable-sized grid, the size of the "slots" can be defined arbitrarily.

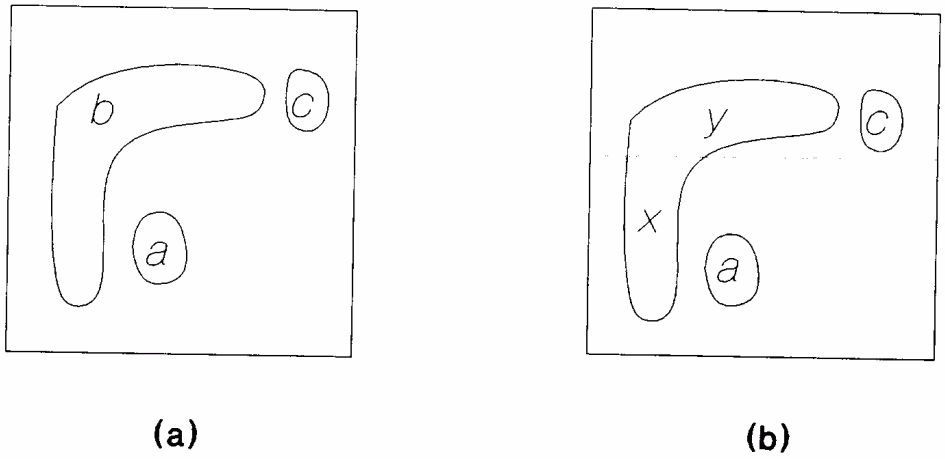


Figure 4.3. Encompassing objects (a) and their segmentation (b).

The 2D string for Figure 4.3(a) is (ab If the object 'b' is segmented into objects 'x' and 'y', as shown in Figure 4.3(b), the 2D string representation becomes (x Planar objects (such as lakes, towns, etc.) can be segmented into primitive constituent objects. Linear objects (such as roads, rivers, etc.) can be segmented into line segments. Point objects, of course, need not be further segmented.

This approach for hierarchical 2D string encoding is outlined below:

```
Procedure 2Dindex(picture,u,v)
begin
  /*object recognition*/
  recognize objects in the picture;
  find minimum enclosing rectangle (MER) of each object;
  /*segmentation*/
  while the MERs overlap
    begin
      segment overlapping objects into constituent objects;
      find MER of each segmented object;
    end
  /*now all objects are disjoint*/
  find centroid of each object;
  find 2D string representation using Procedure 2Dstring
end
```

The segmentation of an object can be done by drawing cutting lines as shown in Figure 4.1 and Figure 4.3(b). A systematic way of drawing the cutting lines will be presented in Section 4.3.

4.3. Definition of Generalized 2D Strings

Based upon the previously introduced three spatial operators, a systematic way of drawing the cutting lines is as follows: First, the extremal points are found in both the horizontal and vertical directions. A example is illustrated in Figure 4.4.

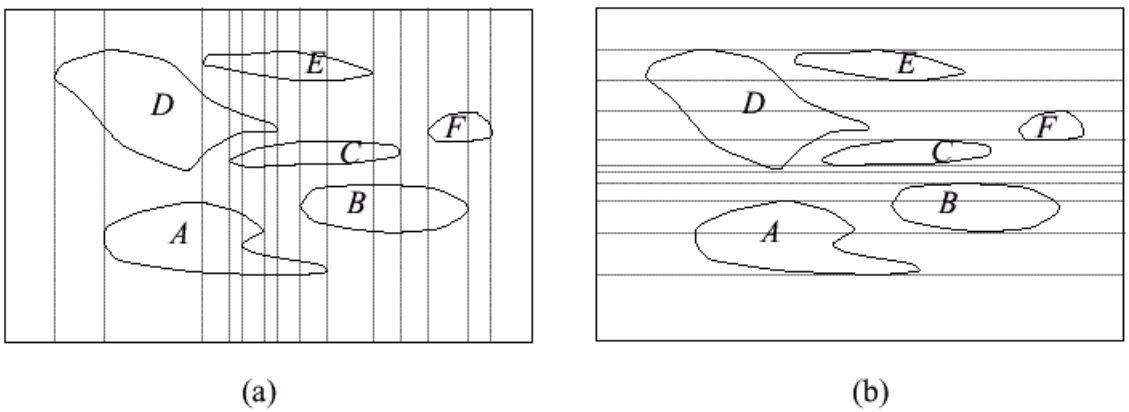


Figure 4.4 Image segmentation using cutting lines.

Next, vertical and horizontal cutting lines are drawn through these extremal points. This technique gives a natural segmentation of planar objects into the constituent parts.

With the cutting mechanism, we can also formulate a general representation, encompassing the other representations based upon different operator sets. This consideration leads to the formulation of a generalized 2D string system.

A *generalized 2D string system* is a five tuple (S, C, E_{op}, **e**, `< , >'), where S is the vocabulary; C is the *cutting mechanism*, which consists of cutting lines at the extremal points of objects; E_{op} = { < , = , | } is the set of *extended spatial operators*; e is a special symbol which can represent an area of any size and any shape, called *empty-space object*; and `< , >' is a pair of operators which is used to describe local structure.

The cutting mechanism defines how the objects in an image are to be segmented, and makes it possible for the local operator `< , >' to be used as a global operator to be inserted into the original 2D strings. The symbolic picture in Figure 4.4 has cutting lines as shown by dotted lines. The *generalized 2D string* representation is as follows:

u: D | A **e** D | A **e** D **e** E | A **e** C **e** D **e** E | A **e** A **e** C **e** D **e** E | A **e** C **e** D **e** E | A **e** C **e** E | A **e** B **e** C **e** E | B **e** C **e** E | B **e** C | B | B **e** F | F

v: A | A **e** B | B < D | D **e** C | D **e** F | D | D **e** E

4.4. Empty Space Object

In the above, the symbol **e** represents "empty space". The term "empty space" was first introduced by Lozano-Perez, to support the full description of a "room". Here we generalize Lozano-Perez's concept and use the special symbol **e** to represent empty areas of any size and any shape. Therefore, the expression "A **e** B" can be rewritten as "A B", and the generalized 2D strings can be simplified:

u: D | A D | A D E | A C D E | A A C D E | A C D E | A C E | A B C E | B C E | B C | B | B F | F

v: A | A B | B < D | D C | D F | D | D E

Furthermore, the expression "B < D" in the v-string can also be written as "B | **e**| D", indicating objects B and D are not touching.

The special empty-space symbol **e** and operator-pair `< , >' provide the means to use generalized 2D strings to substitute for other representations. In other words, we can transform another representation into the generalized 2D string and conversely.

4.5 Projection Types

Symbolic projection is a qualitative method in which various types of projections against the coordinate axis are performed as a means for generation of syntactic descriptions of space in order to identify a large number of object relations. Since the projections of the objects play such an important role it is necessary to have an understanding of the various types of projections that exist and are used in the method. For this reason those types that are most frequently used will be presented.

In the original approach by Chang et al., projections perpendicular to the x- and y-coordinate axis where used. Furthermore, the original approach was characterized by projections of the centroids of the minimum enclosing rectangles of the objects. In later approaches this was changed because that type of projections lead to an object-view that was inherently point oriented and that caused limitations in the way the syntactic object structures were interpreted both by man and by machine. An improved projection type that came to generalize the original approach is the interval projection type, which projects the extremal points of the object, that is the projection of the characteristic attributes corresponding to the minimal enclosing rectangles, discussed earlier in this chapter. This projection technique makes it easier to determine the relations among overlapping object projections. For this reason it was possible to determine a much richer set of object relations than was possible to identify using centroid projections. Centroid and interval projections are illustrated in Figure 4.5(a) and 4.5(b) respectively where the objects are represented with their minimal enclosing rectangles.

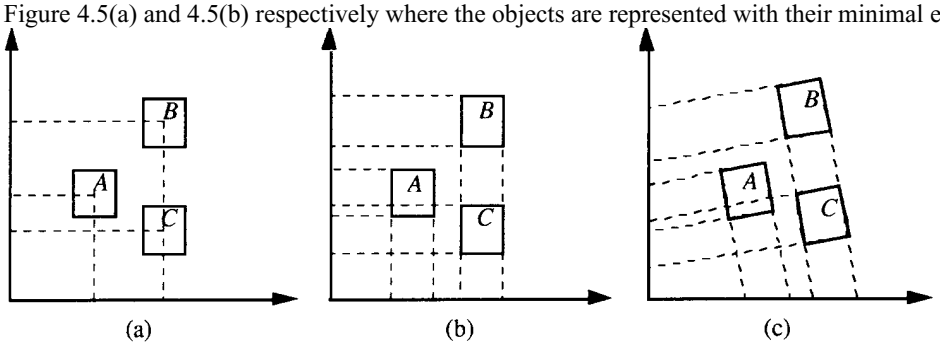


Figure 4.5 Centroid projections (a), interval projections (b) and slope projections (c).

Figure 4.5. Centroid projections (a), interval projections (b) and slope projections (c).

Slope projection is an extension of symbolic projection that came out as a result of the necessity to locate hidden or "badly" located points in space. It has also been used for determination of qualitative directions and distances. However, this technique must be handled with special restrictions since it may give rise to inconsistencies in the syntactic space descriptions. The method may in some cases be uneconomical since it may require extra computations. The slope projections are discussed in depth in Chapter 10 and a simple example

of this projection type is shown in Figure 4.5(c).

The projection types that have been discussed, so far, are the most commonly used, especially, in two dimensional applications. Another type that also is used in 2D applications is the polar projection type, which is strongly related to the slope projections.

The *S*-tree was introduced to support generalization of symbolic projection into higher dimensions. In such applications further extensions on how the projections should be viewed, as well as, how they should be performed are necessary. One approach that has been taken is illustrated in Figure 4.6. This type is more oriented towards projections that correspond to surfaces generated from 3D objects. A consequence of this is that the number of projection directions is increased as can be seen in Figure 4.6(b), where there are three surface projections, one in each plane in the coordinate system, i.e. the x-y-, x-z- and y-z-planes. Normally, the surface projections can be reduced into normal 2D projections, two for each plane in the coordinate system.

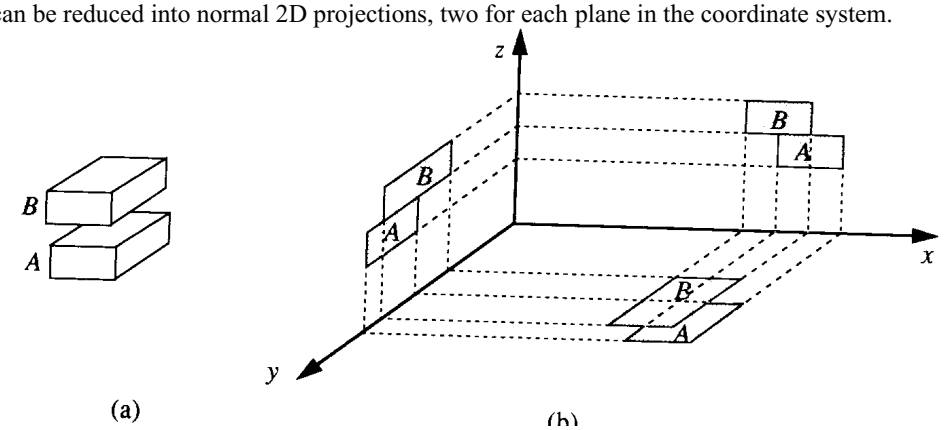


Figure 4.6. A universe including two objects (a) and their corresponding projections onto the y-x, z-x and y-z planes (b).

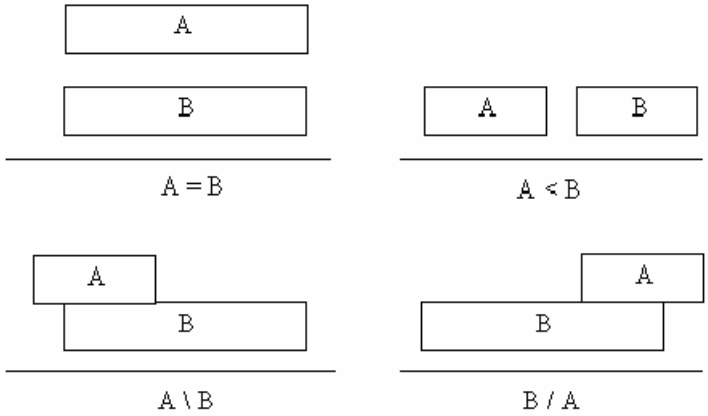
As a further generalization, 3D empty space objects can be introduced. The volumetric 3D empty space object **v** is similar to **e** except that it is three-dimensional. It can be used, for example, in situations where 3D projections are to be generated.

4.6. Extensions of the original operator set

In the original approach to symbolic projection, a limited set of relational operators, i.e., {=,<} was introduced. Later, it turned out that this small set was not sufficient. The main reason for this insufficiency is that the objects turned out to be regarded as point objects and hence the method turned out to be less useful for more complicated problems. A solution to this problem was to expand the number of operators in the original set.

Jungert's extension to the original set of relational operators is strongly related to Allen's temporal relations, i.e. to the temporal intervals described in [11]. This is not a surprise since in symbolic projection each projection string corresponds to a single dimension which is also the case in the temporal intervals described by Allen. For this reason it is easy to see that the relational operators describing completely and partly overlapping intervals must be present in any extension to symbolic projection. Allen defined 13 different temporal relations while Jungert's extension just contains 8. However, a closer look at the two sets shows that Allen's set includes the `inverses' of the relations described by Jungert. From a practical point of view there is hence no real difference between the two sets. The set of operators is {\, %, |=, =|, /, =, <, ~} where the operators are used to specify the spatial relations between the pictorial objects.

As before, U: is the label denoting the string of the symbolic projections along the x-axis while V: corresponds to the symbolic projections along the y-axis. The order of the objects along each coordinate axis is preserved in the 2D string; this correspondence must always be maintained. The order in the V-string is a permutation of the objects in the U-string. The definitions of the operators are



illustrated in Figure 4.7.

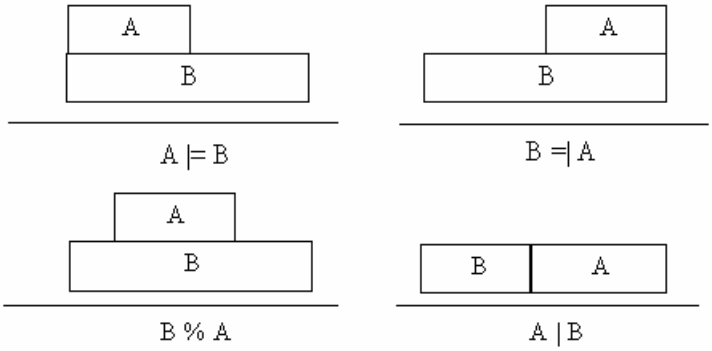


Figure 4.7. The extended set of spatial relational operators projected in one dimension.

The extension to symbolic projection given here was developed to performing local spatial reasoning, i.e., the motivation for extending the original set of spatial relational operators is not merely to give a complete and formalized image description but rather to identify the relations between a small number of objects local to a given area. Among the operators in the extended set, the `|'-operator (edge-to-edge operator} can be used for global reasoning as well.

4.7. Further extensions to the operator set

The extension to symbolic projection was mainly developed for local spatial reasoning and is not the only existing. In section 4.7.1 a split of the edge-to-edge operator called refined edge-to-edge [4], [5], [6] will be demonstrated. This split was originally defined to represent the images as multi-resolution hierarchies similar to quad-trees. In section 4.7.2 a further extension proposed by Lee and Hsu [7] and [12] will be discussed that can be applied on global level and used for similarity retrieval.

4.7.1. Refined edge-to-edge operators

A symbolic picture or simply a picture is an image where some of the slots are filled by picture objects. In pictorial information retrieval, the goal is often to retrieve pictures satisfying certain spatial relations such as, for example, `find the tree to the left of the house'. To specify such spatial relations two spatial relational operators were originally proposed. However, as has already been discussed, this set of basic spatial relational operators must be enlarged. The local edge-to-edge operator is denoted with `|', can be used when two objects are in direct contact either in the left-right or in the below-above direction.

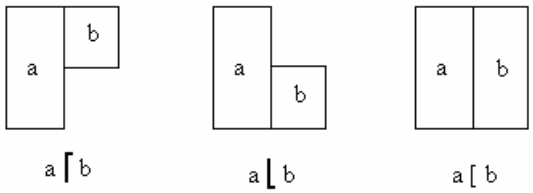


Figure 4.8. The refined edge-to-edge operators.

The set of the three spatial operators {<, =, |} corresponds to the minimal set of spatial operators. They can be used to describe any images in terms of symbolic projections completely although this may not always be efficient or optimal. It turns out that a further extension of the edge-to edge operator is possible, which is illustrated in Figure 4.8 where the edge-to-edge operator has been replaced by three local operators. Using these operators the original "u.v" strings of a symbolic picture can be transformed such that they can represent even more complex spatial relations among the pictorial objects in an image. For example, the "u.v" strings for the picture shown in Figure 4.9 can be transformed from (a < bc, ab < c) into (a < bc, ab ⌈ c). The use of the refined edge-to-edge operators will be discussed further later.

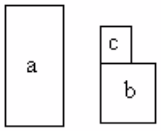


Figure 4.9. A simple picture suitable for description with the refined edge-to-edge operators.

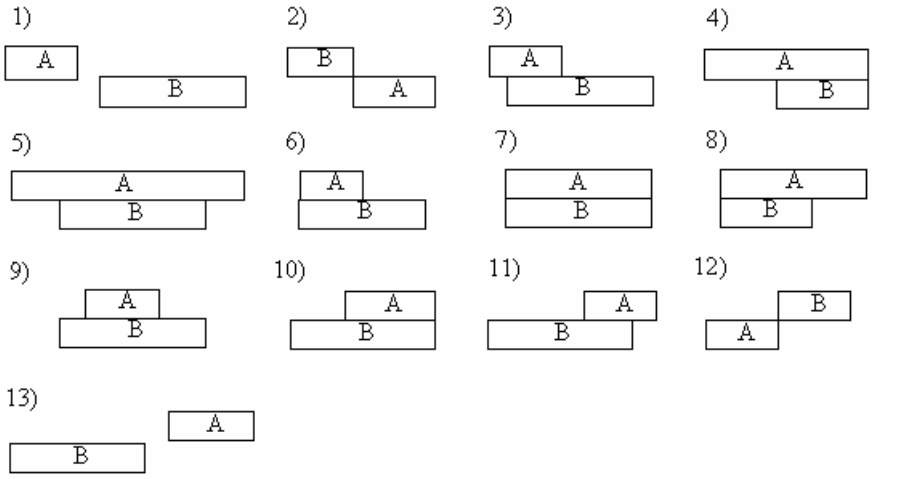


Figure 4.10. The 13 spatial relations according to Lee and Hsu.

4.7.2. The spatial operators according to Lee & Hsu

The extension proposed by Lee and Hsu is in fact an extension of the operator set suggested by Jungert that was described in section 4.7.1. The difference between Jungert's and Lee and Hsu's sets is that the latter also contains all possible inverses and for that reason is identical to Allen's set of temporal operators [11]. Lee and Hsu's set can be seen in Figure 4.10.

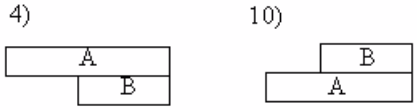


Figure 4.11. A spatial relation and its inverse according to Lee and Hsu.

An important question is whether the inverse spatial operators really are necessary or whether they can be left out. The answer is that they clearly are necessary in Allen's time dependent application which are one dimensional while symbolic projection is concerned with two or three dimensions. In the original work by Chang et al. the order of the projected objects was of no importance, i.e the U-string becomes the same in both alternatives in Figure 4.11, which correspond to relation 4 and 10 in Lee and Hsu's set. Hence, AB is equal to BA. Taking this into consideration there is not always a need for the inverse functions since the difference in order between the objects, A and B, is mirrored in the V-string. Consequently, inserting the inverses in the set of spatial relational operators is the same as including information from the orthogonal string, i.e. from the y-axis. Whether this should be done or not can be debated but there may be applications where this should be allowed as well. Table 4.12 illustrates the notation and meaning as well as the conditions of the operators.

Notation	Condition	Meaning
A < B	end(A) < begin(B)	A disjoint B
A = B	begin(A) = begin(B) and end(A) = end(B)	A equals B
A | B	end(A) = begin(B)	A edge-to-edge B
A % B	begin(A) < begin(B) and end(A) > end(B)	A contains B (different bounds)
A [B	begin(A) = begin(B) and end(A) > end(B)	A contains B (same begin)
A] B	begin(A) < begin(B) and end(A) = end(B)	A contains B (same end)
A / B	begin(A) < begin(B) and end(A) < end(B)	A partly -overlap B

Table 4.12. The definitions of the characteristic spatial operators.

Lee and Hsu have also brought there reasoning mechanism even further by taking the relations pairwise into account, i.e. by combining binary relations along both coordinate axes. Hence, it has been possible to identify 169 different types of spatial relations in two dimensions, which can be seen in Table 4.13.

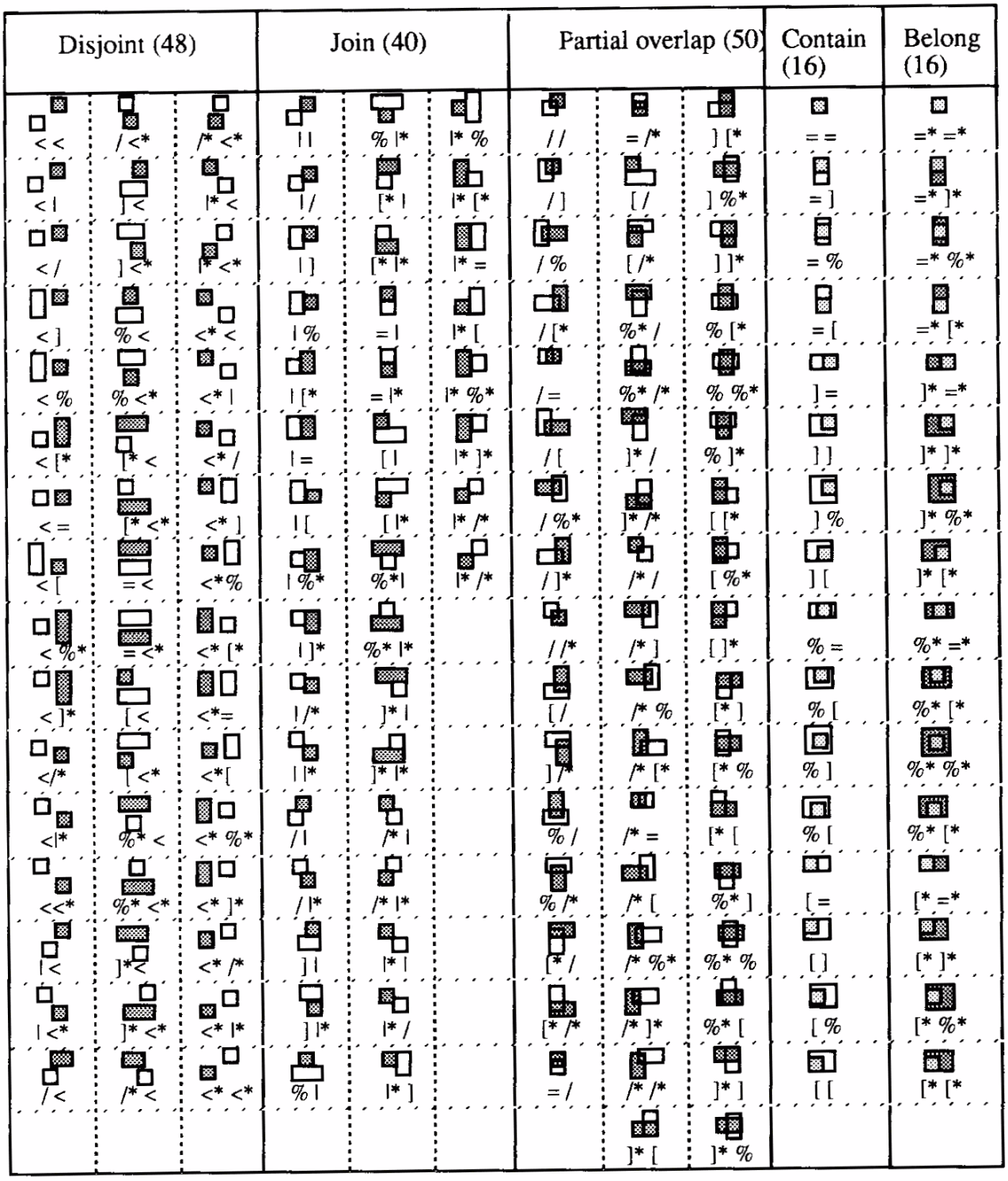


Table 4.13. The 169 types of spatial relations in two dimensions according to Lee and Hsu.

The "contain" and "belong" relations require an extra explanation since for these two cases one of the objects is entirely inside (overlapping) the other object. For this reason the overlapping part is illustrated with a different, brighter, pattern in the table to avoid that two different rela tions will look the same. For instance, the relations =] and]* =* will otherwise look the same.

4.8. The sparse cutting mechanism (Lee and Hsu)

The 2D C-string method introduced by Lee and Hsu [8] is clearly an important extension of the method for local spatial reasoning introduced by Jungert [3]. Their main application is similarity retrieval for which an improved cutting mechanism has been developed as well. The motivation for the new cutting mechanism is due to the fact that the original cutting method with the original set of two operatorsled to too many object slices. Thus Lee and Hsu's cutting mechanism is more economical with respect to the number of cuttings that need be made in a specific image.

The cutting according to Lee and Hsu is performed between partly overlapping objects. More specifically, cutting is performed in such a way that one of the overlapping objects is split into two parts, that is, cutting takes place at the end of the first ending object in a pair of overlapping objects. The object that is not split is called the dominant or dominating object. A simple illustration of this cutting mechanism can be found in Figure 4.14. The picture in Figure 4.14a can thus be represented with A] B] C | B [C which should be compared with the original cutting mechanism of the general string type that gives A | A= B | A = B = C | B = C | B. A comparison

shows that Lee and Hsu's cutting mechanism requires just a single cutting, for the objects in Figure 4.14a, while the original approach requires not less than 4. The number of relations are different as well, Lee and Hsu require four operators in the example while the general string type requires eight. This is due to the different cutting mechanisms. Furthermore, the end-bound point of a dominating object does not partition another other object if the latter contains the former, this is illustrated in Figure 4.14b. In this example object B is dominating C but both B and C are contained in A. Thus the latter object is not cut and the resulting string becomes A % (B] C | C).

Figure 4.14. A is dominating both B and C (a), B is dominating C where both B and C are contained in A (b).

In Figure 4.15 a more complex example that illustrates Lee and Hsu's cutting mechanism is found. Here there are three cuttings along the x-axis and just a single one along the y-axis. This should be compared to the original cutting mechanism that would have required 12 respectively 10 cuttings. From Figure 4.15 the following strings can now be determined:

```
U: D ] A ] E ] C | A = C = E ] B | B = (C [ E < F ) | F

V: A ] B | B < D ] ( C | F < E).
```

In the example just thirteen segmented sub-objects are created along the x-axis, compared to the general string type which gives 26. According to Lee and Hsu their cutting mechanism is always more economical than the original for overlapping objects and at least as economical for non-overlapping objects.

The algorithm for the cutting mechanism, which includes the creation of the 2D C-strings, can now be described. Assume the objects s1, s2, ... sn which are recognized in an image and then enclosed within minimal bounding rectangles. Then the following notations are used for these objects.

```
- pxib corresponds to the begin-bound point of si along the x-axis.

- pxie corresponds to the end-bound point of si along the x-axis.

- pyib corresponds to the begin-bound point of si along the y-axis.

- pyie corresponds to the end-bound point of si along the y-axis.
```

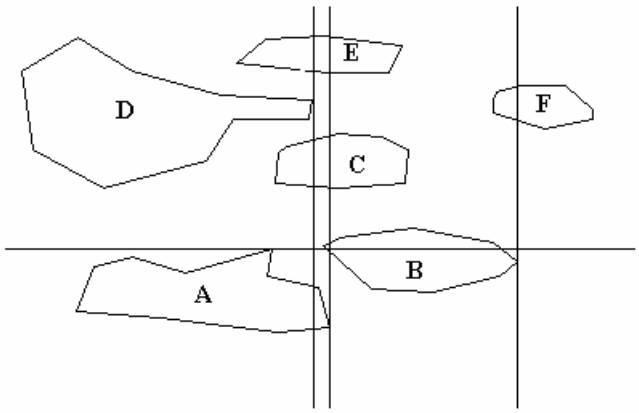


Figure 4.15. An illustration of the cutting mechanism according to Lee and Hsu.

Then the 2D C-string can be constructed from the symbolic picture f by first applying the cutting algorithm. Then the u- and the v-strings are constructed individually since their cuttings are orthogonal. Hence the cutting algorithm can be described as follows.

```
Procedure Cutting (s1, s2, .. sn)

begin

/* the output from this procedure is a 2D C String end-of-comment */

Step 1. Sort all the begin-bounds and end-bounds pxib and pxie of si
for i= 1,2, ... n.

Step 2. For the sorted points, group the points with the same value
into a same value list.

Step 3. For each same value list loop from step 4 to step 9 otherwise
put out the 2D C string.

Step 4. Check whether there are any end bound point in the list if
not go to step 9.

Step 5. Find the dominating objects from the objects in the given
end-bound list cuttings are performed at the end bound points of the
dominating objects.
```

```
Step 6. Cut those objects whose begin bounds are within the range of
the dominating objects.

Step 7. If there are no further unfinished objects situated before or
at the same position as the dominating objects put the dominating
object into the 2D C string.

Step 8. The remaining sub-part of segmented objects in step 6(1) are
viewed as the new objects with the begin bound as the location of
cutting line and the corresponding "edge" flag is marked.

Step 9. Collect the begin bound points of these new objects.

end
```

The cutting is split into three phases.

(1) The objects partly overlapping with the dominant object are segmented. According to the begin bound point values of latter objects, from largest to the smallest, the same begin bound objects are chained by `='-operators. The larger value list is merged into the smaller value list by `]' operators.

(2) The dominating objects, if more than one, are chained by `='-operators. If there further begin bound values which are equal, then the dominating list is extended with that value with a `[`-operator, otherwise the dominating list is merged into the former object by an `%'-operator.

(3). If the containing object in (1) and (2) contains other former objects, the latter object will be connected to the former by `<` or `|' depending on whether the "edge" flag was set in step 8.

4.9. Orthogonal Relations

The original approach to symbolic projection only allowed two types of spatial relations, left - right and below - above. A way of representing more complex spatial relations is first to split the object into segments from which all left - right and below- above relations among the components can be identified. Simple spatial relations can then be transformed into complex ones. Such simple spatial relations, which will be discussed subsequently, are called orthogonal relations [1], [2].

For objects where the minimum bounding rectangles are available, three types of topological relations between the rectangles can be identified:

```
* non-overlapping rectangles
* partly overlapping rectangles
* completely overlapping rectangles
```

The first alternative where the rectangles do not overlap is trivial and will normally not cause any problems because the object relations are simple. The other two might sometimes cause problems, especially when one of the objects partly surrounds the other. Figure 4.16 demonstrates a problem of this type. The fundamental issue here is to find a method that easily describes the relations between the objects. The method is called orthogonal relations, because it deals with spatial relations that are orthogonal to each other.

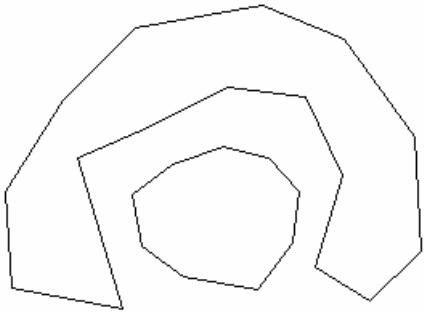


Figure 4.16. Two objects with overlapping MBRs.

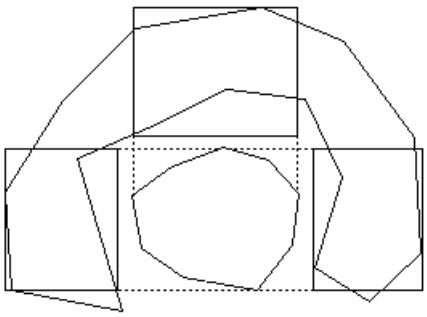


Figure 4.17. The PVO and its corresponding orthogonal relations.

The basic idea is to regard one of the objects as a "point of view object" (PVO). The PVO is projected on the other object in at most four directions (north, east, south, west). Hence, at least one or at most four sub-parts of the other object can be "seen" in the projection directions from the PVO. A part of the object that is actually "seen" is always in that direction where the two MBRs overlap, partly or completely. This is illustrated in Figures 4.17 and 4.18. The sub-objects will in the next step of the method be regarded as point objects,

which correspond to the centroids of those rectangles that enclose the sub-object. It is a fairly simple operation to identify and generate these points if the objects are represented in RLC. Thus in the next step the objective is to identify the relations between the POV and the sub-objects seen by the symbolic projection intervals. It is also of interest to observe that the orthogonal relations are established by a procedure that has similarities with a cutting mechanism, that is, it is correct to view this a local cutting mechanism that is just concerned with a pair of neighbouring objects.

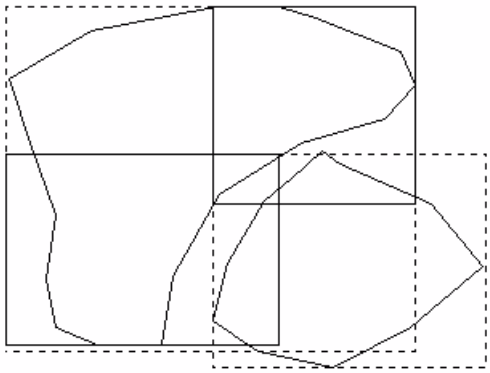


Figure 4.18. The PVO and its corresponding orthogonal relations for partly overlapping MBRs.

Each one of the sub-objects constitutes a sub-object orthogonal the PVO, and, since the sub-objects are regarded as points, a sparse description of the original object is generated. From this viewpoint, it does not matter whether the objects are of extended or linear type. However, it is of importance that the sub-objects are interpreted correctly. Figure 4.19a shows a correct interpretation of a north and a west segment while the interpretation of the same element in Figure 4.19b is erroneous. The natural interpretation is to look clockwise. Hence, nine different combinations can be identified:

```
2 points: N-E, E-S, S-W, W-N
3 points: N-E-S, E-S-W, S-W-N, W-N-E
4 points: N-E-S-W
```

No other interpretations are allowed. It is also possible to regard the elements in between the orthogonal ones. But this is not necessary since enough information is available anyway, see Section 4.7.2 for further discussions of the technique.

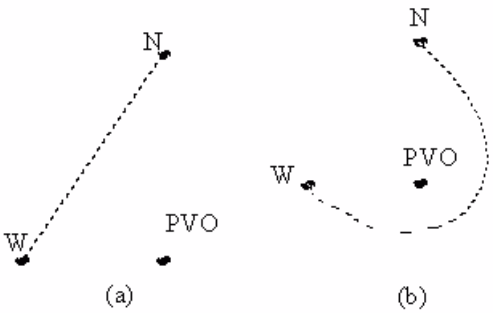


Figure 4.19. A correct (a) and an erroneous (b) interpretation of orthogonal relations.

The technique of finding orthogonal relations is described in the following algorithm.

```
Procedure Ortho (x, y)
begin
 /*this procedure finds the orthogonal relations of object x
   with respect to object y*/

 /*find the minimum enclosing rectangle of x and y*/
   find Mer(x), find Mer (y);

 /*find the four relational objects of object y intersecting
   with the  extensions of object x*/
 y-W = W-extension (Mer (x)), Mer (y);
 y-E = E-extension (Mer (x)), Mer (y);
 y-N = N-extension (Mer (x)), Mer (y);
 y-S = S-extension (Mer (x)), Mer (y);
 return ({y-W, y-E, y-N, y-S});
end
```

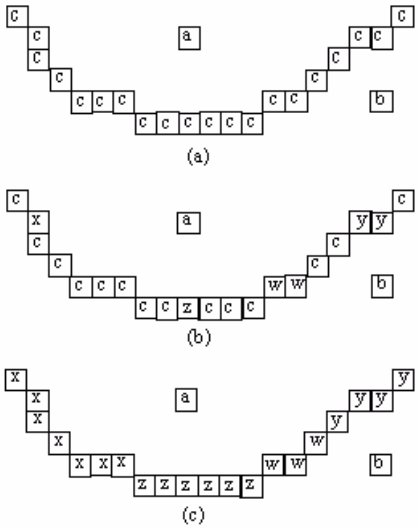


Figure 4.20. Three steps of a segmentation example which is based on the orthogonal relation technique where the original objects (a) are used to identify the orthogonal objects (b) and then the latter are expanded into segments (c).

4.9.1 Segmentation by Orthogonal Relations

The technique of finding orthogonal relations can be applied as a segmentation process to objects in an image. First, the image is pre-processed and the objects are recognized. Then, for each object x, the corresponding orthogonal relational objects in y are found. If the number of orthogonal objects is less than 2, the minimum bounding rectangles (MBRs) of objects x and y are disjoint. Therefore, no further processing is needed. If the number of orthogonal relational objects is greater than or equal to 2, then they will be added to the list Rel (y). After all object pairs have been processed, then for each object y a list of orthogonal objects Rel(y) has been accomplished. The object y can then be segmented into a number of segments corresponds to the number of members in Rel(y). The reference point of each segment will become the centre point of each orthogonal relational object. The symbolic projection strings (u, v) can then be obtained from this symbolic picture, where each segment is regarded as a separate object. The algorithm for generation of the orthogonal segments can be described as:

```
Procedure OrthoSegment (f, u, v)
begin
 /* object recognition */
 recognize objects in the picture f;

 /* initialization*/
 for each object x
 Rel (x) is set to empty;

 /* find orthogonal relations*/
 for each pair of objects x and y
 begin
 find Ortho(x, y)
 /*orthogonal relations of object x with respect to object y*/;
 if | Ortho(x, y) | > 1 then Rel(y) = Rel(y) H Ortho(x, y);
 find Ortho(y, x)
 /* orthogonal relations of object y with respect to object x*/
 if | Ortho (y, x) | > 1 then Rel(x) = Rel(x)H Ortho (y, x);
 end

 /* segmentation*/
 for each object x
 segment x into objects Rel(x);

 /*2D string encoding */
 apply procedure 2Dstring (f, m, u, v, n);
end
```

An example of the segmentation technique is given in Figures 4.20a through c. The original image contains objects "a," "b," and "c," where "a" and "b" are point objects, as shown in Figure 4.20a. After the procedure Ortho have been applied, it is found that Ortho (a, c) = {x, y, z}, and Ortho (b,c) = {w, y}. Therefore, object "c" contains four relational objects "x," "y," "z," and "w," as shown in Figure 4.20b. The segmentation result is illustrated in Figure 4.20c. The object "c" is thus split into four segments, "x," "y," "z," and "w". To obtain these segments, each orthogonal relational object is "grown" until it meets its orthogonal relational neighbours. The centre of each relational object is used as the reference point of each segment. In this way, all orthogonal spatial relations are optained and preserved.

The 2-D string encoding of the picture, in Figure 4.20, is $(u, v) = (x < az < w < by, z < bw < xya)$. To simplify the encoding, the orthogonal relational objects that are adjacent and belong to the same object must be merged. For example, if two orthogonal relational objects are merged into one segment, the reference point of either object can be used, or alternatively the centroid of the merged object can be used as its new reference point, provided that all orthogonal spatial relations are still valid. For example, if the objects "w" and "y" are merged into "y" the encoding becomes < u, v > = < x < az < y < b, z < b < xya >. All orthogonal spatial relations are thus preserved.

4.9.2. A Knowledge-Based Approach to Spatial Reasoning

Symbolic projection and its application to orthogonal relations provide a means that can be used as a basis for spatial reasoning. This will be illustrated in a number of examples below. From the 2-D string representation, spatial relations can be derived without any loss of information. Furthermore, from these spatial relations, even more complex spatial relations can be derived. Therefore, by combining the

2-D string representation with a knowledge-based system, flexible means of spatial reasoning and image information retrieval and management can be provided. The knowledge-based approach to spatial reasoning is illustrated by means of the two examples below.

Example 4.9.1.

Showing an island outside a coastline, Figure 4.21 illustrates the orthogonal relations. The 2-D symbolic projections are then:

```
U : C1 < C2i
V: C2 < C1i
```

The following rule can now applied:

```
 if U: r1 < r2p and V: r2 < r1p
 then
 fact:(south r p)(west r p)
 text:" The object

 is partly surrounded by the object  on its
 south and west sides.
```

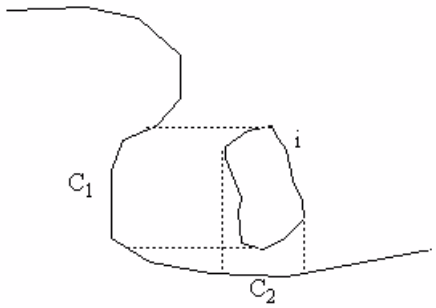


Figure 4.21. An island and its corresponding orthogonal relations in, e.g., a coastline.

When applying this rule to the example, r will be substituted by C and p by i, that is, (south C i) and (west C i). These facts are now stored in the fact database. The textual part can, e.g., be presented to the user.

Example 4.9.2.

Showing a forest near a lake, in Figure 4.22, L1, L2, and L3 illustrate the orthogonal relational objects which are part of a lake. The 2-D symbolic projections are:

```
U: L3 < FL1 < L2
V: L3FL2 < L1
```

This is matched with the following rule:

```
if U: r1 < p r2 < r3 and V: r1 p r3 < r2
then
fact:(west r p)(north r p)(east r p)
text: "The object

 is partly surrounded by the object  or its west, north and east side".
```

In this example, p is substituted by forest F and r by lake L

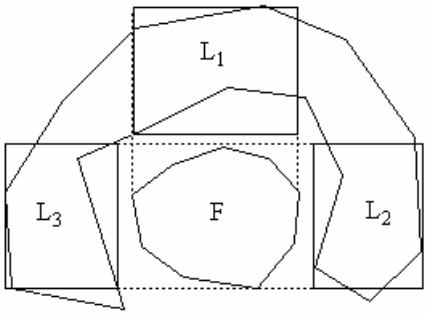


Figure 4.22. The orthogonal relations between a forest and a lake.

By successively applying rules that correspond to each one of the basic orthogonal relation types, it is possible to identify all object-to-object relations. For example, the assertions identified in Example 4.9.2 are (west L F), (north L F), and (east L F).

4.9.3. Visualization of Symbolic Pictures

The knowledge-based approach to spatial reasoning can be expanded further. In the examples in Section 4.9.2, it was demonstrated how spatial relationships between various spatial objects can be inferred from the projection strings. In this section, it will be demonstrated how spatial relationships can be inferred from symbolic descriptions of images.

Consider the problem: given the 2-D string representation of an image, can the original image be recreated from the projection strings in symbolic form? An algorithm for reconstruction of a symbolic picture from its 2-D string representation is can be found in [2], where the objects are assumed to be point objects that are pieces of segmented objects. Somehow these segmented pieces must be reconstructed (and reconnected). Of course, the RLC description can be retrieved from the image database directly and manipulated to visualize the objects. This is, however, time-consuming. An alternative is to apply connection rules to reconnect the segments from a reconstructed symbolic picture. Hence, by successively applying the types of rules discussed in Section 4.9.2 on more complex structures, supplemented with some further rules, it will be possible to generate sequences of fact that can be used to connect the orthogonal relational objects. This method is especially useful when the orthogonal relational objects are expanded into segments. Example 4.9.3 shows the rules used to generate all assertions needed to reconstruct a description of the object.

Example 4.9.3.

The problem is to describe an image and generate an approximate visualization. The image contains three objects, X, Y and Z. X is a linear object, and Y and Z are point objects. From the original image the following 2-D symbolic projection strings have been generated using their corresponding orthogonal relations.

```
X-to-Y projections
U: X2 < X3YX1
V: X3 < X2Y < X1

X-to-Z projections
U: X5 < X4Z
V: X4 < X5Z

Y to Z projections
U: Y < Z
V: Z < Y
```

In the X to Y projection strings, one of the basic rules that describes the applied relations is:

```
if U: r2 < r3 pr1 and V:r3 < r2 p < r1
then
fact:(south r p) (west r p) (north r p)
text: "The object

 is surrounded by  on its south, west and
 north side".
```

Thus the assertions:

```
(south X Y), (west X Y), and (north X Y)
```

can be identified. The relation X to Z uses the same basic rule as in Example 4.9.1 in Section 4.9.2. Hence the following assertions are found:

```
(west X Y) and (south X Y)
```

Finally, for the relation between Y and Z, none of the basic orthogonal relation rules are useful because this relation is even more primitive. The relations between Y and Z are of type non-overlapping rectangles. Hence the following rule can be used:

```
if U: r < p and V: p < r
then
fact: (northwest r p)
text: "The object  is to the northwest of

."
```

It is easy to see that eight further rules of similar type can be identified. The result of the execution of this rule gives:

```
(northwest Y Z)
```

The next step is to verify whether Y and Z reside on the same side of X or whether they are on different. In the latter case a rule from which:

```
(Between X,Y,Z)
```

can be concluded must be identified. However, this is not the case here. This implication is just mentioned because there must be some rules available that describe relations of this kind. Instead, the following rule is applied:

```
if (west r p) and (west r q)
then (west r p q)
```

from which

```
(west X Y Z)
```

is inferred.

By using rules of the type identified so far, it will be possible to infer the assertions that can be used to describe the image. However, it is obvious that general information concerning the types of the objects and the relative distances between them must be available as well when eventually describing the image. The solution to this problem is illustrated in Figure 4.23a where it is assumed that X is a road and Y and Z are buildings. Figures 4.23b and 4.23c, finally, illustrates two similar examples where the images have been reconstructed from the symbolic descriptions. In these two latter cases the projection strings are given as well as the corresponding symbolic pictures. Observe also, that although the forest in both examples are of extended object type, still the orthogonal relational objects, that are of interest, are handled as point objects. It is left to the reader to identify the remaining rules that are needed to solve these two problems and eventually reconstruct the images from the projection strings.

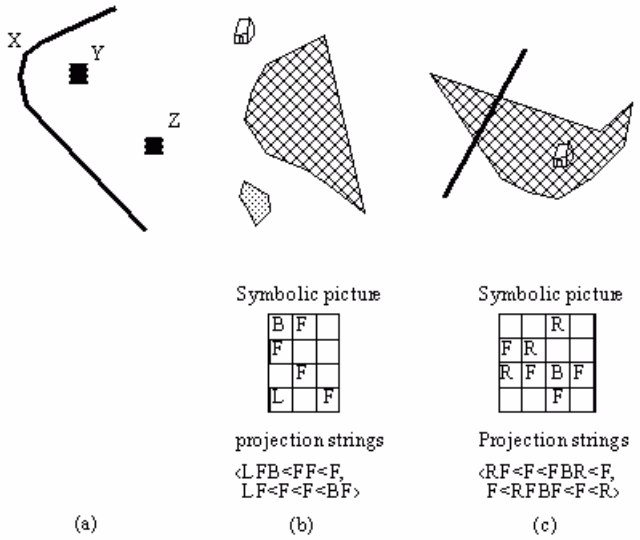


Figure 4.23. The result of connecting orthogonal relational objects: (a) of the road X and the neighbouring buildings Y and Z; (b) a building, a forest and a lake; (c) a road, a forest and a building.

References:

[1] S.K. Chang and E. Jungert, "A Spatial Knowledge Structure for Image Information Systems Using Symbolic projections", Proc. of the Fall joint Computer Conf., Dallas, TX, November 1986, pp 76-78.

[2] S.K Chang and E. Jungert, A Spatial Knowledge Structure for Visual Information Systems", Visual Languages and Applications, T. Ichikawa, E. Jungert and R. R. Korfhage (Eds), Plenum Press, New York 1990, pp 277-304.

[3] E. Jungert, "Extended Symbolic Projections as a Knowledge Structure for Spatial Reasoning and Planning", Pattern Recognition 1988, J. Kittler (Ed.), Springer-Verlag, Berlin Heidel- berg 1988, pp 343-351.

[4] S.-K. Chang and Y. Li, "Representation of Multi-Resolution Symbolic and Binary Using 2DH Strings", Proceedings of the Workshop on Languages for Automation, pp 190-195.

[5] S.-K. Chang, E. Jungert and Y. Li, "The Design of Pictorial Databases Based upon The Theory of Symbolic Projections", Design and Implementation of Large Spatial Databases, A. Buchmann, O. Gunther and T.R. Smith (Eds.), Springer-Verlag, Berlin Heidelberg 1990, pp 303-323.

[6] S.-K. Chang, E. Jungert, "The Design of Pictorial Databases", Journal of Visual Languages and Computing, Vol. 2, No 3, September 1991, pp 195-215.

[7] S.-Y. Lee and F.-J. Hsu, "2D C-string: A new Spatial Knowledge Representation for Image Database Systems", Pattern Recognition, vol. 23, no 10, pp 1077-1087, 1990.

[8] S.-Y. Lee and F.-J. Hsu, "Spatial Reasoning and Similarity Retrieval of Images Using 2D C-string Knowledge Representation", Pattern Recognition, Vol. 25, No 3, 1992, pp 305-318.

[9] M. Tang and S. D. Ma, "A new Method for Spatial Reasoning in Image Databases", Proceedings of the 2nd Working Conference on Visual Database Systems, Budapest, September 30 - October 3, 1991.

[10] M. Egenhofer, "Deriving the Combination of Binary Topological Relations", Journal of Visual Languages and Computing (JVLC), vol. 5, no 2, June, 1994, pp 133-149.

[11] J. F. Allen, "Maintaining Knowledge about Temporal Intervals", Communication of the ACM, vol. 26, no 11, Nov. 1883, pp 832-843.

[12] S.-Y. Lee and F.-J. Hsu, "Picture Algebra for Spatial Reasoning of Iconic Images Represented in 2D C-string", Pattern Recognition Letters 12 (1991), July, pp 425-435.

[13] T. R. Smith and K. K. Park, "Algebraic approach to spatial reasoning", Int. J. on Geo- graphical Information Systems, vol. 6, no 3, 1992, pp 177-192.

[14] G. Petraglia, M. Sebillo, M. Tucci and G. Tortora, "A Normalized Index for Image Databases", S.-K. Chang, E. Jungert and G. Tortora (Eds), Western Scientific Publ. Co, Singapore1995.