COSC 460 Project Report

Department of Computer Science

University of Canterbury

A Diagrammer for the EXSYS Data Model

Supervisor : Dr. R. E. M. Cooper

Peter J. Bromley

October 1985

Table of Contents

1. Introduction 1

2. Background to Problem 2

   2.1. EXSYS 2

   2.2. The EXSYS Data Modeller 4

   2.3. Documentation and Diagrams 4

   2.4. Other Work and Related Areas 6

3. Diagrammer Design 8

   3.1. Overall Design 8

   3.2. Automatic Generation 8

   3.3. Entity Placement

      3.3.1. Overview 10

      3.3.2. Past Techniques 11

      3.3.3. Chosen Method 11

      3.3.4. Improving the Algorithm 13

      3.3.5. Attraction and the Placement Problem 15

   3.4. Relationship Placement

      3.4.1. Introduction 17

      3.4.2. Past Research 18

      3.4.3. Chosen Method 20

   3.5. Aesthetic Requirements and Effects 22

4. Implementation 23

   4.1. User Interface 23

   4.2. Data Structures and Data Collection 24

4.3. Building Trees 25
4.4. Grid Structures 26
4.5. Entity Placement 27
4.6. Relationship Placement 28
4.7. Line Printer Output 30

5. Results and Future Work 32

5.1. Results 32
5.2. Limitations 32
5.3. Future Work 33

6. Conclusions 35

7. References 36

Appendices

Adata User Release on Diagrammer ..... Appendix A
Adata User Manual for Diagrammer ..... Appendix B
changed Entity Placement Complexity Derivation ..... Appendix C
Some Diagrams ..... Appendix D
Listing of Diagrammer Code ..... Appendix E

1. Introduction.

This project involved the design and implementation of a diagrammer for EXSYS, a recent software product for the design and construction of business systems. EXSYS automates the systems analysis process and, through a series of answers supplied by a user, creates a Data Model of the user's business environment. The diagrams created using the proposed diagrammer consist of boxes and lines which represent, respectively, the Entities and Relationships which make up an EXSYS Data Model.

The reasons why a diagrammer is needed for EXSYS, a description of the environment in which the diagrammer will run, and conventions regarding how diagrams are used to represent Data Model information, are covered in Section 2. A formulation of what such a diagrammer should do is primarily dictated by EXSYS' users requirements but some more subtle, and difficult to quantify, expectations arise from the fact that diagrams are meant to be read and understood by humans rather than machines. The theoretical considerations involved in designing the diagrammer, these aesthetic expectations, and their effect upon the design of the diagrammer are discussed in Section 3. A description of the implementation of the diagrammer and some details (such as data structures) of the method of implementation are described in Sections 4.

Section 5 discusses the results of the work done on this project and mentions future work needed both to improve on what has been done and to expand the diagrammer to make it a full product. In Section 6, conclusions are drawn as to what has been accomplished in this project, and the usefulness of the project as seen by the wider world (that is, the intended market of EXSYS).

2. Background to Project.

2.1 EXSYS.

EXSYS is a software product being created and marketed worldwide by Adata Software Ltd. of Christchurch to be used in the field of creating complete business systems without the need for the traditional methods of software developement. EXSYS removes the need for writing and testing large system programs altogether and effectively does away with the usual four-step software development process and replaces it with a single step process.

- Design Module
  - Data Modelling
  - Attribute Detail
  - Data Manipulation & Formulae
  - Condition Modelling
  - Procedure Overview
  - Procedure Modelling

- Runtime System
- Query Facility

Fig. 1. Makeup of EXSYS

As is shown in Fig. 1, EXSYS is made up of a Design portion, a runtime system, and a query facility. The Design portion of EXSYS consists of three main sections :

1. The Data Modeller, which is used to build an Entity/Relationship **Data Model** of a business environment (a discussion of the EXSYS Data Modeller and the Entity/Relationship method of Data Modelling is given in Sec 2.2),

2. the Attribute Detail section which is used to define, for each Attribute in the Data Model, its type (monetary, numeric, etc), its storage form in the User database, and its default screen and report formats and prompts/headers, and
3. the modules used to define and construct the business system to be used in the environment modelled using the Data Modeller (the definition of this system is stored in the **Procedure Model**).

The Runtime system uses the information in the Data and Procedure Models to animate the business system in the same manner as an interpreter for some high level language runs a program. However, this analogy is not completely correct because the Runtime system does not interpret single "statements" as an interpreter does but, instead, works with each Procedure as a unit. Also, once the Runtime System has loaded a Procedure's description from the Procedure Model, it will perform the actions given for that Procedure at speeds comparable to compiled programs. Procedure Descriptions are cached by the Runtime System so that the overhead of loading them only occurs the first time the Procedure is used.

The Query facility, provides for ad-hoc requests about information in the User database. These requests, however, are given in English (that is, a manager can simply sit down and type in an English request like "what are the names of all bachelors in the Purchasing Department?" and Query will answer it to the best of its ability). The ability of the Query system to understand words like 'bachelor' must be supplied by defining such words in terms of items in the Data Model (for example, Employee such that Employee Marital Status = "SINGLE" and Employee Sex = "M").

2.2 The EXSYS Data Modeller.

Of particular relevance to this report is a description of the EXSYS Data Modeller and the philosophy behind its construction. The EXSYS Data Modeller is based upon the Entity/Relationship view of Data Modelling, formalised in Information Engineering disciplines (see [15]). The Entity/ Relationship philosphy is the view that a user's world-picture being modelled is made up of things (eg. Customer, Branch, etc) called **Entities**, characteristics of these Entities (eg. Customer Address) called **Attributes**, and relationships between Entities (eg. Customer makes a Sale) called **Relationships**.

The EXSYS Data Modeller has built in the expertise of experienced systems analysts which is used to create a Data Model by asking the user questions about the business environment he or she is modelling. This Data Model correctly matches the user's perception of this environment and is easily changed if the user is not satisfied. The Data Model is also fully normalised (to 5th Normal Form) as the Data Modeller detects attempts by the user to define a non-normalised Data Model and helps them to redefine it better. The Data Model created using the Data Modeller is used as the basis for a database for user data (the User Database).

2.3 Documentation and Diagrams.

The Data Model being built by a business must be documented to give information both about the current state of the Model, and also about how the Model has changed through the modelling process. The documentation can be given in two, complementary, fashions. The first method is to list all aspects of the Data Model (Entities, Attributes, and Relationships). This gives a detailed, comprehensive listing of the Data Model but is not helpful in showing the structure of a Model. The second method is to draw a diagram. This method is very useful for displaying the structure of a Model and for displaying relationships in such a manner so that the information specific to a relationship can be quickly assimilated and

understood.

The idea of using diagrams to represent information is not new. Conventions for representing Entity/Relationship Data Models using diagrams are given in Martin & Finklestein [15] and are described below (see Fig 2). To a person acquainted with these conventions, a great deal of information about a particular Data Model can be conveyed by a diagram of that Model.

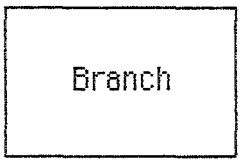


Fig 2a

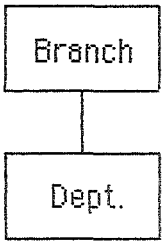


Fig 2b

In reality this connection has been changed (see p. 30)

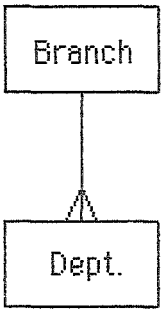


Fig 2c

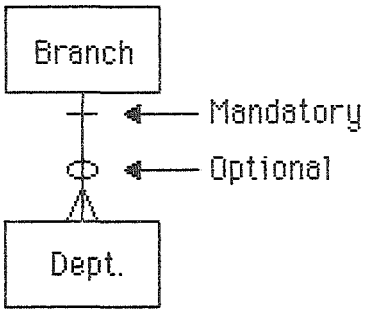


Fig 2d

An Entity is represented by a box with the Entity's name within (Fig 2a). Relationships are represented by lines between the boxes of the two Entities involved in the Relationship (Fig 2b). Information concerning the nature a particular Relationship is drawn at the points where the Relationship lines join onto the Entity boxes using lines, circles, and crows feet.

A Relationship can be defined in two directions (eg, "**Branch** is made up of **Department**", and "**Department** is part of **Branch**"). In each direction, a Relationship has two pieces of information which define its

nature, namely :

One/Many : represented by a crows foot (or the lack of), and

Optional/Mandatory : represented by, respectively, a circle or a line across the Relationship line.

The One/Many nature of a Relationship (in one of its directions) defines whether an instance of one of the Entities involved in the Relationship maps onto "one at most" or "many" instances of the other. This is best illustrated using the example above (and Fig 2c) :

"A **Branch** is made up of many **Departments**", and

"A **Department** is part of a **Branch**".

The Optional/ Mandatory nature defines whether an instance of one of the Entities involved in the Relationship must map onto "at least one" instance of the other or not. This is illustrated using the same example (and Fig 2d) :

"A **Branch** is made up of (possibly) many **Departments**", and

"A **Department** is part of (neccessarily) one **Branch**".

Other conventions exist that define how special Relationships such as Mutually Exclusive or Mutually Contingent Relationship pairs are diagrammed. However, these are not required to be discussed in this report since the diagrammer does not at present deal with diagramming special Relationship information.

2.4 Other Work and Related Areas.

The use of diagrams to represent Data Models is standard practice with systems analysts. However, an aid to generating such diagrams, be it interactive or automatic, is an area that has been neglected. Of course, the interactive construction of diagrams is similar in concept to many other interactive applications such as logic simulators [1], [2].

Automatic construction of diagrams has, on the surface, many similarities with some aspects of CAD (eg. printed circuit board design,

chip layout). When studied more deeply, however, it becomes apparent that the automatic construction of diagrams involves not only the problems recognised and dealt with by these fields but is also affected by the major considerations of the readability of diagrams, how informative they are, and their aesthetic value. These considerations have a large influence on the methods used to solve the obvious problems dealt with in those other fields (see Sec. 3.5), but, due to the lack of work in the area of the generations of data model diagrams, no research can be found which identifies and deals with them.

3. Diagrammer Design.

3.1 Overall Design.

The overall design of the functions of this diagrammer was largely determined by the expectations of its intended users. Because of the documentation purposes of diagrams, the users require that diagrams are able to be generated quickly (either interactively or automatically) and then displayed or printed on some output device like a terminal or a printer. Also, the user requires that diagrams are able to be stored for periods of time, and later retrieved, so that modifications can be performed on them, or comparisons made to assess the effect of changes to the Data Model. Thus the user views the ideal diagrammer as in Fig. 3.

Load / Save / Delete Diagrams

Generate a Diagram

- Automatic
- Interactive

Modify a Diagram

Output a Diagram

- Line Printer
- Plotter
- Graphics Terminal

Fig. 3. Functions of the Ideal Diagrammer

3.2 Automatic Generation.

Given the user expectations of the features that a diagrammer should provide, the next step is to decide the best methods for an implementation of such a diagrammer to use in fulfilling them. The primary function that a diagrammer should provide is the facility to generate a diagram. Additional facilities such as storage, retrieval, and output of diagrams, although neccessary for a complete product, do not involve any serious

theoretical problems. Similarly, the development of a graphics system for interactive generation and modification of diagrams belongs to an area that has been dealt with by many researchers [1], [2] and, beyond the specification of user commands and actions, is also a task with no real problems to be solved.

The automatic generation of diagrams, on the other hand, was recognised soon after beginning work on this project to be a problem of considerable size and scope. It has many similarities with such aspects of CAD as Printed Circuit Board design, Integrated Chip layout, and Interior Design. Two problems are generally recognised by researchers in these fields :

1. The determining of positions of objects to minimise some distance function on the lengths of connections between them (eg. circuit part, transistor, and room positions), and
2. The construction of the connections between these objects so as to minimise such things as vias, corners, and crossings.

In the automatic generation of diagrams, the corresponding tasks are the positioning of the Entity boxes, and the construction of the Relationship lines between those boxes.

Since diagrams are intended for human consumption, the automatic generation of diagrams also involves problems in attempting to duplicate the mental processes involved when, say, a draughtsman draws up a diagram of some Data Model. These mental processes must be identified and duplicated in order to generate diagrams that not only "look neat" and "look right", but also in which the information about the structure of a Data Model is most readily conveyed. Section 3.5 attempts to define these problems, and discusses the effect of the methods used to solve them upon the methods chosen to place a Data Model's Entities and Relationships on a diagram.

3.3 Entity Placement.

3.3.1 Overview.

The first problem in the automatic generation of a diagram is that of determining the positions of Entity boxes on the diagram. We require that the positions of Entity boxes are chosen so that, because of the need to later place the Relationship lines between Entities, the distance between Entities is minimised. This is an application of a problem recognised in many fields (such as management science, engineering, and combinatorics) which is known as the **Quadratic Assignment Problem** (QAP). Various restrictions on this problem give rise to such related problems as the Component Placement Problem [9], the Backboard Wiring Problem [22], and even the Travelling Salesman Problem [9]. The QAP is also researched under the name of the Placement Problem [24].

The Quadratic Assignment Problem is usually stated as below:

Given : a set of locations $L = \{ x_1, \ldots, x_n \}$,

a set of elements $S = \{ s_1, \ldots, s_n \}$ to be placed,

a connection matrix $C = [c_{ij}]$ to represent

connections between elements of S,

find : a permutation of location assignments $P = \{ p_1, \ldots, p_n \}$

(p_i is the location of element s_i)

so that

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij} * d(P_i, P_j)$$

is minimised (where **d** is some general distance function which computes the "distance" between two locations).

This, in printed circuit board terms, may correspond to finding positions for elements so that the total wire length is minimised.

3.3.2 Past Techniques.

The QAP was first recognised by Koopmans & Beckman [12] in 1957. At first research on the QAP was concentrated on finding exact solutions using branch-and-bound techniques (eg. Gilmore [8], Lawler [13]). It soon became obvious to researchers that, due to the exponential nature of any general procedure to gives exact solutions, progress in this area would be slow except for a class of highly restricted sub-problems (exact procedures become computationally ineffective when the size of the problem (n) > 15). Effort was soon diverted to finding fast, good, sub-optimal heuristics for this problem.

Sub-optimal methods that have been proposed for the QAP are :

(1) Constructive methods, which place one or more elements at a time to build up a complete placement,

(2) Improvement methods, which seek to improve a given placement by exchanging several elements,

(3) Relaxation methods, which use the "forces" on an element due to its connections with other elements to reach a point of minimum tension, and

(4) Inexact Branch and Bound methods, which develop bounds on the cost of assigning an element along with backtracking if the search along a branch of the decision tree being created does not prove to be useful.

An excellent review of the work to 1970 is given in Hanan & Kurtzberg [24] and Burkard & Stratmann [25] review the work to 1978.

3.3.3 Chosen Method.

As stated already, the diagrams generated by the diagrammer are intended to be viewed by humans. A consequence of this is that the arrangement of the entities and relationships on a diagram is required to be such that related entities are close to one another and that relationship lines generally cross as little as possible and are as short and straight as possible. For maximum clearness in a diagram, the latter (Relationship)

aims are more important than the former (Entity). This means that, although a reasonably good method of placing Entities is required, beyond a certain point the degree of difficulty involved in placing difficult Relationships is not reduced by improvements in the optimality of the placement of Entities.

Most of the algorithms developed to provide a solution to the QAP have been developed with the purpose of giving a solution that is faster and gives more optimal results for certain standard test problems than all previous algorithms. The increased optimality of such an algorithm is usually only a small gain and, given the reasons for which a QAP algorithm is required, is of dubious quality compared to the added complexity of a program written to implement it.

With this in mind, the method that was chosen for use within the diagrammer is a modification of the algorithm proposed by Garside & Nicholson [7] in 1968. Their method is essentially an iterative construction algorithm followed by a pairwise exchange improvement algorithm. It was intended to implement the iterative construction algorithm and to add the pair-wise exchange algorithm later if it proved to be a worthwhile addition. However, the time complexity of the pairwise exchange algorithm proved to be a slow $O(E^3)$ per iteration with an unknown number of iterations which was decided to be too costly for the very small improvement provided. (E and R are used in this report to stand for, respectively, the number of Entities and Relationships to be placed in the diagram).

1. Choose the Entity most related to those already placed.
2. Test for placement at each position
   (a) if empty - compute the cost of placing the Entity here,
   (b) else - test the Entity already there at all empty positions and compute cost of the placement plus displacement.
3. Place the Entity at the position which gives the lowest cost.

Fig. 4. Garside & Nicholson's Algorithm

The iterative construction algorithm taken from Garside & Nicholson's paper is given in Fig 4. Taken literally, this algorithm has a time complexity of approximately $O(E^4)$ (cf. Gilmore's N^4 and N^5 algorithms) and, applied to the situation in EXSYS where a very large number of Entities are allowed and must therefore be accounted for in the design of a diagrammer, definitely needs improvement before being used.

3.3.4 Improving the Algorithm.

The iterative construction algorithm given by Garside & Nicholson can be improved in two ways. In any general method for placing Entities, the number of positions for placing Entites in must be at least as many as the possible number of Entites able to be placed. EXSYS allows a maximum of 900 Entities at present so as many possible Entity positions on a diagram must be made available. To check every position would not only be slow but is also intuitively wrong since when a new Entity is placed (which is related to others on the diagram) its final position will neccessarily be near or adjacent to those already placed. This means that an obvious improvement to the algorithm is to limit the attempts at placing an Entity to positions within extremes given by the positions of those Entities already placed.

The second improvement arises from the fact that the original algorithm tests if the displacement of an Entity already placed on the diagram will improve the final diagram. However, the earlier an Entity is placed on the diagram, the less likely that its displacement will be profitable. This is because, for any Entity, the positions of all Entities placed later will be dependent upon its position. A way of determining which Entities are worth attempting to displace (and hence removing wasted effort on those which are not) could improve the algorithm by up to an order of E.

A good method of accomplishing this is to first build an augmented breadth-first spanning tree of the graph that is equivalent to the diagram

and then to restrict displacement attempts to those Entities whose depth in the tree is greater than or equal to the depth of the Entity being placed. This breadth-first tree is constructed by following relationships from the Entity involved in the most Relationships (the root) to construct each layer of the tree. This means that Entities at the second level of the tree, say, are a minimum of two Relationships distant from the root Entity. The Entities at each level of the tree are connected to those in the above (father) and below (son) levels and to those on the same (brother) level. This covers all relationships between Entities in the tree. An example of a Data Model and the corresponding tree is given in Fig 5.

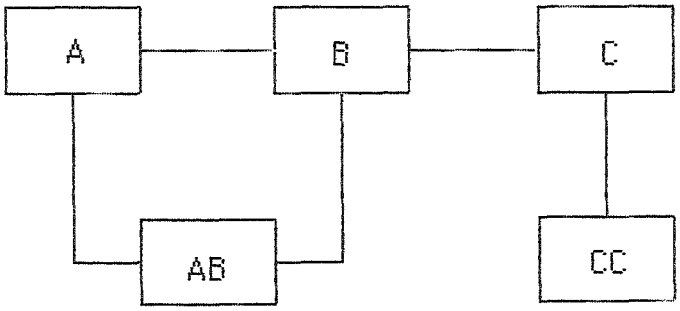


Fig 5a. A Data Model

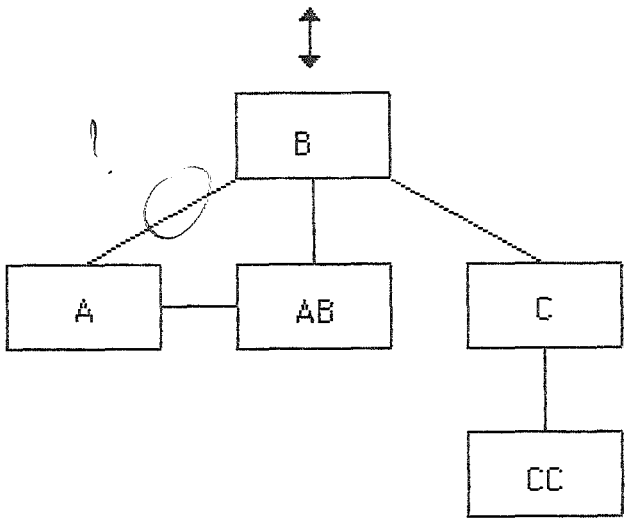


Fig. 5b The equivalent tree

This method of constructing a breadth-first tree is quite effective at reducing the attempts at displacing Entities because there is a rough

correspondence between the occurence of Entities in the tree and the order in which they are placed on the diagram.

The final algorithm used for Entity placement is given below in Fig 6.

1. (Preprocess). Determine the number of Relationships each Entity is involved in.
2. (Tree build). Build a breadth-first tree (with cross links) of the Data Model by the method discussed above. Each Entity has a depth associated with it.
3. (Placement). Repeat E times.
   (1) Choose the Entity most related to those already placed.
   (2) For each position in the area given by the minimum and maximum x and y positions of those already placed.
       If empty - compute the cost of placing the Entity there.
       Else if the Entity there is of a higher depth than that being placed - try next position.
       Else - for each position as before with nothing placed in it - compute the cost of moving the Entity to this position and add it to the cost of placing the first Entity in the second Entity's old position.

Fig. 6 Final algorithm for Entity placement

3.3.5 Attraction and the Placement Problem.

The time complexity of the improved algorithm used by the diagrammer to place Entities is $O(E^{5/2})$ (see Appendix C for a derivation of this estimate). This results in slow execution when E is large (upward of 50 Entities) and a better method would be appreciated.

A new approach to the Placement Problem (an alternative formulation of the QAP), based on Force equations derived from Classical Mechanics is proposed by Quinn [19]. This method was found recently while researching

for good routing methods for the problem of placing Relationships. It is similar to relaxation methods [6] given for the QAP but the notion of virtual area is new. This method, given some reasonable initial configuration, uses iteration to derive a stable final configuration with changes in element positions calculated using force equations based on attraction between connected elements and repulsion between "overlapping" elements. Damping is applied so that an equilibrium position will be achieved.

The attractive forces are based on the idea of a spring between element that are connected and increase linearly with the distance between connected elements. The repulsive force can be based on either repulsion between all elements according to the inverse square law or can be based on repulsion between elements whose virtual areas overlap. An element's virtual area can be compared to personal space, is proportional to the number of connections from the element, is always larger that the actual size of the element. With this virtual space repulsion the size of the repulsive force is proportional to the overlapping area.

Some method of determining that, by and large, most elements' virtual spaces do not overlap in the equilibrium position is needed to complete Quinn's method (this can be achieved by slowly increasing the strength of the repulsive force due to overlaps). A demonstration of the forces caused by connections and overlaps and the result of those forces is given in Fig 7.

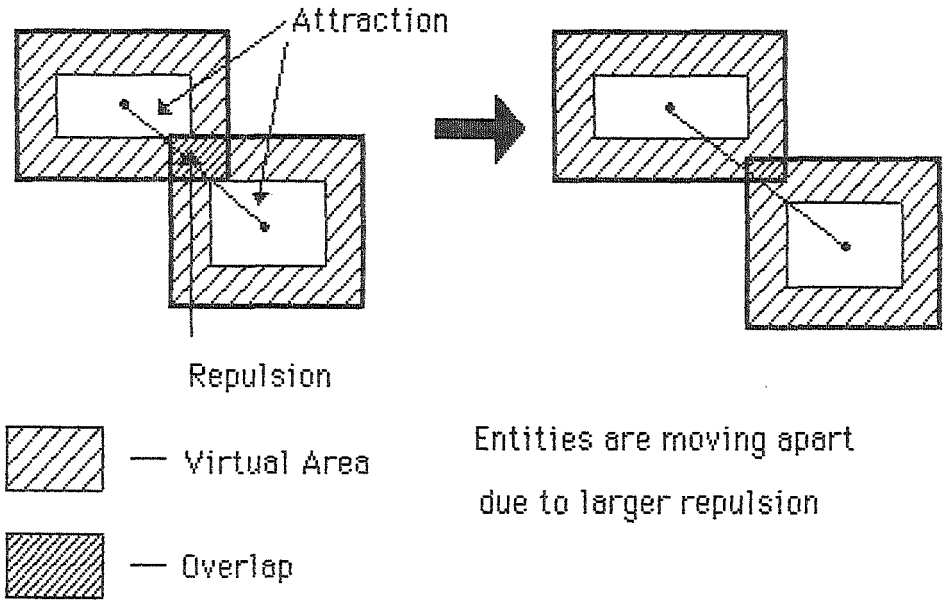


Fig. 7 Forces and their Effect on Entity Positions

This method appears very promising due to its order estimate ($O(E^2)$ per iteration with few iterations needed) and to the fact that the virtual area concept (allowing more room around the more highly related Entities) will allow room for situations which using the present method of inflexible grid positions would become very congested with Relationship lines. Unhappily, due to time and priority limitations, it was not possible to implement this method to test its usefulness.

3.4 Relationship Placement.

3.4.1 Introduction.

The second of the two problems that make up automatic generation of diagrams is the problem of constructing the line segments that make up Relationship lines connecting the Entity boxes. This problem is one that is well known in CAD in such applications as PCB design, routing of runs in VLSI chip design, and maze running. This problem has a sub-problem (also recognised in the design of computer boards incorporating many chips) which involves deciding at which position the ends of the Relationship lines join onto the sides of the Entity boxes. This problem is called the Pin

Assignment Problem by Mory-Rauch [16] who applies it to the field of computer printed circuit board layout. However, it seems that this problem has more relevance to the field of diagram construction than in the field she applies it to since the position and function of pins on chips are usually predetermined by the manufacturers.

3.4.2. Past Research.

Much research has been performed in the field of routing lines : first in the area of maze runners and later for the purpose of automating the layout of integrated circuits. This research all originates from the algorithm proposed by Lee [14] in 1961. This and other early algorithms using the Lee technique and/or wavefront strategies [4], [21] were slow and consumed large amounts of storage. The next major advance came from Hightower's [11] escape line algorithm. This, using backtracking when a dead end is reached, calculates the lines perpendicular to and crossing a particular line segment to work out the line segment to be the next part of the line being routed. Once a line has been routed, a series of improvement are performed to isolate the corners of the line and then to remove overshoots.

The eventual descendant of Hightower's escape line algorithm is the line expansion algorithm developed by Heynes, Sansen & Beke [10] in 1980. This algorithm is based on the expansion of a line segment in the direction perpendicular to it to give an expansion zone (see Fig 8). This expansion zone contains all the points that can be reached by following a line perpendicular to the line segment being expanded and its edges will be either existing line segments or new "active lines" to be expanded further. Expansion is carried out from each of the two endpoints of the line to be routed and stops when the expansion zone of one of the endpoints meets the expansion zone of the other. The line segments of the routed line can then be constructed by tracing back through successive expansions from the meeting point (or "solution area"). The expansion of lines can be

carried out on one or two layer boards and will find a solution (if one exists) in which the number of corners in the line are minimised.

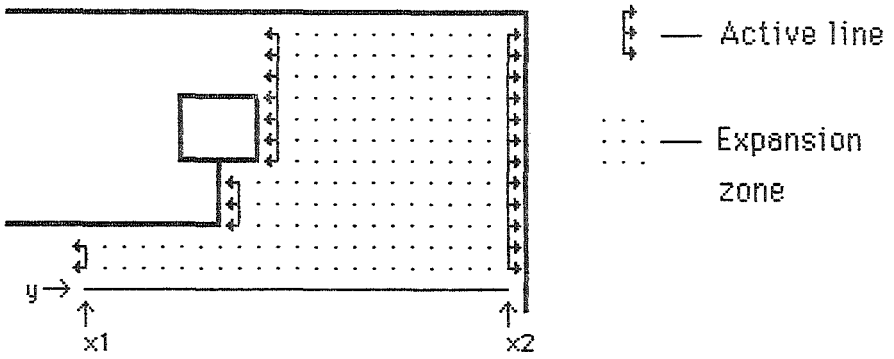


Fig 8. Applying Heyns' Algorithm

Heynes' algorithm needs some modification before it can be applied to the area of placing Relationship lines for the automatic generation of diagrams. This is because when working on one layer the algorithm allows no crossings whatsoever and when working on two layers crossings are allowed since the crossing lines will be on different layers. The ideal situation envisaged for placing Relationship lines would allow crossings if their existence were justified by an appropriate reduction in corners and distance.

This line expansion method was under study for implementation at the termination of this project.

Other new methods that are being developed for use in the area of routing and maze running are multi-contouring algorithms and channel and river routers [18], [23]. The mutli-contouring algorithm of Dobes [3] places each line with no regard for obstructions and then 'stretches' the line to avoid them. The channel and river routers, which seem to be most useful for the placing of wires on computer PCB's, deal with wires to be routed by deciding which "channels" (or gaps between chips) the wire may be laid along and by the use of quick 'rip up and relay' algorithms finds solutions that cause no crossings.

3.4.3. Chosen Method.

The main intent of an automatic line router such as that proposed for this project is to successfully route all lines while minimising such things as vias, crossings and corners. For the purposes of the diagrammer, the lines representing Relationships are required to be constructed so that the resulting diagram is as clear as possible. To implement the construction of Relationship lines it decided that work should first be done on a direct, deterministic scheme to be augmented and/or replaced eventually by a general line router able to place all lines regardless of complexity (the complexity of a line corresponds quite closely to the distance between the Entities at each end of the line). There were three reasons behind this choice :

1. No line router of any repute had been found at the time work started on Relationship placement,
2. The direct placement scheme would give some insight into the arrangement of Entities on the diagram for future work on a general router, and
3. The methods used to place the simplest Relationship layouts could be used later instead of invoking the general router.

The eventual general router must be able to balance the three limiting factors of the length of the line, the number of corners in the line, and the number of crossings the line causes against one another to give the best placement for each Relationship.

The direct router takes its design from the fact that an inflexible grid is used for placing Entities. This grid means that the boxes representing Entities will appear on the diagram like chips on a computer PCB. Since Entities are placed so that related Entities are more likely to be close together, each Relationships in the diagram can be classed according to the positions of the two Entities involved in it. The classes that Relationships are divided into are (using Fig 9) :

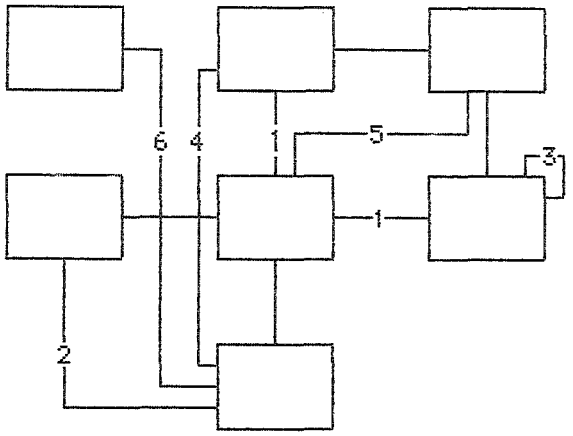


Fig. 9 Relationship Types

(1) Straight line,

(2) One corner,

(3) Self Relation,

(4) Same row or column, not adjacent,

(5) Next row and next column,

(6) Next row or column, far away,

(7) Other (which the router does not attempt to place).

This division of Relationships is sufficient for smaller Data Models and gives a very fast, useful line placement method. However, as the number of Entities in the Data Model increases to above around fifty, the number of Class 7 Relationships and the number of Class 4 to 6 Relationships not placed due to congestion begin to increase rapidly. The original intention was to write a more general router to place the rejects of the direct router but it was later decided that such an general router should be written (when one was found) to place all relationships in the same manner (the direct router could still be used then for placing very simple Relationship like types 1 and 2).

3.5 Aesthetic Requirements and Effects.

Beyond the aspects of Entity and Relationship placement covered by methods for solving the QAP and routing problems are the additional considerations of readability, informativeness, and aesthetic value. Any attempt at defining methods to deal with these considerations will neccessarily duplicate assumptions made by a person when they draw a diagram. It was decided that there are two ways in which the diagram could better fulfil the above requirements :

1. Allow the Relationship(s) between Entities to influence the relative positions and orientations of the Entities, and
2. Move blocks of Entities around to create room so that more Relationships are able to be drawn simply.

The first of these goals can be satisfied by creating "biases" for each of the Entity/Relationship/Entity combinations that require a particular relative positioning. These biases can be added to any algorithm for placing Entities by returning an artificially high distance value between two Entities when their relative positioning is not that required.

It should be noted that the biases chosen for use in the diagrammer are purely experimental and may not effectively reflect the assumptions actually made by systems analysts.

The second of these goals was found to be too difficult to deal with effectively for two reasons :

1. It requires the ability to move large blocks of Entities around on the diagram with the aim of reducing the complexity of Relationships, and
2. It requires the diagrammer to possess knowledge about the real world (for example businesses) which is not directly obtainable from the information stored in the Data Model.

The use of this knowledge is displayed by a systems analyst when he or she groups together Entities which seem to have no obvious link but go together because they describe similar aspects of a business.

4. Implementation.

4.1. User Interface.

The diagrammer allows users to generate four different types of diagrams which differ in the selection of the Entities and Relationships to be diagrammed. These four types of diagrams, Entity Centred, Entities involved in a Procedure, Entities in a Functional Area, and Complete Data Model, were chosen to provide a grading of size and complexity of diagrams. Two of the diagram types are simple to explain out of the EXSYS context : these are Complete Data Model and Entity Centred diagrams .The Entity Centred diagram is a diagram of the Entities and Relationships within a certain number of Relationships distant from a particular Entity (the user inputs both the Entity to centre the diagram on and the number of Relationships to follow out from that Entity). The other two types of diagrams (the Procedure and Functional Area type diagrams) need to be explained in the EXSYS context. In the Procedure Model in EXSYS are defined Functional Areas (an equivalent concept to schema or sub-schema) and Procedures (which must belong to a Functional Area). Entities are involved in the Input and Output of of a Procedure (it is these Entities that are diagrammed for an "Entities involved in a Procedure" type diagram). The Entities in the "Entities involved in a Functional Area" type diagram are all those Entities involved in each of the Procedures that belong to a particular Functional Area.

The user, after choosing the type of diagram they wish to do, must answer questions to completely define the diagram (that is, things like a Procedure name or an Entity name). The steps involved in this are described in [5]. Once all these questions are answered the automatic diagram generator is invoked and the Entities and Relationships are placed. Once a diagram has been generated the line printer output routine is called and the diagram is written to a disk file and may be printed at any time after that.

There are two reasons why the diagrammer has been packaged as a

complete unit for the automatic generation of diagrams at this time - there was no time to implement an interactive method, and Adata wanted the diagrammer to be sent overseas as a part of EXSYS.

4.2. Data Structures and Data Collection.

To implement the Entity and Relationship placing algorithms for the automatic generation of diagrams, various data structures are required. There are two main divisions of these data structures. The first is the top level of the data structures (both for Entities and Relationships) which stores information gathered from the EXSYS Data Model about all Entities and all Relationships. This information is gathered upon the first entry to the diagrammer and is neither changed or deleted while the user is using the diagramming package. The second level of the data structures is for information generated by the data gathering section and the placing of Entities and Relationships and include data structures that hold information about the corners and line segments that belong to a Relationship. These second level data structures last only for the duration of the construction of a diagram and a new set is constructed for each diagram that is generated.

The data structures used by the diagrammer are set up so that any data relating to Entities or Relationships is able to be accessed from anywhere in, respectively, the Entity or Relationship data structures. This is accomplished by using pointers to traverse down the data structures and ID numbers of Entities and Relationships to get to the top level of them. The algorithms implemented in the diagrammer require some methods of getting from a Relationship to the two Entities involved in it and from two Entities to the Relationship that exists between them (if one exists). It is a simple matter to store information in the Relationship data structures that says which two Entities are involved in it. However, to find a (possible) Relationship between two Entities a hashing method must be implemented. The hashing function used in the diagrammer is given

below :

Given two Entity ID's in character form, (e_1, e_2) each zero-padded and of three character length, the hash value of each one is given by

$$h(e) = e[0] * 5 + e[1] * 7 + e[2] * 11.$$

These hash values are then multiplied together modulo 3607 (which is the first prime number greater than 3600 (the number of Relationships allowed by EXSYS)) to give the bucket a Relationship between these two Entities should reside in. The linked list of buckets originating from that index is then searched to look for the wanted Relationship and, if found, its ID is returned.

4.3. Forestry.

The first stage in the Entity placement algorithm is the construction of a breadth-first spanning tree to enable the other, major part of the algorithm to reduce the number of attempts at displacement of Entities already placed when placing an Entity. The root of the tree is either the most related Entity of those selected (for Complete Model and Functional Area type diagrams) or a chosen Entity (for Entity Centred diagrams and for Procedure diagrams (here the root is the input (or output) Target Entity of the Procedure)).

After determining the root Entity, successive calls to the function gr_forest are made which add another layer of Entities to the tree. These calls stop when either no more Entities are added to the tree or, in the case of Entity Centred diagrams, when a predetermined limit is reached. The function gr_forest adds a new layer of Entities by traversing the tree in a breadth-first manner, adding Entities when at a leaf of the tree.

If working with a Complete Model or Functional Area type diagram, and the first tree does not include all Entities to be diagrammed then successive new trees are created until all Entities to be diagrammed are in a tree. The roots of each new tree are chosen in the same manner as the

root of the first tree : that is, each new root is the most related entity not yet a member of a tree. The situation where more than one tree is created corresponds to the situation where the graph equivalent to the Data Model is disconnected.

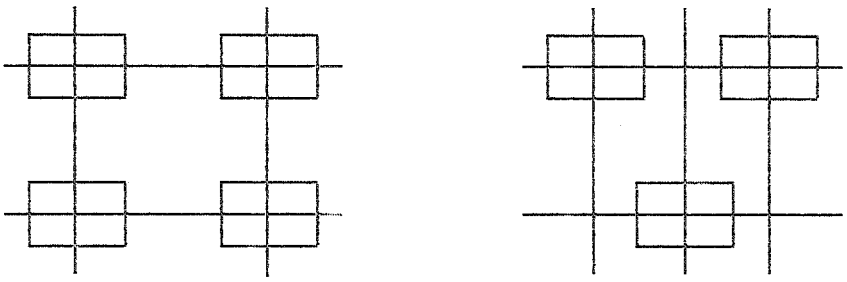
4.4 Grid Structures.

Due to limitations imposed both from the external world and from the design methodology of the diagrammer the Entity boxes and the Relationship lines cannot be placed just anywhere. Entity positions are restricted because the direct placement scheme for Relationships can cope only with the Entities in predictable grid positions. To give more flexibility to the positions of Entities and to attempt to design a general Relationship placement scheme without knowledge of any work done elsewhere would have made the automatic generation of diagrams sufficiently complicated to take it outside the scope of an Honours Project.

The positions of Relationship corners and line segments are limited to positions on a grid because of the nature of the most often used, and most limited, output device - the line printer. The smallest unit of size in a diagram must be the same as, or larger than, that of a line printer (the printable character). This means that arbitrarily close Relationship lines can only be printed by "blowing up" the diagram until the lines are one or more character position apart on the line printer. Due to this, and since a diagram should take up some minimum space, some lower limit to the positional difference between Relationship lines must be imposed - this is a grid. Relationships are also constrained to follow only horizonal and vertical lines (Manhattan layout) since a printer cannot cope with diagonal lines.

Along with the grid defined for Entity placement (50 x 50) there was also work done on a grid of 100 x 50. This grid allowed Entities to be placed in "half positions" on the normal grid as in Fig 10. This grid was

developed to allow more freedom to Entities but created too many types of Relationships for the direct Relationship placement scheme too cope with so it was shelved indefinitely. Also, this grid gave no real increase in the clarity of the diagrams when used for Entity placement.



Normal Grid Grid with half positions

Fig. 10 Grids

4.5 Entity Placement.

The Entity placement algorithm in Sec. 3.3 is implemented in the diagrammer at page 3 of the listing (Appendix E). There is only one point worth reporting in regard to this section. As part of the attempt to make the automatic diagram generator create diagrams that fulfil the aesthetic and information expectations of users, a bias on the relative positions of related Entities has been implemented as part of the distance function between the Entities. This bias function has several different bias types implemented, which are :

(1) Ordinary : the two Entities have no preferred relative positioning,

(2) SidebySide : they prefer to have the same y position,

(3) Vertical : they prefer to have the same x position, and

(4) UpandDown : one Entity prefers to be above the other but they don't care what relative horizontal positions they are in).

These biases are derived from various types of Relationships, and are implemented by causing the distance function to return an artificially high value (multiply the ordinary distance between the two Entities by some large number like 10) if the two Entities are not in the preferred relative orientation. There is no definite proof that the bias functions relate at all to user expectations but, since the current method of determining which Relationships types cause which biases is purely experimental, user feedback will refine the assumptions used in the current implementation.

4.6 Relationship Placement.

Relationships are placed on the diagram by a direct placement scheme. These Relationships are required, for the sake of nice looking diagrams, to have as few crossings and corners as is possible. A way to accomplish this is to place the Relationships in order of simplicity (this order is given in Sec. 3.4) so that the relationships with the least number of corners are placed first. As given in Sec. 3.4, each Relationship is given a type depending on the positions of the two Entities involved in the Relationship. Each type of Relationship has an area of the sides of the boxes representing the Entities involved in a Relationship of that type which are preferred locations for joining the ends of the Relationship line to the Entity boxes. These preferred areas (which have not been shown here due to reasons of space) have been chosen to minimise the chance of causing crossings and the direct Relationship placement routines choose the point closest to the favourite position that is part of the preferred area and is not already being used.

For type 5 Relationships, if there are more than one Relationship between the same two Entities the methods of choosing the points of attachment to the Entities cause unneccessary crossings. In this situation, the points of attachment on the uppermost Entity of these Relationships are rearranged to remove these crossing as in Fig. 11.

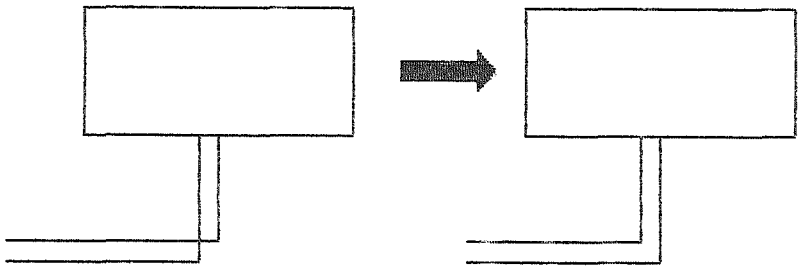


Fig 11.

For types 4 and 5 Relationships, there are two alternative constructions of these Relationships (see Fig 12). To ensure that the most profitable alternative is finally placed, both of them are tested and the one which causes the least crossings is retained. When each alternative is tested, the effects of the test construction must be undone before testing the next or choosing the best (this includes the rearrangement of the points of attachment as described above).

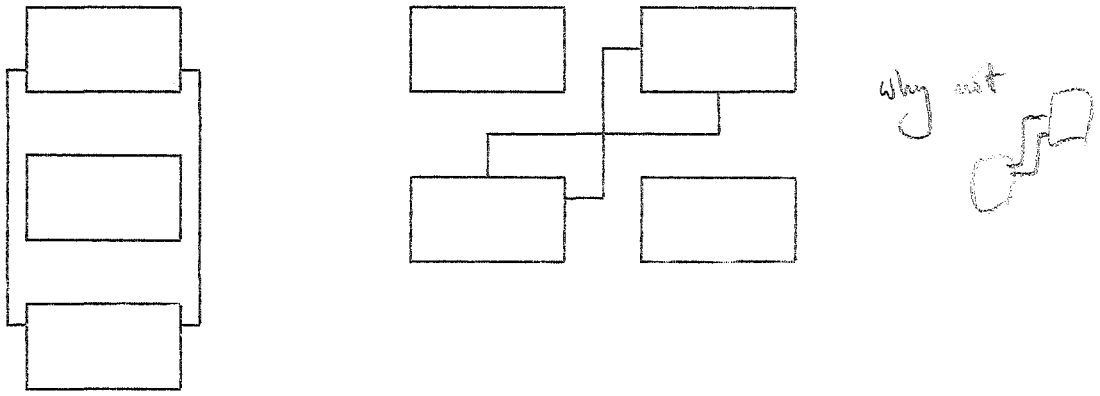


Fig 12. Alternative for type 4 & 5 Relationships

When the construction of a line runs into trouble (as when there is no room left on the preferred section of an Entity box or in a gap between Entity boxes) the partial effects of the construction must be removed. If this happens, the Relationship placement is abandoned and the Relationship is flagged as "not placed" so that later, when printing the diagram, a list of unplaced Relationships can be printed.

4.7 Line Printer Output.

When a diagram is finally constructed it must be written to a file to be printed. A virtual image of the diagram is used on which diagram information is written in positions corresponding to Entity and Relationship constructions. In this virtual image, one horizontal grid position is equivalent to three character positions and one vertical grid position to one character. Entity names are centered in the boxes that represent Entities and are split into more than one line according to their length (EXSYS names are allowed up to 70 characters). The symbols used to represent information regarding the one/many/optional/mandatory nature of Relationship departs from the conventions described in Sec. 2.3 (see Fig 13) since it was found to be too difficult to draw crows feet, etc on a line printer. Instead, * is used to represent many, and **O** and **M** are used to represent optional and mandatory (they are printed inside the Entity box if there is no room to print them on the Relationship line. Multiple Relationships between two Entities are labelled with a number and a list of labelled Relationship are printed below the diagram to enable them to be told apart.

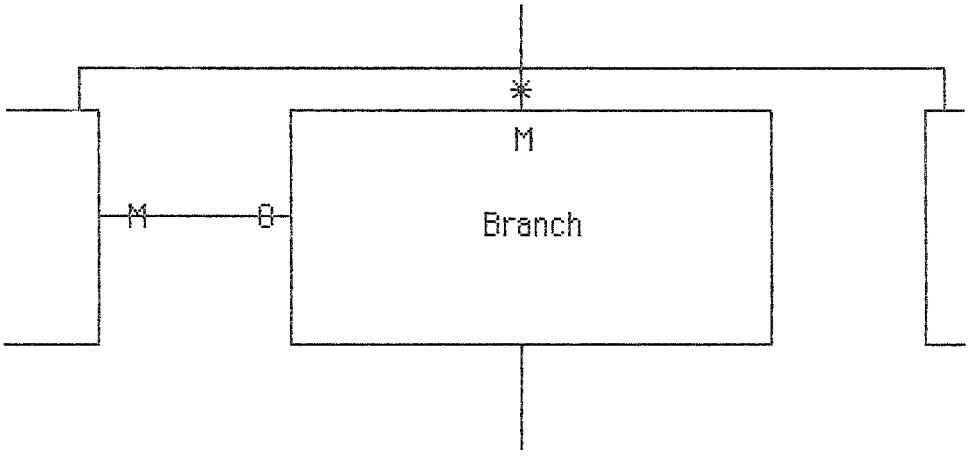


Fig. 13. An Example of Line Printer Output

Mutually Contingent and Exclusive Relationships are not represented using the diagrammer yet as this was felt to increase the complexity of both the Entity and Relationship placement methods beyond an acceptable

level (much less would have been achieved by the termination of this project if this was included). This is because these Relationships are required to adjacent for at least part of their length so that information labelling them as these type of Relationships can be drawn between them.

The printing of the diagram cannot be done by the obvious method of printing each row of the diagram in its entirety and moving to the next row, because of the fact that useful information on the virtual image of the diagram may be (and usually will be) more characters wide than the width of the printer the diagram is intended for. The method developed to print a diagram like this is to divide the diagram up into vertical strips the width of the printer and print each of these in turn. These strips are combined into a complete diagram by cutting and pasting them. The diagram is prefixed by the number of blanks needed to centre this complete diagram.

5. Results and Future Work.

5.1. Results

The features of the diagrammer envisaged at the beginning of this project that have been implemented are : an automatic Entity Placement scheme, a Relationship Placement scheme, and the Line Printer Output routine. Although this is considerably behind the schedule projected at the beginning of the year, this is not a problem as this schedule was drawn up before attempting to study any of the problems associated with graphic design. A major achievement is the fact that a automatic design system sophisticated enough to be accepted as part of EXSYS was completed in August. This diagramming system give the users of EXSYS the capability to generate diagrams based on four different Entity selection schemes.

As yet, there have been few empirical results returned from use of the diagrammer but from what has been returned the conclusion has been reached that for diagrams involving more that around 20 Entities the output produced turns out to be very untidy and unhelpful with quite a few Relationship lines not placed. However, the opposite seems to be true for diagrams involving fewer than 20 Entities with excellent, informative diagrams being generated. This seems to agree with the opinion that diagrams should be kept as small as possible with large Data Models represented in some manner by several smaller diagrams.

5.2. Limitations.

There exist three limitations to the diagrammer in its current state, of which only two can be avoided. These limitations are mostly to do with the grid structures used in placing the Entities and Relationships that make up a diagram.

The first limitation, one that is not removable but can be virtually completely ignored, is a limit on the number of Relationships an Entity is allowed to be involved in. A value for this limit (currently 32) must be given for the programs that make up the diagrammer to compile. However,

if this value proves to be too low it can be reset to a higher value quickly (Notice that EXSYS defines an average of 4 Relationships per Entity but in most models there will be some Entities that are related to far more than that many).

The second limitation is closely tied to the first and, likewise, can virtually be ignored. This limitation is the definition of the number and distribution of the points at which a Relationship line can be joined onto an Entity box. The current limit allows 32 (as above) of these points (9 on both the top and bottom of an Entity box and 7 on the left and right sides). This limitation has a little more importance to the user than the first as it can be the cause of the non-placement of some Relationships in larger Models.

The third limitation is caused by the coarse grid used for the positions of Entities. This use of this grid means that the spaces between the Entities for the placement of Relationships are all of the same size. Due to the facts that some Entities are more related than other, and therefore some areas of the diagram will have more Relationship lines passing through them, some of these areas will become congested. This limitation is caused by the design of the Entity and Relationship placement schemes and will cause some Relationships in large Models to become unplacable. The only real solution to this limitation is to allow more flexibility to Entity positions and to use an intelligent general Relationship line routing algorithm.

5.3. Future Work.

Due to the open-ended nature of the project and to the sheer size of implementing all the aspects of the diagrammer, there still remains much untouched work to be done. Important sections of the projected finished diagrammer are : the interactive diagram generator / modifier, the loading and saving of diagrams, and the plotter output routine. Although this work has not yet been started much of the background such as general data

structures has already been done and will make this future work much easier to implement.

The work that remains to be done also includes modifications to existing work. Possibilities include preparing and implementing the algorithm mentioned in Sec. 3.3.5 and implementing the general Relationship router described in Sec. 3.4. Also, the automatic placement algorithms of the diagrammer have quite a few parameters (such as weighting schemes for Entities, the biasing of Entity positions (and the penalty for positions which violate these biases), and the method used to determine the distance between grid positions for Entities). It should be possible to improve the automatic generation of diagrams and to make the layout of diagrams correspond more closely to user expectations by tuning these parameters.

6. Conclusions.

The main purpose of this project was to write a diagrammer for EXSYS, with both interactive generation and modification of diagrams and automatic generation of diagrams. This has been partially achieved with an automatic generator of diagrams successfully implemented. The size and complexity of the problem of designing an automatic generator was quite unexpected. The generator produced as a result of this project, as stated in the results, seems to be very useful for generating diagrams of 20 Entities or less as there is little intrinsic complexity in Data Models of this size.

However, some customers of Adata's envisage Data Models involving up to 200 Entities and would like to use the diagrammer for documenting their Model (if intended customers can not successfully use the diagrammer then is it little more than a toy for use at demonstrations of EXSYS?). Clearly, the automatic generator in its current state cannot deal reasonably with a Data Model of this size and complexity so the question must be asked - What needs to be done to successfully use the diagrammer to generate diagram for such large Data Models. There are two solutions to this situation envisaged :

1. Write the interactive diagram generator and implement the new algorithms for automatic generation.
2. Write the interactive diagram generator with hooks into the automatic generator for automatically placement when required.

The second alternative looks more useful in the extreme cases presented by real Data Models. It is envisaged that the user can select the Entities do be diagrammed by hand so that the Data Model can be constructed out of several smaller diagrams each of a managable size and each diagram of clusters or groups of Entities as are recognised by the user.

In conclusion, it is felt that the diagrammer has the capability to be a useful tool for use in aiding users designing systems in EXSYS to document their Data Models.

7. References.

[1] H. M. Bayegan, E. Aas,
"An Integrated System for Interactive Editing of Schematics, Logic Simulation and PCB layout Design",
Proceedings of the 15th Design Automation Conference, 1978, pg 1.

[2] P. Carmody, A. Barone, J. Morell, A. Weiner, J. Hennesy,
"An Interactive Graphics System for Custom Design",
Proc. 17th DAC, 1980, pg 430.

[3] I. R. Dobes, "A Multi-contouring algorithm",
Proc. 14th DAC, 1977.

[4] L. Dysart, M. Koifman, "An application of branch and bound to automatic printed circuit board routing",
Proc 16th DAC, (1979), pg 494.

[5] EXSYS User Manuals, Vols 1 & 2, EXSYS Inc.
Published by Print Marketing Ltd.

[6] C. J. Fisk, D. L. Caskey, L. L. West,
"ACCEL : Automated circuit card etching layout",
Proc. IEEE, v55 (1967), pg 1971.

[7] R. G. Garside, T. A. J. Nicholson,
"Permutation Procedure for the Backboard Wiring Problem",
Proc. IEE, v115 (1968), pg 27.

[8] P. C. Gilmore, "A solution to the module placement problem",
Rep. RC430, IBM, Yorktown Heights, N.Y., 1961.

[9] C. H. Heider, "An N-Step, 2-Variable Search Algorithm for the Component Placement Problem",
Naval Research Logistics Quarterly, v20 (1973), pg 699.

[10] W. Heyns, W. Sansen, H. Beke,
"A Line-Expansion Algorithm for the General Routing Problem with a Guaranteed Solution",
Proc. 17th DAC, 1980, pg 243.

[11] D. W. Hightower,
"A Solution to the Line-Routing Problem on the Continuous Plane",
Proc. 6th DAC, 1969, pg 1.

[12] T. C. Koopmans, M. Beckman,
"Assignment Problem and the Location of Economic Activities",
Econometrica, v25 (1957), pg 53.

[13] E. L. Lawler, "The Quadratic Assignment Problem",
Management Sci., v9 (1963), pg 586.

[14] C. Y. Lee, "An Algorithm for Path Connection and its Applications",
IEEE Transactions on electronic computers, (1961), p 346.

[15] J. Martin & C. Finklestein, "Information Engineering" Volumes 1 & 2,
Published by Savant Research Studies,
2 New Street, Carnforth, Lancashire.

[16] L. Mory-Rauch, "Pin Assignment on a Printed Circuit Board",
Proc. 15th DAC (1978), pg 70.

[17] R. H. J. M. Otten, "Automatic Floorplan Design",
Proc 18th DAC, (1981), pg 261.

[18] G. Persky, C. Enger, D. M. Selove, "The Hughes Automated Layout
System - Automated LSI/VLSI Layout Based on Channel Routing",
Proc. 18th DAC, (1981), pg 22.

[19] N. R. Quinn, "The PLacement Problem as Viewed from the Physics
of Classical Mechanics",
Proc. 15th DAC, (1978), pg 173.

[20] C. F. Shupe, "Automatic Component Placement in the Nomad System",
Proc 15th DAC, (1978), pg 162.

[21] J. Soukup, "Fast Maze Router",
Proc. 15th DAC, (1978), pg 100.

[22] L. Steinberg, "The Backboard Wiring Problem: A Placement Algorithm",
SIAM Review, v3, 1961, pg 37.

[23] T. Yoshimura, "An Efficient Channel Router",
Proc. 21st DAC, (1984), pg 38.

[24] M. Hanan, J. M. Kurtzberg,

"A Review of the Placement and Quadratic Assignment Problems",

SIAM Review, v14, 1972, pg 324.

[25] R. E. Burkard, K. Stratmann,

"Numerical Investigations on Quadratic Assignment Problems",

Naval Research Logistic Quarterly, v25, 1978, pg 129.

Appendix A

HOW TO GENERATE DATA MODEL DIAGRAMS

To obtain a diagram of part or all of your Data Model, enter the Model Listing facility of Data Modeling, choosing the option to produce a listing of a Complete Data Model. There, you may take option 2, a Diagram of the Model.

This will offer you a choice with which you effectively define the scope of your diagram. You may obtain a diagram for

- an Entity and its surrounding related Entities
- the Entities involved in a Procedure
- the Entities in a Functional Area
- the complete Model.

Diagrams consist of boxes for Entities and lines from/to them for Relationships. Currently there is a single restriction on the size of Model which can be managed. The number of lines entering/leaving a box is restricted to 9 at top and bottom and 7 on each side. This means that up to 32 Relationships can be drawn to an Entity, although a few less may be allowed in practice due to variations in physical arrangement (on paper) of the related Entities.

If you choose the first sub-division (an Entity and its surrounding related ones), you are asked for the depth of Relationship attachment; that is, how many Relationships distant from the central Entity you require. Up to 9 is allowed; for most Models, 2 - 4 will suffice.

When you have established the required diagram contents, EXSYS will commence building it. The time to do this depends on Model size and machine loading. Informative messages appear as the process continues.

You are then asked to indicate the dimensions (in characters) of your paper sheet. It is desirable that the printer used be a modern one, with Upper and Lower Case, and compressed print mode. If it has compression capability, it is strongly recommended that you use it.

The default paper size is 132 x 66. The vertical dimension (66) denotes form length, and should match that set up for the AOS/VS system print control. For compressed mode printing, we recommend dimensions of 218 x 88.

If the Model diagram is too wide, EXSYS will automatically divide it into an appropriate number of vertical strips, printing them sequentially. You may then cut and paste to produce the final diagram.

To simplify cutting and pasting, set the top of form at line 1, not line 4. This stops implicit form feeds over the perforation from creating gaps in the diagram. It also makes the page used by this program (66 lines, say) for centering a diagram vertically the same as that used by the printer.

Finally, EXSYS informs you of the filename to which the diagram was sent. You may print this from CLI.

Most of the diagram's details will be self-explanatory. Entity names are centered in boxes. "Association" and "Repeating Group" Entities have an A and RG printed in the top left-hand corner, because this information may be quite significant.

Relationship detail is embedded in the lines, as follows -

```
*----O-----(2)------M----
Many                    One
```

The O indicates "Optional", the M "Mandatory".
Occasionally these have to be omitted, if the line is too short, cramped, or contorted.

If multiple Relationships exist between a pair of Entities, they are
labelled with bracketed numbers (example (2) above) near the line centre, and a list of numbered Relationships appears at the end of the diagram. Apart from this, no Relationship names are printed on diagrams, to avoid needless over-crowding.

Appendix B

6.8 Complete Model Details

Option 8 of the LISTING menu allows you to produce listings of your entire Model in two different ways.

The first is simply a combination of all other options: a complete documentation of your Model.

The second is different, however. **EXSYS** allows you to produce diagrams of your Model.

When you choose option 8, EXSYS presents a choice of the types of output. Note that both are produced only upon a printer.

"1. Printout of All Data Model Details
2. Diagram of Data Model

Enter choice: [1] "

These options are further described in the following sections. Note that pressing CANCEL/EXIT returns you to the LISTING menu.

6.8.1 List the Complete Model

If you choose this option, **EXSYS** will ask you if you are sure that you want this listing.

"Do you want a complete listing of your Model: [_] "

If you answer 'Y', EXSYS will give you a listing which is really a combination of all of the other options in the LISTING menu, except option 4.

Notes

1. A Complete Model listing will always be sent to the printer.

2. A Complete Model listing will usually be of considerable size and this is worthy of some consideration.

   It is, however, a necessary prerequisite during active Model development, as it is a way of recreating your Model if it should be lost. We therefore recommend that you obtain a Complete Model listing periodically.

6.8.2 A Diagram of the Model

EXSYS allows you to produce diagrams of your Data Model using option 2 of the 'Complete Model Details' menu. These diagrams consist of the Entities and Relationships involved.

You may include your entire Model, or you may wish only to diagram a restricted part of it. EXSYS offers four choices.

DATA MODEL DIAGRAM

"1. Centre Diagram on an Entity
2. Entities involved in a Procedure
3. Entities involved in a Functional Area
4. Generate Diagram for the complete Data Model

9. Return to the Data Model Listing

Enter choice: [9] "

These options are detailed below.

1. Centre the Diagram on an Entity

You may wish to prepare a diagram of your Model with reference to a particular Entity. EXSYS allows you to do this with option 1.

Firstly you are asked for the name of the 'central' Entity.

"Enter the name of the Entity on which to centre the diagram:
_____ "

You should enter the name of an Entity that is present in your Model. Remember that you may use the INDEX (SHIFT F2) key to obtain Entity names.

EXSYS then asks you to indicate how much detail you require.

"Enter depth of surrounding Relationships: [9] "

This question is essentially asking you to indicate how large the diagram should be. If you enter '1', you are asking for 1 level of Relationship around the central Entity - you will get a diagram that contains only those Entities directly related to the central Entity.

Increasing this number means that the number of Entities included in the diagram will substantially increase. Entering '9' will, almost certainly, yield a diagram that contains all Related Entities in your Model.

When you have entered a number, EXSYS will prepare the diagram. After accomplishing this, you are asked to give details of how the diagram is to be printed.

PREPARING DIAGRAM FOR PRINTING

"Enter dimensions of paper:
Width: [132]
Height: [66] "

The default values will produce a diagram on a standard draft printer. If your draft printer has 'compressed' capability you should use this by choosing a width of '218' and a height of '88'. Note that EXSYS produces a file which must be dumped to the printer. Your System Administrator should be able to accomplish this for you.

When you have entered values, EXSYS will inform you of the name of the file in which the diagram is contained.

2. Entities in a Procedure

You are able to diagram the Entities involved in a Procedure. Choosing option 2 means that EXSYS will ask you to specify the name of the Procedure.

You are firstly given a choice of two different ways to make this specification.

"1. Nominate a Procedure
2. Step through Procedures

Enter choice: [2] "

Choosing to nominate a Procedure means that EXSYS will simply ask you for the name of a Procedure.

"Procedure name:
_____ "

Remember that you may use the INDEX (SHIFT F2) key to obtain the names of Procedures.

'Stepping through Procedures' means that EXSYS will present each of the Procedures in your Model (in turn) and ask you if you wish to create a diagram of the Entities involved in it.

EXAMPLE

"For Inquiry Procedure 'INQUIRE ON A CUSTOMER' (Y/N): N "

Whenever a Procedure is chosen, EXSYS will prepare a diagram. If you choose to diagram the Entities involved in a PROCESS, EXSYS will ask you if you wish to involve the Input or Output parts.

You are then asked for the dimensions of the paper, as in option '1'.

3. Functional Area

As with diagramming the Entities involved in a Procedure, EXSYS asks you for the Functional Area that the diagram is to be based upon. You are given a choice of two methods of specifying a Functional Area.

"1. Nominate a Functional Area
2. Step through Functional Areas

Enter choice: [2] "

---

4. Complete Model

If you choose option 4, EXSYS will create a diagram of the complete Model. For large or complex Models, EXSYS may not be able to fit all of the Relationships on to the diagram. All such Relationships are noted as a part of the diagram.

After choosing to diagram your complete Model, you are asked for the dimensions of the paper, as in option 1.

better (?) title is the contents specification on p. 2

Appendix C: Derivation of $O(E^{5/2})$ for Entity Placement

There are E Entities to place.

After placing i Entities the area of the diagram delimited by Entity positions is roughly

$$(\sqrt{i}+1)^2$$

allowing a gap of 1 position around the outside of those Entities placed.

This gives an estimate of the number of attempts at placing Entity i+1 of ~~$2\sqrt{E}$~~ $2\sqrt{i}$ positions with i computations of • costs at each attempt.

This gives $\sum_{i=1}^{E} 2\sqrt{i}\; i$ computations for attempts at placement

$$= O(E^2\sqrt{E})$$
$$= O(E^{5/2})$$

For displacement attempts, there are $2\sqrt{i} \times i$ computations for attempting to replace the Entity in the position being tested. However the number of attempts at displacement is limited to $O(1)$ attempts per attempt at placing an Entity.

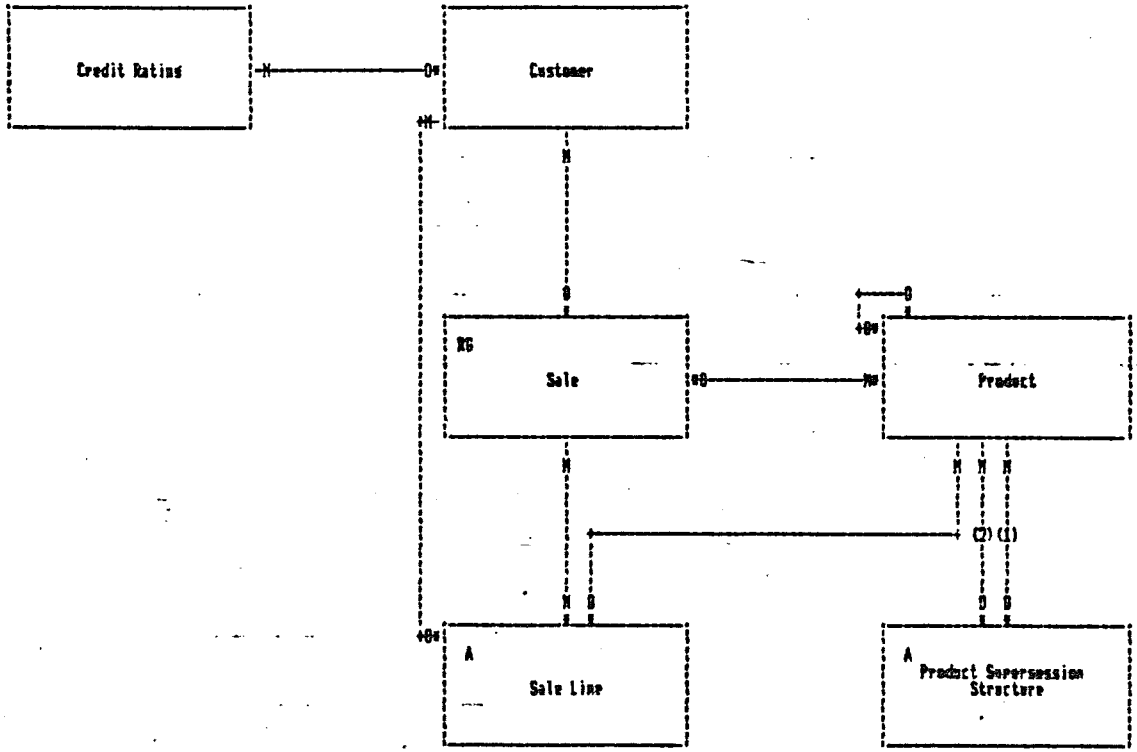
This gives $\sum_{i=1}^{E} \sqrt{i}\; i$ computations for displacement attempts.

$$= O(E^2\sqrt{E})$$
$$= O(E^{5/2}) \text{ also.}$$

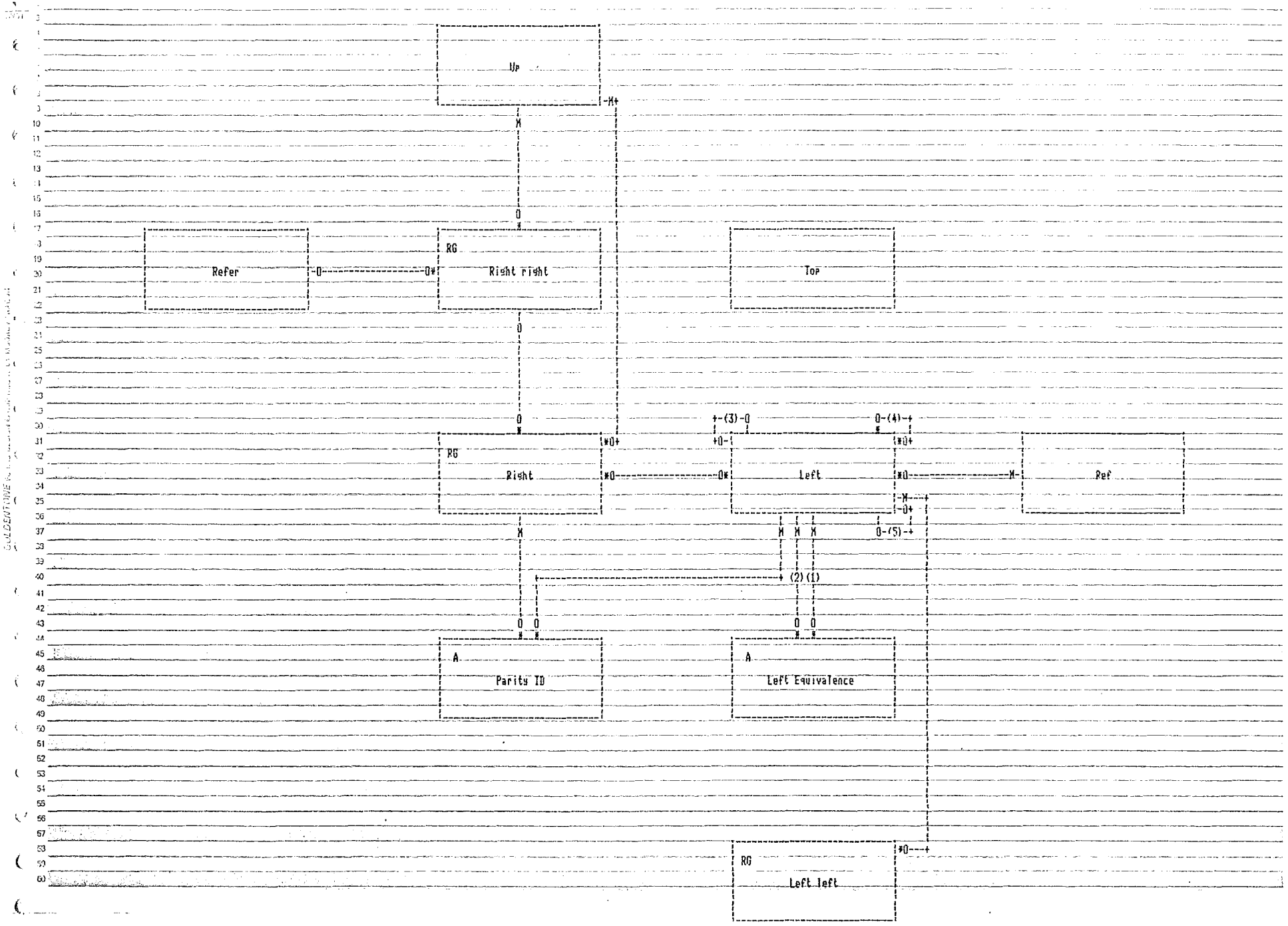
So the order estimate for the Entity placement algorithm is

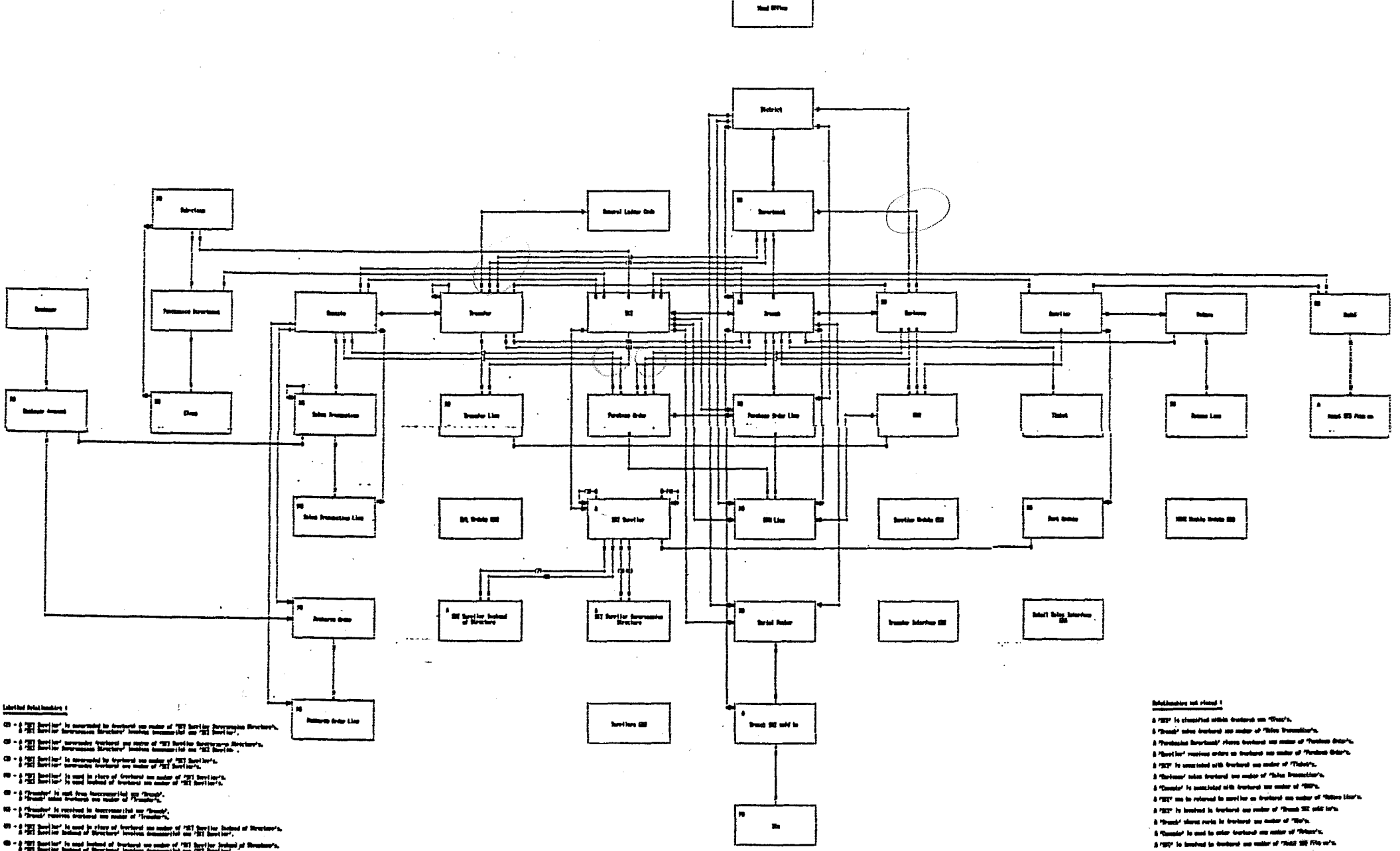
$$O(E^{5/2})$$

Complete Model Diagram



Appendix D : Some Diagrams


Up
Refer
RG
Right right
Top
RG
Right
Left
Ref
A
Parity ID
A
Left Equivalence
RG
Left left



Labelled Relationships :

- 01 - A [illegible] 'Service' is associated to (business) one member of [illegible] 'Service Organization Structure'. A [illegible] 'Service Organization Structure' involves (consumption) one [illegible] 'Service'.
- 02 - A [illegible] 'Service' (organization structure) one member of [illegible] 'Service Organization Structure'. A [illegible] 'Service Organization Structure' involves (consumption) one [illegible] 'Service'.
- 03 - A [illegible] 'Service' is (consumption) by (business) one member of [illegible] 'Service'. A [illegible] 'Service' (consumption) (business) one member of [illegible] 'Service'.
- 04 - A [illegible] 'Service' is used by (share of) (business) one member of [illegible] 'Service'. A [illegible] 'Service' is used (business) of (business) one member of [illegible] 'Service'.
- 05 - A 'Transfer' is used from (consumption) one 'Branch'. A 'Branch' makes (business) one member of 'Transfer'.
- 06 - A 'Transfer' is received by (consumption) one 'Branch'. A 'Branch' receives (business) one member of 'Transfer'.
- 07 - A [illegible] 'Service' is used by (share of) (business) one member of [illegible] 'Service Ledger of Structure'. A [illegible] 'Service Ledger of Structure' involves (consumption) one [illegible] 'Service'.
- 08 - A [illegible] 'Service' is used (ledger) of (business) one member of [illegible] 'Service Ledger of Structure'. A [illegible] 'Service Ledger of Structure' involves (consumption) one [illegible] 'Service'.
- 09 - A 'Warehouse' is the source for (business) one member of 'Transfer'. A 'Transfer' has its source in (consumption) one 'Warehouse'.
- 010 - A 'Warehouse' is the destination for (business) one member of 'Transfer'. A 'Transfer' is destined for (consumption) one 'Warehouse'.
- 011 - A 'Branch' is used to order (business) one member of 'Purchase Order'. A 'Purchase Order' is ordered by (consumption) one 'Branch'.
- 012 - A 'Branch' holds [illegible] records stored (business) one 'Purchase Order'. A 'Purchase Order' is connected with (business) one 'Branch'.
- 013 - A 'Purchaser' orders (business) one member of 'Purchase Order'. A 'Purchase Order' is ordered by (business) one 'Purchaser'.
- 014 - A 'Purchaser' authorizes (business) one member of 'Purchase Order'. A 'Purchase Order' is authorized by (consumption) one 'Purchaser'.

Relationships and phrases :

A 'POS' is classified within (business) one 'Class'.
A 'Branch' makes (business) one member of 'Sales Transaction'.
A 'Purchasing Department' places (business) one member of 'Purchase Order'.
A 'Department' receives orders on (business) one member of 'Purchase Order'.
A 'POS' is associated with (business) one member of 'Tickets'.
A 'Customer' makes (business) one member of 'Sales Transaction'.
A 'Customer' is associated with (business) one member of 'POS'.
A 'POS' can be returned to (service) on (business) one member of 'Returns Line'.
A 'POS' is invoiced to (business) one member of 'Branch POS sold to'.
A 'Branch' stores route in (business) one member of 'Site'.
A 'Customer' is used to order (business) one member of 'Orders'.
A 'POS' is invoiced to (business) one member of 'Branch POS Site sold'.
A 'Distributor' stores route in (business) one member of 'Site'.
A 'General Ledger Code' is entered in (business) one member of 'Sales Transaction'.
A 'POS' has VAT's also recorded on (business) one member of 'Sales Transaction Line'.
A 'Department' is invoiced to (business) one member of 'POS Services'.
A 'Branch' places (business) one member of 'Purchase Order'.
A 'Department' receives (business) one member of 'Purchase Order'.