A network display program

Susan Jones*

LSE, Houghton Street, London WC2A 2AE

This paper gives an account of the development and use of software to manipulate and display networks. The work was done as part of the LSE LEGOL Project in Information Systems Analysis and Design, but the programs were written to be generally applicable; so that they might be equally suitable for displaying the results of statistical processes like single-linkage cluster analysis and path analysis, or producing structure diagrams for program or system components.

(Received April 1977)

The idea of treating a large system with many interrelated parts as a directed graph is common to many applications (See Busacker and Saaty, 1965). In particular, another LSE research project has produced software to perform well defined algorithms (e.g. partitioning into subgraphs, forming lists of successors and predecessors) on large precedence networks representing complex information systems (Waters, 1976). The original aim in the case of the present work was to find a method for displaying a network automatically, and for tracing particular paths through it in a selective way. This proved possible, using an interactive graphics terminal and a program which allowed a network of nodes and connections to be built up progressively on a screen, under user control. In the last year, facilities for partitioning, merging and reducing networks have also been included.

The first section of this paper discusses networks in the context of the LEGOL project with the help of examples; the second section outlines the capabilities of the programs developed so far.

1. The use of networks within the LEGOL project

LEGOL is intended as a formalism for describing the rules which define an information system. It has been developed mainly as the result of the study of those complex but well defined systems specified by Acts of Parliament or other legislation. The aims of the LEGOL project as a whole are described fully in Stamper (1977) and elsewhere. For the present, it is sufficient to make the point that the formalism consists of two kinds of 'rules'. What are called first-order rules are those which make some particular prescription, for example:

Section 1. 'Subject to the provisions of this Act, there shall be paid by the Minister for every family which includes two or more children, and for the benefit of the family as a whole, an allowance in respect of each child in the family other than the elder or eldest at the rate of eight shillings a week in respect of the first child other than the elder or eldest and ten shillings a week in respect of each other such child.'†

Second-order rules act upon first-order rules by specifying their dependence upon and influence over one another; they are the counterpart of such phrases as:

'Subject to the provisions of Section 7 of this Act . . .'
'Without prejudice to the provisions of paragraph 1 of Schedule 7 to the Insurance Act . . .'

Within any piece of legislation there is a complex web of interrelationships, both explicit and implicit, for which the idea of a precedence network appears to be appropriate. The Family Allowances Act of 1965 has been examined from this point of view. It is a short, comparatively self-contained Act, but one which implies the necessity for an administrative system to carry out its provisions. Below are some general observations about the problems involved in setting up a precedence network based upon this Act.

1. *Precedence*

The first question to be discussed is what exactly is meant by the word 'precedence' in this context. One of the purposes of the LEGOL formalism is to translate, with the least possible distortion, the 'static', descriptive form of an Act, stating which rules apply in which circumstances, into a more procedural language, which expresses itself in terms of operations to be performed in a certain order. But this ordering must be based upon the underlying relationships of dependence to be found in the original prose version. At the textual level, the relationships may be expressed in many different ways, as the following examples show:

Section 24. 'Notwithstanding anything in the Government of Ireland Act, 1920, the Parliament of Northern Ireland shall have power . . .'

The Family Allowances Act overrides the Government of Ireland Act in this instance.

Section 11(6). 'Where a person is entitled in respect of a child to a guardian's allowance under Section 29 of the Insurance Act . . .'

The Insurance Act provides some of the information required to administer the Family Allowances Act correctly.

Section 10(3). 'In the application of this Section to Scotland—for the reference in Subsection (1) to the bankruptcy of a person there shall be substituted a reference to the sequestration of the estate . . .'

Section 10(3) conditionally modifies Section 10(1).

Section 4(1). 'Allowances for any family shall belong . . .
Section 4(1)(b). in the case of the family of such a man as is mentioned in Section 3(1)(b) of this Act, to him;'

Paragraph 4(1)(b) comes into force only if the conditions set out in paragraph 3(1)(b) apply.

†This extract, and all subsequent statutory quotations, are taken from the Family Allowances Act, 1965.
*Now in the Department of Computer Science, The City University, Northampton Square, London EC1V 0HB

Downloaded from https://academic.oup.com/comjnl/article/22/2/98/428910 by guest on 19 April 2024

Many other variants could be illustrated, in which one section of text affects the interpretation of another, by widening or narrowing its scope of application, and placing it precisely in relation to the rest of the Act, and to other relevant Acts. What the above examples have in common is that the contents of the logical 'predecessor' must be taken into account before its 'successor' can be applied correctly. At the level of the LEGOL formalism, precedence relationships will determine the sequence in which the operations specified by first-order rules must be performed. In general this will not be a simple linear sequence but a partial ordering; some parallel processing may be implied. On the other hand, some sequencing rules will be conditional or based upon criteria not always applicable. So, in order to simulate the effect of the Act in any particular circumstances, it will be necessary to trace a path selectively through first-order rules according to those precedence relationships which are actually relevant. The software described in Section 2 is intended to allow the selective 'exploration' of a network by a user, as an aid to the analysis of problems of this kind.

2. *Network components: nodes*

An Act of Parliament in prose form is split into parts and schedules, further divided into sections and subsections, which in turn may be subdivided into smaller units when separate cases or conditions are itemised. Textual units at any of these levels may be treated as nodes in a network. There are some difficulties about this, as the discussion below indicates; nevertheless, textual units appear to be the only objective framework within which to examine precedence relationships initially. Each node can be identified uniquely by a key which may specify:

1. Act
2. Year
3. Part or Schedule number
4. Section number
5. Subsection number
6. Item letter (a, b, c, etc.)
7. Item number (i, ii, iii, etc.)

However, the textual units making up an Act of Parliament differ greatly in function. A first attempt to categorise them led to the following six types:

2.1 *Defining*

These items put a precise meaning on some of the words and phrases used in the Act.

Section 2(1). 'A person shall be treated for the purposes of this Act as a child—

Section 2(1)(a). during any period whilst he is under the upper limit of the compulsory school age . . .'

2.2 *Prescriptive*

These items state what is to be done, who is to do it, when, where, etc.:

Section 4(2). 'Sums to be paid on account of an allowance for the family of a man and his wife living together shall be receivable either by the man or by the wife.'

Examples of the above types may contain references to other parts of the text, but they have some substantive content of their own. The remainder are more bound to the internal structure of the text.

2.3 *Modifiers*

These items specify changes (usually conditional) to be made in another part of the Act:

Section 14(5). 'In its application to Scotland this Section shall have effect as if—

Section 14(5)(a). in Subsection (2) the word "summary" were omitted.'

2.4 *Links*

These items point to some other part of the Act and state in what circumstances it is applicable:

Section 3(3). 'The provisions of the Schedule to this Act shall have effect as to the circumstances in which a man and his wife living together . . . is (sic) to be treated as maintaining a child . . .'

2.5 *External References*

Most Acts will make numerous references to other Acts, in whole or in part (see first two examples in paragraph 1) and these references should also be treated as separate nodes in the network, although sharply distinguished from the pieces of text to which they refer, which could be of any of the other five types.

2.6 *Empty*

These items have no text associated with them but are merely 'place markers' within a hierarchy. Section 3 of the Family Allowances Act has no content which is not part of Subsections 3(1), 3(2), etc. but it may be referred to as a whole from other parts of the Act, and so requires to be separately identified.

Clearly, there are difficulties in attempting to set up a network in which such disparate elements are treated on equal terms. At the level of the LEGOL formalism some nodes will eventually produce first-order rules, some second-order rules, some both or neither. Compared with orthodox programming languages, for instance, legal prose mixes together the functions of data definition, text editing, specifying procedures, transmitting arguments, etc. in a very free way. Moreover, as we have already seen, an Act is hierarchical in its structure of sections, subsections and so on, but precedence relationships may operate between nodes at any level. It does not seem to be feasible to represent both hierarchical and precedence relationships simultaneously in visual form, but it is necessary to be aware, when examining a singel-level network, that a connection to, say, Section 3, implies a connection to all its constituent parts.

3. *Network components: connections*

A connection between two nodes implies some sort of precedence relationship between them. How can we derive these relationships from an examination of an Act? This section discusses some possible criteria.

3.1 *Sequential*

In a trivial sense one paragraph is the 'predecessor' of another if it comes before it in the text. However, legal prose probably relies less on the cumulative effect of a sequential scan than prose of any other kind, and textual ordering is a poor indicator of the order in which operations must be performed to carry out the provisions of an Act correctly. The Act under discussion begins by enunciating a general principle (see the quotation above) and then proceeds to further levels of detail. To apply it to any particular cases, the detailed provisions would need to be considered before deciding whether the general principle was applicable. In this particular example, various definitions of the meanings of words and phrases used throughout the Act are placed near the end, in Sections 17-19. In a 'procedural' interpretation, such definitions would have to be taken into account first. In saying that textual precedence

Downloaded from https://academic.oup.com/comjnl/article/22/2/98/429907 by guest on 09 April 2024

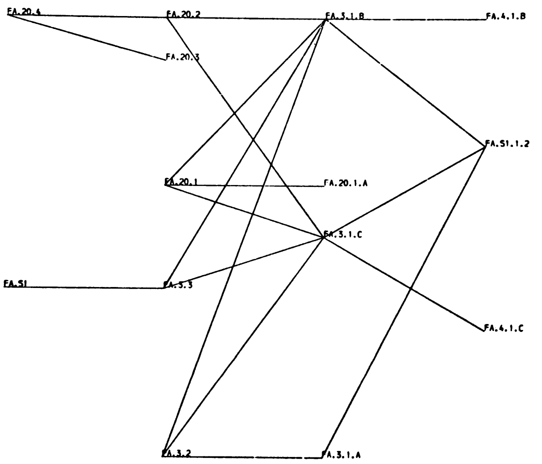

Fig. 1 Family Allowances Act, 1965 cross reference network: Sections 3, 4, 20 and the Schedule

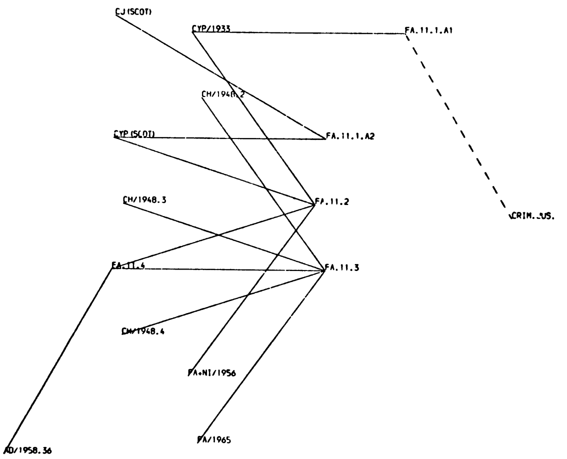

Fig. 2 Family Allowances Act, 1965 cross reference network: Section 11, Children's and Young Persons Acts, Criminal Justice Acts, etc.

does not imply logical precedence, no criticism is intended of the draftsman, who sets out the Bill in the way which will best expedite its progress through Parliament, rather than to act as a specification for an administrative system. But for our present purpose, a consideration of the sequential relationships between different parts of the text is not particularly useful when attempting to discover logical relationships.

3.2 *Explicit*

A great many of the important logical connections are signalled by explicit cross reference from one piece of text to another. All the examples given in paragraph 1 are of this kind. However, the interpretation of cross references for the purpose of identifying precedence relationships is not entirely straightforward. Sometimes a reference is made to another Act for the purpose of *denying* any such relationship:

Section 10(2). 'Sums receivable by any person on account of an allowance shall *not* be included in calculating his means for the purpose of Section 5 of the Debtor's Act, 1869.' (sic).

Some references are very general in scope, making it difficult to link the node to the network at any precise point:

Section 12. 'Section 91 of the Insurance Act (which makes provision for the furnishing by registrars of births, marriages and deaths information for the purposes of that Act and for the obtaining of birth, marriage or death certificates for those purposes) shall apply for the purposes of this Act as it applies for the purposes of that Act.'

Some references specify a connection without a definite precedence relationship, like the one between the two Acts mentioned in the following extract:

Section 8(3). 'Where, in the case of any person, any sum may . . . be recovered by deduction from any payment under this Act, it may instead be recovered from him . . . by deduction from benefit under the National Insurance Act. . . .'

In short, it is not enough to record simply the fact that a cross reference has been made; a detailed examination of context is essential to determine its effect. Examples of the phraseology used to indicate precedence have already been given in paragraph 1. Some parts of a network for the Family Allowances Act, the connections of which represent explicit textual cross references, are given in **Figs. 1 and 2.***

The first diagram shows the links between various parts of Section 3, (dealing with the definition of a family) Section 4, (stating to which member of the family the allowance belongs), Section 20, (giving the residence qualifications for the allowance to be payable to the family), and Schedule 1, (in which more detailed information is given concerning whether a child can be considered part of a family). This example illustrates the usefulness of a cross reference network but also its limitations. In the text, Subsection 3(1) gives three definitions of a family:

(*a*) a man and his wife living together plus children

(*b*) a man plus children

(*c*) a woman plus children

Subsection 4(1) states to whom the allowances for the family belong in these three cases. Paragraphs 4(1)(b) and 4(1)(c) make explicit reference to the corresponding paragraphs 3(1)(b) and 3(1)(c) but paragraph 4(1)(a) makes an implicit reference by the repetition of the phrase: 'a man and his wife living together' and so this particular connection, though just as necessary as the other two, does not appear in the network.

The other example centres on Section 11 of the Act, which details those circumstances in which a child shall not be treated as a member of its family for the purposes of assessing the allowance, because he is in the care of the local authority, at an approved school, etc. This section refers to the Children's and Young Person's Act, 1933, and the Children's and Young Person's (Scotland) Act, 1937, the Criminal Justice Act, 1961 and the Criminal Justice (Scotland) Act, 1963. It also refers to two sections of the Children's Act, 1948, the Family Allowances and National Insurance Act, 1956, and the Adoption Act, 1958. Note that all these external references are regarded as predecessors of the relevant subsections, since their provisions must be taken into account before Section 11 can be applied correctly.

3.3 *Implicit*

It is not possible to build up a complete picture of an Act's structure from cross reference alone. Equally important are those implicit relationships based upon the definition and use of words or phrases referring to entities and their 'properties'. The idea of semantic analysis is central to the whole LEGOL project. That is, any attempt to translate an Act into the LEGOL formalism must be preceded by a detailed identification of those entities (objects, persons, events, relationships) dealt with by the legislation. In the present case, we are

*In these, and in all subsequent diagrams, the 'predecessor/successor' relationship is normally shown as a left-to-right connection (but see Section 2).

Downloaded from https://academic.oup.com/comjnl/article/22/2/98/428910 by guest on 19 April 2024

talking about entities such as 'children', 'families', etc. which are explicitly defined in the Act, making use of such concepts as 'school-leaving age', 'issue', 'apprentice', which in their turn are also defined. Clearly the dependencies implied by such definitions form another precedence network.

However, they are only the most conspicuous examples of those intrinsic logical relationships which must operate to make a piece of legislation 'work' correctly. For instance, we have already seen from previous examples that Section 3 of the Act under examination defines when a family is deemed to exist, and Section 11 details those circumstances when a child is not to be treated as part of his family. There is no explicit link of any kind between these two sections and yet, because a knowledge of Section 11 may be required to interpret Section 3 correctly, there is a precedence relationship between them.

The example in **Fig. 3** illustrates a network for the Family Allowances Act based upon logical connections such as those described above. It is expressed at a high level of generality, showing only connections between complete sections of the Act. Nodes in this network are labelled with the section number and a single mnemonic summarising that section's contents. The following detailed commentary describes the network in terms of the series of steps which would be required to build up the complete diagram using the network display program described in Section 2 of this paper.

Commentary

We begin with Section 1, which states, in general terms, that an allowance is payable to families in respect of all children except the first. Logically, however, this is not the beginning of the story, since a great deal of the Act is involved with specifying precisely what conditions must be fulfilled for the entitlement to be valid. So initially the search for connections will go backwards.

Step 1

Section 1 has two immediate predecessors: Section 3, which defines what combinations of parents and children constitute a 'family' within the meaning of the Act, and Section 20, which makes the qualification that an allowance is payable only to families resident in Great Britain.

Step 2

Section 20 has no predecessors but Section 3 has three: Section 2, which gives the conditions, in terms of age limits and occupations, for qualifying as a 'child', Section 11, which deals with the exclusion from consideration as part of the family of children in the care of the local authority, etc. and Section 17, which contains 'provisions as to special circumstances affecting the operation of Section 3'.

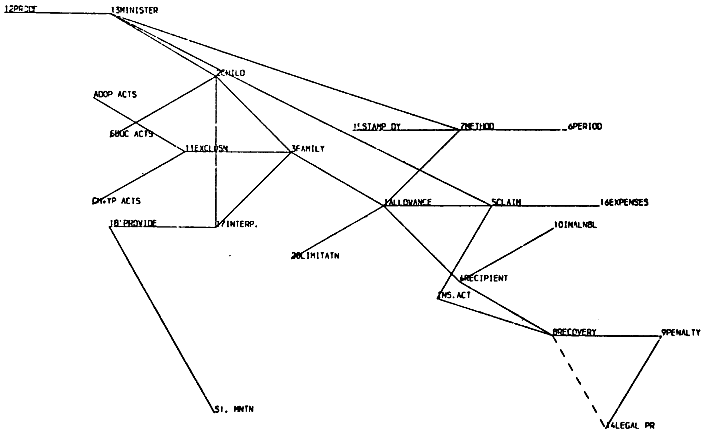

Fig. 3 Family Allowances Act, 1965 logical network

Step 3

Section 2 has two predecessors: the Education Acts, which legislate for the upper limit of the school-leaving age (one relevant criterion for 'childhood') and Section 13, which states among other things that the Minister (of Pensions and National Insurance) may make regulations for specifying the circumstance in which a person is to be treated as a child. Section 2 also connects with Section 17.

Step 4

The definition of a family includes the condition that a child is 'maintained' by its parents and the notion of maintenance is expanded in Section 17, using the term 'providing for' a child. Section 18 gives the meaning of 'providing for' and the Schedule to the Act defines what level of maintenance, in terms of monetary contributions, satisfies the required conditions.

Step 5

Section 11 appeals to Adoption Acts and various Children's Acts when declaring when a child is to be excluded from a family for the purposes of the Family Allowances Act.

Step 6

Section 18 is also a logical predecessor of the Schedule. Section 13 gives the Minister the right to require persons to furnish him with information of facts affecting rights, and its predecessor, Section 12, states, in general terms how documentary evidence of births, marriages and deaths can be obtained.

Step 7

The predecessors of Section 1 are now exhausted and we wish to examine the sections of the Act which follow when the right to an allowance has been granted. There are three immediate successors: Section 4, which declares to whom the allowance belongs and by whom it may be received, Section 5, which states to whom the claim for an allowance is formally made (the Minister) and Section 7 which deals with the method of payment.

Step 8

The new predecessors of Section 7 are: Section 13 which states that the Minister may extend 'the period limited by Section 7 for obtaining payment' and Section 15, which says that stamp duty is not chargeable on the payment of allowances. It has one successor: Section 6, which goes into more detail about the period for which allowances are to accrue.

Step 9

Section 5 has two new predecessors: Section 13 which gives the Minister the power to prescribe the manner in which claims may be made, and the National Insurance Act. We have seen how Section 13, which deals in general with the powers of the Minister, is connected at several different points in the network. If the network had been specified at a level of detail corresponding to subsections or paragraphs, each of these different functions of Section 13 would have been represented by a separate node. The National Insurance Act impinges on the Family Allowances Act at many points, but not all its connections have been recorded to keep the picture simple. Section 16 is the only successor of Section 5; it states that the expenses incurred in carrying out the provisions of the Act shall be paid out of moneys provided by Parliament.

Step 10

The successors to Section 4 are Section 10, stating that rights to a Family Allowance cannot be transferred to creditors in the event of bankruptcy; and Section 8 which relates to the recovery of allowances wrongly paid.

Downloaded from https://academic.oup.com/comjnl/article/22/2/94/428910 by guest on 19 April 2024

Step 11
Section 8 has two predecessors; the Insurance Act, and Section 14, which covers provisions as to legal proceedings for recovering wrongly paid allowances. Its successor is Section 9, which is about penalties for receiving payment wrongfully.

This concludes the example. The remaining sections of the Act, which are somewhat less procedural, have not been included, and some connections which could have been made were omitted in the interests of simplicity.

We have seen that the simple notion of precedence covers a number of different relationships, some derived directly from the text, others more indirectly by consideration of the intention and effect of a piece of legislation. A LEGOL version of an Act of Parliament will need to take account of and reconcile these relationships in an orderly way. The software described in the following section is a tool intended to help the investigations necessary to achieve this end.

2. Software to handle networks

The programs described below are written in FORTRAN for the CDC 6400 machine at the University of London Computer Centre, running under interactive Intercom. For graphical output, a Tektronix 4014 terminal is used, linked to the 6400 via a PDP 11/10 and controlled by routines from the Tektronix PLOT10 subroutine library. The London permanent file manager is used to store and accsss data on disc.

There are two main routines, a preprocessor and a display program, although some subroutines are common to both programs. The preprocessor is used to arrange and store details of networks on a disc file, either by reading new data or by manipulating existing networks. It can be run interactively or in batch mode. The display program accesses details of networks set up by the preprocessor and outputs them on the screen of a Tektronix graphics terminal, according to options specified by the user. Modified versions of some display routines can also be used to plot network diagrams onto microfilm.

The FORTRAN used is non-standard in that it includes special routines for bit and character manipulation and for reading and writing an indexed file. The programs have been written so that storage areas can be re-allocated for each new network processed. This means, for example, that a large but sparsely connected network may occupy the same amount of space as one which is smaller but denser. For this reason, it is not possible to give a figure for the maximum network size with which the programs will deal; it depends upon key length, number of connections, etc. The display program, which must be run interactively, must execute within the store allocated to an Intercom job, about 35K. This will accommodate information about networks containing 500-600 nodes. Bigger networks can be set up using the preprocessor in a batch job with a larger store allocation and partitioned or reduced so as to be suitable for interactive display.

1. *The preprocessor*

This program has four main functions:

1.1 *Set up new networks*

A new network is presented to the program in two related sequential files. The first contains a list of nodes. Associated with each node are three pieces of information, a label (essential), a type and a key (optional). The label is the character string printed out when the network is displayed. Possible meanings for types and keys in the context of LEGOL are mentioned in Section 1.

The second file contains details of connections, consisting of a list of predecessor/successor pairs with optional specific connection labels. The program reports when any two nodes are mutually dependent but does not detect longer cycles. Before details of any network are recorded on disc, the user is given the opportunity of having the details printed out, and either accepting or rejecting it.

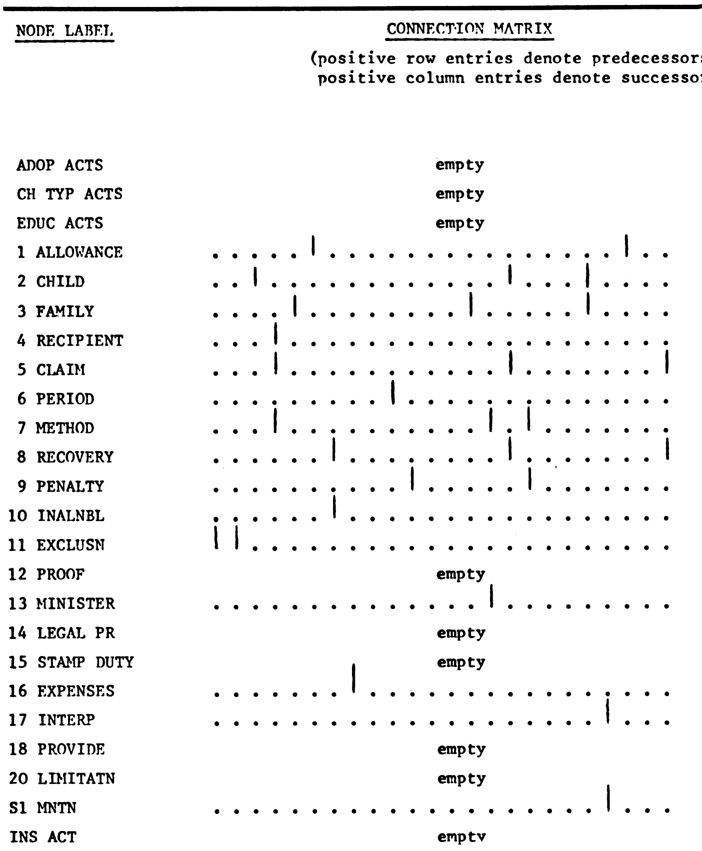

Fig. 4

The network of connections is represented in store as a binary matrix, with empty rows suppressed. This format is economical on the CDC machine, which has a 60-bit word, and efficient for the required Boolean operations. A number of matrices can be held on file for the same set of nodes, each matrix representing a different type of connection. These matrices may be superimposed on one another, using a logical 'OR' operation, so that connections of different types can be displayed on the same diagram. **Fig. 4** shows the node label list and associated matrix for the network in Fig. 3.

1.2 *Partition networks*

The node set for any network can be partitioned, either by node type or by division into connected subgraphs. Each new network so formed may be examined, accepted, or rejected by the user. There is an option to reject automatically any network with fewer than a specified number of nodes. Specific connection labels are retained in the newly created network.

1.3 *Combine nodes*

Nodes within a particular network may be combined by two different methods. The first is based upon key identity down to a specified level. For instance, where a network was originally set up as described in Section 1.2; with nodes identified down to the subparagraph level, the program can reduce it by combining nodes relating to the same section or Act. The program assigns to the composite node the label for the first constituent node it encounters. As the program stores nodes in key order, this should be the one representing the appropriate point in the hierarchy.

The second method of combination is based on lists of nodes (disjoint sets) input by the user. In this case new labels must be

Downloaded from https://academic.oup.com/comjnl/article/22/2/98/428910 by guest on 19 April 2024

assigned to the composite nodes and all key information is discarded.

With both methods, the user is asked for a composite type code to be assigned to the newly created nodes. Specific connection labels are not retained in the newly created network. The program performs a logical 'or' between the rows and columns of the connection matrix corresponding to the nodes combined, and thus several differently labelled connections may be reduced to one. **Fig. 5** shows a network based on cross references as in Fig. 1, but now simplified by combining together all nodes within the same section of the Act.

1.4 *Merge networks*

Two networks based upon the same node set but with different connections may be merged together. If the node sets are not identical, the user is asked whether they should be combined by intersection or union. Intersection or union is performed in relation to node keys, not labels; if the two node sets happen to be labelled differently, the labels retained in the merged network are those for the second network specified to the program by the user.

Once again, specific connection labels are discarded in the resultant network, since the merge is performed by a logical 'or' of corresponding rows of the two connection matrices. However, there is a way of retaining this information, if required. As a by-product of union or intersection, the program produces two compatible networks based on the new node set but the old connection matrices and labels. The user has the option of keeping these intermediate products as well as the merged result, and since they are compatible, they can be used together by the network display program.

Fig. 6 shows a partial diagram produced in this way. It is the result of merging two networks already illustrated, (Figs. 3 and 5). In this example, connections are labelled according to their origin. Arcs labelled 'SS' represent logical connections and those labelled 'R4' represent cross references. A label 'xx' indicates that the connection was common to both original networks.

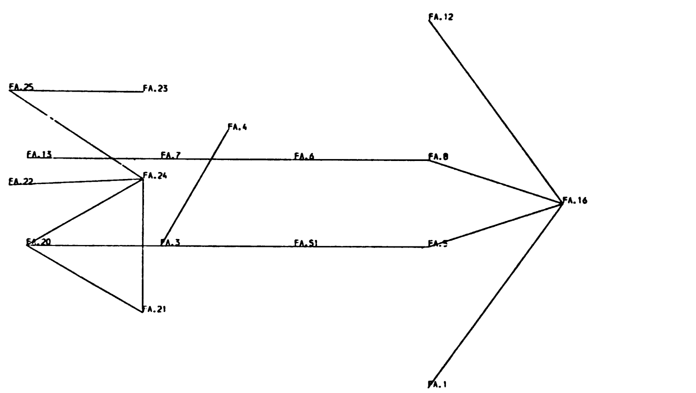

Fig. 5 Family Allowances Act, 1965 network of cross references at section level

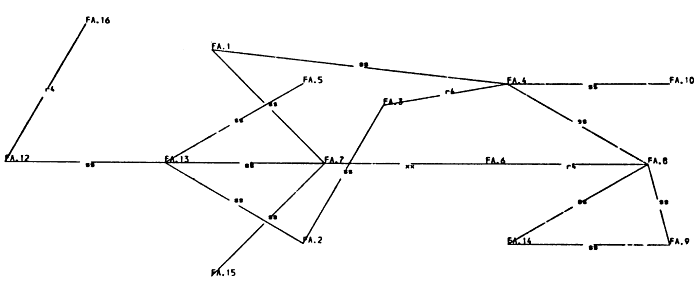

Fig. 6 Family Allowances Act, 1965 partial display of merged networks

2. *The network display program*

This program outputs network diagrams on to a Tektronix screen, under interactive user control. The kind of options available are indicated briefly below.

2.1 *Choice of network(s)*

The network to be displayed is selected by name. Two or more networks may be combined into a single display, provided their node sets are compatible (see paragraph 2.1.4). The user may terminate the display of one network and start another, during the same program execution.

2.2 *Choice of starting node and its position*

Any node in the network can be selected as a starting point. By convention, a node's successors are displayed to the right of that node, and its predecessors to the left. Depending upon whether the search is to go forward or backward through the network, or in both directions, the starting node may be placed on the left, the right, or the centre of the screen. Choice of successors, predecessors or all connections can be made for each subsequent node. (Occasionally the relative positions of nodes in the screen will not correctly reflect their precedence relationship. When this happens, the connection between the nodes is drawn as a broken line.) There are examples in Figs. 2 and 3.

2.3 *Mode of operation*

If 'automatic' mode is selected, the program will attempt to draw the connected graph containing the first node, halting only when all nodes and connections have been displayed or the diagram reaches the edge of the screen. This method is adequate where a subgraph contains only a few nodes. For more complex networks, the user may build up the diagram selectively, step by step, indicating the next part to be extended by 'pointing' with a cross-hair cursor.

2.4 *Shape and size of display*

There are two algorithms to select screen positions for the successors or predecessors of a given node. One method calculates a fixed distance between connected nodes, the other a fixed increment on the x axis only. Each method has its disadvantages, and which is more suitable for any diagram can only be decided by trial and error. By default, the fixed distance used is two inches, on a screen 10 inches by 15. The user has the opportunity of altering this default and so drawing the diagram on a larger or smaller scale.

2.5 *Connection labels*

The nodes of a network are always labelled when a diagram is displayed. The arcs connecting nodes may also be labelled if necessary. Specific connection labels will be used if they were recorded when the network was set up, otherwise the network name will be output as a label when this option is requested.

2.6 *Relocating the diagram*

If a new node or connection to be added to a diagram will not fit on to the screen, the program will print an informative message. The user may either clear the screen and begin again with a new starting point, or redraw the existing diagram in a different position. The cross-hair cursor is used to indicate the extent and direction of the movement. Relocation may cause some nodes in the network to be 'pushed off' the screen. However, the program will remember them and their relative positions so that they will reappear if the diagram is moved back to its original place. Thus it is possible to examine, in parts, a network which is too large to go on the screen as a whole.

The layout of a diagram appearing on the screen can be stored on a stack (a sequential disc file) and retrieved later. So a user

Downloaded from https://academic.oup.com/comjnl/article/22/2/98/488910 by guest on 19 April 2024

building up a network by a series of steps can record his current state, or return to a previous state. The information saved in this way can also be filed, and used to produce microfilm copies of the screen image.

3. *Applications*

The work described in this paper can be considered in two ways. A piece of self-contained software has been produced, intended for use by researchers in any discipline who have networks which they wish to display and explore. It has been used, for instance, to represent patterns of co-occurrence between words in natural language texts (Jones, 1976) and, by contrast, to illustrate social interactions within a small group of people. Whatever the application, the user has the task of identifying the individual elements to be associated and what the connections between them represent. The programs can provide useful manipulations of this data.

However, from the point of view of the LEGOL project, the software is just an aid to the process of examining legal rules in terms of networks. Section 1 discussed the problems of analysing an Act of Parliament into textual elements and connections but such an analysis is only a first step. The textual units must be translated into automatically interpretable rules and the precedence relationships between them converted into instructions about the order in which such rules ought to be applied. The application of rules by the LEGOL 'interpreter' should cause operations to be performed on stored data representing typical cases, e.g. persons and their relationships in order to generate results, e.g. details of families satisfying the conditions specified in the Act and the amount of allowance to which they would be entitled. The relationships of dependency between the different data elements used and created by LEGOL rules may also usefully be represented as a precedence network.

In the context of the LEGOL project then, software to handle networks is intended to assist the difficult process of moving from the textual to the operational level of specification, of producing an automatically interpretable set of rules which retain the essential structure of the legislation from which they were derived. As such it is a very small part of a system which may eventually prove useful to lawyers or parliamentary draftsmen.

Acknowledgements

I should like to thank Alex King of the LSE Computer Services Unit for assistance with the graphics routines.

Downloaded from https://academic.oup.com/comjnl/article/22/2/98/428910 by guest on 19 April 2024

References

Busacker and Saaty (1965). *Finite Graphs and Networks*, McGraw-Hill.

Jones, S. (1976). Word Collocation as a Principle of Classification, *LSE Papers in Informatics* (T21). Paper given at meeting of Classification Society at the Cambridge Language Research Unit, April 1976.

Stamper, R. K. (1976). *The LEGOL Project: A Survey*, IBM UK Scientific Centre Report No. UKSC 0081, Peterlee, 1976.

Stamper, R. K. (1977). The LEGOL 1 Prototype System and Language, *The Computer Journal*, Vol. 20 No. 2, pp. 102-108.

Thornton (1970). *Legislative Drafting*, Butterworths.

Waters, S. J. (1976). CAM01: A Precedence Analyser, *The Computer Journal*, Vol. 19 No. 2, pp. 122-126.

Book reviews

An Introduction to Mathematical Modelling, by Edward A. Bender, April 1978; 256 pages. (*John Wiley*, £11·95)

This is an interesting book, one can learn from it how to formulate and tackle a variety of problems in environmental, biological and social sciences using elementary mathematics and statistics. These fields are different from the traditional physical sciences, where mathematics has been most successfully applied in the past; a different outlook is needed, the problems are not well structured, the solutions are often qualitative rather than quantitative, the problems are practical and do not fit the conventional standard methods. The book deals with these problems by identifying the more important features while ensuring that the ignored details do not invalidate the broad results obtained. Computer methods do not feature prominently but they are used where appropriate.

The first chapter illustrates the main features of model building by considering the problems of population growth and the number of salesmen a firm should employ. Chapter 2 uses arguments based on proportionality, scale and dimensional analysis to solve problems on cost of packaging, shape of racing boats, size of animals and pendulum period. Chapter 3 applies graphical methods to problems on the missile arms race, the number of species on an island, the theory of the firm, stability in economics and group dynamics. Chapter 4 uses optimisation methods in problems concerned with inventory control, geometry of blood vessels, fighting forest fires, bartering economics and caste formation in ants. Chapter 5 applies probability to problems in population studies, sex distribution, the psychology of choice and learning, simulation of a doctor's waiting room, sediments and river networks. Chapter 6, titled Potpourri, studies temperature control in the body of the desert lizard, election procedures, respiration and carbon dioxide elimination. Chapters 7, 8 and 9 introduce differential equations and apply them to problems in the pollution of lakes, driving hazards on road curbs, polymer chains, towing a water skier, stability problems, species interaction and population size, Keynsian economics and the dynamics of car following in heavy traffic. The last chapter studies stochastic models in radioactive decay, facility location and particle size in sediments.

The mathematics used is first year college level, each chapter is supplemented by further exercises, problems and references. There is an appendix on probability and a classification by subjects of the 90-odd problems considered in the book. It would be a pity if, as I suspect, this book does not fit easily the teaching program of mathematical specialists, but it should prove a useful source of material for teaching mathematics to non specialists.

I. M. Khabaza (London)

Discrete Mathematics in Computer Science, by D. F. Stanat and D. F. McAllister, 1977; 401 pages. (*Prentice-Hall*, £12·80)

This book is an attempt to gather together the parts of mathematics that are used in the various branches of computer science. It contains chapters on mathematical models, mathematical reasoning, sets, binary relations, functions, counting and algorithm analysis, infinite sets and algebras. Each chapter introduces the definitions and theorems necessary for discussing the particular topic and where possible attempts have been made to provide solutions to related computing problems using an ALGOL-like programming language. There are a large number of mathematical problems set throughout the text as well as some problems that require the reader to produce solutions to programming problems.

Many students of computer science who are required to do courses on pure mathematics may well find this book will give them some indication of which parts of mathematics are useful tools in their computing studies.

M. Flower (Bristol)