March 1971

ISUPPOSEW--A COMPUTER PROGRAM THAT FINDS REGIONS IN THE PLAN MODEL OF A VISUAL SCENE

by

Kazuhiko Masuda

Artificial Intelligence Group

Technical Note 54

SRI Project 8973

The research reported herein was sponsored by the Advanced Research Projects Agency and the National Aeronautics and Space Administration under Contract NASW-2164.

ABSTRACT

This technical note describes the nature and structure of the computer program ISUPPOSEW and some of its results. ISUPPOSEW is designed to enable a robot to make conjectures, on the basis of its visual information, about elements of its environment that it cannot see. The process of conjecture employed is analogous to that which a human employes in similar circumstances.

I INTRODUCTION

Suppose you visit someone's house and your visit is confined to one room, say the living room. After you have returned home, it may be interesting to conjecture, on the basis of your memory of the visual information acquired from seeing only one room, where the other rooms of the house are located. Similarly, we often guess the locations of elevators or exits in places such as department stores or halls. As a matter of fact, the results remain as conjectures unless one finally confirms the locations by seeing for oneself. A person tries to reduce the problem by conjecturing as reasonably as possible with the help of his empirical knowledge.

The computer program described in this technical note provides a means for conjecturing how the environment of a robot is constructed of regions by taking into account the unseen elements of the plan model of the scene that the robot now has as its model of the environment.

Let us first consider some applications. Suppose that you command a robot located in a large room to do a job that requires some information that the robot does not yet have. For example, you might give the command, "Turn in the corridor to the right and go into the third room," but the robot does not know where the corridor is. If, however, he can guess the most likely location of the corridor through his already-known information, he goes there, confirms its location, and solves the problem. If he finds that his first conjecture is wrong, he moves to the second possible point and looks for the corridor. We can consider another example. Suppose a robot asks the receptionist at the entrance of the university building, "Where is Professor K's office?" The receptionist may answer, "Turn to the right at the corner, and go straight on. You'll see a big office behind a smaller office in front." The robot must find the large room with a small office in front. When he finds an office that fits the description, he conjectures that he has solved the problem.

This type of conjecture will not be done by only one means. The exit of the building to the outside will be more easily conjectured by sound, wind, or light, but the visual model may also be important for that purpose.

Buildings usually consist of comparatively regular structures of a particular type. That is, very few homes are built with round rooms. A theater, however, may be circular, with the corridor surrounding the hall; hence, when a person is in a theater, he applies different conjectures with the knowledge that he is now in a theater. This process must be something like a global conjecture based on an elementary one by adding a different strategy and different information to the basic conjecture method.

Now, the author thought that it would be valuable to try to construct the more complete environmental model of a robot from the visual scene by following as closely as possible the process that a person does in making such conjectures.

The estimation of the results is related so much to the purpose of action and accumulation of empirical knowledge that simple programization is difficult. Regretfully, this program conjectures only by following several elementary rules given to the program beforehand and does not include any estimation of its procedure and results. For this reason the author named the program ISUPPOSEW--"W" means "DOUBLE," for the program and for the author. In addition, since the program technique of the author is very rudimentary, algorithms are elementary and need to be imporved. Although the program requires a rather long running time, example data examined are rather more complicated than actual data, for the author expects that this type of conjecture must be limited to several local areas of the model.

II HEURISTICS

As described in the Introduction, this program does not follow any theorem or axiom, nor does it have any estimation function to monitor the procedure of conjecture. It follows only the human way to conjecture as naturally as possible. Consequently, there may be some people who doubt the results. For those people, several rules that the program follows are given below.

First of all, the basic terminology--EPOINT, VIEWZONE, and VIEWLINE--is explained. Figure 1 shows a part of the plan model of visual scenes. Triangles constructed by three points including RB such as Δ(RB P1 P2) and Δ(RB P3 P4) are called VIEWZONEs, where RB signifies the location point of a robot. The lines constructing those triangles such as $\overline{\text{RBP1}}$, $\overline{\text{RBP2}}$, and $\overline{\text{P1P2}}$ are called VIEWLINEs. Lines such as $\overline{\text{P1P2}}$ are elements of models at the same time. Points such as P2, P3, P4, and P5 that are edges of only one element of a model are called EPOINTs.

The rules of the program are given as follows:

Rule 1--The elements of a model are thought to be related to each other by right angles or parallelism.

Rule 2--The conjecture procedure is applied only to all EPOINTs.

Rule 3--Conjectures elements of a model must not be drawn in VIEWZONEs, except in the special case of Rule 4(a).

Rule 4--There are three kinds of conjectures applied to EPOINTs:

(a) When two lines that contain the opposite EPOINTs are colinear with each other, these two EPOINTs are connected.

(b) The line that contains an EPOINT can be extended as long as the extended line segment does not violate Rule 3.

(c) In the case where Rule 4(b) cannot be applied because an EPOINT is in contact with a VIEWZONE, the line that contains the EPOINT is turned with a right angle to the direction in which the extended line does not cross the VIEWZONE and is extended in the same way as in Rule 4(b).

Rule 5--The extended or turned and extended line from an EPOINT is connected to the line that crosses the former one or may cross it if extended at the closest point to the EPOINT on the former one.

Rule 6--The conjecture procedure is repeated until no new conjectured line is created with regards to all EPOINTs.

Rule 7--The region surrounded by a single closing curve is thought to be a structural region of a model.

ISUPPOSEW is an algorithm that carries those rules into effect. A brief explanation about rules is added below.

In application of Rule 4(a), we can consider four cases shown in Figure 2. Figures 2 (b) and (d) indicate cases where there are several VIEWZONEs between opposite EPOINTs with lines colinear to each other. If a strict definition such as "GATE" or "DOORWAY" is preferred, Rule 4(a) may have to be applied only to cases (a) and (b), but because of inconvenience described later, in ISUPPOSEW, Rule 4(a) is applied to all four cases. However, in a case such as (d), since EPOINTs should be considered to be connected to line L, ISUPPOSEW treats those paired EPOINTS as NGATE (a structure that is not like a gate) and considers application of Rule 4(b) to them simultaneously.

Consider the configuration of Figure 3. With regard to points P2, P5, P7, Rule 4(b) can be also applied, but as is seen in the case of P5, infinite extension of the line is not allowable. The same thing may be considered also with regard to P2, or P7. P8 is the point to which Rule 4(c) is applied. The extension of the line must be done after turning to the right at the point, P8, and the line is connected to the extended line of P7. At P1 and P4, any conjecture is impossible. When only those points are left, the algorithm terminates.

Figure 4(a) illustrates Rule 6. Rule 4(a) is thought to be applied to P1, but unless P2 and P3 are connected, the extended line from P1 will cross a VIEWZONE. The sequence of consideration of EPOINTs is optional, and so we must withhold any conjecture about P1 until consideration on P2 or P3 is completed. This suggests that conjecture process must be repeated.

When line L3 is extended from P4 in Figure 4(b), it must terminate at the crossing point of the extended line of L1, rather than at that of L2. That is what Rule 5 explains; and this makes the program rather conservative.

III PROGRAM

A. Structure

This program can be divided into two parts: Part 1 consists of functions EX2IN, CONJECT 1, and CONJECT 2, and Part 2 consists of functions EX2IN* and RGNFND. Part II is the program that creates a new model of scenes by drawing possible conjectured lines in a given data model by following the rules: Part II is the program that separates a created model into several closed regions. Each part of the program is explained below.

B. EX2IN

Since the author dealt with only hand-written experimental data, a program to transform input data into internal format is needed. EX2IN is the program that transforms the input data shown in Figure 5 and Table 1 into the internal format shown in Table 2.

Table 1

EXAMPLE INPUT DATA

```
(  (RB (15.0 13.0))
    L1 (P2 (21.0 21.0) P3 (21.0 12.0))
    L2 (P1 ( 5.0  6.0) P5 (18.0  6.0))
    L3 (P8 (18.0 12.5) P5 (18.0  6.0))
    L4 (P4 ( 0.0 18.0) P6 (18.0 18.0))
    L5 (P4 ( 9.0 18.0) P7 ( 9.0 12.0))
    L6 (P1 ( 5.0  6.0) P9 ( 5.0 11.3))
    L  (POO( 0.0  0.0) POY( 0.0 99.0))
    T  (POY( 0.0 99.0) PXY(99.0 99.0))
    R  (PXY(99.0 99.0) PXO(99.0  0.0))
    B  (PXO(99.0  0.0) POO( 0.0  0.0))
```

Table 2

INTERNAL FORMAT

```
VIEW--POINTS--(P1 P2 ... P9 POO PXO PXY POY)
   |
   |   LINES--(L1 L2 ... L6  L   T   R   B)
   |
   |
   |
 RB--XCOR--15.0
   |
   |   YCOR--13.0
   |
 P1--XCOR--5.0
   |
   |   YCOR--6.0
   |
   |   TYPE--C
   |
   |   NLNS--(L2 L6)
   |
   |   NPTS--(P5 P9)
   |
   |  NVZNS--((RB P9 P1)(RB P1 P5))
   |
   |
   |
   |
POY--XCOR--0.0
   |
   |   YCOR--99.0
   |
   |   TYPE--C
   |
   |   NLNS--(L T)
   |
   |   NPTS--(POO PXY)
   |
   |  NVZNS--(NIL NIL)
   |
   |
 L1--ORT--(P2 P3)
   |
   |
   |
   B--ORT--(POO PXO)
```

The input data must have an assumed boundary region that covers all the territory of an original model. It is a square region surrounded by four straight lines L, T, R, and B, that connect points POO, POY, PXY, and PXO, where POO is not necessarily the origin; the origin is allowed anywhere.

The first line of the input format clarifies the location of a robot (RB signifies ROBOT). The values of X-Y coordinates of RB are put in parentheses next to RB. Then the arbitrary names of lines that are the elements of a model including assumed boundary lines are listed in optional sequence, followed by a list of end points of each line and their X-Y coordinate values after each name of a line.

The original data are named VIEW, and all the lines and points except RB are put into the property lists of VIEW with identifiers LINES and POINTS. Each line has one property list identified by ORT, where the list of both end points of the line are propped. Each point has property lists identified by XCOR, YCOR, TYPE, NLNS, NPTS, and NVZNS, but RB has only the properties XCOR and YCOR. XCOR and YCOR are values of X-coordinates and Y-coordinates of each point, respectively. TYPE indicates the type of points such as E for EPOINT and C for corner. NLNS is the abbreviation of neighbor lines; the list of lines diverging from the point is put into NLNS. NPTS means neighbor points, and it identifies the list of points connected to the point by NLNS. Finally, NVZNS signifies neighbor viewzones. One point in a model always has one viewline that is attached by two viewzones on both sides. The list of those viewzones is put under the identifier NVZNS. In the case of P2 in Figure 1, NVZNS has the list, ((RB P1 P2), (RB P3 P4)). EX2IN computes NVZNS of each point at its final stage, using the already transformed internal format.

ISUPPOSEW outputs EX2INED when the transformation is completed.

C. CONJECT 1

This is the program that applied Rule 4(a) to the model. Before entering CONJECT 1, ISUPPOSEW prepares the data list named ELIST, which is a list of all points whose types are E. CONJECT 1 creates all the possible pairs of EPOINTs and judges whether or not the pairs meet Rule 4(a). If such a pair is found, the new line that connects both points is created in such a manner that the function GENSYM names the line, the paired points are put into the property list of the new line (identified by ORT), and the new line is APPENDed to the list MODEL, which was prepared beforehand. The list MODEL is constructed in the form of the list of lines and their end points, though coordinate values of points and the list of RB are not listed.

The implementation of Rule 4(a) is done by the function named GLISTF. First of all, both points of a pair must have NLNS colinear to each other, and the pair connected in the original data--namely, elements of a model--or the pair that includes other colinear lines between them is deleted. Then whether or not both points have one common NVZNS is checked. If so, they are the pairs shown in Figures 2(a) and (c). Next, to distinguish case (b) from case (d), the function checks whether or not the connecting line of points crosses more than one viewline.

CONJECT 1 deletes the EPOINTs that meet Rule 4(a) from the ELIST and puts the left ones into the new list, E*LIST, but the EPOINTs considered to be NGATE are still left in E*LIST for further consideration.

ISUPPOSEW outputs the new current model created by CONJECT 1.

D. CONJECT 2

CONJECT 2 is one program that applies Rules 4(a) and (b), Rule 5, and Rule 6 to the EPOINTs listed in E*LIST. The algorithm is shown in Figure 6.

According to Rule 5, the repeated conjecture is required for the EPOINTs from which adequate conjecture is not extracted through each path. Consequently, E**LIST is set for those points, and the same process is repeated until the contents of both E*LIST and E**LIST become the same; namely, no more conjecture can be extracted.

The following are brief explanations of each stage of the algorithm.

1. Setting Candidate Lines

As is shown in Figure 7, we prepare three lines, LX, LY1, and LY2, starting from P and terminating at crossing points of border lines. Each of these lines is at a right angle to the next one. That procedure is done by three functions: CANDLX creates the list of three lines, LX, LXB1, and LXB2, computing the crossing point, XNP1, of the extended line, LX, with one of border lines, L,T,R, or B; the function CANDLY1 does the same computation with regard to the imaginary line, LR, which is the assumed line of L shifted to the right in a right angle against L with the center, P; and the function CANDLY2 does the same with regard to the assumed line, LL, the shifted line of L to the left. Three sets of lines (LX, LXB1, LXB2), (LY, LY1B1, LY1B2), and (LY2, LY2B1, LY2B2), are prepared. The internal format of those lines and newly created points such as XNP1, XNP2, XNP3, PR, and PL are temporarily made with property lists of ORT for lines and XCOR and YCOR for points for convenience of computation.

2. Determining Whether a Candidate Line Crosses NVZNS

The function CUTNVZNS determines whether a candidate line crosses NVZNS. One of three lines, LX, LY1, and LY2, that does not cross NVZNS of P is chosen. The priority is given to LX first and then either to LY1 or to LY2. In the case of P5 in Figure 3, LX crosses the NVZNS of P5, but it is possible to extend the line as far as the crossing point. Therefore, the criterion of CUTNVZNS is whether or not the extended line crosses only one VIEWLINE of NVZNS.

One extended line chosen from LX, LY1, and LY2 is called a candidate line of P and its set, i.e., (LX, LXB1, LXB2), is bound to CDLNS, and each element of the CDLNS is called CDLN, CDLN1, and CDLN2, respectively.

3. PROCESSK

This is a program to apply Rule 5, namely, to modify the current model, using the result of the judgment of whether the proposed CDLNS must be used as new elements of the model or whether they must be modified. The algorithm is shown in Figure 8. First of all, the list LNS, which consists of all lines picked up from the current MODEL, is prepared. Then, picking up each line from the list, the program judges the relationship between the line and CDLN. This judgment is carried on by the following four functions.

OVERLAPL (Figure 9(a))--This function judges whether or not CDLN overlaps the line L. The definition of OVERLAP here is either that both ends of L are inside CDLN or that only one of them is inside. Both lines must be colinear with each other. If so, the new line whose new end is the closer NPTS of L to P is bound to CDLN, and CDLN1 and CDLN2 are bound to NIL.

XNL* (Figure 9(b))--If the line L crosses the CDLN, the crossing point is calculated, and the new set of CDLN, CDLN1, and CDLN2 is created instead of the old ones, as is shown in Figure 9(b).

XN2L (Figure 9(c))--This function is a little complicated. When the extended line of L crosses the current CDLN, unless the extended segment crosses or overlaps any other line or crosses any VIEWLINE on its way to the assumed crossing point of CDLN, new CDLN and CDLN1 are created, and CDLN2 is set to be NIL, as is shown in Figure 9(c).

COVER (Figure 9(d)--This is also for a very special case. The value of this function becomes T in the reverse case of OVERLAPL, namely, when the CDLN is overlapped by L because the already created line, L, in the current model connecting P to the other, exists. This situation sometimes occurs for EPOINTs such as the part of NGATE judged by CONJECT 1 or the point already chosen in the current model as the opposite part against the crossing point of the CDLN1 when XN2L worked through the process of conjecture on the other EPOINT before. So, as is shown in Figure 7(d), the new set of CDLNs is the shifted one as the point P' is conjectured by XNL*, for convenience of computation.

After the above judgments have been made, the new set of candidate lines, CDLNS, is formed, or it may be the same as the original one. Then, it is checked as to whether or not the CDLN crosses any VIEWLINE in the model. This seems ridiculous, but it must be checked after all the above conjectures or simultaneously, because otherwise, it makes the above conjectures insignificant. Consider P in Figure 10. Perhaps, if EPOINT P1 has not been conjectured yet, P will be connected to the line L only with the above four conjectures.

All through the process, all new elements of CDLNS and their created crossing points, if any, are named by the function GENSYM but at this stage no internal format for them is made.

When any CDLN is negated and we find that the questionable point is not to be conjectured on the current MODEL, the point is put into E**LIST and prepared for the second path of CONJECT 2.

ISSUPOSEW outputs the newly created MODEL after CONJECT 2 is completed.

E. EX2IN*

As for the list MODEL, created by CONJECT 1 and CONJECT 2, only ORT of new lines and values of X - Y coordinates of new points are put in their property lists in internal format. Consequently, EX2IN*, almost the same function as EX2IN, works on MODEL to make properties NPTS, NLNS, and TYPE of points.

EX2IN* results in the new internal format such that the old points have properties XCOR, YCOR, new NPTS, new NLNS, and NVZNS, the new points have properties, XCOR, YCOR, NPTS, and NLNS, all the lines have their ORT, and the point RB remains the same. The property lists of VIEW, LINES, and POINTS are left as they were, although the elements of LINES must have different properties from the old ones. (The author has not yet developed a function to modify LINES.)

The new property of points, NPTS, is different from the old one in a way. EX2IN* has the function called FOOP, which lists the neighbor points of a certain point in the manner of traversing clockwise, as shown in Figure 11. The algorithm is shown in Figure 12.

F. RGNFND

The function RGNFND works on the list of all points of the model to find out the closed region in the model.

Whenever one closed region is found, the name of the region is given by GENSYM and put into the property list of MODEL identified by REGIONS. Each created region has the property list of all lines and the list of all points that define that region listed in the order that we can see the region on the left as we follow the contour. ISUPPOSEW outputs the lists of all regions and its points as shown in the results (see Figure 13 to 17).

The RGNFND program is still incomplete; it has not yet been developed so that it can delete the region outside the border lines and unite two regions created by outer and inner boundaries when one large region holds the small one.

IV RESULTS

The examined data and their results are shown in Figures 13 to 17. Figure 16 shows the whole output of ISUPPOSEW.

Some questionable points of the program should be considered.

A. Necessity of NGATE and Singularity of Solution

Even in the case of Figure 2(d), CONJECT 1 connects both EPOINTs in pairs. Strictly speaking, they may not have to be connected. However, the author gave the program the characteristic that it make as many closed regions as possible. See Figure 18. If CONJECT 1 is defined strictly, the region R2 in Figure 18(a) and the region R6 in Figure 18(b) may be left as open regions, for the lines L and L$'$ are not created in some circumstances; whether they are created depends on the sequence of EPOINTs that CONJECT 1 is given. Case (a) may be thought to be natural without R2, whereas in case (b) the preference is to close the region R6. The present CONJECT 1 closes both regions, unfortunately, and deletes the trouble of singularity of the result of the kind caused by the proposed sequence of EPOINTs.

If we examine the result of DATA 3, we see that the regions 98 and 99 were created, but the situation is very similar to the case in Figure 18(a). That is, these regions have no evidence such as VIEWZONES inside them. Regions 98 and 99 are also created on account of the sequence of EPOINTs given to CONJECT 2. That type of difference of solution is not excluded from the program. ISUPPOSEW cannot guarantee the singularity of solutions of this kind in the present stage. The program that defines the relationship between regions must be added to. Then inconvenient regions may be deleted from the results or may be merged by the other (as they are most likely to be), but it must be another problem concerned with the global conjecture.

B. The Problem of Overlap of Lines

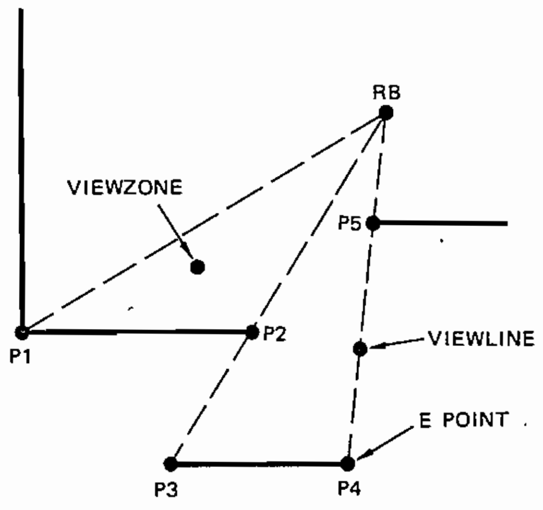
The algorithm of CONJECT 2 has the problem of overlap of created lines. ISUPPOSEW always replaces the old line in the current model by the newest line. It does conjecture procedures evenly on all the EPOINTs listed in E*LIST at first, without deleting any that happen to be considered to be connected as the result of the conjecture of other points. The case (b) in Figure 18 has three points, P1, P2, and P3, to be dealt with by CONJECT 2. Assume the sequence given in (. . . . P1. . . P2. . . P3 . . .). When CONJECT 2 works on P1, the set of lines such that the extended line of P1 crosses L' is created in the model, CONJECT 2 is applied to P2, and the result becomes the same. ISUPPOSEW replaces the old set in the model with the new result. This is the way that ISUPPOSEW avoids the overlap problem, but it drives the program to meaningless calculations.

C. Questionable Conjecture, Impossible Conjecture, and Necessity of CONJECT 3

The basic concept of ISUPPOSEW includes the idea that an EPOINT like P9 in DATA 1 seldom exists, which has no opposite EPOINT to which it can be connected, in the building of the average kind. This gives the basic reason to case (a) in Figure 19 that the line must be extended in the direction shown by an arrow, for if that point is one part of the doorway, it must be connected to the opposite one by CONJECT 1. The problem occurs when such opposites cannot be found because of obstacles. Such a case occurs in rather complicated data, as is shown in Figure 19(b). Both points PK and PL must cross NVZNS to make closed regions, because the opposite EPOINTs of them may be somewhere behind LO. This is one point at which the author fears that CONJECT 3 is necessary.

We can consider more impossible conjectures. P3 in DATA 1 is one of them. ISUPPOSEW has treated only EPOINTs. Consequently, although they are in almost the same situation, conjecture on P5 in DATA 1 is different from other points such as P11 and P10 in DATA 1.

Suppose the line LO in Figure 19(b) is connected to the other by CONJECT 1. There is left no possibility that the conjectured line L is drawn. Then point PM has no chance to be connected to any other point. Furthermore, who can guarantee that line L is right? The best answer may be that the line L connects to LO on the central point of LO. Now, it is impossible for ISUPPOSEW to make the above conjectures, and the author thinks it is the limit of ISUPPOSEW and that of human beings at the same time so long as we consider only the plan model of a visual scene. God only knows.



TA-710531-7

FIGURE 1 PART OF THE PLAN MODEL OF VISUAL SCENES

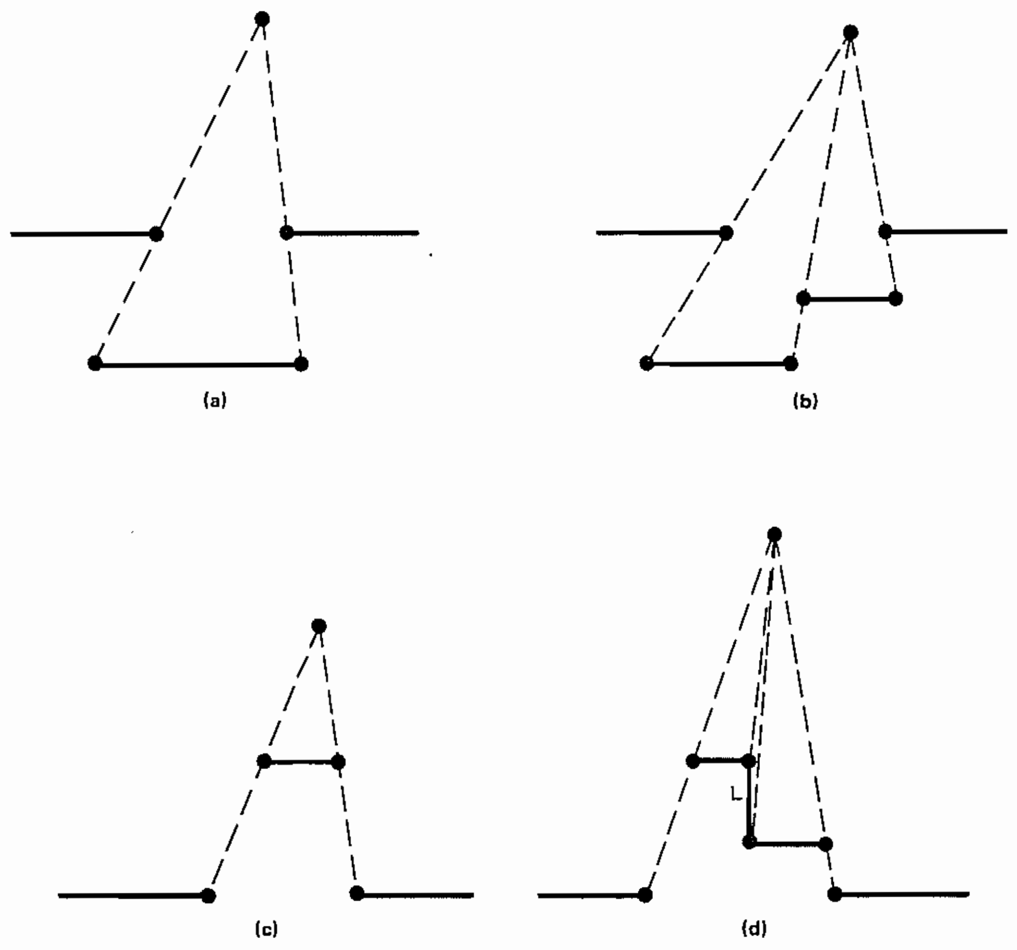


FIGURE 2 CASES FOR APPLICATION OF RULE 4(a)

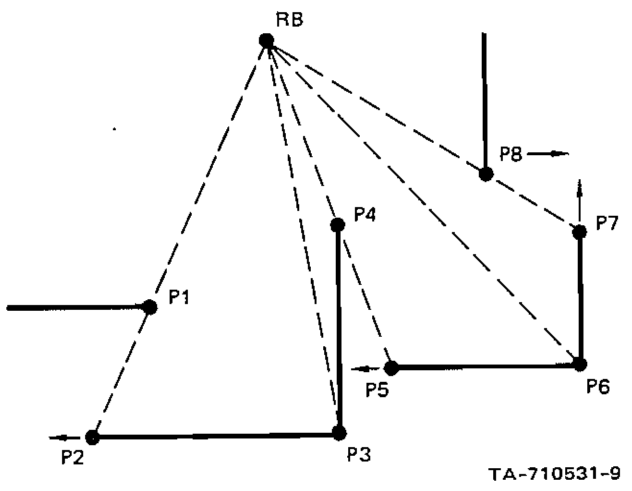


FIGURE 3 CASE FOR APPLICATION OF RULE 4(b)

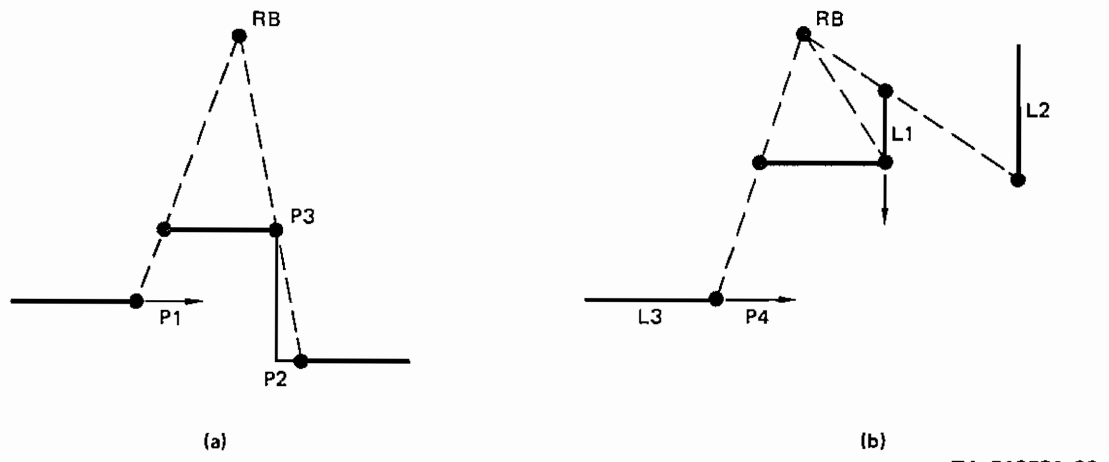


FIGURE 4 DIAGRAM ILLUSTRATING RULES 5 AND 6

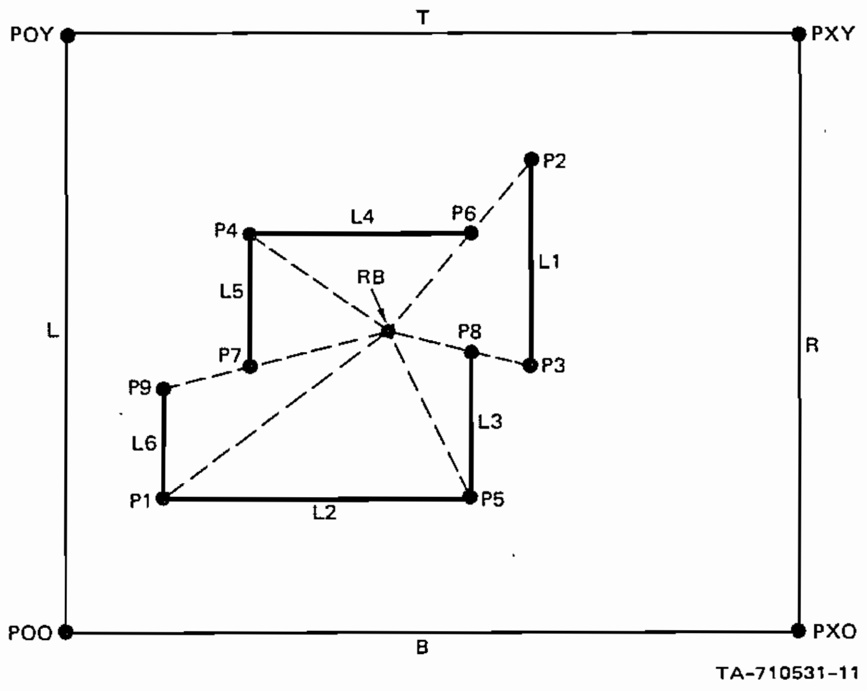
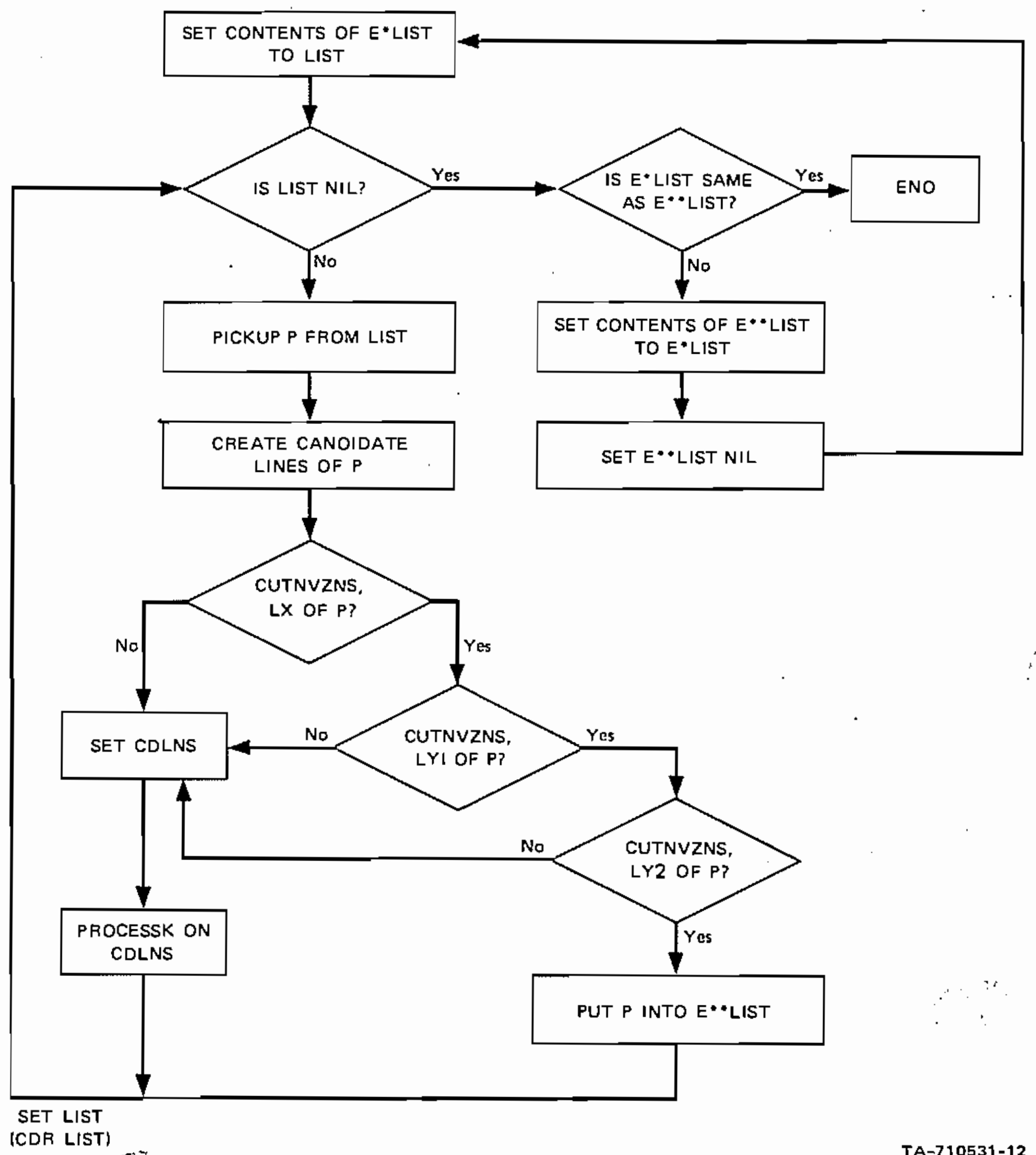


FIGURE 5 EXAMPLE INPUT DATA



TA-710531-12

FIGURE 6 ALGORITHM OF CONJECT 2

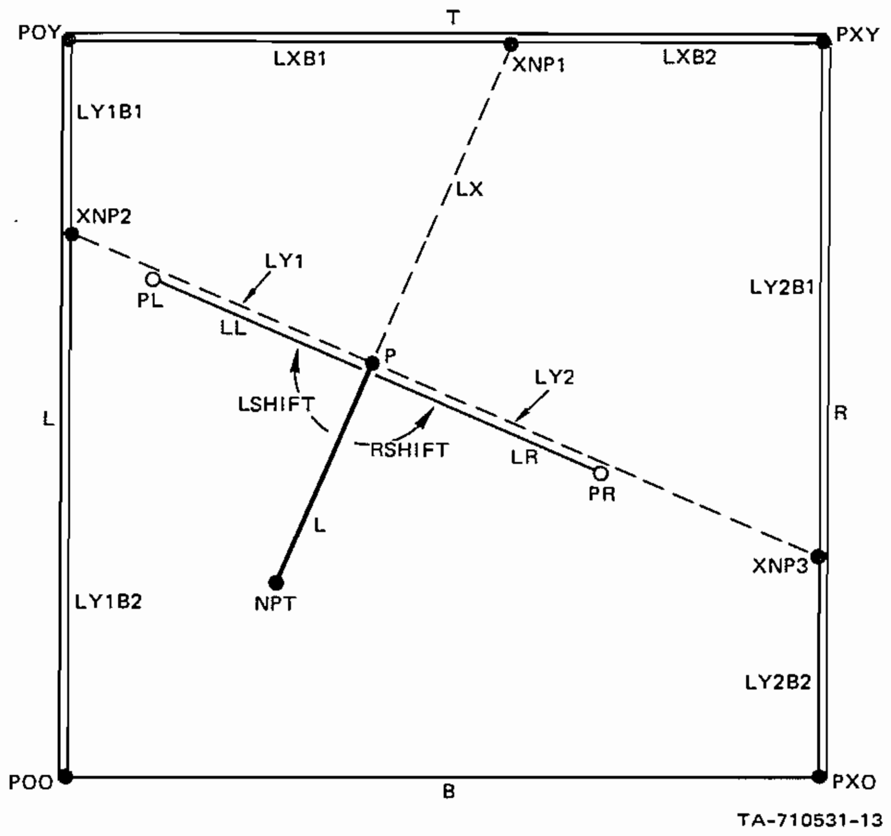


FIGURE 7 EXPLANATORY DIAGRAM FOR CONJECT 2

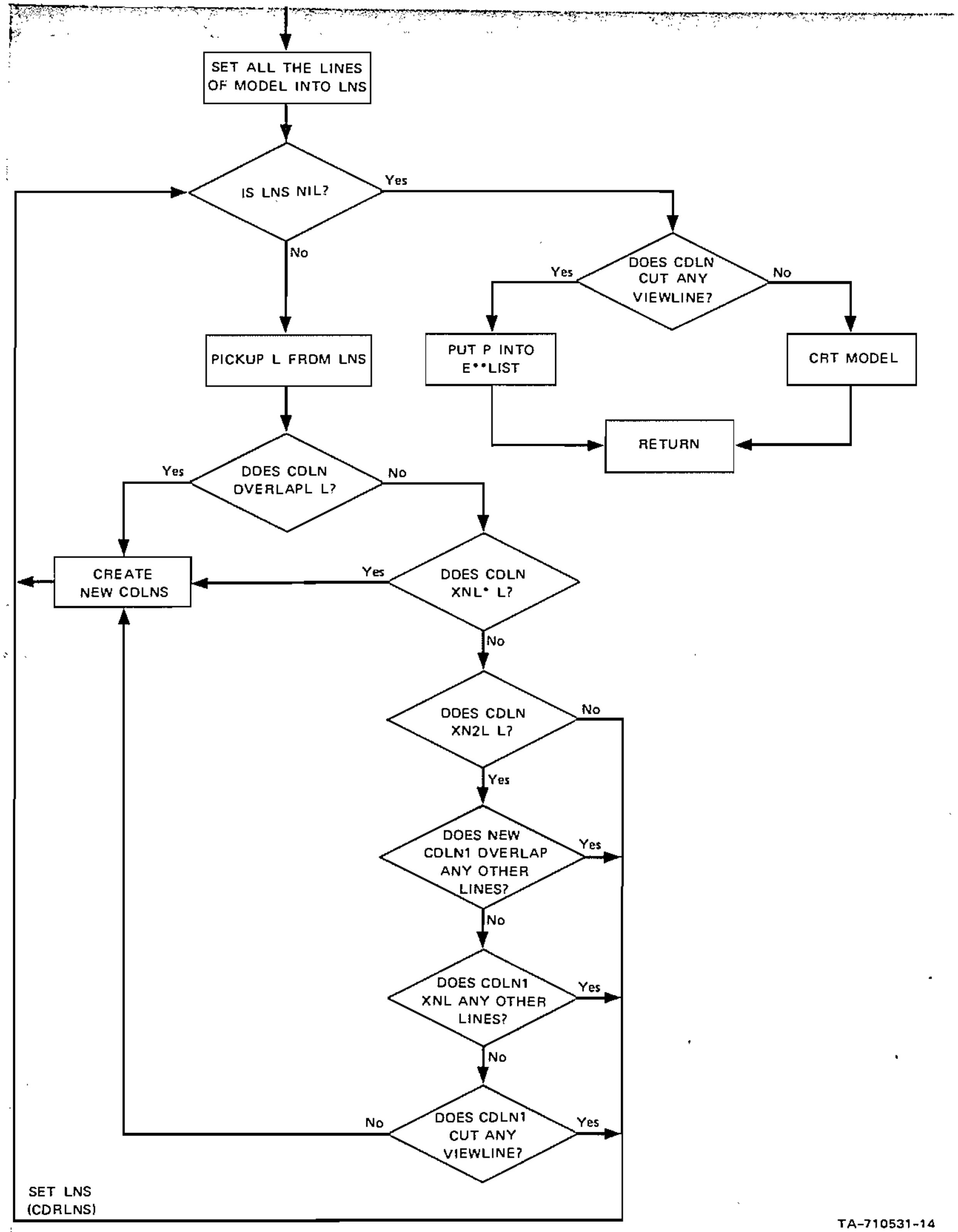
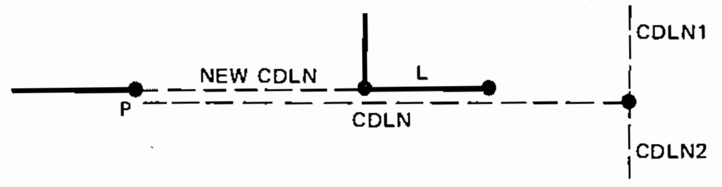
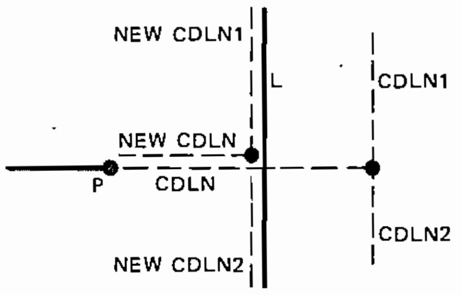


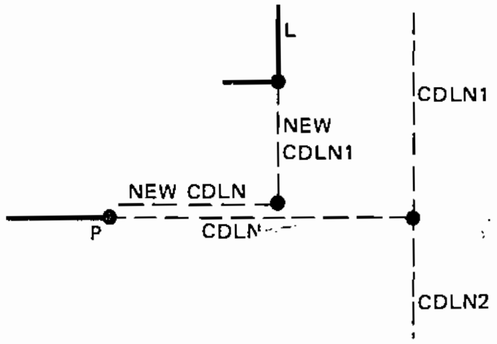
FIGURE 8 ALGORITHM OF PROCESSK



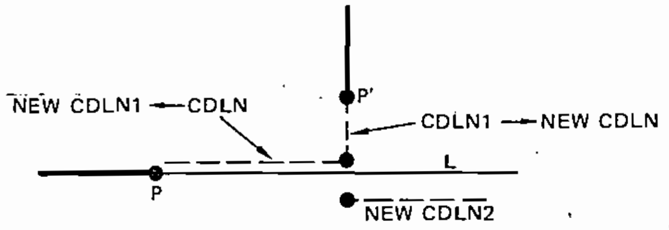
(a) OVERLAPL



(b) XNL*



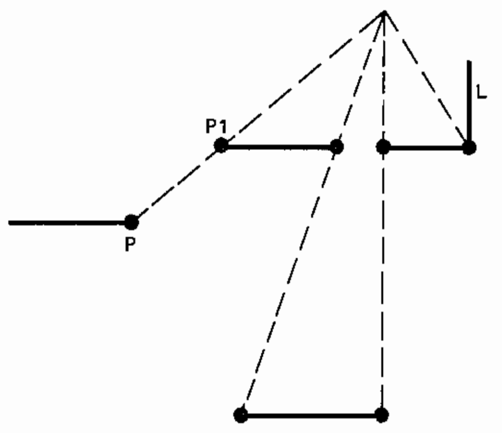
(c) XN2L



TA-710531-15

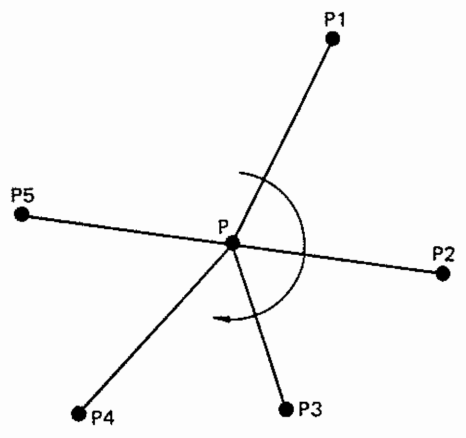
(d) COVER

FIGURE 9 FUNCTIONS FOR JUDGING THE RELATIONSHIP BETWEEN A LINE AND CDLNs



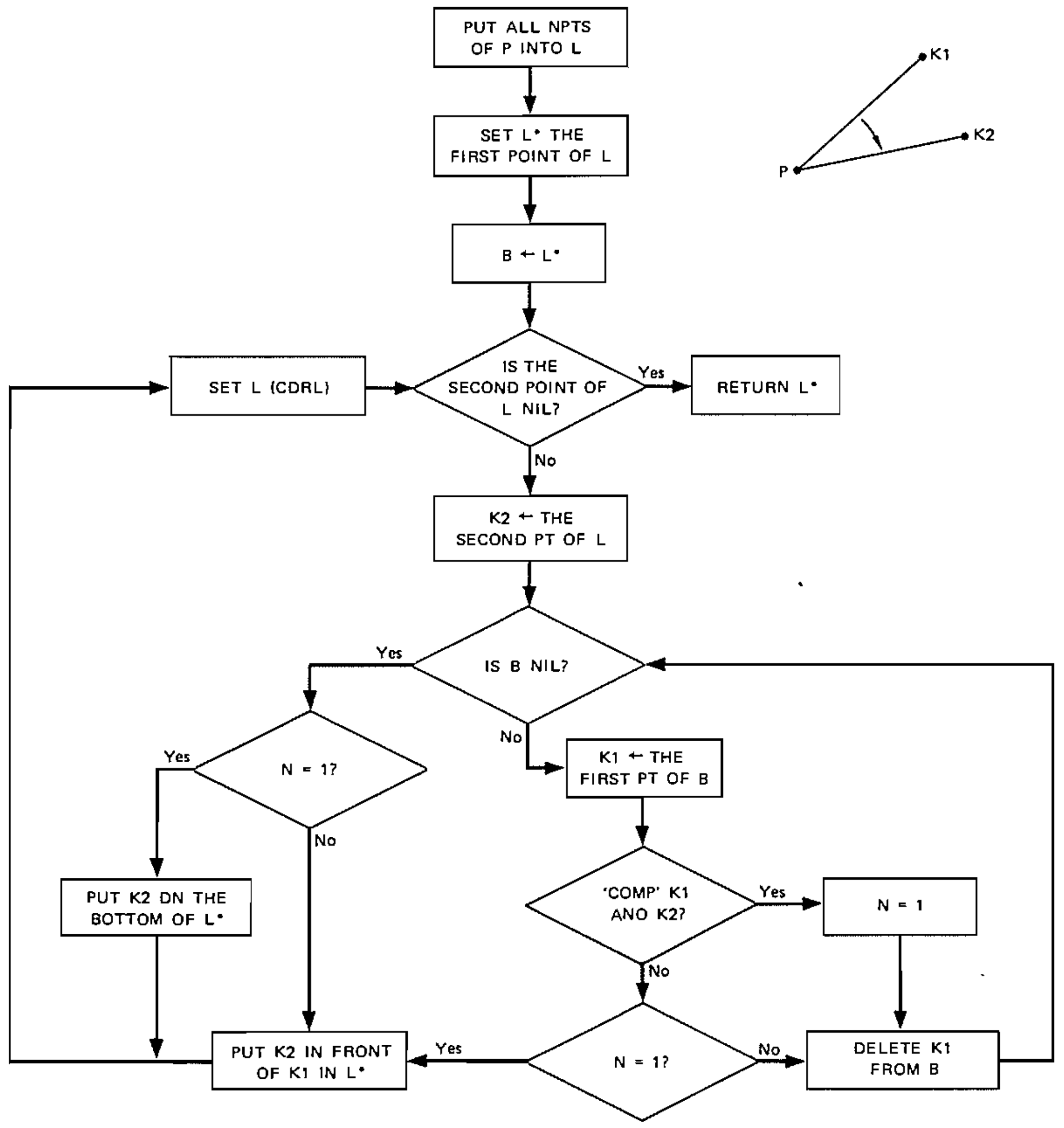
TA-710531-16

FIGURE 10 DIAGRAM ILLUSTRATING THE NECESSITY OF CHECKING WHETHER CDLN CROSSES ANY VIEWLINE



TA-710531-17

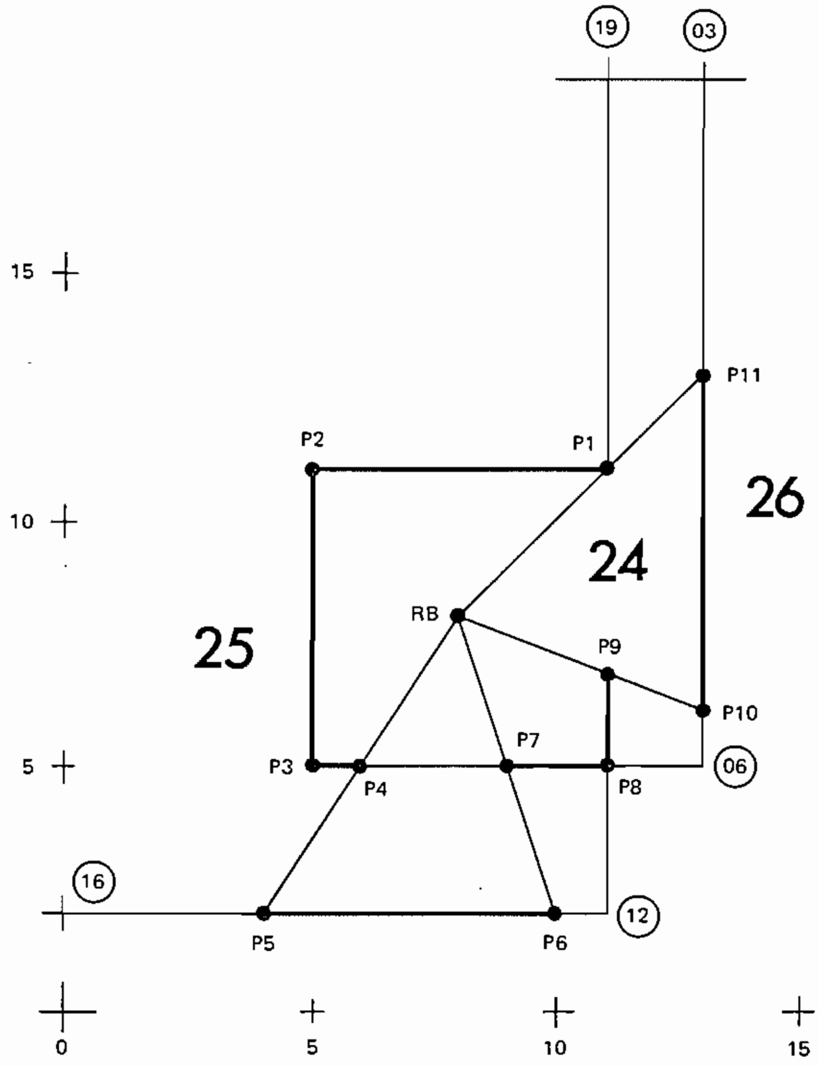
FIGURE 11 EXPLANATORY DIAGRAM FOR FUNCTION FOOP



NOTE: 'COMP' is T when angle $<$K1PK2 is less than or equal to be π with regard to the direction shown by an arrow.

TA-710531-18

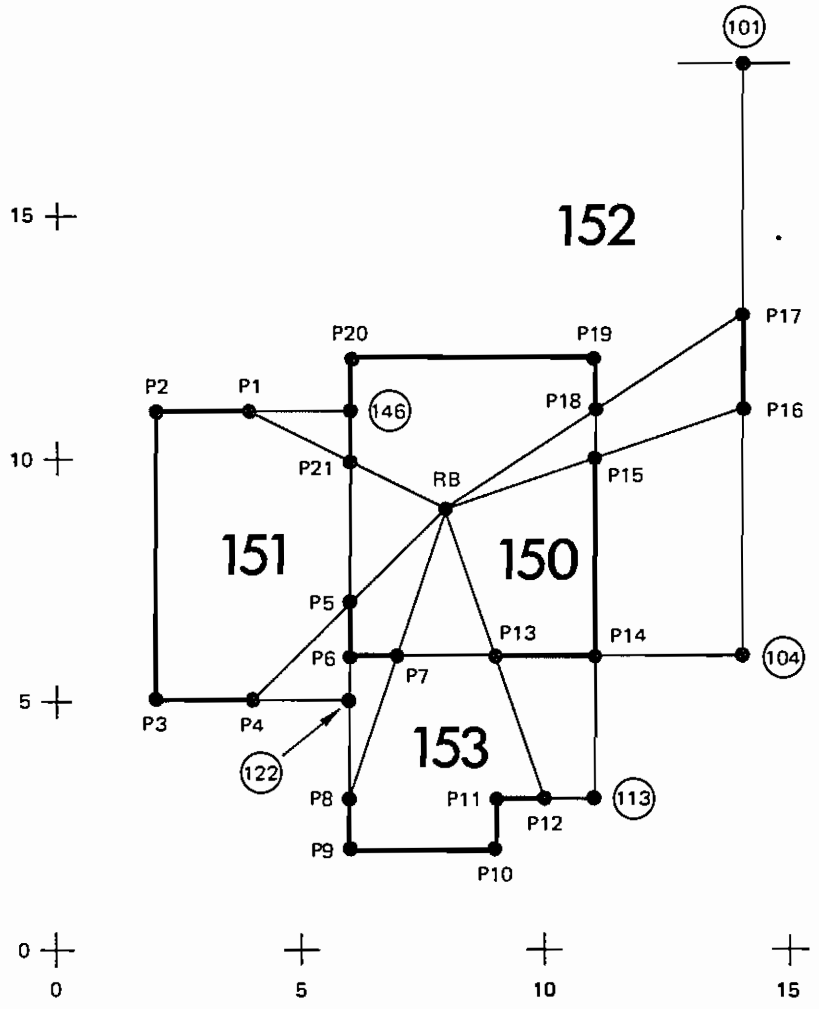
FIGURE 12 ALGORITHM OF FOOP



```
RESULTS:  [G0026 (G0016 P00 PXO PXY G0003 P11 P10
                G0006 P8 G0012 P6 P5))
           (G0025 (G0019 POY G0016 P5 P6 G0012 P8
                P7 P4 P3 P2 P1))
           (G0024 (G0019 P1 P2 P3 P4 P7 P8 P9 P8
                G0006 P10 P11 G0003))
           (G0023 (G0019 G0003 PXY PXD P00 G0016
                POY))]
```

TA-710531-19

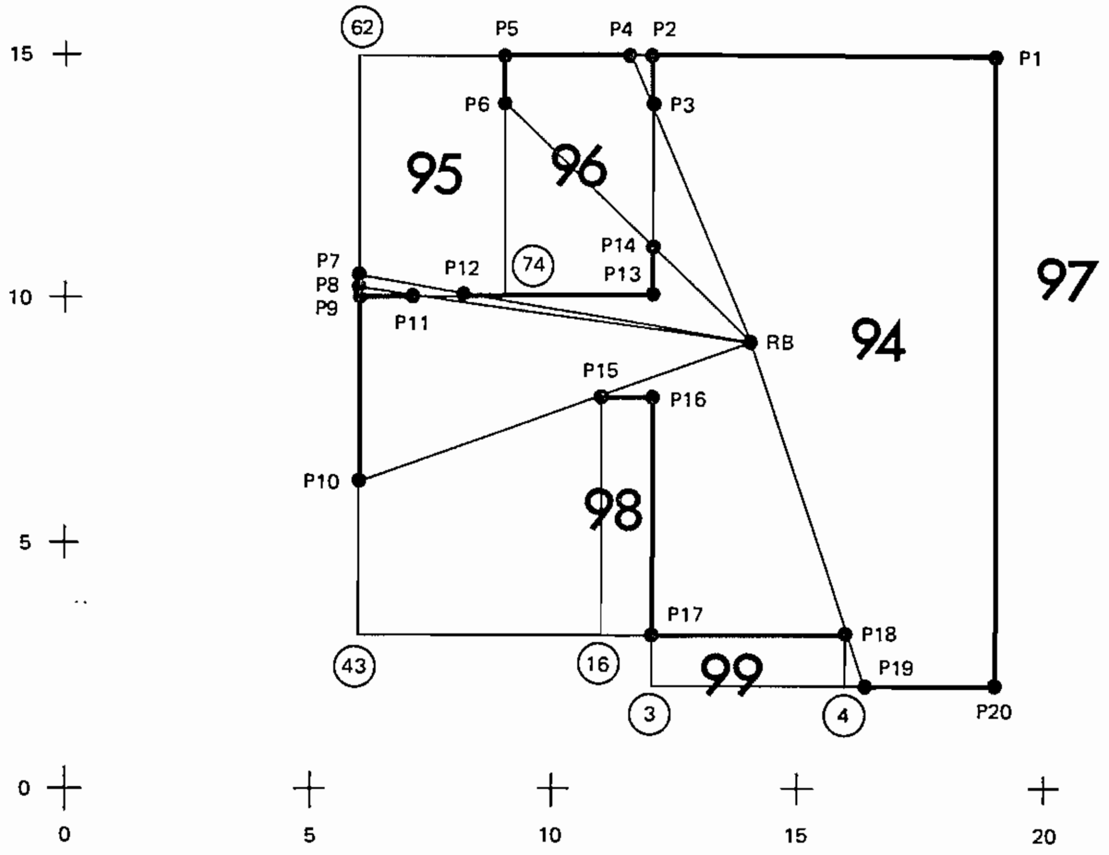
FIGURE 13 DATA 1 RESULTS



```
RESULTS:  [(G0154 (G0101 PXY PXO P00 POY))
           (G0153 (G0122 P8 P9 P10 P11 P12 G0113 P14
               P13 P7 P6))
           (G0152 (G0146 P20 P19 P18 P15 P14 G0104 P16
               P17 G0101 POY P00 PXO PXY G0101 P17
               P16 G0104 P14 G0113 P12 P11 P10 P9
               P8 G0122 P4 P3 P2 P1))
           (G0151 (G0146 P1 P2 P3 P4 G0122 P6 P5 P21))
           (G0150 (G0146 P21 P5 P6 P7 P13 P14 P15 P18
               P19 P20))]
```

TA-710531-20

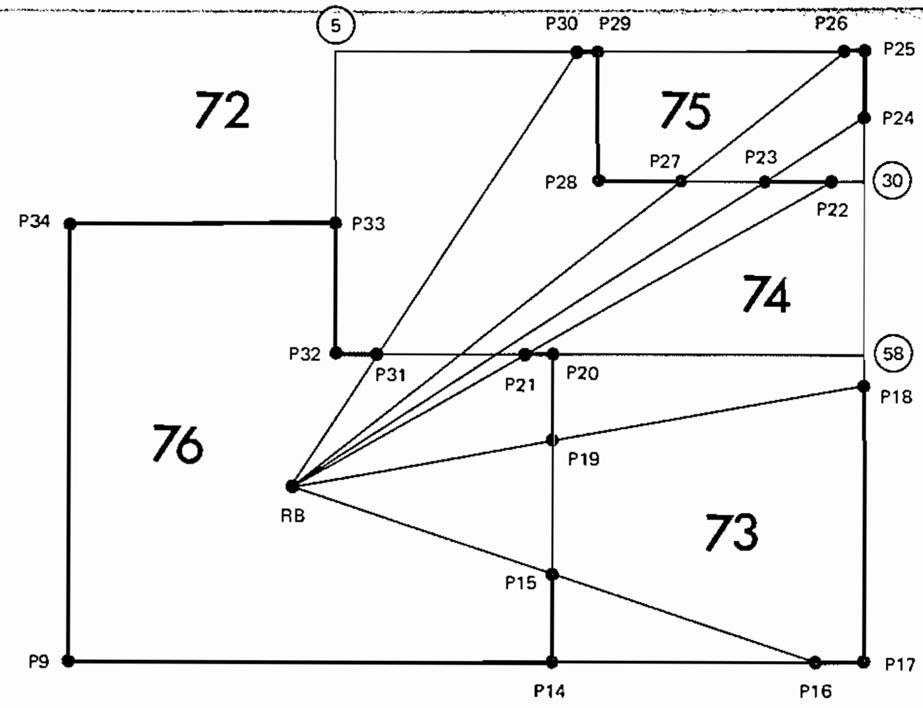
FIGURE 14   DATA 2 RESULTS



```
RESULTS:  (MAPCAR (FUNCTION RGNLIST) (RGNFND (GET VIEW @POINTS$
          ((G0101 (POY POO PXO PXY)) (G0100 (POY PXY PXO POO))
          (G0099 (G0009 P1 8 P17 G0003)) (G0098 (G0016 P17 P16 P15))
          (G0097 (G0062 P5 P4 P2 P1 P 20 P19 G0009 G0003 P17 G0016 G0043
          P10 P9 P8 P7)) (G0096 (G0074 P13 P14 P3 P2 P4 P5 P6)) (G0095
          (G0074 P6 P5 G0062 P7 P8 P9 P11 P12)) (G0094 (G0074 P12 P11 P9
          P10 G0043 G0016 P15 P16 P17 P18 G0009 P19 P20 P1 P2 P3 P14 P13)))
```

TA-710531-21

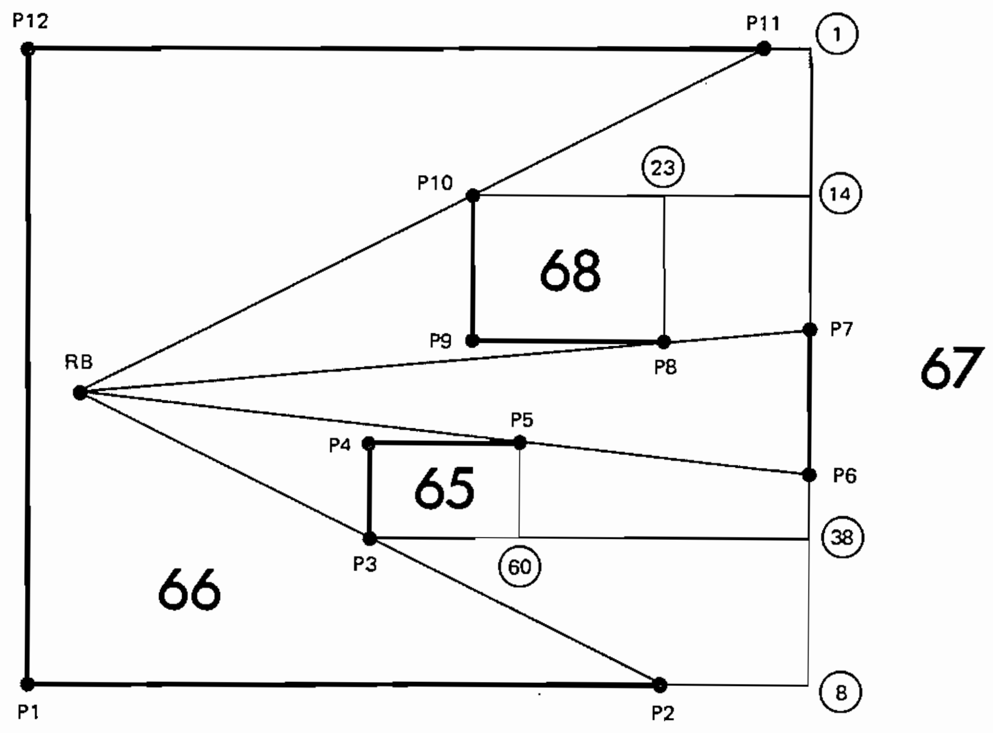
FIGURE 15   DATA 3 RESULTS



```
RESULTS:  *(DSKIN DATA4S
          ***
          *(I SUPPOSEW DATA4S
          EX2INED
          MOOEL
          (B (PXO POO) R (PXY PXO) T (POY PXY) L (POO POY) L23 (P33 P34)
          L22 (P32 P33) L21 (P31 P32) L20 (P29 P30) L19 (P28 P29) L18 (P27
          P28) L17 (P25 P26) L16 (P24 P25) L15 (P22 P23) L14 (P20 P21) L13
          (P19 P20) L12 (P17 P18) L11 (P16 P17) L10 (P14 P15) L2 (P9 P14)
          L1 (P34 P9))
          CONJECT1
          (B (PXO POO) R (PXY PXO) T (POY PXY) L (POO POY) L23 (P33 P34)
          L22 (P32 P33) L21 (P31 P32) L20 (P29 P30) L19 (P28 P29) L18 (P27 P28)
          L17 (P25 P26) L16 (P24 P25) L15 (P22 P23) L14 (P20 P21) L13 (P19 P20)
          L12 (P17 P18) L11 (P16 P17) L10 (P14 P15) L2 (P9 P14) L1 (P34 P9)
          G0001 (P31 P21) G0002 (P27 P23) G0003 (P24 P18) G0004 (P19 P15)
          CONJECT 2
          (B (PXO POO) R (PXY PXO) T (POY PXY) L (POO POY) L23 (P33 P34)
          L22 (P32 P33) L21 (P31 P32) L20 (P29 P30) L19 (P28 P29) L18 (P27 P28)
          L17 (P25 P26) L16 (P24 P25) L15 (P22 P23) L14 (P20 P21) L13 (P19 P20)
          L12 (P17 P18) L11 (P16 P17) L10 (P14 P15) L2 (P9 P14) L1 (P34 P9)
          G0001 (P31 P21) G0002 (P27 P23) G0004 (P19 P15) G0006 (P30 G0005)
          G0007 (G0005 P33) G0020 (P26 P29) G0031 (P24 G0030) G0040 (P22
          G0030) G0060 (G0058 P20) G0061 (G0058 G0030) G0059 (P18 G0058)
          G0071 (P16 P14))
          EX2INEO*
          REGIONS
          ((G0078 (POY POO PXO PXY)) (G0077 (POY PXY PXO POO)) (G0076
          (P9 P14 P15 P19 P20 P21 P31 P32 P33 P34)) (G0075 (G0030 P24 P25
          P26 P29 P28 P27 P23 P22)) (G0074 (G0058 G0030 P22 P23 P27 P28 P29
          P30 G0005 P33 P32 P31 P21 P20)) (G0073 (G0058 P20 P19 P15 P14 P16
          P17 P18)) (G0072 (G0058 P18 P17 P16 P14 P9 P34 P33 G0005 P30 P29
          P26 P25 P24 G0030)))
```

TA-710531-22

FIGURE 16 DATA 4 RESULTS



```
RESULTS:  [(G0070 (POY P00 PXO PXY))
           (G0069 (POY PXY PXO P00))
           (G0068 (G0023 P10 P9 P8))
           (G0067 (G008 P2 P1 P12 P11 G001 G0014 P7 P6
               G0038))
           (G0066 (G0060 G0038 P6 P7 G0014 G0023 P8 P9
               P10 G0023 G0014 G0001 P11 P12 P1 P2
               G0008 G0038 G0060 P3 P4 P5))
           (G0065 (G0060 P3 P4 P5))]
```

TA-710531-23

FIGURE 17   DATA 5 RESULTS

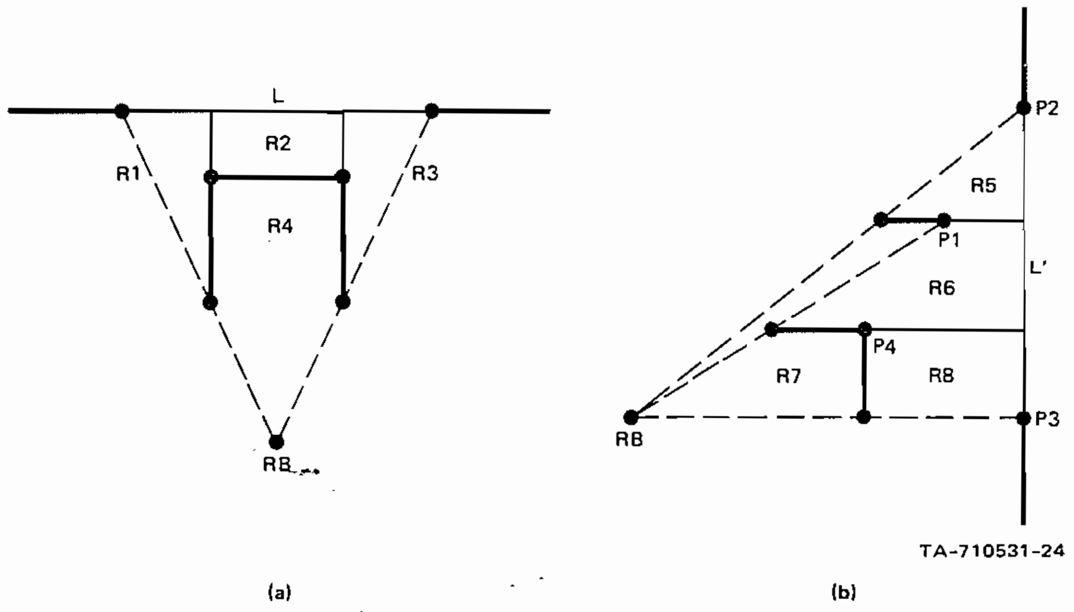


FIGURE 18 EXPLANATORY DIAGRAM FOR CLOSING REGIONS

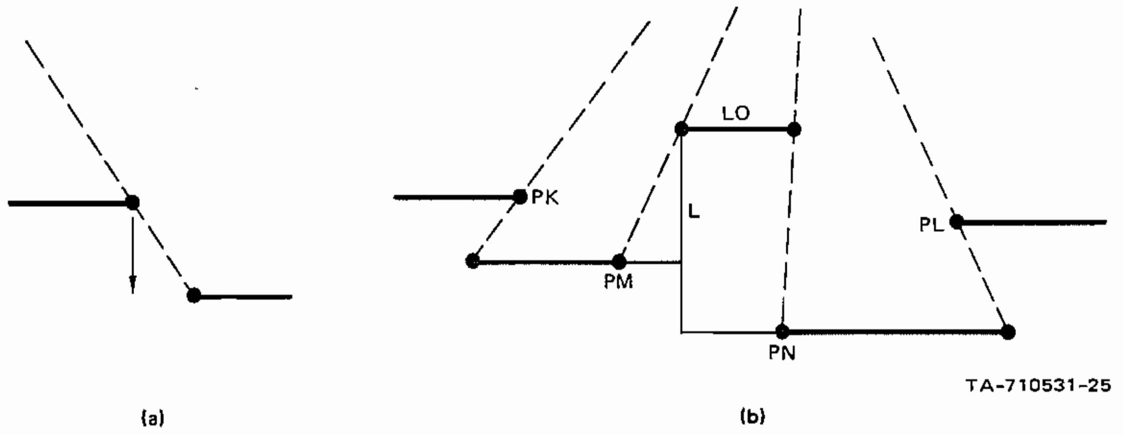


FIGURE 19 DIAGRAM FOR QUESTIONABLE CONJECTURE