VICTORIA UNIVERSITY OF TECHNOLOGY

AN EXPERT SYSTEM FOR A LEGAL OFFICE

A DISSERTATION SUBMITTED TO

THE FACULTY OF THE SCIENCE

IN CANDIDACY FOR THE DEGREE OF MASTER OF APPLIED SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS

BY

PHILIP MASSEY

FOOTSCRAY

MAY, 1995

Abstract

The subject of this thesis is the use of Information Technology (IT) to assist in the resolution of cases that appear before the Family Law Court in Australia. IT is used to assist in two processes. The first process is the intelligent gathering and preparation of information for Family Law cases. The second process is the modeling of case-decisions by the Family Law Court. The first process is covered in this thesis and the second process is part of the Andrew Stranieri's doctorate research at La Trobe University. Legal Interaction Charts (LIC) are developed to model procedures a Family Law solicitor carries out when gathering information and preparing a case. Sequenced Event Charts (SEC) are developed to implement LICs on a computer.

TABLE OF CONTENTS

Disclaimer

This thesis does not contain any material from any other research publication or any other form of publication other than where reference is indicated in the thesis and the source is fully described in the Bibliography.

Introduction

What are the reasons and purpose for the research in this paper and does it follow the maxim of Professor Fred Hollows, *"no research without service"* ? This research looks at the use of IT in a particular area of law where computers may seem particularly inappropriate, it is an area of particularly human concerns; the area is the breakdown of the marriage between a couple and the resolution of dispute that may occur in their separation. There are over 40,00 couples divorcing each year in Australia. The Family Law Court, where the formalities of marriage dissolution are enacted, is the largest court in Australia. There is a place for IT in this court and in these proceedings.

We are researching the use of computers to model a solicitor's skills or a judge's decision in this area of separation disputes. The computer is not going to replace the solicitor or the judge. The computer tools we are developing are to assist a solicitor to advise people.

The first chapter describes the present use of IT in legal businesses and the particular IT tools that could be used for this project. The second chapter looks at on the ground experience gathering the domain knowledge of a Family Law solicitor and the tools that proved useful in this task. The third chapter describes the tools developed and looks at how the legal expert would use the tools to model legal knowledge and how the software developer would use them to build a useful and powerful expert system. Chapter Four investigates the tools themselves and their capabilities to perform the required tasks. Chapter Five concludes this paper and describes the present state of the research of which this paper is a part.

Chapter 1 - Information Technology and Law

Computers are used in legal offices today but mainly in administrative and accounting services. There are computer models being developed for particularly legal problems and areas of knowledge. We look at where this project fits into the current developments.

Legal Systems and Computers

Simpler divorces on the way

Opening the first national conference of the Family Court of Australia, Chief Justice Nicholson said the court was trying to make itself more accessible to the public in order to reduce litigation as well as reduce legal costs. Justice Nicholson said he recently tried, as an experiment, to fill in a divorce application and was 'bewildered' by its complicated legalese [The Weekend Australian, 3/7/93]. In this article Justice Nicholson describes changes in the Family Court to reflect the changes occurring in society. He outlines a new voluntary mediation program to ***resolve*** family law disputes through early intervention and thereby avoiding litigation, saving time and reducing costs. He proposes information charts be placed on the courts' walls which in simple language explain the court's role and explain procedures. He recommends television cameras in courts.

Using computers to assist people to resolve their disputes is part of the process of change in the organs of our late twentieth century society. The present use of information technology in law is primarily in the processing and preparation of information in legal matters. Using present technology there are many practices that are made simpler, more easily achieved and more clearly understood with the use of computers. The computer application, **Split-Up**, that the research of this paper is contributing towards, was prompted by Mr. Justice Graham Q.C., a judge of the Family Court of Australia. Justice Graham sees the uses of computers to assist solicitors and Family Court officials and those couples in dispute in the FC by providing a couple with expert advice without the physical presence of the expert and by giving a prediction, with some foundation, of the resolution of the case in the FC without the actual consideration of the case by a FC judge.

Family Law

The only criterion necessary for a judge to award a divorce within the Family Law Act of Australia (1975), is irreconcilable breakdown of marriage. Thus litigation is, in general, confined to property and the welfare of the children. Split-Up aims to model the various stages that a person, whose marriage has broken down, may go through in seeking a to reach a settlement with the other partner

to the marriage. The final stage that may be reached in Australia is the division of property by a judge in the FC, the percentage split of the Common Assets Pool..

There are two stages in the design, analysis and implementation of Split-Up, the Legal Expert System (LES) made to assist couple resolve their assets dispute:

Stage 1. modeling the procedures a solicitor goes through with a client seeking to resolve the dispute with the assistance of the Family Law Court, particularly determining what property is included in the Common Pool.

Stage 2. predicting, with justification, the percentage spit-up of the property in the Common Pool as would be decided by a judge of the Family Law Court.

Stage 1 is the subject of this thesis. The Domain Expert (DE) is a solicitor who specializes in Family Law. The LES being built is to provide an interface that enables the DE to build a representation of the procedures involved in the Question and Answer (Q&A) sessions between the DE and the client seeking to resolve their marriage split-up difficulties. This representation must be easily maintained by the DE to keep abreast of continual changes occurring in Family Law and in the Court procedures. The interface the LES presents to the user must quickly and easily takes a client through the steps that will lead to either:

Resolution 1. resolution of the dispute through mediation arriving at an agreement of property division.

Resolution 2. preparation of the case for hearing in the Family Law Court.

Stage 2 is the subject of research by Stranieri A., and Zeleznikow J., at La Trobe University. Stage 2 will use the information gathered in Stage 1.

Stage 1 models the domain knowledge of a solicitor and would be used by a paralegal to ask the client the appropriate questions and prepare the documents that would, in the opinion of the solicitor, be required to reach Resolution 1 or 2 for a client.

Filing for Divorce with the LAC

Split-Up is being developed with the assistance of a solicitor, Renata Alexander, who is working at the **Legal Aid Commission (LAC)** office in the Melbourne.

A client filing for divorce may go to the LAC to apply for Legal Aid. Administration staff assess the initial eligibility of the client. A form is filled out detailing income, disabilities etc.

If the person is eligible for aid then a solicitor assesses the possible outcome, in the solicitor's opinion, of the case in the Family Law Court. A two to three hour interview between the solicitor and client takes place and a summary file is prepared. The solicitor arrives at a plea, a list of demands, that is sent to the client's partner. No court documents are filled out. The partner's solicitor responds with a counter offer. The solicitor has a second meeting to advise the client of the counter offer. If both partners are close to agreement they may then negotiate a settlement and fill-out an official form and notify the court to approve it. If agreement is not possible then the solicitor initiates legal proceedings and fills out **Form 7** and sends this to the Family Law Court. It is the aim of the court to resolve rather than to convict and the evidence required to be presented to the court is kept to the 'pleadings'. Conciliation must be attempted between the partners to resolve the property conflict by seeing a family counselor. If no conciliation is reached then the solicitor prepares the case for court and submits **Form 17** to the Family Court.

Form 17 details the assets that are in contention between the couple. Form 17 also provides the judge with the basis for deciding the percentage split of the assets that are included in the Common Pool for the couple and includes the financial circumstances, income, expenditure, electricity bill etc. of the partners.

Computers in Law

Computer Systems in Legal Offices Today

Use of Computer Systems

	City	Country	Suburbs
Practice management accounting	92%	16%	15%
National on-line services	11%	16%	5%
International on-line	4.5%	2%	2.5%
Litigation support	11.9%	-	3.9%
Access to Govt. databases	23%	51%	21%
Computers not used	9%	12%	3%

Table 1. A survey of computer use (Daniel 1994)

The data in Table 1 is based on two surveys, 1988 and 1993. Several hundred firms (595 and 340) responded to questions about the use of computers for office administration (word-processing, time-keeping, accounting) and their application to professional work (computerized legal information storage and retrieval, litigation management, expert systems).

Computerized legal information storage and retrieval

In 1988 11 per cent of firms subscribed to one or more computerized legal information services. Maynelaw, CLIRS and ESTOPL, in that order of frequency, were cited. LEXIS at that time had been taken up by one per cent of firms. **Expert systems** were claimed by a very few as part of office infrastructure.

Many more legal information databases are now available, but utilization has increased only marginally. CD ROMs (compact disc read-only memory), regarded in 1988 as the cost-effective information technology of the future, are still not widely used.

The most likely users of legal information retrieval systems are librarians, clerks, administrative assistants and students (in that order of frequency).

In 1988 the in-house expert systems, installed in the biggest firms in the UK and USA, were regarded as a promising innovation, a way of ensuring immediate access to the corporate legal wisdom and know-how of the firm. But, there has been little implementation of such expert systems in Australia. A range of software is available, but it is not cheap; the technical skills and knowledge of law applied to a system's set-up are extensive; the amount of data entry involved is expensive. (Daniel 1994)

Computers in the Legal Aid Commission

The LAC office is using computers only for word-processing. The use of a database system for basic client information is planned for 1994. The information flow in the office is in text documents. A high degree of skill is required by office staff, legal secretaries, to extract information from numerous files to prepare legal documents.

Using Intelligent Computers in Law

The IT support tools used in Law , such as case and statute databases, litigation support tools and document creation systems, are the adaptations of the computer programs developed for manipulating numbers and records in business and are non-intelligent.

Is it possible with computers to reproduce legal reasoning and to replicate the way in which lawyers approach a given problem? The La Trobe LES research group believe "it is at present unrealistic to build general purpose legal reasoning systems. Instead projects aim to build intelligent legal tools which do not seek to replicate human reasoning, but rather assist lawyers to improve their performance. To do so we need to reason with statutes, heuristic knowledge and precedents" [Vossos, 1993].

First generation legal expert systems were based on production rules. They took the form of 'IF condition THEN action'. Such systems fail to take into account problems with natural language ambiguity, open texture, and do not address the issue of precedent within common law. There are a number of techniques that have been used to enhance legal knowledge systems, by reasoning with experience. These include **Case Based Reasoning (CBR), Neural Nets (NN)** and extending **Rule Based Reasoning (RBR)** systems to interface to CBR or NN. CBR provides flexibility and allows for machine learning and mimic many features of legal reasoning. NNs are currently incapable of providing the reasoning behind their conclusion, the "why" of the argument. Lawyers are very much concerned with explaining why the case or form of argument is relevant [Warner , 1990].

Case Based Reasoning

A LES that infers from cases must have a database of cases to work on. The cases must be classified into a format by schema that allow storage and retrieval from the database in a way that is appropriate to the *current context*. A *knowledgebase* rather than a database with the operations of *classification* and *recognition* rather than the storage and retrieval of the records with clearly defined types and sizes.

One method being explored is to recognizes cases in a very early stage of problem solving. The aim of the method is to find a case in the case base which is sufficiently similar to the current case to be able to function as a model in problem solving. Since problem solving also implies gathering information, i.e. posing questions, the recognition mechanism should be able to suggest a case even before all information is available.

The interpretation of raw data is inherently subjective, and indeed, such interpretation is a major component of the expertise of the system. Taken to its logical extreme literal accuracy in the representation of case law can only be achieved by full-text incorporation in the knowledge base of all the operative facts - effectively transforming the system at hand from an expert system to a database. Any other approach is always open to a charge of misrepresentation [Edwards, 1992].

Rule Based Reasoning

A LES that uses rule based reasoning requires that the laws being modeled are able to be fully interpreted into a technical format. This requires a formal representation of the contents of the law. The possibility of representing formally the content of laws solves in part and supplies the interaction between Information Technology and legal systems. Formal representation of content has several objectives:

- to render possible the direct use of content by means of computers.

- to reduce the ambiguity of the interpretation of the same law by a number of different participants

- to support interpretations that are uniform.

If a formal representation of an area of law is possible then it would be possible to use a formal language developed in some other area of knowledge. The analysis of laws by means of **Petri Nets** is one such formal language that is being explored.

The principal difficulty is the nature of the content of the law that has to be represented formally. "The reason why laws specify chiefly the desired behaviour of systems in a procedural manner (even if the objective, that is the desired consequence of the activation these processes, are often indicated) is perhaps connected to the impossibility of tying behaviour uniquely to objectives." [Antoni, 1982]

The Appropriate LES for Family Law

There are many differing areas of law with many types of information and involving a multitude of processes. The form of logic and classification that a LES is to use will depend on the area of law it is concerned with. The researchers working on Split-Up have found the placement of the area of law being modeled on a dichotomy representing the extremes of the content and the processes of the law that the LES models, to be a useful technique to test the appropriateness of the methods of the LES to that area law. The dichotomy is defined in such a way as to serves as a means of defining the appropriateness of the computer model to the legal model.

Two dichotomies are drawn out:

- The reasoning and procedures followed are Highly Structured at one extreme and Flatly Procedural at the other.

- The evidence and language used is Open Textured at one extreme and Closely Defined at the other.

Laws that are very technical and refer to specific things, such as legislation on trading goods, imports and exports, have Highly Structured rules and Closely Defined terms. Laws concerning human emotions and have broad aims, such as in the children's court, have Flatly Procedural rules and Open Textured terms.

Stage 1 of Split-Up concerns the Common Pool Determination of the assets in a marriage. The questions asked concern date of purchase, category of asset, use of asset, etc.

Stage 2 of Split-Up concerns the percentage split of the Common Pool by a judge and concerns such matters as the contributions (in financial and in home-duties terms) to a marriage, future financial needs of a person, projected superannuation pay-outs etc.

- Stage 1 terms are more closely defined than Stage 2 terms.
- Stage 2 involves more complex considerations of interconnected elements.

In figure 1 the types of LES, Stage 1 and Stage 2 of Split-Up are placed on these two dichotomies.

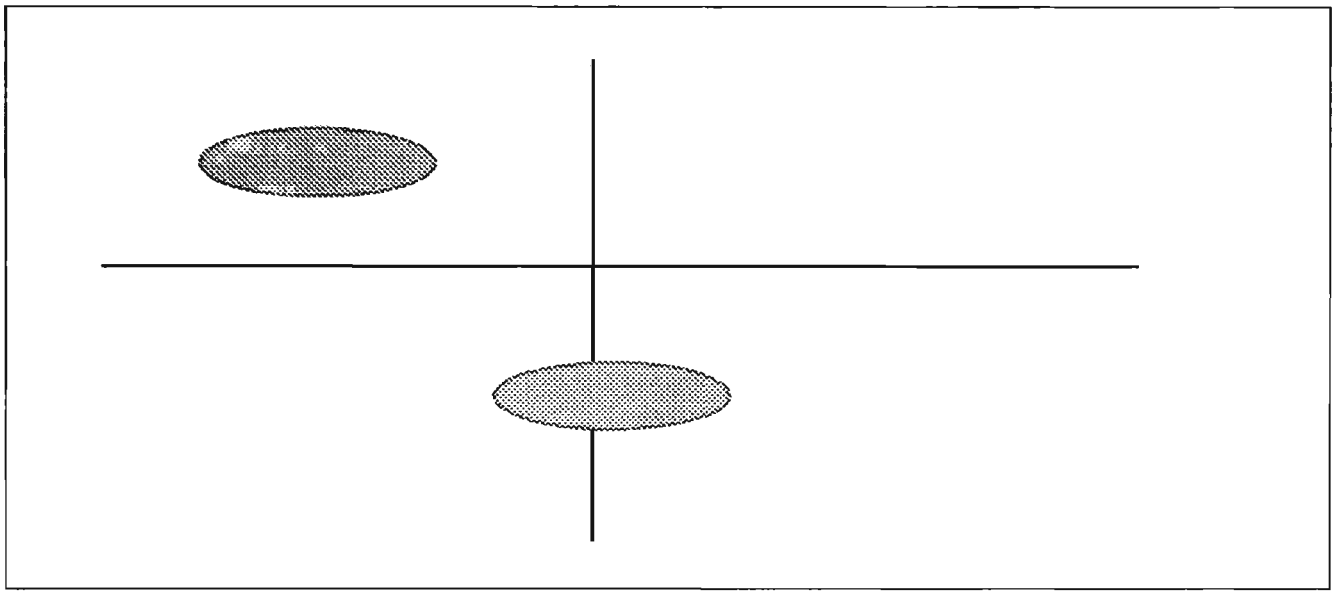


Figure 1. Dichotomies of areas of law.

Stage 1 has elements of CBR, where a DE reasoning on the possible outcome of a case will look to previous similar cases. The judgments of the Family Law Court are largely based on heuristics. However as most cases are very similar, having close to the same ingredients, it is possible to construct rules from the heuristics employed in the court.

Split-Up Stage 1 - RBR, CBR and Heuristics

The determination of Common Pool membership for most assets can be represented by rules because there is little contextual knowledge required and a limited number of alternatives for each type of asset.

However, there are a limited number of circumstances within the common pool determination that are difficult to model with rules. For example, there is a heuristic, not specifically mentioned by the Act but certainly in use by the Court that assigns assets gifted to one party into the common pool. However

the Court will not assign the asset into the common pool if the donor clearly intended the gift to benefit only one partner but not both. The determination of intention to benefit solely one partner is difficult to represent using rules as the donor's intention is often dependent on the nature of the gift. A rare stamp gifted to a stamp collection husband is likely to have been gifted to benefit the husband alone. However, if the stamp is mounted to be hung on dining room wall the intention to benefit only one party is not clear. This situation is where the rules have 'run out'.

The retrieval of similar cases using a CBR approach is useful for end users. Similar past case indicate the likely decision for the present case and provide a rationale for a decision.

The implementation of Stage 1 must allow rules to be defined, the fulfillment of premises results in and/or triggers conclusions, but Stage 1 must also allow the builder to nominate exceptions and define the appropriate actions for exceptions.

Chapter 2 - Tools for a Legal Expert System

People today do not see any use for computers in personal, emotional problems. The professional people and the people seeking to negotiate a settlement to a marriage breakdown do not look to computers to resolve the human conflict in the dissolution of the relationship and the nitty-gritty of how to share children and property. The way to solve these problems is seen to involve processes of mediation and assistance by human beings, by people with experience and wisdom. Computers are not perceived as being of any use in this field of personal problems. The LES we are designing for the LAC must be very, very user friendly. The processes it performs should be transparent. The DE must feel very comfortable with the processes the LES is performing and must be confident that he or she knows what the LES is *"doing"*, i.e. the operations of the LES. In this project as we gathered the DE's skills and then represented these procedures in a model to be implemented on computer we have designed tools that enable the Knowledge Acquisition, Representation and Maintenance processes to be done as one. In this chapter we explore a computer interface that will allow the DE to build the knowledge base by directly representing the expert knowledge in charts.

Design and Implementation as One

Knowledge Acquisition, Representation, Manipulation and Maintenance

A large part of building any expert system is building the knowledge base that will be used by the inferencing process. The domain knowledge of the DE is the ES knowledge base.

Extracting knowledge from the Domain Expert represents a number of problems:

- it is heterogeneous.
- it is complicated and imprecise.
- it is qualitative rather than quantitative.
- much of the knowledge is context-dependent.
- much of the procedures used to acquire knowledge are dependent on the individual DE and may change with changes in personnel.

The acquisition process of the knowledge base may benefit from a more direct participation by the DE, the 'hand-crafted' approach with the DE acting as the designer and maintainer of the ES. This requires an *external* knowledge interface providing the DE with tools to manipulate the domain-

familiar abstractions, terms and objects and to map the model so built to an *internal*, computational representation which is hidden from the user. This enables non-programming personnel to create and refine rules, without the mediation of the knowledge engineer [Edmonds, 1993].

We are able, in Split-Up Stage 1, to build a computer model that enables the DE to directly represent the procedures and specifications required to complete the interview, assessment and advice process for the more general Family Law cases. The DE uses the LES as a computer charting package to build flow charts modeling the procedures of interviews and document preparation. The charts so built direct computer procedures to load and "run" appropriate charts that are the interview procedures. The procedures display the appropriates sequences of questions that follow in response to answers given and so arrive at concluding actions.

The hands on building of the computer ES Knowledge Base as a series of charts by the DE enables the DE to complete the procedures of knowledge acquisition and representation required to build the ES for this domain. The expert has direct control over the way the ES will model procedures and so the DE can manipulate and maintain the ES.

The Knowledge Base as Charts

The interface of an ES should vary according to the type of user. A DE wants to see a different aspect of the system than does a clerk and will be prepared to use a technical approach and a formal language to handle the complexity that design may require.

For the end-user information presented in natural language is often the most accessible. It is convenient to the user to be able to express his or her request in terms that he or she may understand. The chief challenge of LES design is to make the front end accept queries posed in something close to natural language. In relational Data Base Management Systems the formats for displaying data are usually rigid row and columns. In Knowledge Based systems the data or knowledge is more complex and deserves strategies that go beyond the tabular.

The graphical representation of the structure of systems is used as a formal analytic tool in many areas of knowledge today. It provides a means to encode information at various levels in an accessible fashion. The increasing sophistication of graphical knowledge representation tools available in the computers Windows environment proved to be a central tool in gathering and modeling the domain knowledge for Split-Up.

Flow charts

The basic model of flow charts is a graphical representation of nodes and links with text attached. The symbol for a place, usually some box shape, may also classify the place and so has meaning. Similarly

the symbol for an arc between places, usually an arrow, may classify the connection in some way, mostly it signifies the direction to follow to go from one place to the next and thus to "read" the chart.

Firstly the computer members of the Slit-Up design team starting using flow-charts to model the legal procedures being detailed by the legal member of the team. The flow-charts were easily built and maintained using MS-Windows packages that are increasing in the sophistication of structural representation possible with features such as links between graphs. These flow-chart models proved successful in not only in capturing the information for computer design but also in assisting the legal member to represent knowledge for legal purposes independent of computer modeling. This lead the members to design a computer model that followed directly from the flow-chart model for by retaining the flow-chart appearance of the legal knowledge in computer representation of this knowledge we also retain the ability of the legal member to build and maintain the computer model, i.e. the DE can build and maintain the computer model.

Instead of using a flow-chart as a top down approach and as an abstract description of operations that may require many lines of code to be implement, we use the flow-chart as a direct tool to be manipulated by DE to build a chart that represents the procedures required to be performed by the LES. Where the greater the complexity of the procedures required for an operation then the more levels of charts there will be required to arrive at an action. The closest term to describe this type of chart in current chart theory is ***a discrepancy net***. We named the charts used by the DE to represents the interview and assessment process **Legal Interaction Charts (LIC)** and named the charts used in the computer implementation of a LIC **Sequenced Event Charts (SEC)**.

Stage 1 of Split-Up is to assist a paralegal to question a client and so gather information that will assist in resolving the client's conflict with their previous partner.

This procedure was modeled by Andrew Stranieri, Split-Up's primary designer, and Renata Alexander, a family law expert with the Legal Aid Commission of Victoria. A question and answer process (Q&A) was modeled as a set of procedures governed by heuristics built up in this domain by the direct experience of the DE. Each process started by focusing on a particular topic that may contribute to the resolution and followed this topic through to an appropriate conclusion for a Q&A session.

A graphical representation of this elicitation of a Q&A provides a tool for the representation of the heuristics in this domain in a way that is acceptable to DE and in a way that is directly useful to the ES programmer. Other lawyers commented on the usefulness of the flow-chart model to follow the strong procedural content of a great deal of law. Hybrid flow-chart are used by experts in many fields to elicit and represent essentially the flow of events in the system being modeled. In a Q&A session the flow of events are lines of questions and answers leading to conclusions. The conclusion need not

be deducible from nor even supported by the preceding statements. The conclusion is formed, defined and placed by the DE.

Legal Interaction Charts

The particular type of flow-chart developed for our legal Q&A sessions is called a **Legal Interaction Chart** (LIC). A LIC serves to model the interaction between the client and the expert but do not aim to develop a model of the reasoning process alone. Thus, a LIC represents the combination of heuristic knowledge and procedural knowledge and implicitly represents the logical structure of the reasoning, according to the DE.

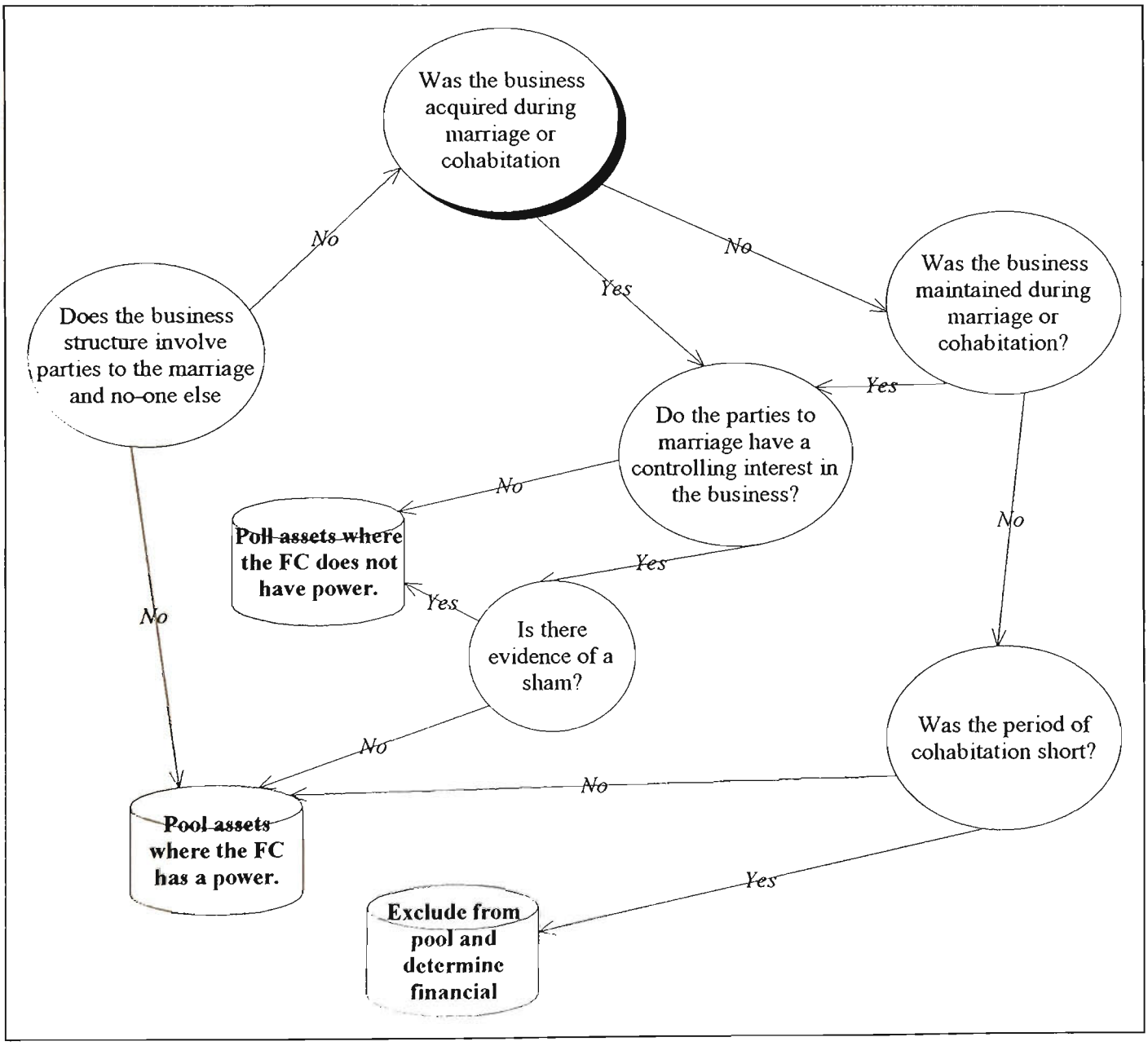


Figure 2. Legal Interaction Chart for the common pool determination of a business.

LIC features

An LIC consists of nodes and arcs:

- there is one entry point to a LIC.
- there is one or more terminal.
- *elliptic nodes* have the text of the question asked at this stage in a Q&A session.
- *arcs* have the text of the answer to the 'from' node question that results in 'going to' the next node.
- *terminal nodes* have varying shapes to represent varying conclusions.
- the number of arcs from a node is not limited.
- more than one arc can go to a node thus a number of paths **'join into'** one.
- *shaded nodes* indicates a **'not sure'** answer which is resolved to the appropriate answer by a set of questions, for the "business acquired during" question in figure 2 the 'not sure' is resolved in figure 3.
- a LIC is basically understood as a **directed acyclic graph**, having the intention of working as a **discrepancy tree**, however cyclic paths are not excluded but this would only have meaning if some variables were manipulated thus changing the meaning of the text at a node.

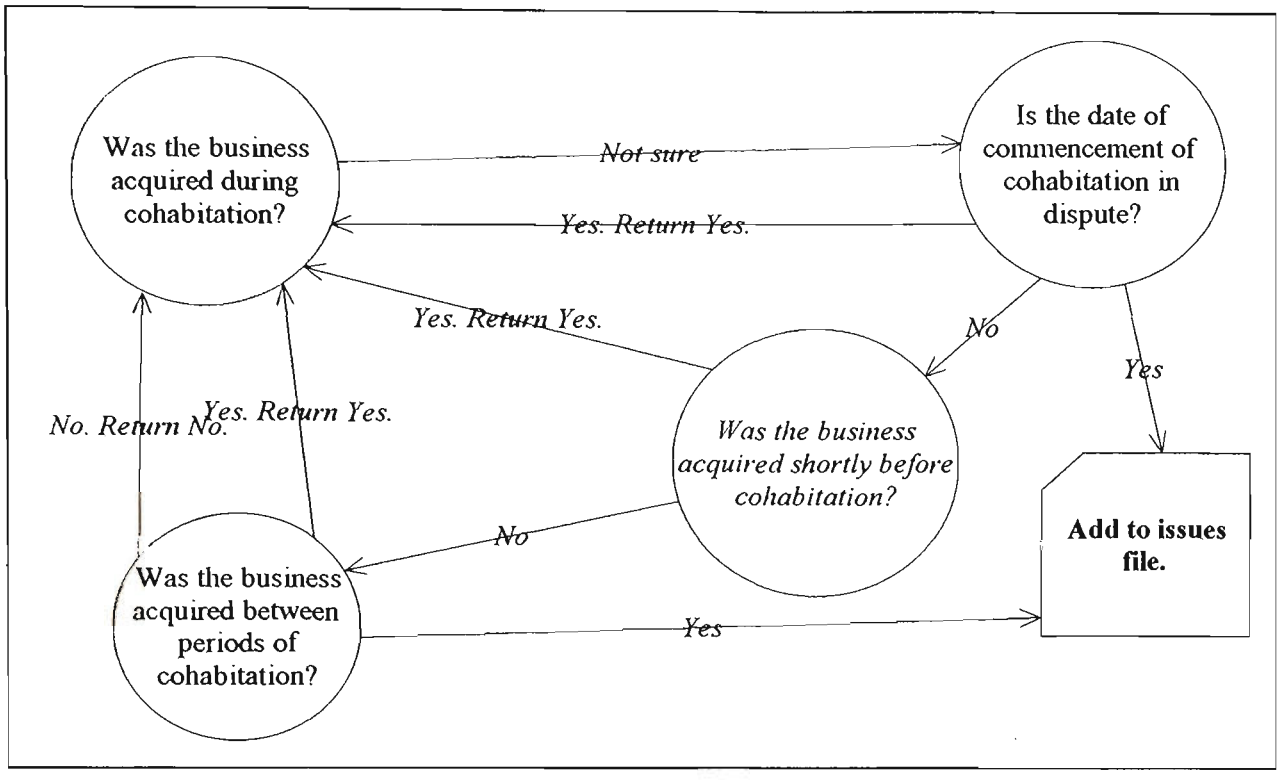


Figure 3. Resolving 'not sure' to the appropriate answer for the "acquired during" question.

The major Q&As in Spit-Up concern the common pool determination of the assets of the couple. This provides each Q&A with a definite starting point and a definite set of conclusions for an asset. If an asset exists it is assigned some 'common pool status', e.g. included, excluded, include where court has jurisdiction, etc.. Other interim conclusions must be allowed as part of the information gathering process, where information is incomplete or uncertain for example, but these are more procedural considerations.

LICs for all types of assets were derived in thirteen, sixty minute sessions of concentrated discussion and process modeling between Andrew Stranieri and Renata Alexander; knowledge engineer and legal expert. During this acquisition phase, Andrew would draft a basic LIC from a first discussion with Renata of how she interacts with her clients for a *particular matter* of the Family Law Court. In following discussions both knowledge engineer and legal expert would modify the LIC fully represent the possible processes that Renata carry out to complete the client's case. Fifty-one Charts containing a total of 230 nodes were elicited.

The LIC are the means of communication between the DE and the ES builder. The knowledge engineer would be required to extract the logic from the LICs and build the rules for a rule based LES. The logical structure of the reasoning process used by the DE is not made explicitly evident by the LIC nor is it implicitly contained. To logically represent the reasoning processes used by the DE in following these paths of Q&A would require building sets of rules with the logical features of completeness, coherence, compatibility, etc..

LIC as the LES Knowledge Base

When the LICs proved adequate to the task of knowledge representation for a LES we investigated the possibility of a computer environment in which the DE can directly build LICs which then serve as the knowledge base of the LES.

The computer environment for a LES in which the DE can directly build the knowledge base as LICs requires two parts, a part in which the non-programmer can construct models and a part in which the programmer can enable the first part. We use the metaphors **compartments** and **containers** to describe these two parts of the LES. A compartment is the part of the ES that the DE and user sees. It uses the computer. A container is designed by the computer programmer to be specific to a compartment. It makes computer tools available to the compartment and those tools are to be the most appropriate for that compartment and to be easily comprehended and used by the builder of the compartment. Into the compartment the domain expert can place all the information about an object that is relevant to reaching a conclusion for the object. Into the containers the computer programmer places all the variables and methods required for that object to exist in the compartment.

Compartments

The domain expert directly enters text describing behaviour that will reach this or that conclusion and joins one description of behaviour to another to make up a contextually sensible series of sentences. The joined descriptions should "flow" and a ***sequence*** of descriptions should make sense when read and should justify the conclusion reached. A well filled compartment will contain the text to provide an adequate justification to "Why", "Why not", and "How" the conclusion was reached. The encoding required to make the rule base is transparent and intuitively understood. It is simply the underlying structure of one sequence follows another and so on to some conclusion. This syntax is in fact an aid to understanding the structure of a Q&A session and is graphically represented as a flow chart. This is easily understood by the domain expert and can be used not only to understand the syntax of the procedure but also as an interface to build and edit a Q&A model on a computer. There is no inference mechanism and a conclusion is reached from a sequence not by inference but because the designer of the sequence has designed it so.

Containers

The simple functional capability of the compartments does not offer the designer and user of this proposed system much incentive to use it. It is in the power of the containers to provide the designer and user with access to the resources of the computer environment that this model has potential. The container paradigm to the design of building models to utilize computer resources is one of the cornerstones of **Object Oriented** modeling (see below). The model of a container performing

calculations using a single tape is at the heart of computer theory (Church's Theorem). The potential for a model of a container performing operations from a sequence of inputs is well worth investigating.

Consider that the container is to implement all the computer related tasks to allow the user to use the contents of a compartment. The text of question will be displayed and the user can then answer. The text of the appropriate next question is displayed, the user answers and so on to the final display of the conclusion. A record of questions and answers should be saved. Information relevant to other compartments should be stored. Some form of explanation to the conclusion reached should be available to the user.

How much information is there required to be in the structures to direct and control a container that is displaying them as questions and to then display the next question appropriate to the user's answer? This information needs to be "understood" by a container. The only information in a **sequence**, as we will now call the structure carrying the question, that a container needs to understand is the identity of the sequence and where the text is in the sequence. If there is an identity number unique to a sequence then the container can use this to retrieve the sequence and a "string" variable within the sequence that holds the text to be displayed as a question. A list of string variables in the sequence can then be displayed as possible answers to the question. The user selects on of the answers, this is understood by the container as the index position of the answer in the list. The container then concatenates this number onto the identity of the present sequence and then uses this new identity number to retrieve the next sequence. If this is how the identifiers are assigned when the sequences of a compartment are being built then the syntax of the identifiers will be valid.

The container does not "understand" the text in the compartment and so is unaffected by the text. If the container has functions and variables built into it to enable it to use the potential of the computer environment then there must be other layer(s) to a sequence so that it can communicate with and instruct the object in the use of these features. These extra layers, particularly those that are computer dependent, may require skills that are outside the compartment builders skills. This difficulty may be alleviated by careful design of the object by the programmer so that it assist the builder of its compartment by the domain expert. The object should handle the technical considerations at a level that the domain expert can understand.

Sequenced Event Charts

A Sequenced Event Chart (SEC) is a LIC directly modeled on the computer.

The goal of the design of the SEC system is :

- If the DE can model procedures with a flow chart then the DE can build the LES to enact those procedures without any assistance from the software engineer.

The SECs are the Compartments, the structures modeling the information in a Q&A session, and the term ***ChartObjects*** is used to describe the Containers, the processors, with associated processes, that carry the information in these charts or nets.

Logic in a SEC.

The designer of a SEC does not have to ensure that the conclusions at the end of paths are logically exclusive. there is not a rigorous logical structure to a net. However as much as a SEC makes sense so it capture some structure in the information it models. The chart structure enables some basic inferencing by chaining forward or backward along the chains of events to and from places, such as;

- what are the possible paths to a conclusion?
- what conclusions cannot be reached from this place?
- what are key junctions to determining which conclusion is reached?

Some **explanatory** function may be possible with SECs. There are three distinct types of explanation in expert systems.

Type 1. Matching data with Local Goals involves examining appropriate segments of runtime behaviour of the system. This type of explanation can best be made by maintaining a trace of the rules fired, or some other representation of the path that inferencing takes. The path to a conclusion in a SEC can be *played back* as an explanation of **how** this conclusion was reached.

Type 2. Knowledge known as Justification. Justification of the conclusions involves recourse to a deeper level of understanding of the purpose or the intent underlying rules. This is beyond the SEC.

Type 3. Knowledge as Understanding. Comprehension of the complete system involving cognitive concepts such as balance and harmony. A SEC may assist human understanding of a system by graphical representation of knowledge.

Chapter 3 - Using the Tools to build and run a LES

The wide use of flow-charts to model procedures and communicate information indicates a wide acceptance and understanding of the graphical representation of procedures. However there are good and bad flow-charts and there are skills to be learnt in designing and building flow-charts. So there will be methods and skills to the design of SECs to make a useful LES.

Designing a SEC

Categories and Relations

The SEC approach to representing the Expert's knowledge does not require the encoding of information in machine understandable symbols. The SEC system provides an open mechanism for the builders of an Expert System to model the Q&A process. The structure of the Q&A session should be clear as the *textual surface* of the SEC. What the questions are about, how questions are related and the goal of questions should be made clear. The grouping and categorization of information into subjects, into topics, under headings etceteras and relations between these categories is a powerful tool to achieve understanding.

The Compartment concept fundamentally groups like things. The tree structure of the SEC focuses the design of a chart to make sense of relations between Compartments. A design that continues to branch at each juncture will result in a *discretionary tree*, drawing finer and finer distinctions and categorization of objects. However a SEC may also have *joining* paths to model inclusive categories.

Most of the Q&A sessions to be modeled in Part A of the Split-Up project are to determine the Common Pool Status of assets in the marriage.

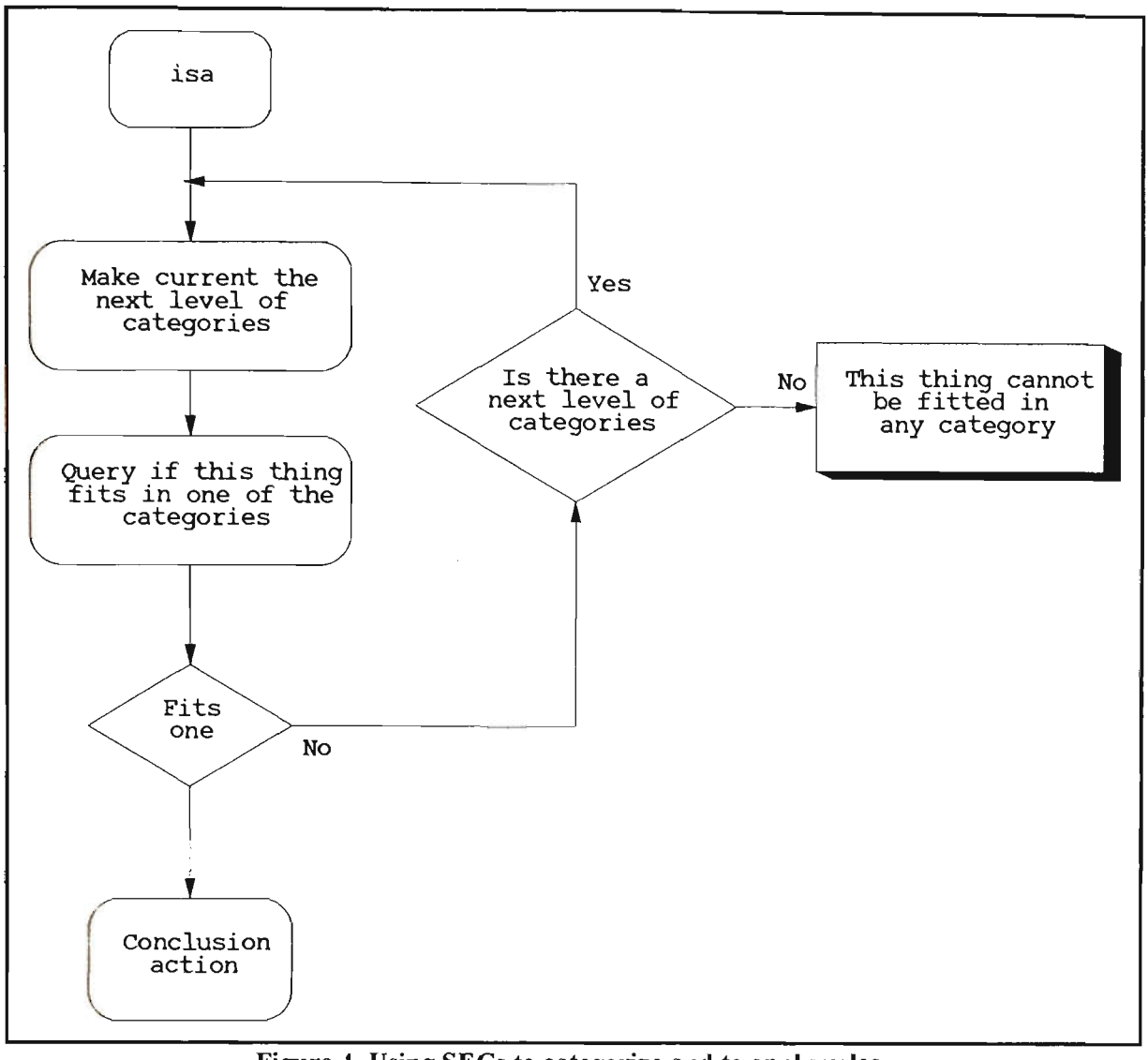


Figure 4. Using SECs to categorize and to apply rules.

Broad groupings applying to all cases

There are broad groupings, not exclusive, that may apply to all or some of the things we are to examine. Each of these grouping has a set of rules that may be applied to anything that fits into it and with an outcome to the rules that is appropriate to anything that fits into a grouping, irrespective of what further definitions may be made to that thing.

In Split-Up the questions asked and the action taken are the same if the item is a gift irrespective of any further classification of the item as jewelry, real estate, business etc. The same applies to things that can be classified as an inheritance or if the item is in dispute.

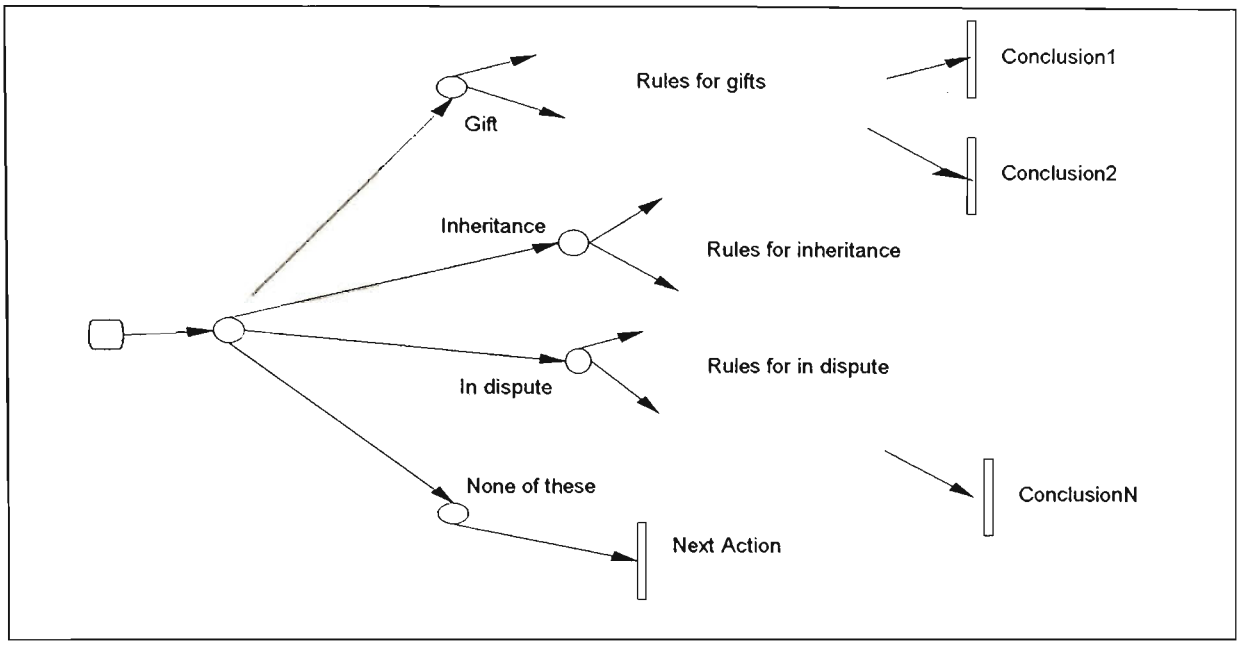


Figure 5. Broad Grouping of Assets to Gift, Inheritance, In dispute or to next level

An Item In A Broad Grouping

There must be a set of rules that define the properties of an item for it to be classified into a group, for example is it a gift. A broad grouping by its very nature will have a very few explicit rules that require simple and clearly understood information. The information can be gathered directly from those concerned by direct questioning. If there is uncertainty or disagreement then an explicit set of procedures will have to be invoked.

In a SEC the choice *'not sure'* provides a set of questions specific to defining what choice could be correct.

Rules for exclusive and defining categories

Some things can be categorized by their nature, e.g. a chair is a piece of furniture, a shop is a business, a ring is an item of jewelry.

There are *common properties* within each category that may further define and group items in the category. The properties that occur regularly may be outlined in rules, legislation, or be taken as read from their occurrence in many prior cases.

IF the thing has these *common properties* THEN this is the usual result.

There are properties at *variance* to the common properties that require more specific reasoning.

There are properties *over and above* the common properties that require more specific reasoning.

exclusive properties:

A the house was purchased before the marriage

B the house was purchased after the marriage

IF *A* **THEN ....**

IF *B* **THEN ....**

compatible properties:

A the house was maintained during the marriage

B the house was improved during the marriage

C the house was improved after marriage with both parties finances

IF *A or B or C* **THEN** *D*..

conjunctive properties:

A the house was purchased before the marriage

B the house was purchased with both couples savings

IF *A* and *B* **THEN** *C*

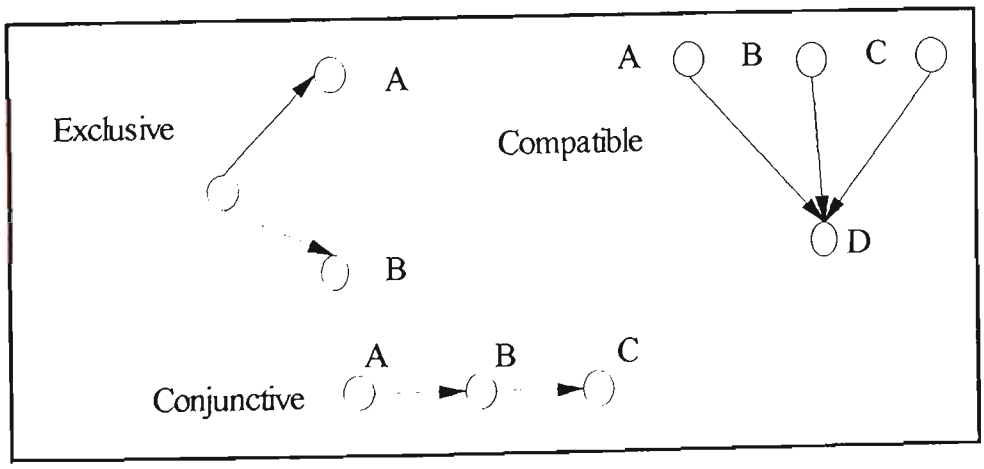


Figure 6. Properties to categorize an asset.

The three types of properties, disjunction, conjunction and compatibility are fully contained in the logic of a SEC. The formal description and computer implementation of this logic is looked at in the appendix **Logic Programming.** The use of a chart as part of a formal language to model these properties and their implications as *events* and *places* is looked at in appendix **Petri Net and SEC.**

SEC as Tools in the Computer Environment

SEC functions

A system running on a computer should be able to use the computer to perform computer-tasks, such calculation and retrieval of information within in the computer system. In the SEC system this task is assigned to ChartObjects.

ChartObjects

The information that must be computer dependent, such as dynamic data, names and personal details etc., is placed within the context of a SEC by the use of the ChartObject. All operations performed in a SEC are performed by ChartObjects, mostly by the ChartObject that is 'current' to the SEC.

Building

The basic building operations required for a user to build a SEC can be described in a

few short step as follows. Further refinements such as joining and 'not sure' features are detailed in appendix *Petri Net and SEC.*

STEP 1. For the current Sequence

A sequence is one of:

- FUNCTION:

  1. A list of the available functions for this object is displayed.

  3. Select the function required.

  4. Select the function parameters from the appropriate list of objects and fields displayed.

  5. Go to the next step.

- TERMINAL:

1. A list of the available SEC files is displayed.

2. Select the SEC that will run when this SEC finishes, e.g. the main menu SEC.

3. This path of the SEC has ended.

- DEFAULT

  Go to the next step.

STEP 2. Enter text for a sentence in the form of BEFORE CHOICES and AFTER.

STEP 3. The system will go to STEP 1 for each of the choices entered.

The System to build a SEC

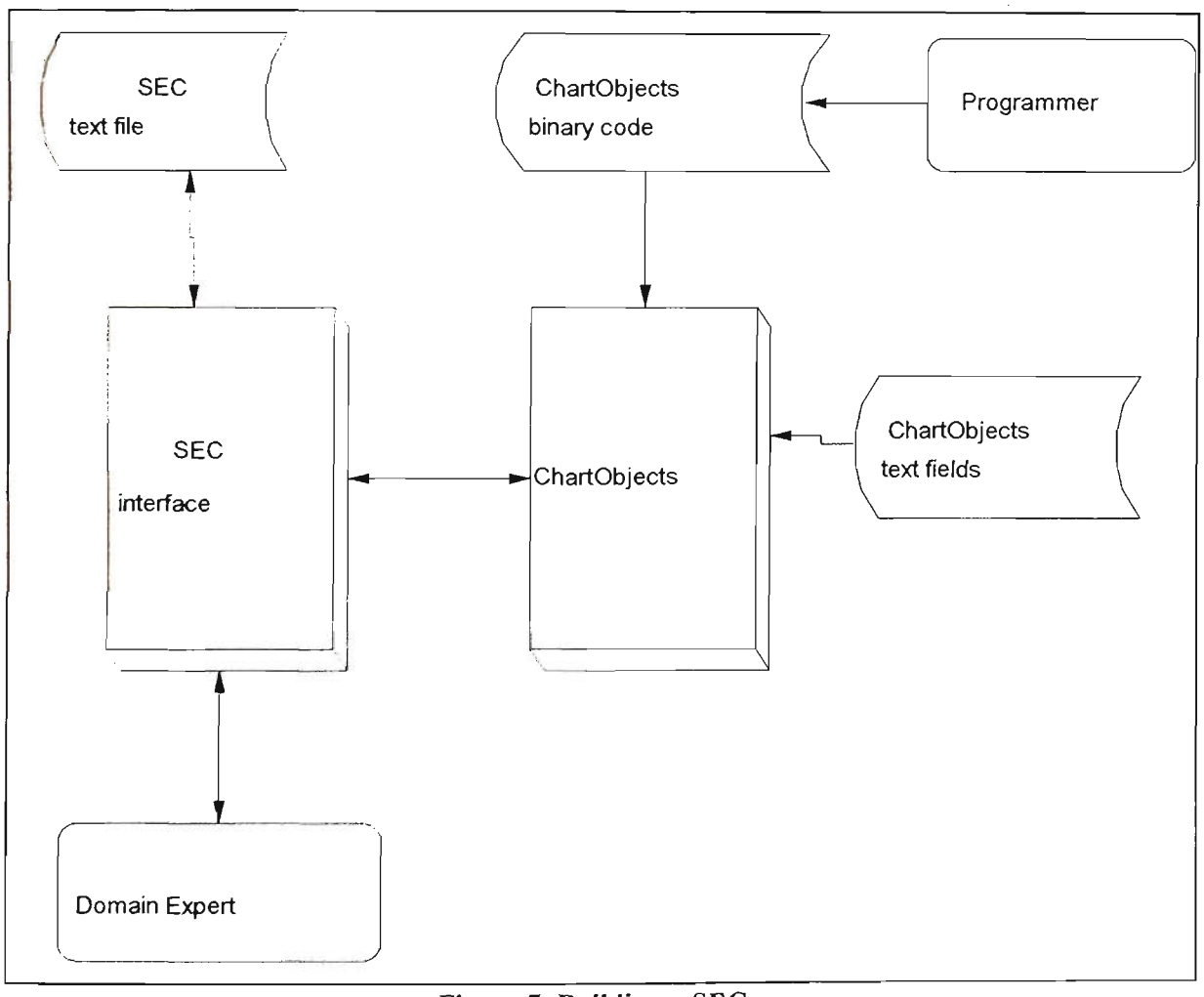


Figure 7. Building a SEC.

Editing

Any Sequence can be edited. The rest of the chart is updated by re-sequencing, deleting paths and maintaining joins as appropriate. A graphical interface to editing is essential with point and click access to any Sequence on the SEC.

Running

STEP 1. Retrieve the next Sequence for this path.

A Sequence is one of:

- FUNCTION:

  1. Perform the function which returns a number corresponding to the index of the appropriate choice in CHOICES, e.g. i.

  2. Display the Sequence BEFORE CHOICES[i] AFTER.

  3. Go to the STEP 1

- TERMINAL:

  1. save the path for this SEC to this SEC's object

  2. close this SEC

  3. get the SEC file name that a terminal Sequence must have

  4. assign this SEC file name as the next SEC to run

  5 RETURN

- ELSE go to next step

STEP 2. Display the Sequence's BEFORE CHOICES AFTER.

STEP 3. User selects choice.

STEP 4. Append the index number of the choice onto the current path.

STEP 5. Go to STEP 1.

A System to run a SEC

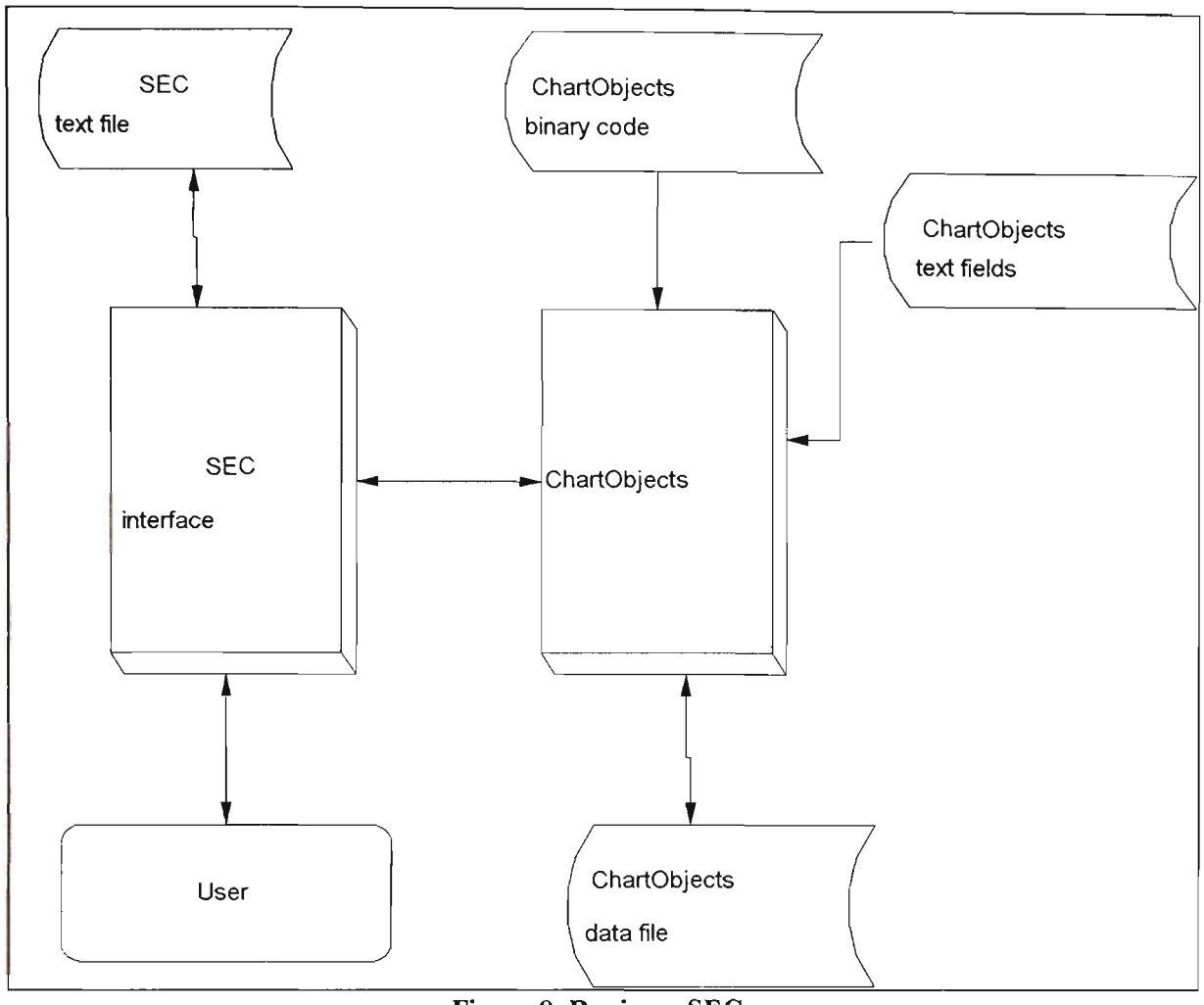


Figure 8. Runing a SEC.

Chapter 4 - Tools or Toys

The aim of this research may have been fulfilled. A computer package has been designed that allows the DE, with little computer skill, to build and maintain the ES. We have yet to trial these tools to know how much time, effort and skill will be required by the DE to build SECs. We must find out if SECs have some structure that models some structure(s) found in the interview process? Are they only a *spaghetti* of heuristics and so are of no use as a tool for understanding the interview process and are a nightmare for a person to design and to maintain? Chapter 4 outlines a justification for the claim that SECs do model structures that are relevant to the interview process. We also consider the limitations that the SEC format places on the design of charts. The performance of the SEC computer model on a computer is outlined.

Using SEC for Human Activities

We have concentrated on the Q&A sessions, between a solicitor and client to determine the common pool status of assets from the marriage. The purpose of these sessions is to gather the information required to make an offer to the client's partner or to assess whether to accept the partners present offer or to proceed to the next stage in the process outlined by the Family Court. The circumstances that define the information to be gathered for each case are specific to each client and each client has a unique set of circumstances of his or her marriage. However in their humanity and in their society there are many common features. This commonality of people's circumstances allows us to look to science and natural law for tools and methods to use in modeling Q&A sessions.

Natural Law

'No one would do science unless he or she believed two things:

a) the sequence of events that we observe in the external world is not entirely whimsical or capricious

b) the relations that govern this sequence of events is, at least in part, apprehensible to the human mind.' [Rosen, 1987]

Natural Law is a set of "fundamental truths" about the world we live, i.e. truthful representations of things in the world made in terms of human cognition. These fundamental truths can be used as building blocks to build a model that will "truthfully" replicate something in the real world.

In Natural Law

- causality is the name given to the relations between *events*

- relations between events can be mirrored in corresponding relations between *propositions* describing these events

If the propositions describing the events can be identified, described and coded then they can be used to build a rule base. If the relations between the propositions can be likewise encoded then an inference engine can be built to use the rule base. The rule base and inference engine will output new propositions from a set of input proposition. The system has an "Artificial Intelligence" that infers valid new information that is not explicit in the information that is presented to it.

Human Law

The relations between events in the animate world are many and complex and the propositions to describe them are often unformulated or are contentious and changing. The Artificial Intelligence Systems that have been built to handle human processes are only capable of modeling very elementary processes. The difficulties of extracting "machine understandable" information from natural human language are enormous. In a Q&A session we must handle natural human language.

SEC - Structure in Q&A Sessions

A rule base system appropriate for a Q&A system would require analysis of natural language and technical domain specific terms into underlying elementary rules and their relations and then encoding these rules and relations into a machine "understandable" format. This was seen as too large a task to build into Split-Up simply to gather the information required in determining the common pool status of assets. Instead we do not try and break apart the Q&A process into elementary propositions and the relations between the propositions but rather we used a graphical representation of these Q&A procedures. The DE uses these graphical representation to represent procedures and at the same directly implement these procedures on a computer. Whether this representation is limiting in the complexity of connections that can be understood in a two-dimensional form is to be explored using Split-Up in the real world.

However if there is structure in a Q&A session and a SEC models the session so there will be structure in SEC. How much more than a two-dimensional format will be required to model the Q&A's structure will be explored by building models of Q&As.

SEC and other graphical representations of knowledge

The current use of spatial representations of knowledge shows *where* humans find this means of knowledge representation useful and for *what forms* of knowledge. However the present use does not

define the future use. The limitations of technology may be defining present usage and as developing technology enhances and facilitates graphical means of communication so the use of graphical representation may grow.

The following examples are widely used in education, commerce and industry as tools for analysis and manipulation of information:

- Flow Charts
- Stochastic Processes, Markov Chains
- State Transition Machines
- Petri Nets

Computer Performance considerations

SEC size

The size of a the data in a typical Sequence in a SEC is around 20 words (100 bytes).

A typical SEC contains about 30 Sequences (3,000 bytes).

The data structures required to manipulate a SEC are:

- Strings
- Arrays
- Btree -Optional

These are low level structures requiring little computing overhead. Using Borland's ClassLib Objects the Strings functions compiled to 35 Kbytes, the Array to 54 Kbytes and the Btree to 44 Kbytes.

SEC performance

One SEC, at a time, is loaded from file into a Btree structure in memory, taking less than one second on a 486 SX 25. This memory is dynamically allocated as the file is read from disk and then freed before a new SEC is loaded.

The location of a Sequence from the SequenceTree containing up to 10,000 items will be achieved in three seeks.

ChartObjects

All ChartObjects are loaded into an array at the beginning of a Session, 20 objects plus array using 100 Kbytes. If the number of object became large, 100 plus, then some objects could be loaded only when required to run "their" SEC.

Overall

A SEC occupies little more RAM than the actual text it consists of. The only critical operation is reading the SEC from file into memory. The use of a database application to handle the storage and retrieval of SECs may be appropriate.

ChartObjects are a bundle of operations for handling SECs. The Object Oriented code model is highly suitable to their design and coding and results in a high performance model.

The data structures and code to manipulate the structures results in high performance program running on a P.C.

Chapter 5 Conclusion

We have yet to implement Split-Up on site to assist people with the settlement of marriage property disputes and so are yet unable to claim *service* for this *research*. However the SEC tools described in this paper fill the requirement for Split-Up part one.

The SEC tools make possible the computer implementation of a LES to build, maintain and run a LES to assist people in the process of mediation and resolution of conflicts.

The ChartObject model will enable the software engineer to give the DE access to operations and features of the computer environment that will assist them in their work.

The overall model makes possible the use of many features found in today's graphical computer environment.

The SEC model

The SEC model has the very valuable feature that the vocabulary of the knowledge representation is defined by the DE and this text can be changed readily. The usefulness of such a simple structure that a SEC represents is found in the external properties that the text embodies and the conclusion it justifies. This surface is to carry all the meaning for the process that the SEC models. The basic form of the knowledge structure is transparent and easily understood.

The syntax is simply a sentence with a defining phrase. The limitations that this may place on the DE's ability to model an interview and on the user's ability to follow the flow of questions and answers is to be only discovered with trialing of the system. The presentation of English sentences in a clear print-quality format with a 'click' selection of choices from a list is the central feature of the interface. The powerful tools of today's graphical users interface in IT offers many means of communication on a computer screen. The potential for meaning to be associated to different fonts and colours, icons, symbols and patterns grows with the increasing sophistication of imaging provided by computer technology .

Building and editing a SEC is like drawing and modifying a flow chart, a task that is greatly assisted by a computer GUI interface. The complexity of flow paths resulting from the possible branching of paths at certain junctures and then the possible joining of divergent paths, is greatly assisted by the graphical representation of a these paths. The identifying feature of a SEC as a type of flowchart is that the *meaning of a choice* made at a juncture fits into and is part of a *passage* of text that has

contingency for the DE. The choice made fits and defines the eventuality of the case, this answer to this question turns to these possible outcomes rather than those possible outcomes.

The meaning of the SEC is to be clearly seen in the surface of text. The logic in the sentences is brought out by following the path of connected sentences. The rules that determine if a sequence of connected sentences is valid are embodied in the path that makes a sequence. Note: the validity of a system composed from the combination of a number of rules is not being modeled by the combination of a number of sequences to make a *net*, where net carries the implication of a model with an integral validity of structure. The concepts of coherence or consistency are not implicit in the ability to *graphically join* two or more sequences in a SEC. The validity of composing a sequence and of joining sequences is dependent on the decision(s) of the DE thus the term *chart*, with the implication that the validity of the model drawn is dependent on the maker(s) of the chart.

At what level of abstraction can the DE model a SEC and yet make it worthwhile to use the model? If it is at the highest level of abstraction then the information captured is no more than text manipulation such as a word processor does. However simply recording a sequence of English sentences can capture sufficient information to justify a conclusion. It is the value of that conclusion that is the worth of the system.

Even where a SEC is no more than a word processor it is a very good manipulator and presenter of this form of text:

- A series of conjunctions of sentences and choices leading to conclusions entered only as text is hard to set-up and maintain without some controlling procedures for checking connections, edits, additions and deletions.

- The graphical representation of this form of complexity is an aid to understanding as shown by the use of the flow chart models.

- The format of presenting a sentence, as the next sentence in a paragraph of sentences, with a list of choices, noting that each choice may be from one word up to a complete sentence, is a format easily read by the user. In fact the context achieved by this format may be of great import in a person's understanding of the conclusion that is reached, a very human format.

SEC and ChartObject

The Compartment and Container metaphor appropriately suggests the modular nature of these two parts that are designed to fulfill the information gathering task for Split-Up. The SEC enables to the DE to build and modify the Q&A sessions to reflect the changing world of the Family Law Court. The SEC achieves a contextual organization of information. The right operation is triggered in the right place and time. The right information is stored or retrieved in the right place.

The operations in a ChartObject add power to what a SEC can achieve in its computer environment. The independence of code and the implementation in the OO environment enables different ChartObjects to be loaded and used for different environments to run the same SECs. The computer dependent operations are separated from domain dependent information and can be changed to suit different devices, databases and interfaces etc. with no change to the SECs.

The Implementation of Split-Up

The code for the ChartObjects has been developed in C++. The interface is entirely line by line text. The graphical representation so vital to the ease of use in building and maintaining nets would be the next stage in the development of the project.

The operations performed by the ChartObjects are at the data entry stage. No reports, such as the Forms 7 and 17 required by the Family Court, are being produced. It is planned that another module would produce these reports from the data gathered.

The SECs are written and read from flat files with a delimiter marking the beginning of each Sequence. Considering the increasing complexity of the data structure required in the Sequences to store instructions for and respond to the ChartObjects, it is advisable that the SECs are mapped onto a database. A relational data structure completely fulfills the requirements for the SEC structure.

Glossary

DE	Domain Expert
ES	Expert System
IT	Information Technology
LAC	Legal Aid Commission
LES	Legal Expert System
LIC	Legal Interaction Chart
Q&A	Question and Answer
SEC	Sequenced Event Chart
CBR	Cased Based Reasoning
RBR	Rule Based Reasoning
CK	Control Knowledge
NN	Neural Nets

Bibliography

Amble, T., *'Logic Programming and Knowledge Engineering'*, pp 232, 1987. Addison-Wesley Publishing Company.

Antoni, G. D., *'Analysis of laws by means of Petri Nets: Motivations and Methodolgy'*, Artificial Intelligence and Legal Information Systems. Vol 1, pp 273-299, 1982. North-Holland Publishing.

Chandrasekaran B., Tanner M., & Josephson J., *'Explanation: the Role of Control Strategies and Deep Models'*, Expert Systems: the User Interface, 1988. Ablex Publishing Corporation. New York.

Croft, W.B., Smith, L.A. & Turtle H.W., *'A Loosely Coupled Integration of a Text Retrieval System and an Object-Oriented Database System'*, SIGIR92 pp. 223-232, ACM Press New York.

Daniel A. *'Legal information technology increasing in sophistication but little change in patterns of use'* Law Society Journal,.pp 54-57, March, 1994.

Edmonds, E. A., O'Brien, S. M., and Bayley, T., *'Constructing end-user knowledge manipulation systems'*, Int. J. Man-Machine Studies Vol 36, pp 51-70. 1993.

Edwards, L. and Huntley, J. A. *'Creating a Civil Jurisdiction Adviser'*, Law, Computers & Artificial Intelligence, Volume 1 Number 1, 1992.

Kayman, M.L. and Kim, M.P. *'Expert Systems in Alternative Dispute Resolution'*, Proceedings of the Third Interanational Conference on Artificial Intelligence and Law, Oxford, , pp207-214, June 25-28 1991. ACM Press.

Lakoff, G., *'Fire, Women and Dangerous Things'* ACM Press, 1987. ACM Press.

Murata, T., *'Petri Nets: Properties, Analysis and Applications'*, Proceedings of the IEEE. Vol. 77, No. 4 April, 1989.

Rosen, R., *'The Scope of Matlhematics and Science'*, in Casti, J. L., Karlquist A., (eds), Real Brains, Artificial Minds, North Holland, 1987.

Stanieri, A. & Zelenikos, J., *'SPLIT-UP Expert system to determine Spousal Property distribution on Litigation in the Family Court of Australia'*, in Adam, A. and Sterling, L. (eds.), *Proceedings of A192,* World Scientific, pp. 51-56, 1992.

T Potter, *Law and Information Technology*, page 4, Section 4, July 3-4, 1993. The Weekend Australian.

Warner, D.R. *"The Role of Neural Networks in Law Machine Development"*, 16 Rutgers Computer and Technology Law Journal, 129, 1990.

Appendix 1 - Logic programming

Propositional calculus

The simplest form of logic formalism is **propositional calculus** or **propositional logic**, which is another name for **Boolean algebra**.

The elemental formulae in this calculus can be represented by Sequences of a SEC:

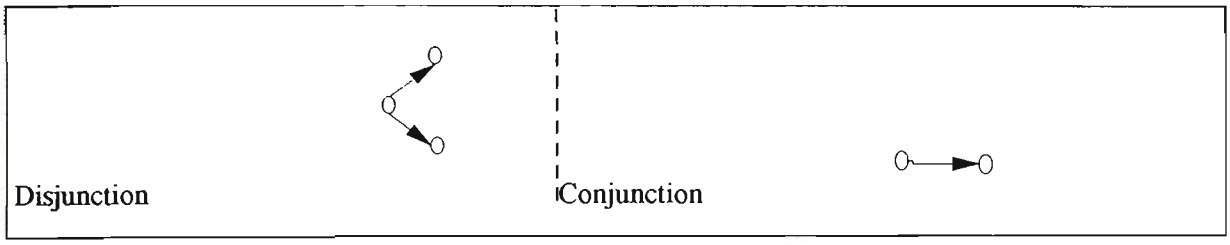


Figure 9. Propositional calculus formulae as SEC Sequence.

However a choice of the Sequence of a SEC is not implicitly a state with logical properties defined in relation to other states reached by making other choices. Choices at a Sequence do not necessarily lead to mutually exclusive states. The logic in propositional calculus that allows TRUE and FALSE values to be deduced for variables from combinations of other variables is not contained in a SEC and rules built in propositional calculus cannot be built in a SEC. As figure 10 shows the result of the rule can be modeled but the logic is not represented.

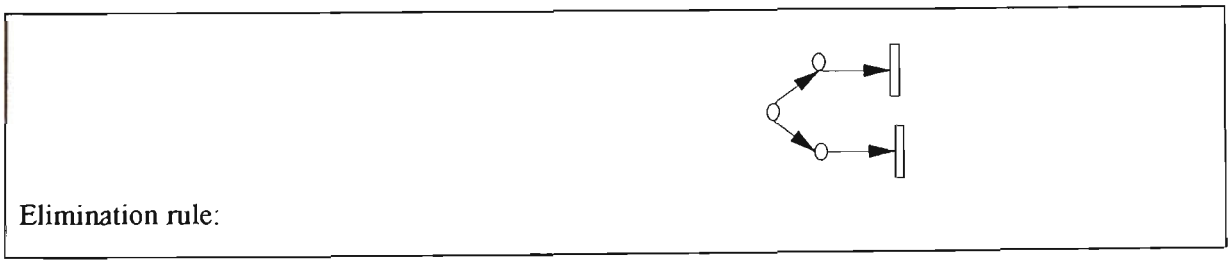


Figure 10. SEC interpretation of propositional calculus elimination rule.

The places that are meant to have definite properties in a SEC are the terminals of a path and the more mutually exclusive properties that each 'conclusion' of a path has then the more meaning there is implicit in the explicit paths of the SEC.

However a propositional calculus statement containing only conjunctions can be fully modeled in a SEC, figure 11.

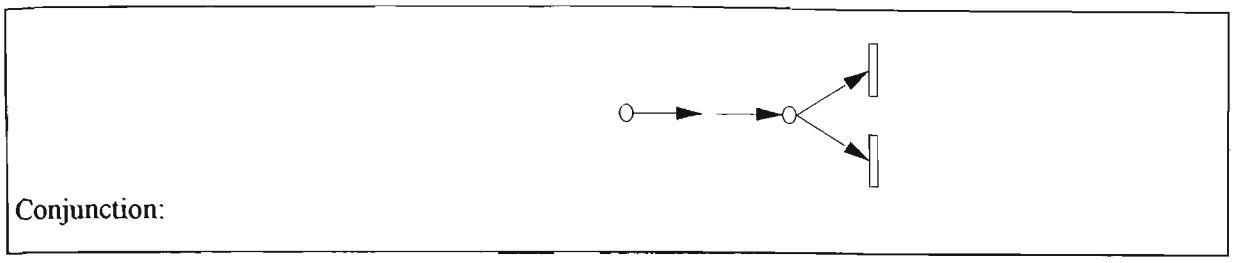


Figure 11.

Logic programming

In **logic programming** logic rules govern information retrieval. The rule IF *B1 and ....and Bn* THEN *A1 or ... or Am* is fully represented by the SEC. Therefore the SEC can be used to represent rules that are used in logic programming.

First order predicate logic

Propositional calculus is a subset of a more general logic system called first order predicate logic. By introducing logical variables into propositional calculus, first order predicate logic is entered.

In the sentence 'Clyde is an elephant', something is said (being an elephant) about something (Clyde). 'Being an elephant' is a **predicate**, while that being talked about is called an **argument**.

In symbolic logic, the standard is to write the simple predicates as identifiers, followed by the arguments enclosed in parentheses. Mathematically, a predicate is a function delivering only truth values; for example:

elephant(Clyde)

A predicate with arguments is called a **literal**.

To make a predicate-argument of the rule:

All elephants are mammals.

a logical variable, *x*, is introduced which may represent any argument.

For all things *x*,

x is a mammal if *x* is an elephant.

Any clause formula containing logical variables is supposed to be true for all possible values of the variables in the formula.

mammal(x)← elephant(x)

The example in figure 11 for one A:

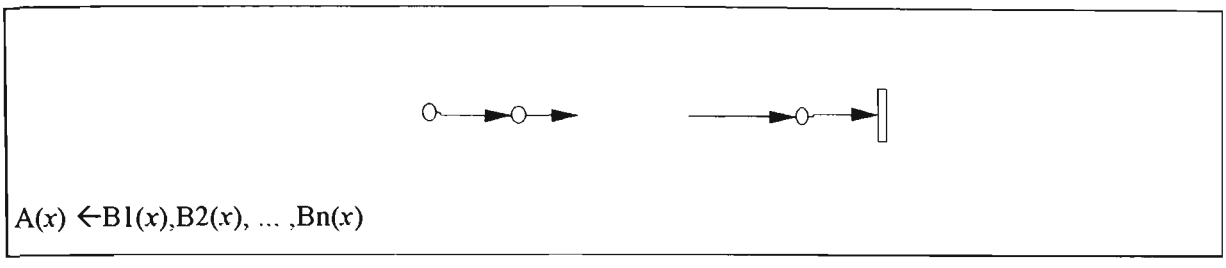


Figure 12.

A choice is a predicate, its argument is the *identifier* for the Sequence B1, B2 ...Bn, it is TRUE when this choice is made. Appendix 0 is a comparison of a portion of a rule base built, from an error tree, for an expert system in Prolog and a SEC of the rules. The procedural information that is contained in the error tree is lost in the Prolog code. The expert's knowledge is represented and maintained with the error tree. The SEC can replace the error tree. The Prolog code can be generated from the SEC by a very simple parser.

There are two obvious difficulties:

1. The tie-up between the formal language of Prolog and the text of the Sequences may prove too restrictive on the text used in the SEC. Instead an underlying text of actual Prolog code may be attached to each Sequence for each choice. This would require some understanding of Prolog syntax and functions.

2. The problem of establishing the text that is the identifier, the argument of the predicate, is more complex than a parsing problem. How can a rule be built with an identifier when that identifier is established at runtime? When the argument being referred to is one of many and its 'id' is a dynamic key then it is impossible to directly refer to this in the rule.

Identifiers

In Prolog a **constant identifier** is used to denote an atomic entity or concept. It cannot be substituted by any other term and it will have the same interpretation in all clauses or any derivations of them. The example in figure 12 the logical variable x could be denoted as applying to a *current identifier* that the rule is being run for. This is restrictive in the use of the rule and the design of the rules, logical variables x, y, z etc. maybe required.

"Formal logic usually defines some knowledge. From a logical point of view, this is a declaration of things that are held to be true. It represents the set of facts that are stated, or can be deduced. This is

the **declarative interpretation** of clausal logic. On the other hand, when the same clauses are combined with the top-down procedure, this gives a procedure that solves problems. This is the **procedural interpretation** of clausal logic. Usually, instances of the logical variables that made the proof possible must be collected, because these are the solutions".

"Clausal logic is a specification language of **universal expressivity**, with an automatic deductive mechanism for solving any problems specified. The property of being a very powerful specification language, and at the same time a very high level programming language makes clausal logic a very strong formalism. The procedural interpretation of clausal logic makes it into one of the most powerful programming languages ever conceived.

"The use of predicate logic as a programming language is called **logic programming**. The key idea underlying logic programming is programming by description. A traditional program consists of a sequence of operations to be performed in solving a problem. The assumptions on which the program is based are left implicit. In logic programming, these assumptions are made explicit in logic, while the sequence of operations is implicit".

Error Tree

[Amble, 1987]

The places A to I are parts of a television set. The branches from the places correspond to the state of the parts - OK or DEAD. The terminal leaves have the same state, ERROR, indicating this part is a cause of the previous symptoms.

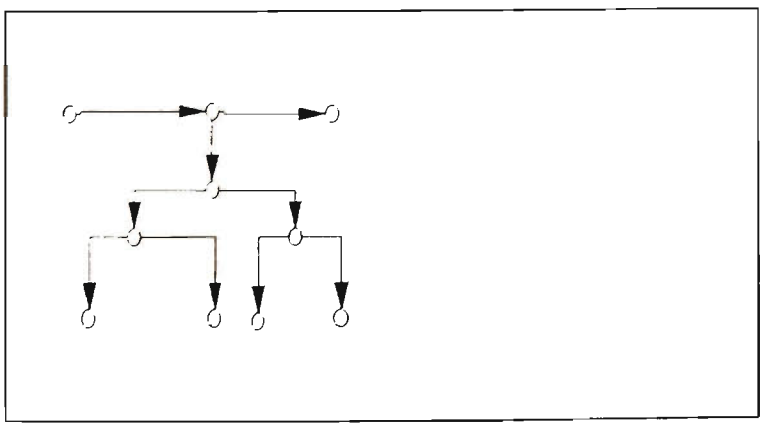


Figure 1.

Prolog Code

if A and OK(B) and OK(C) and OK(D)

then ERROR(F)

if A and OK(B) and OK(C) and DEAD(D)

then ERROR(FG

if A and OK(B) and DEAD(C) and OK(E)

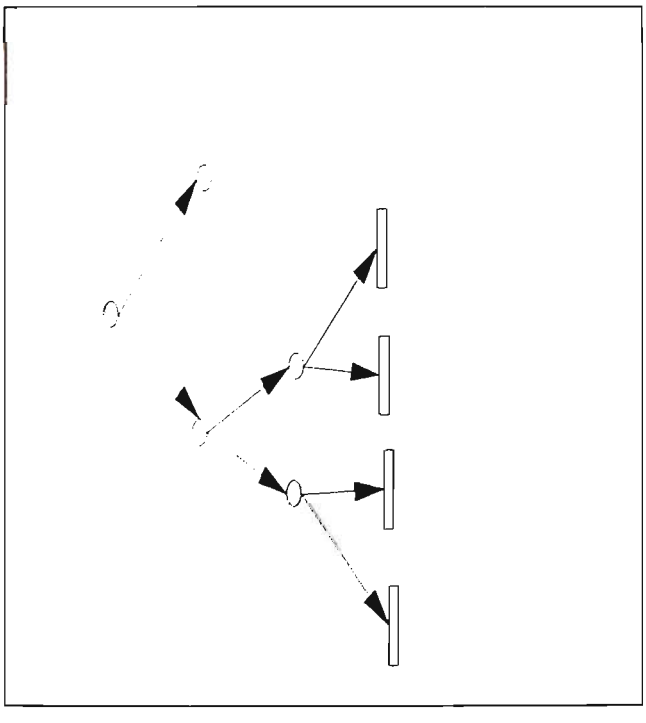
then ERROR(H)

if A and OK(B) and DEAD(C) and DEAD(E)

then ERROR(I)

SEC

Sequences



ID	Before	choices	After
0	The B 0	is	OK.
		I is not	

01	The C	0	is		OK.
			1	is not	
011	The D	0	is		OK.
			1	is not	
010	The E	0	is		OK.
			1	is not	

Conclusions.

0111	Error F
0110	Error G
0101	Error H
0100	Error I

A graphical interface to the SEC that allowed full editing of the chart would enable design and maintenance of the logic program by someone who only need understand the error tree.

Parsing the SEC

A template could be used convert the SEC to Prolog code

For each conclusion:

if < choice(Identifier) > **and** ... < choice(Identifier) > **then** conclusion(Identifier)

For the conclusion 01111:

if is(B) and is(C) and is(D) then ERROR(F)

For the conclusion 01110:

if is(B) and is(C) and is not(D) then ERROR(F)

Appendix 2 - Petri Net and SEC

"Petri nets can be applied informally to any area or system that can be described graphically like flow charts and that need some means of representing parallel or concurrent activities" [Murata, 1989]. There appears to be little concurrent activity in a SEC. A SECis modelling a series of questions with a set of answers to each question, basically this is a disjunctive activity with differing paths of questions and answers to differing conclusions. There is some concurrence where differing sequences of questions and answers reach the same conclusion and where differing sequences are joined to one continuing sequence. In all cases it may be useful to be able to use the formal language for graphical models that Petri-net theory has developed.

Comparison of SEC and Petri net

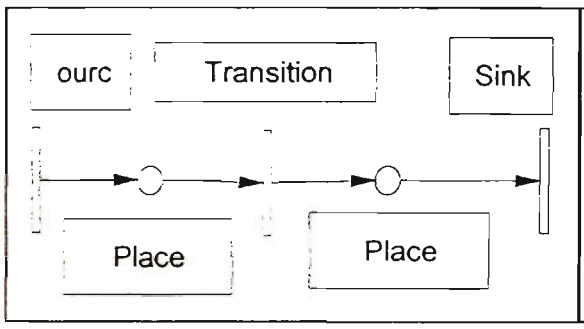


Figure 12. Petri net

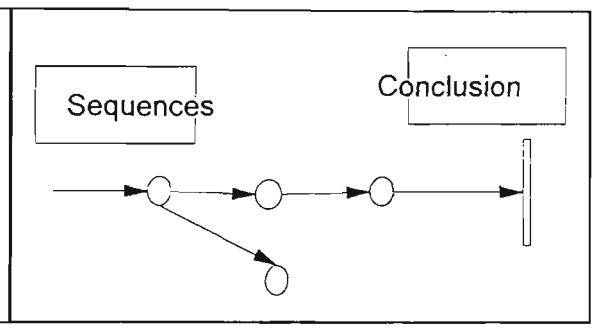


Figure 13. SEC

The interpretation of Transitions and Places in a Petri net model of the question and answer process:

- A *place* is a **state** where a condition is made or fulfilled such as the conditions of a question or the consequences of a particular answer to a question. A distinction can be made between input and output places.

An *input place* is the **conditions** of a question.

An *output place* is the **conclusion** of a question from the answer given.

- A *transition* is the **event** of satisfying a rule and fulfilling the conditions, such as the act of answering the question.
- A *token* passing through a transition to a place indicates the conditions of the place are fulfilled or that this is the answer given to the previous question.

A SECis a set of paths to conclusions. Each path is a complete set of Sequences. Each question on a path is a Sequence. A conclusion is made where sufficient conditions are fulfilled to stop this line of questioning. An SEC can be modelled as a Petri net, a Sequence is a place and a transition is put between each Sequence.

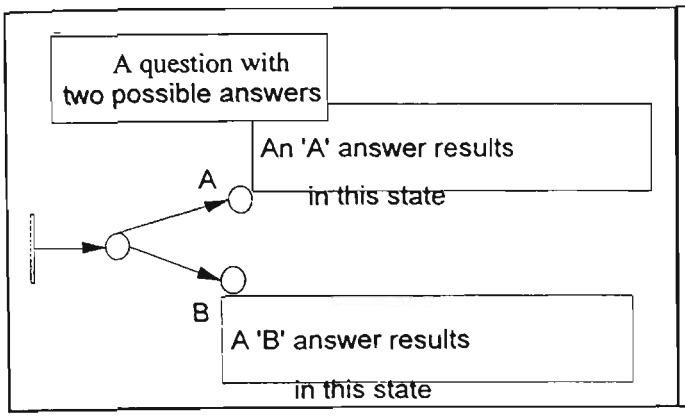


Figure 14. SEC

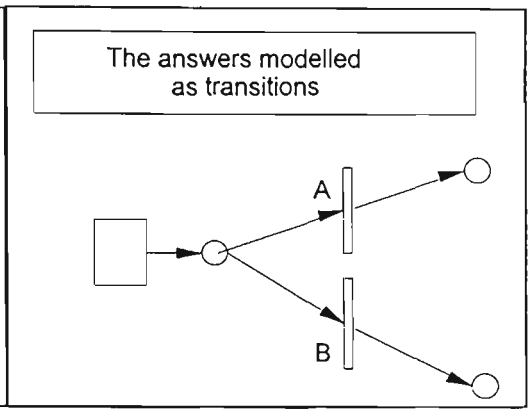


Figure 15. SEC with transitions

Controlling Concurrent Events

Concurrent events are controlled in a Petri net by requiring a *transition* to be enabled, i.e. a place with an arc to this transition has a token on it, before it can fire and a token can pass. A *transition* can have one or more one incoming arcs and each of these incoming arcs must have a token on it for the transition to be enabled and for a token to pass through this transition to its output arc(s).

Concurrency does exist in the system being modelled by the SEC. The Path to a conclusion in a SECis a series of concurrent events. To reach a particular conclusion *a* particular answer is to the first question **and** *a* particular answer to the second **and** ... so on to this conclusion. In figure 16 this is modelled as a Petri net. The conjunction of answers is modelled as the concurrent conditions required to be true, i.e. the **places** A, B, C and D must have tokens in them , for the Conclusion to be reached, i.e. the **transition** to fire.

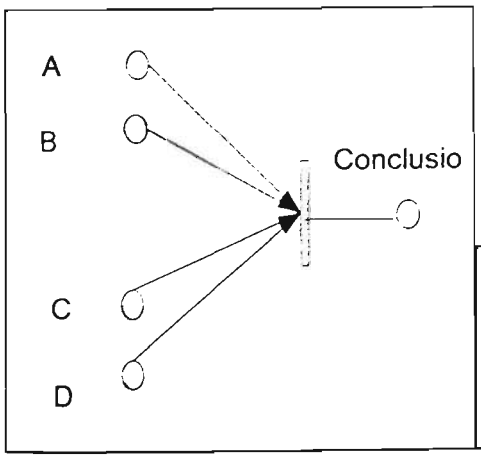


Figure 16. The path to a conclusion as concurrent conditions A, B, C, D in a Petri net.

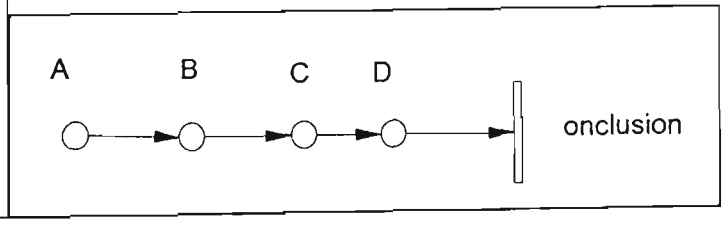


Figure 17. The path as modelled in a SEC.

The graphical models of the concurrent conditions required to reach a conclusion ,as shown in the Petri net, figure 16, and as shown in the SEC, figure 17, are very different. The series of questions A, B, C and D are asked in a definite sequence in the SEC while in the Petri net there is no connection between the places.

However there are two points to note:

1. The sequence A, B, C and D in the SEC may only be procedurally linked and not logically necessary conditions of the conclusion.

2. A Petri net that looked like the SEC in figure 17 would achieve the same effect, asking the questions and proceeding to the next place dependent on the answer. But the logic that the conditions A, B, C, and D must be fulfilled to reach this conclusion would not be implicit in the model.

Building a Petri net from a SEC

The paths to conclusions in a SEC can be used to build a Petri net. The paths 00, 010, ... from the SEC in figure 19 indicate the same conclusion in the Petri net in figure 18. The Petri net has only one transition, answering the question, after the source transition, not shown, to the sink transition, the conclusion. There are two places on each path, firstly the question and then the answered questioned. There is a correlation in that a longer path of Sequences in a SEC appears as more inputs arcs to a sink transition in a Petri net.

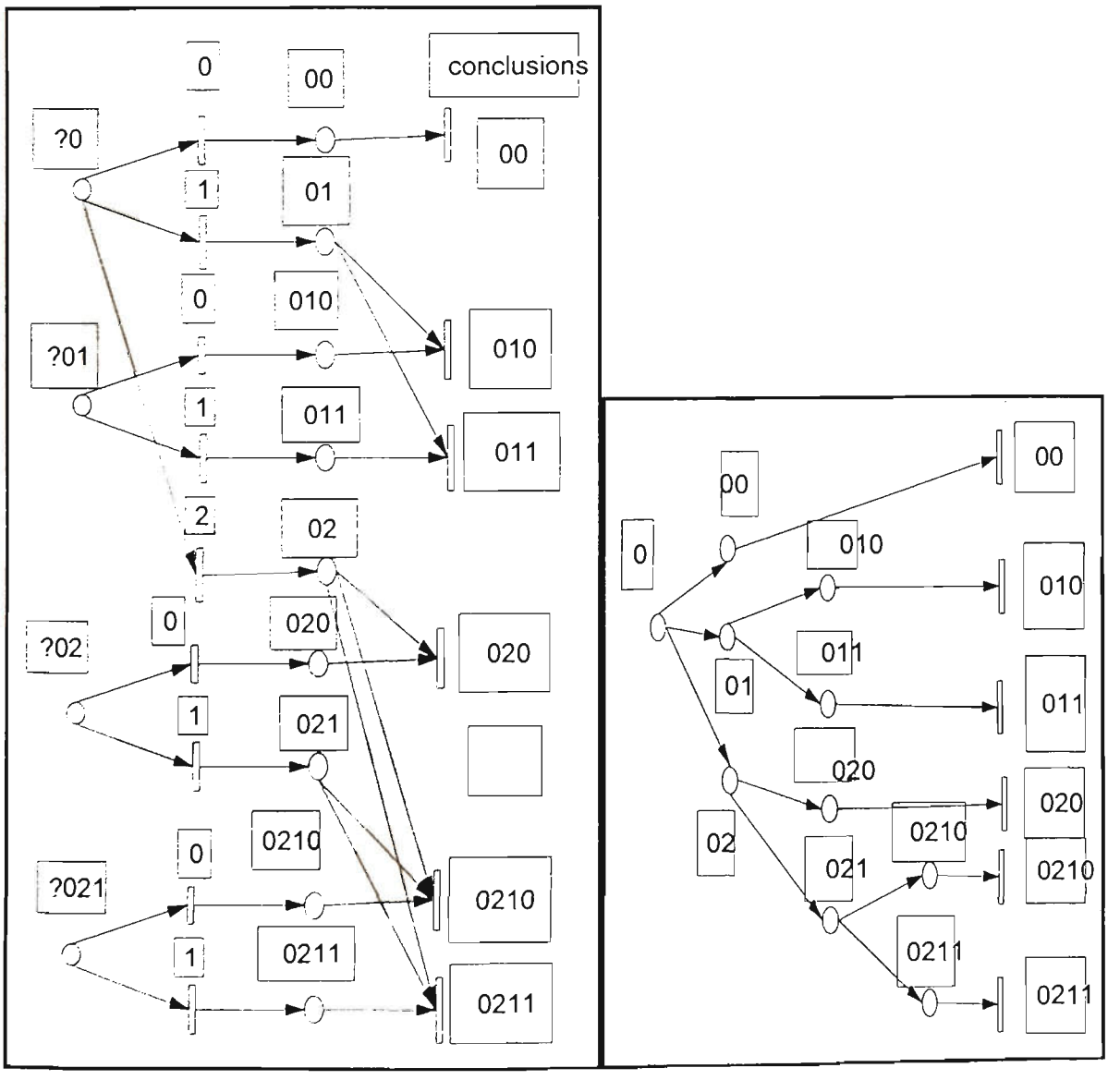


Figure 18. Petri net built from SEC paths.

Figure 19. The SEC.

Joins in a SEC

Joins to a place

A SEC may have more than one input arc to a *place* where a number of paths *join*, i.e. different sequences of events can result in the same state occurring and a common continuing path. This is similar to the Petri net model of the flow of tokens into a place - a place can receive a token from any input arc with no distinction between incoming arcs and with no consequence to selection of outgoing arcs. Similarly for a SEC the continuing sequence is not affected by which of the previous sequences was taken before the *join*.

Joins to a transition

It may seem appropriate to join two or more SECs. If a question has been asked in SECA then get the answer for SECB by joining to that answer in SECA, assuming a flag is set as a record when SECA is run.

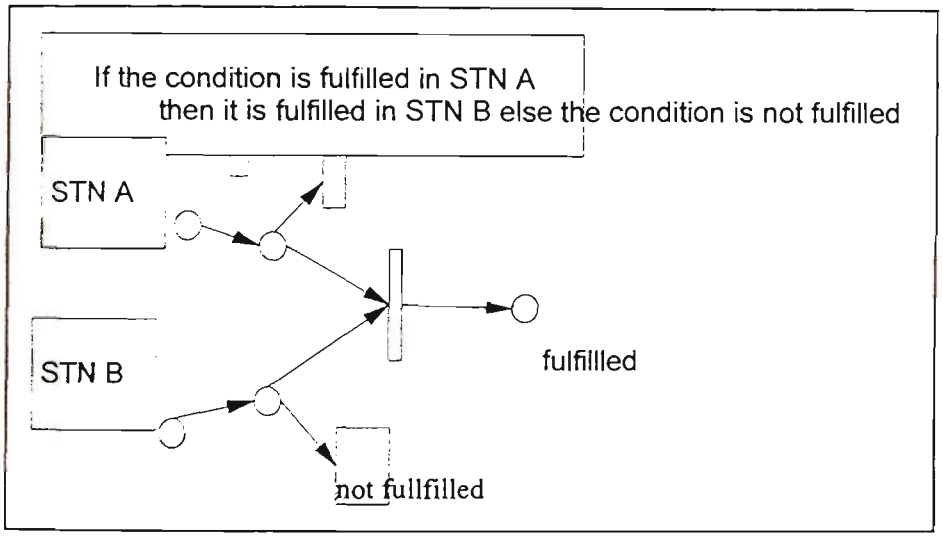


Figure 20. SECs joining.

However although SECA may not reach this point that does not mean that this condition would not be fulfilled and that this event would not occur. The syntax of a SEC does not make each place an exclusive event. Therefore it is not useful to join two or more nets. What is being modelled in a Petri-net transition is the existence of a relationship. In a SEC the relationship is part of a series of events that exist in sequences in a context. The current SECis to be as appropriate as possible to the current context. The information in the SEC should only be referred to in this current context and thus referring from one SEC to another as containing an independently existing fact is not appropriate.
If there is something that is sufficiently independent, identifiable and meaningful to all the sub-contexts of the SECs used in a system then it is to be written and read to an independent knowledge base. If the

context of the information is sufficiency independent then its content could be contained in a flat tables database.

A SEC can have a function at a place to assist the user with access to, and manipulation of, this independent information . This would assist in avoiding asking the same question twice and in deducing answers from information given, such as age from birthdate or one date before another.

The use of units of this independent information may be modelled as the input arc and output arcs of a transition.

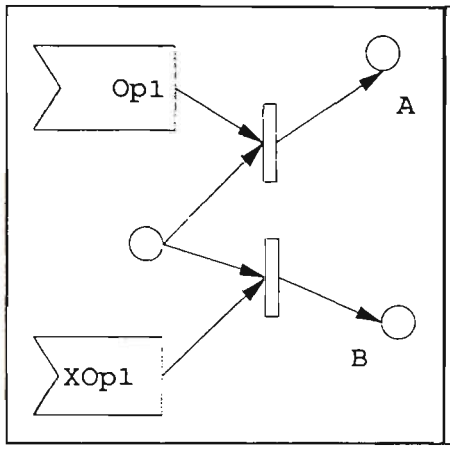


Figure 21. Petri net

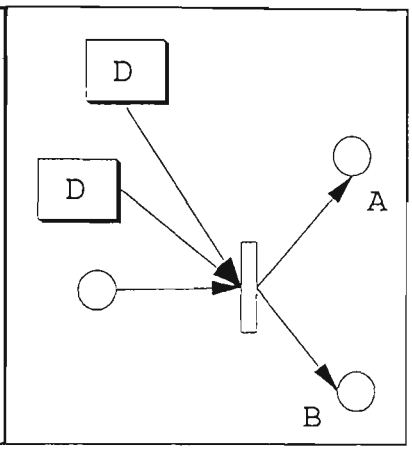


Figure 22. SEC

Existence: If the numeric values D1 and D2 both exist then go to A else go to B.

Comparison: If D1 *compare* D2 then go to A else go to B where the comparison may be equality, greater than etc.

This is modelled as a Petri net in figure 21: If Op1 returns True then go to A else Op1 is False go to B. This is modelled as a SEC in figure 22. The place preceding A and B has an action attached to it, the action returns the sequence go to A or go to B depending on the manipulation of D1 and D2.

Petri nets and SECs Summary

Both Petri nets and SEC model systems that may use flow charts in analysis and as a means of communication for analysis and design. The formal language that can be used with Petri nets is a powerful tool that enables the Petri net of a system to be used as an abstract model to investigate the system.

Both have a procedural process of going from node(s) to node(s) depending on certain conditions; a SEC has only one node 'active' at a time, a Petri net can have more than one. There is only rule one has to learn about Petri-net theory: the rule for transition enabling and firing. There could hardly said to be any rules to a SEC just a very simple algorithm to run one: retrieve the next Sequence based on the current Sequence plus the current choice.

Appendix 3 - -Comparison Of SECs And Frames

Frames are a current technique in ES technology for knowledge management and inference.

Frames are an AI tool closely related to the object oriented software model. They are a technique for the organization and maintenance of a rule-based inferencing expert systems with a very large number of rules. A frame enables the software designer to cluster rules and associate them into classes by connecting them to frames. A frame has a some context and thus specifies where, in what context, a rule is used. The connection of frames with rules and rules with frames aims to build an integrated system for knowledge representation and for inferencing. The rules in a system now refer to frames. The slots in the frames behave like variable parameters in the rules.

Frames in Expert Systems

Frames, as used in Expert Systems, are a generalization of a record structure, containing a variable number of named slots which in turn may contain many different kinds of information. Frames can be used to represent entities and events, and as such are a form of object. Frames may inherit common facts and properties from other frames in the same class in a hierarchically structured model.

Frames can have rules (known as attached predicates) that control storage and retrieval of information to maintain the integrity of the knowledge base and provide for dynamic information management.

Frames do enable the vocabulary of the knowledge representation to be defined by the user, who can define predicts and frames as deemed appropriate. Thus the basic vocabulary is extensible and can be customized to the needs of the user. The syntax is English-like and readable.

SEC and Frames

A Sequence in a SEC is a structure to hold data that is to be used in deciding the branch of the path to choose at this juncture, the data is in context. The data structure of the information matches the Object the SEC is design for (and is designed by). The Compartment is made by the Container. The surface text of a SEC is entered by the designer in sentences that are to flow and be sensible. The functions built into the Net Objects/Containers can control storage and retrieval of information and can be triggers to events.

When a human being perceives a fact situation, he or she appears to fit the case immediately to a description that captures the structure and expectations inherent in the case [Lakoff, 1987]. One of the basic ideas in cognitive psychology to explain this phenomenon is that memory is organized in large

and highly structured conceptual models. During problem solving, or whatever task a human being is performing, appropriate memory modules are activated to guide the interpretation of the situation the problem-solver encounters.

Many have argued that modules like these are the basic building blocks of cognition [Lakoff, 1987]. This general idea has been is seen the AI models. The use of frames that organize knowledge in a hierarchy of frames or scripts that describe a stereotypic sequence of actions.

Appendix 4 - Legal Interaction Graphs

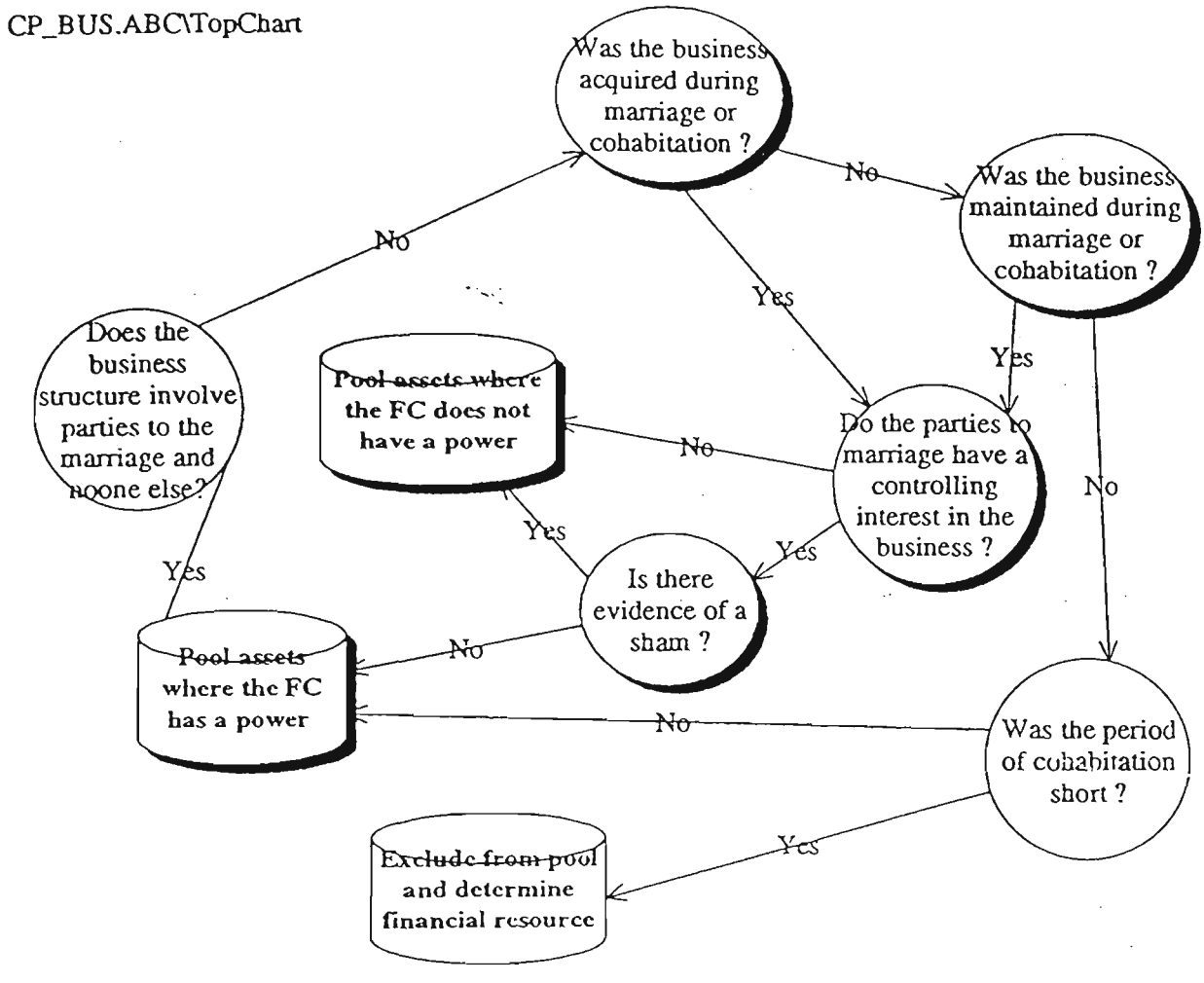
Business

Furniture

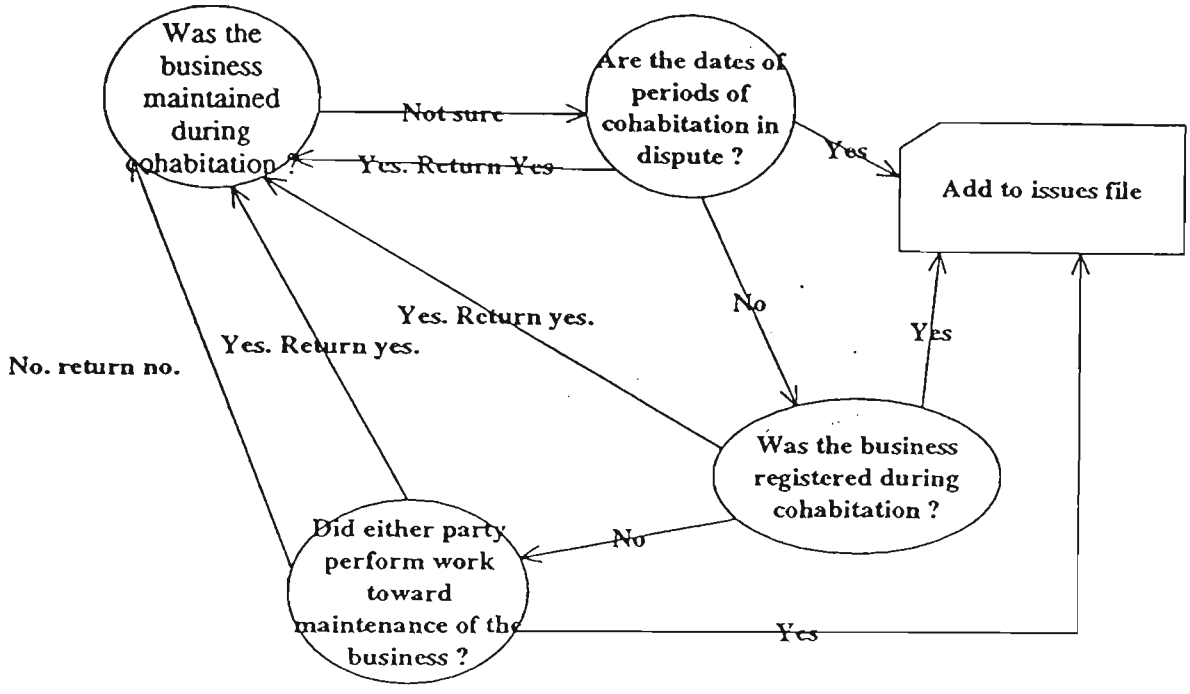
Liability

Superannuation

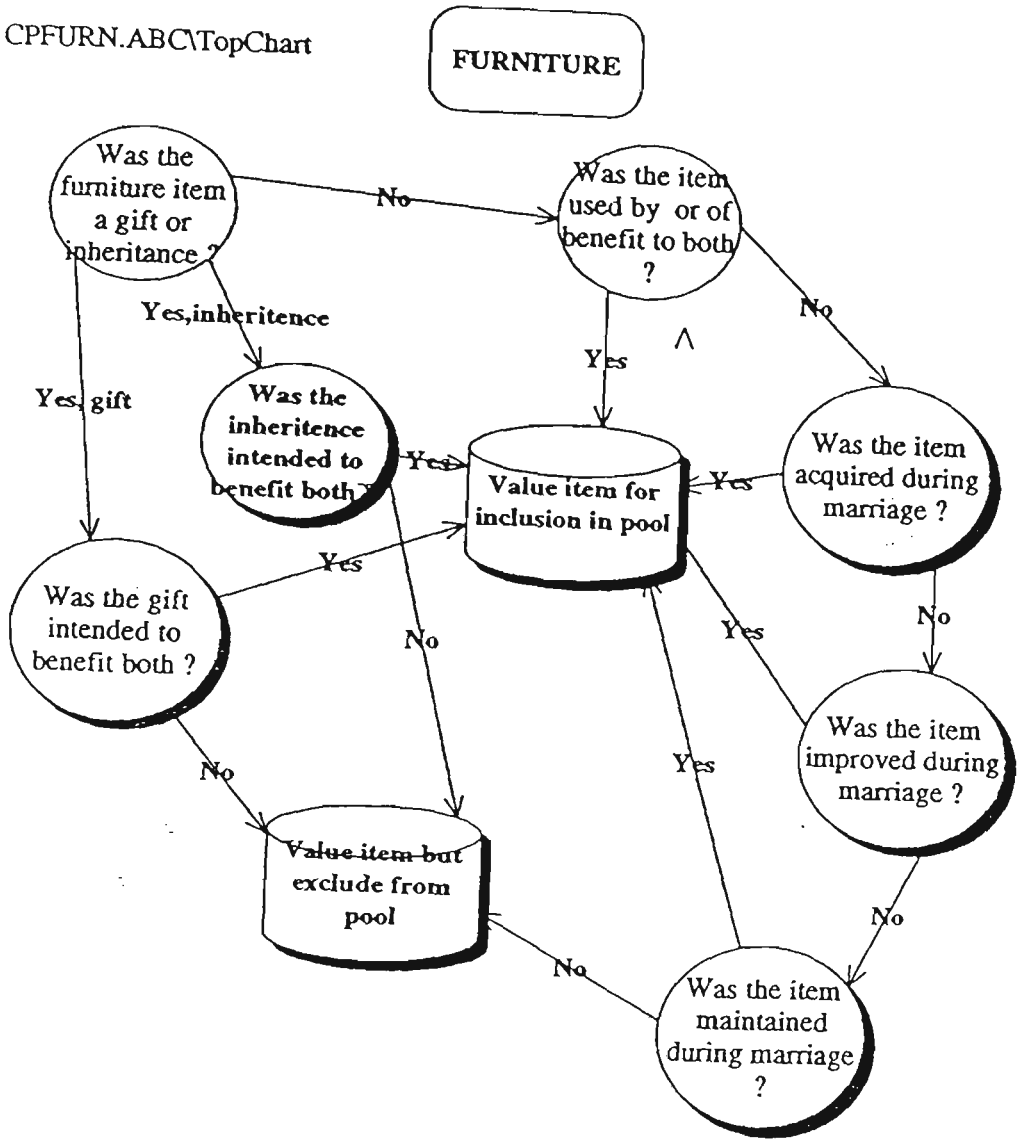
CP_BUS.ABC\TopChart



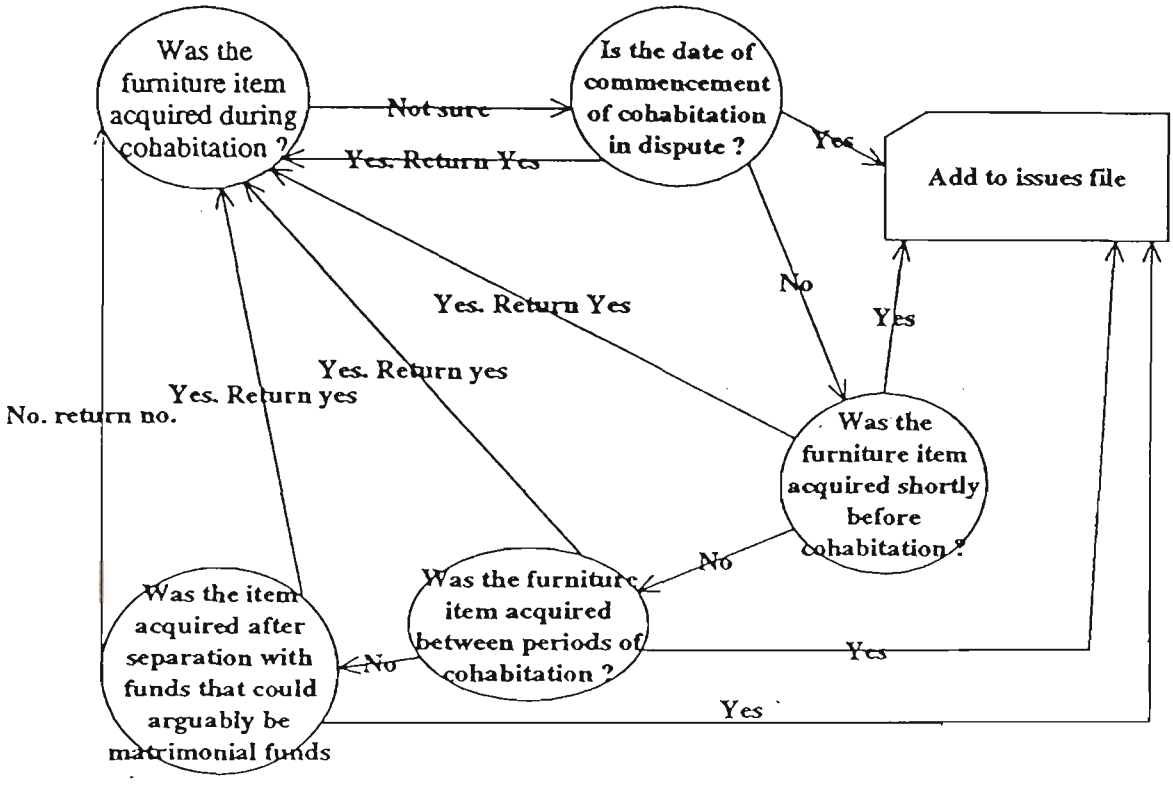
CP_BUS.ABC\TopChart\Bus. Maintained during



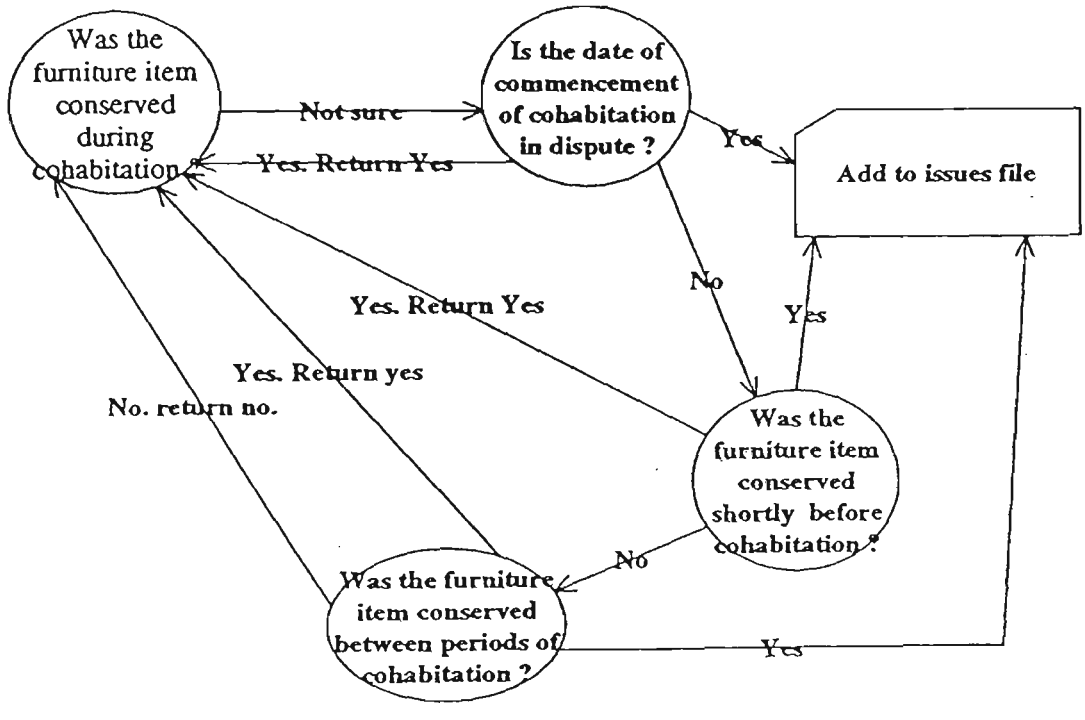
CPFURN.ABC\TopChart



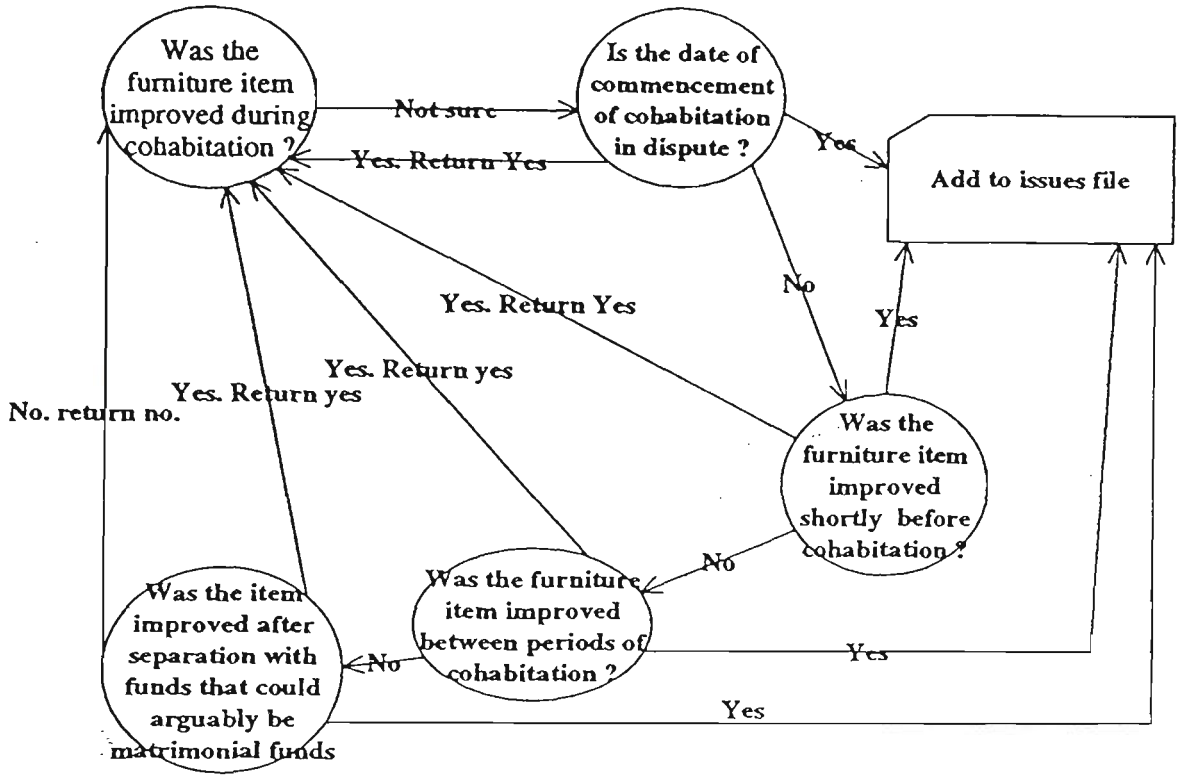
CPFURN.ABC\TopChart\Furn acquired during



CPFURN.ABC\TopChart\Furn conserved during

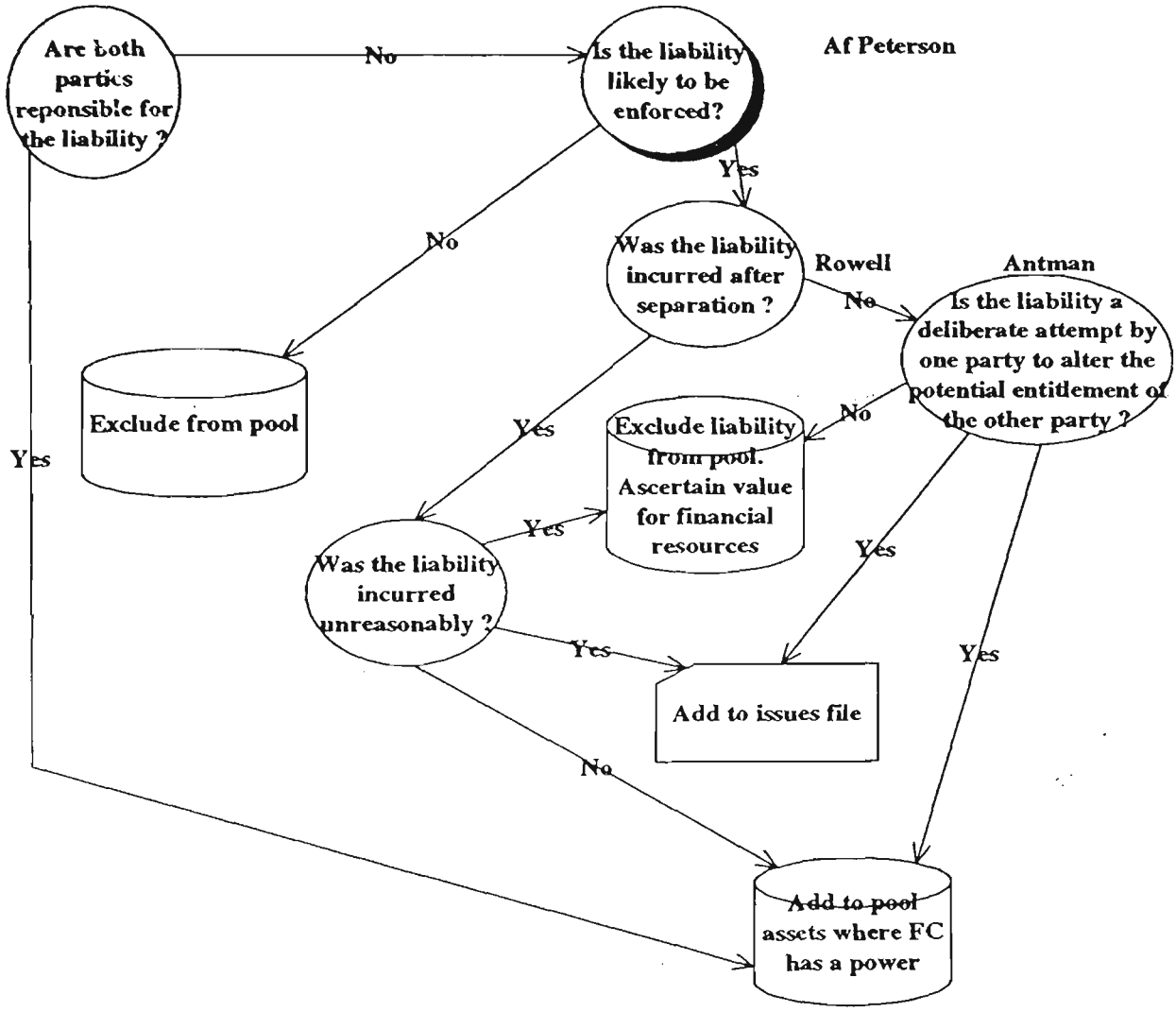


CPFURN.ABC\TopChart\Furn improved during

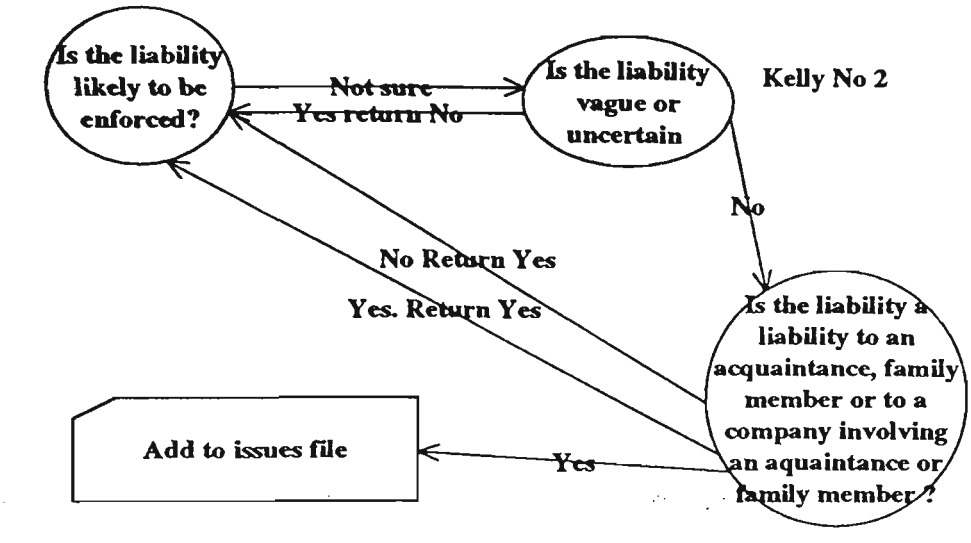


TopChart

Liabilities not associated with an asset

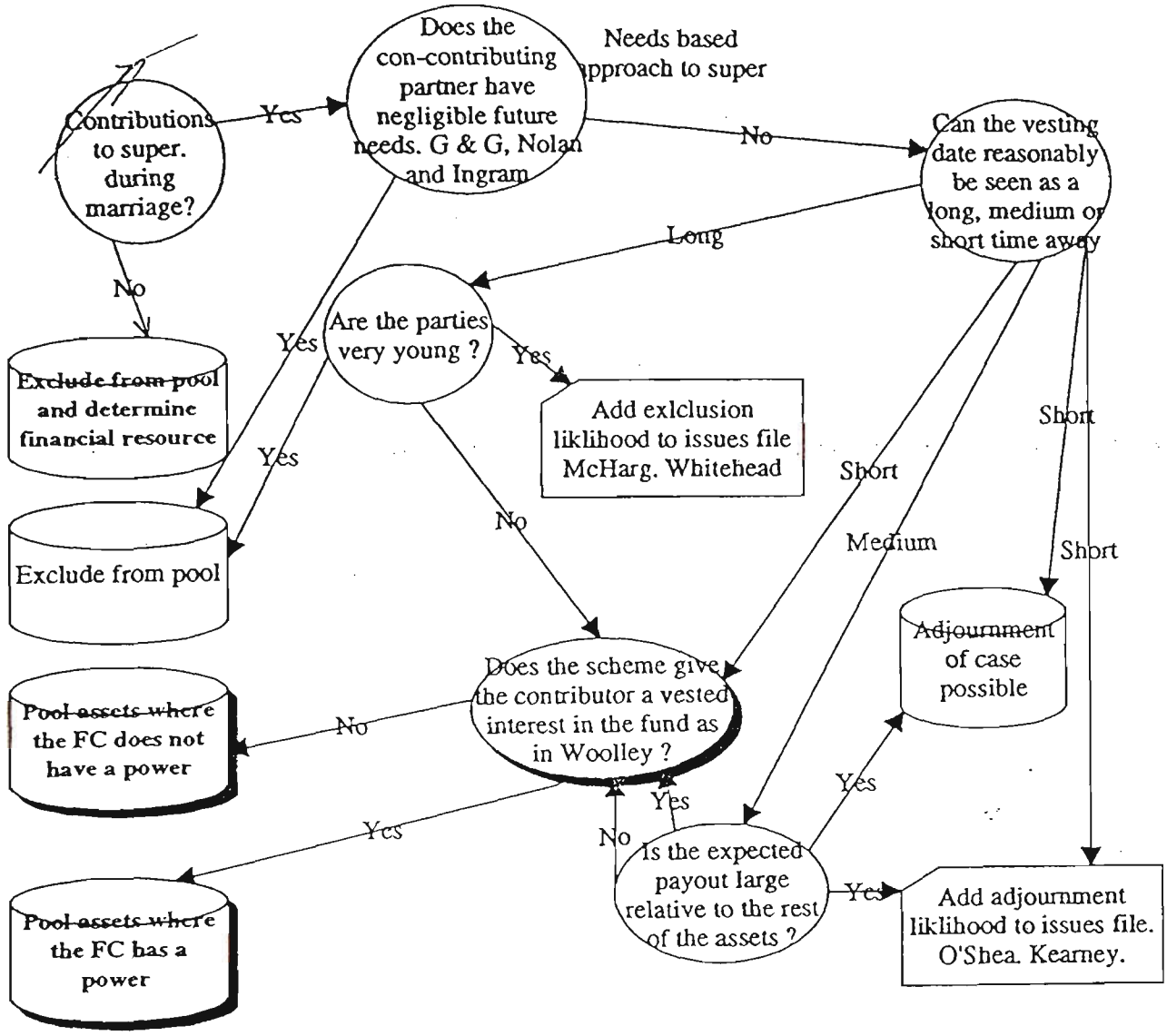


Debts enforced



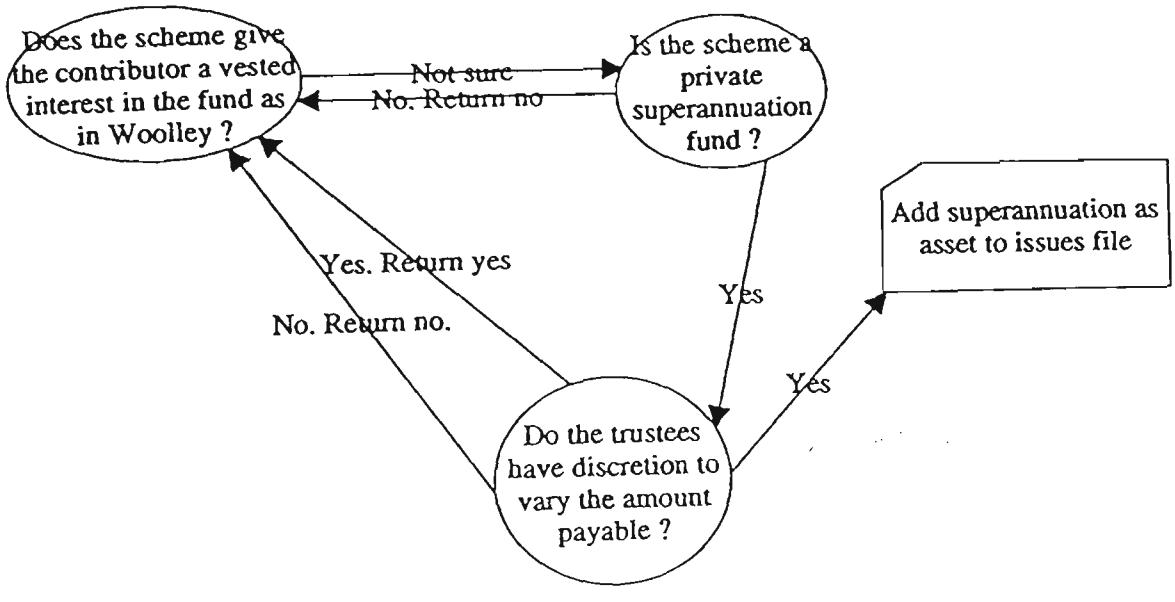
TopChart

SUPERANNUATION

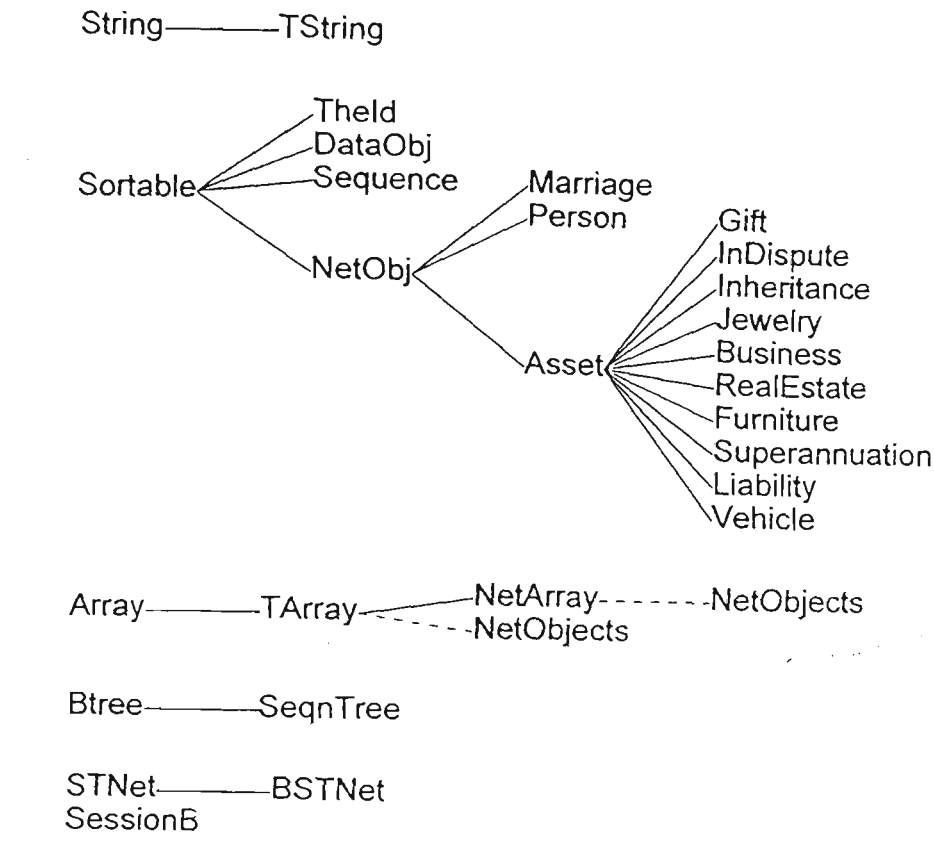


Super property

Wooley [2] 90-011 (1981)



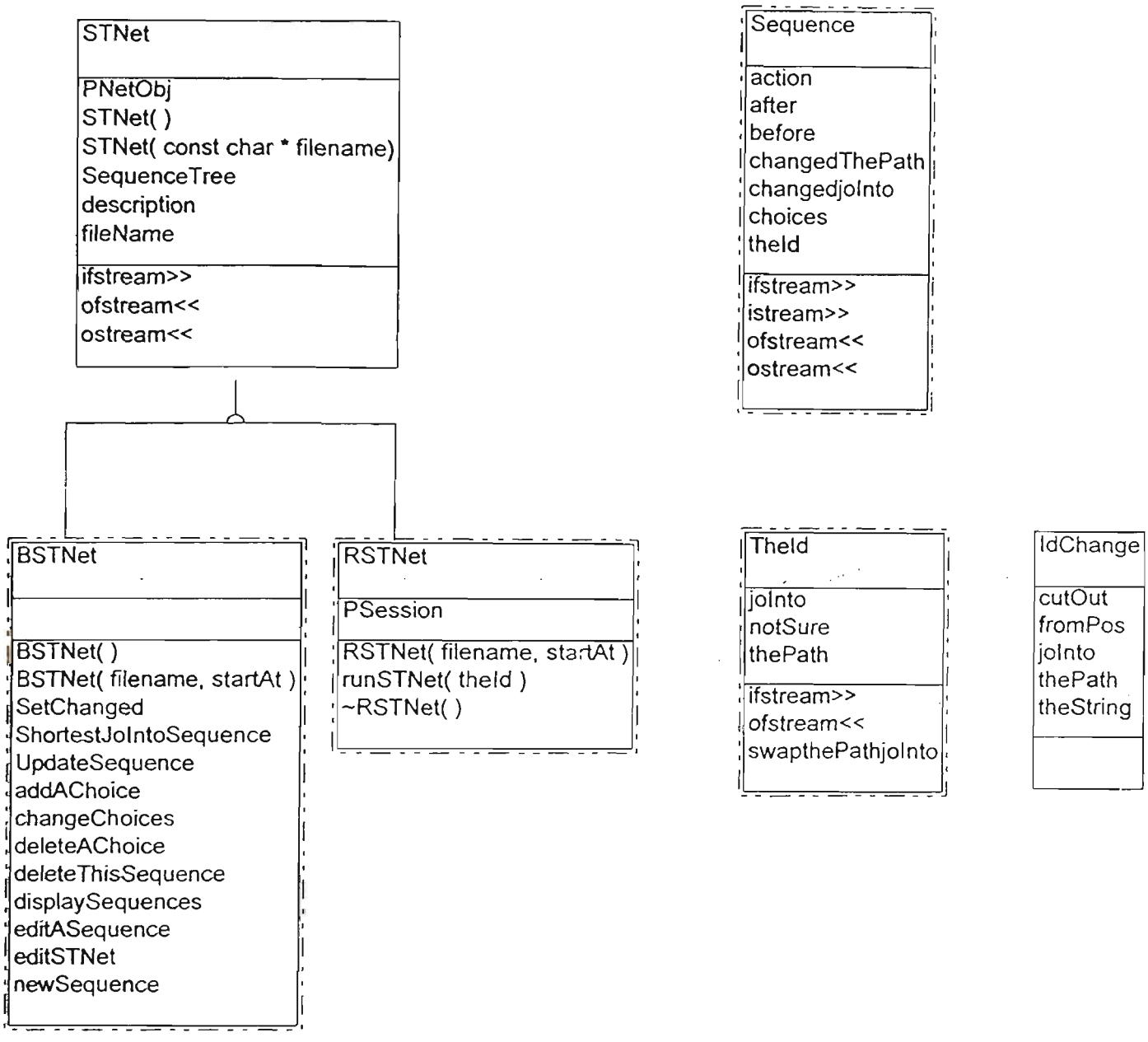
Appendix 5 - Object Models



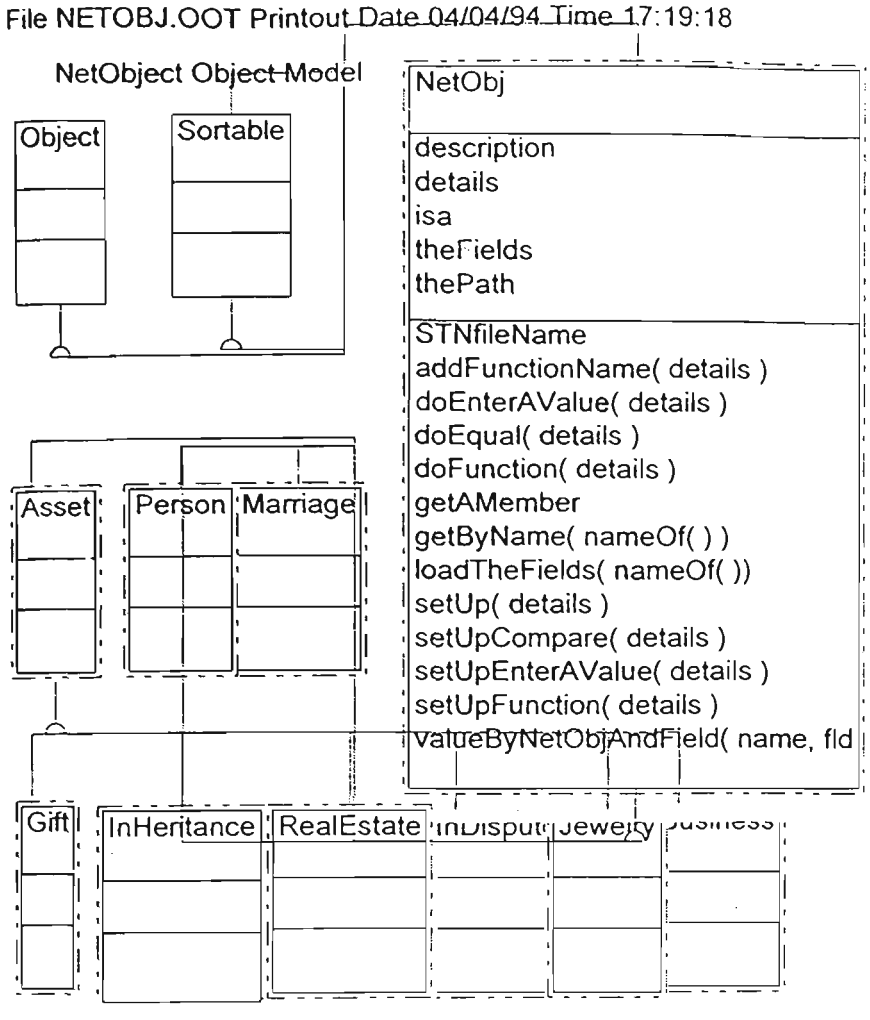
Date: 04/04/94, 04:45 00

File STN.OOT Printout Date 04/04/94 Time 17:24:28

STN Object Model



File NETOBJ.OOT Printout Date 04/04/94 Time 17:19:18



OBJECT: BSTNet

Purpose:
Build and edit ChoiceLines

General:

Use:

ASSOCIATIONS WITH

(Gen) STNet

ATTRIBUTES

SERVICES

BSTNet( )

Function:
Destructor for the BSTNet.
Query to the user to write the SequenceTree to file.

Parameters:

Returns:

Algorithm:

BSTNet(filename, startAt)

Function:
Constructor for a build and editing a STN. Invokes the STN constructor. This loads the SequenceTree from the file exists else sets fileName to NULL, BSTNet wil then open a new file and call newSequence(startAt) to build STN.

Parameters:
filename - the file name of the STN.
startAt - the id to start the STN with.

Returns:

Algorithm:

SetChanged

Function:
Set the changed attribrute of each CL in the net to a character.

Parameters:
The character

Returns:

Algorithm:

Function:
Place the shortest joInto, if there is one, in IdC.theString

Parameters:

Returns:

Algorithm:

UpdateSequence

Function:
When a CL choices are edited if there is changes in the sequence of transitions then the each sequence in tl
that has thePath through this changed transition must be changed to the new sequence

Parameters:

Retums:

Algorithm:

addAChoice

Function:
Add a choice to a Sequence.choices then update the any changed sequences in the net.

Parameters:
the reference to the CL to add choice to.

Returns:

Algorithm:
Make a copy of the Sequence.choices.
Add the choice.
Iterate through the Sequence.choices comparing the newSeq with the oldSeq.
If changed then load data into a IdChange struct and call UpdateSequence for the SequenceTree.
Then call newSequence for the added choice.

changeChoices

Function:

Parameters:

Returns:

Algorithm:

deleteAChoice

Function:
At a Sequence the user is prompted for the number of the choice to be deleted. This choice is removed fro
Sequence.choices and the STN is updated.

Parameters:

Returns:

Algorithm:
A copy of Sequence.choices is made - oldChoices.
The chosen choice is deleted from Sequence.choices.
Then iterate through oldChoices with counter oldSeq, comparing each choice with the current array of choi
If the sequence of a choice has changed then call UpdateSequence for the net.
For the delete choice call deleteThisSequence to remove all Sequences continuing on from this choice. A c
made for other paths joining into this continuing path. If so, the shortest joining path is made the continuing path.

deleteThisSequence

Function:
Delete the Sequences that continue on from a deleted choice of a Sequence.

Parameters:

Returns:

Algorithm:
Find the shortest joInto this sequence if there is one then change the before part of thePaths up to the first
thePath of the joInto and delete the Seqn that is the JOIN. Also change any other joIntos to the new path.

displaySequences

Function:
To enable the user to browse through the existing Sequences, to stop at a Sequence and get thePath of thi
Sequence.

Parameters:
thePath

Changes:
Only thePath argument.

Returns:
thePath of current Sequence.

Algorithm:
Handed thePath, retrieves CL from CLines, displays the CL and all choices. Prompts user to make a choice
stop. Returns thePath of current CL if stop else recurse on thePath += seq.

Notifies user if at a join
automatic return at a terminal.

editASequence

Function:
Change the text of Before, After or a choice or to reset a joInto of an existing Sequence.

Parameters:

Returns:

Algorithm:
For a choice delete the old choice, calling deleteAChoice, then add the new choice and update the net.

editSTNet

Function:
Allow the user to choose a Sequence to be edited and then to branch to the appropriate edit functions with reference this current Sequence.

Branch to:
addAChoice
deleteAChoice
editAChoice

Parameters:

Returns:

Algorithm:

newSequence

Function:
Add a new Sequence, Before, After and Choices, and then for each choice in the recurse to build a new Sequence.
Ids are built by adding the sequence, from 0, of the alphabetically sorted choices, to current Id.thePath

Sequences are be defined as:
RESOLVING a 'not sure' choice,
JOINING a sequence onto another sequence,
TERMINAL - this stops the recursion,
DEFAULT - normal.

Summary:
Buffer

Values:

Constraints:

SERVICES

Purpose:
The class function forEach requires a void*. The IdChange structure is passed along with the function, such UpdateSequence, with the information to identify the sequence to change, the position of the change and the chara to insert/replace.

General:

Use:

ASSOCIATIONS WITH

ATTRIBUTES

cutOut

Summary:
The number of characters to cut-out of thePath for the string function replaceInStr

Values:

Constraints:

fromPos

Summary:
The position in thePath to start inserting characters when using the string function replaceInStr

Values:

Constraints:

joInto

Summary:
thePath of the sequence that this TheId joins into

Values:

Constraints:

thePath

Summary:
Value of transition sequences for a path in a STNet

Values:

Constraints:

OBJECT: RSTNet

Purpose:
Run an STNet

General:

Use:

ASSOCIATIONS WITH

(Gen) STNet

ATTRIBUTES

PSession

Summary:
Pointer to the SessionR

Values:

Constraints:

SERVICES

RSTNet(filename, startAt)

Function:

Parameters:

Returns:

Algorithm:

runSTNet(theId)

Function:

Parameters:

Returns:

Algorithm:
Retrieve the Sequence from SequenceTree by theId.
Switch on action:
JOINING
TERMINAL
FUNCTION
DEFAULT

~RSTNet( )

Function:

Parameters:

Returns:

Algorithm:

Purpose:
The object containing ChoiceLines for a STNet.
Services: stream operators to read & write CLs.

General:

Use:
Base class for building and running STNets.

ASSOCIATIONS WITH

(Spec) BSTNet(Spec) RSTNet

ATTRIBUTES

PNetObj

Summary:
Pointer to a NetObj. The NetObj is to be considered as current and local for the STN that is being run. The N is to contain the appropriate data and functions for the STN.
The reference to the object is made by the constructor searching the global array NetObject uses a match of STN filename to object.nameOf( ).
Values:

Constraints:

STNet( )

Summary:
The destructor for a STN.
Flushes the SequenceTree.

Values:

Constraints:

STNet(const char * filename)

Summary:
Constructor taking the name of the file for the SequenceTree. If file not opened sets its attribute fileName to NULL, as a flag.
Uses the filename (ex extension) as the name of the NetObj to reference in the global array NetObjects, if a specific obect is not found then it uses NetObj.

Values:

Constraints:

SequenceTree

Summary:
Container for Sequences.

Values:
The container must have specific stream operators for Sequences.

Constraints:

description

Summary:
A description for this STN.

Values:

Constraints:

fileName

Summary:
The file name for the Sequences in the SequenceTree.
Passed into the constructor for file opening and reading.
Used by the destructor for file writing.
Type:
TString
Values:
const char*

Constraints:
DOS filename

SERVICES

ifstream>>

Function:
Read Sequences from fill into the SequenceTree

Parameters:

Returns:

Algorithm:

ofstream<<

Function:

Parameters:

Returns:

Algorithm:

ostream<<

Function:
To display the Sequences in the SequenceTree

Parameters:

Returns:

Algorithm:

Purpose:

General:

Use:

ASSOCIATIONS WITH

ATTRIBUTES

action

Summary:
To flag specific operations for a Sequence

Values:
Actions are enum in the Sequence class.
{base, error, terminal, joining, noAction, upper}
0 1 2 3 4 5
Constraints:

after

Summary:
Text string for the 'after' part of the sentence.

Values:

Constraints:

before

Summary:
Text string for the 'before' part of the sentence.

Values:

Constraints:

changedThePath

Summary:
A flag when updating the Sequnces in the net after changing the sequences of a transition eg, deleting a ch

Values:

Constraints:

Summary:
A flag when updating the Sequnces in the net after changing the sequences of a transition eg, deleting a ch

Values:

Constraints:

choices

Summary:
Alphabetically sorted array of choices for the sentence. The sorted order provides a match of choice to a nu
to be used in building the path of the Id of each Sequence in a net.
Values:

Constraints:

theId

Summary:
A TheId structure containing the path, the joInto and notSure required to navigate a STN.

Values:

Constraints:

SERVICES

ifstream>>

Function:

Parameters:

Returns:

Algorithm:

istream>>

Function:

Parameters:

Returns:

Algorithm:

ofstream<<

Function:

Parameters:

Returns:

Algorithm:

ostream<<

Function:

Parameters:

Returns:

Algorithm:

Purpose:
Contain the relevant information to identify CLs in a STNet

General:

Use:

ASSOCIATIONS WITH

ATTRIBUTES

joInto

Summary:
thePath of a sequence in a STNet that this sequnce is joining

Values:

Constraints:

notSure

Summary:
The length of the current sequence of thePath where a 'not sure' choice was made.

Purpose:
The 'not sure' sequence joins back to the 'not sure' CL. It should resolve the 'not sure' choice to another choi back at that position on the sequence.

Constraints:

thePath

Summary:
String of the starting point and subsequent sequenced transitions of CLs in a STNet

Values:

Constraints:

SERVICES

ifstream>>

Function:

Parameters:

Returns:

Algorithm:

Appendix 6 - C++ Code (on disk)

Building a Chart

Run the file BUILDSTN.EXE.

The user is presented with a list of the SEC file in the current directory and asked to

Edit a previous SEC, or a New SEC, or Quit (E / N / Q):

Building a new SEC

At the opening screen the user enters 'N'. The program then lists the ChartObjects available. If a SEC Chart exists for the object then this will be displayed.

Edit a previous SEC, or a New SEC, or Quit (E / N / Q): **N**
The ChartObjects (Containers) available to you are:
1 ChartObject ChartObject
2 Person person
3 marriage Marriage
4 Asset asset
5 Gift gift
6 Inherita inheritance
7 InDisput in dispute
8 Business business
9 Jewelry jewelry
10 RealEsta real estate
11 Furnitur furniture
12 Superann superannuation
13 Liabilit liability
14 Vehicle vehicle

Enter the sequence number for the object: **6**
Building new SEC, filename **Inherita.stn**
Enter the SEC description :**Chart to resolve Common Pool Status for Inheritance.**

The Start of all Sequences in this SEC

This decision is 'C'oncluding, 'R'esolving, 'J'oining,
<Return> to continue the path or 'Q'uit:
Enter before :**The inheritance**
Attach a function to this question (Y/N): **N**
Enter words that define a choice : **was**

Another a choice (X - end choices): **was not**
Another a choice (X - end choices): **X**
Enter the words to complete choice:**intended to benefit both parties.**

This completes one sequence of a Chart. The program then continues, recurses, with the continuing sequences for each choice of the current sequence until each choice reaches a conclusion.

For the choice :The inheritance was intended to benefit both parties.
This decision is 'C'oncluding, 'R'esolving, 'J'oining,
<Return> to continue the path or 'Q'uit:

Editing a SEC

At the opening screen the user enters 'E', moves to the required sequence, selects the part of the sequence to be changed and enters the new detail.

Edit a previous SEC, or a New SEC, or Quit (E / N / Q): **E**
Enter Add a choice , Delete a choice , Edit a Sequence or Quit:
Enter (A / D / E / Q): **E**
Enter id or Browse (B):**B**
The business 1 `does' 2 `does not' 3 `not sure' involve parties to the marriage and no-one else.
Enter 1, 2 .. the sequence or Stop : **S**
The buses 1 `does' 2 `does not' 3 `not sure' involve parties to the marriage and no-one else.
For this Sequence do you wish to :
edit the B efore, A fter or C hoice
the J oin (reset a JOIN or make a Seqn into a JOIN)
or Q uit: **B**
Enter the new BEFORE :**The business**

Running a Chart

Run the file RUNSTN.EXE.

The user is presented with a list of the DAT file in the current directory and asked to

The main menu for all Sessions is the SEC "SESSIONR.SEC"
The program is Menu/Choice driven
Some control is affected by the following keys
ESC to return to MAIN MENU.
'~' to go back one SEQUENCE.
'x' to end a SESSION.
'x' to end a SESSION.
The previous Sessions are:

1 GIFT.DAT Example of gift.
2 FURNITUR.DAT Example of furniture.
'N'ew, 'L'oad or 'V'iew prevoius session or 'Q'uit (N/L/V/Q):

Running a New Session

The use enters 'N' at the main menu and continues to enter appropriate menu selections to step through the appropriate Chart, in the following example this is the BUSINESS SEC.

'N'ew, 'L'oad or 'V'iew previous session or 'Q'uit (N/L/V/Q): **N**

MAIN MENU : 1 `Assets Common Pool Determination' 2 `Marriage details' 3 `Personal details' .
Enter 1, 2, .. the sequence of your choice **1**
ASSETS MENU : 1 `Financial' 2 `Gift' 3 `none of these' .
Enter 1, 2, .. the sequence of your choice **3**
ASSETS SECOND MENU : 1 `Business' 2 `Furniture' 3 `Jewelry' 4 `Real estate' .
Enter 1, 2, .. the sequence of your choice **1**
The business 1 `does' 2 `does not' 3 `not sure' involve parties to the marriage and no-one else.
Enter 1, 2, .. the sequence of your choice **2**
The business 1 `was' 2 `was not' acquired during the marriage.
Enter 1, 2, .. the sequence of your choice **1**
The parties 1 `do' 2 `do not' have a controlling interest in the business
Enter 1, 2, .. the sequence of your choice **1**
There 1 `is' 2 `is not' evidence of a sham.
Enter 1, 2, .. the sequence of your choice **2**

The concluding sequence is :

Add to Pool

The user is shown the conclusion for this asset. The user enters **X** at the Main Menu and is prompted for a description for this session. This session is saved in the DAT file.

Viewing a Previous Session

The use enters 'V at the main menu and select the previous session from the list and then enters the name of the asset they wish to see. The sequence of steps taken in the session for this asset is 'played back' and the conclusion displayed.

'N'ew, 'L'oad or 'V'iew previous session or 'Q'uit (N/L/V/Q): **V**

1 BUSINESS.DAT Business example.
2 FURNITUR.DAT Example of furniture.
3 GIFT.DAT Example of gift.
Choose the number of the Session **1**

From the Net Objects run

business

Enter the name of the ChartObjectect to view trace (the name is the left column above) :**business**

The business does not involve parties to the marriage and no-one
The business was acquired during the marriage.
The parties do have a controlling interest in the business
There is not evidence of a sham.
The business does involve parties to the marriage and no-one else

The conclusion is :Add to Pool.
Enter to view another or Q uit:

STN File Structure

The ChartObjects for a chart are held in SEC files. The following example is from BUSINESS.SEC.

Determine Common Pool Status for a business	*Chart description*
~	*delimiter*
0	*object id*
0	*join into*
0	*not sure*
The business	*before text*
involve parties to the marriage and no-one else.	*after text*
3	*number of choices*
not sure	*choice*
does not	*choice*
does	*choice*
~	*delimiter*
01	*object id*
etc	

The C++ code for this structure is defined in the files THEID.H, SEQUENCE.H.

DAT File Structure

The DAT files contains a list of all the available ChartObjects. The BUSINESS.DAT file for the above session is as follows:

Business example.	*description*
~	*delimiter*

```
1                                   chartobject identifier
ChartObject                         chartobject name
0                                   path through chart
~
2
        :
        :
8                                   chartobject identifier
business                        chartobject name
01001                               path through chart
~
9
jewelry
0
~
```

Appendix 6 - C++ Code (on disk)

Building a Chart

Run the file BUILDSTN.EXE.

The user is presented with a list of the SEC file in the current directory and asked to

Edit a previous SEC, or a New SEC, or Quit (E / N / Q):

Building a new SEC

At the opening screen the user enters 'N'. The program then lists the ChartObjects available. If a SEC Chart exists for the object then this will be displayed.

Edit a previous SEC, or a New SEC, or Quit (E / N / Q): **N**
The ChartObjects (Containers) available to you are:
1 ChartObject ChartObject
2 Person person
3 marriage Marriage
4 Asset asset
5 Gift gift
6 Inherita inheritance
7 InDisput in dispute
8 Business business
9 Jewelry jewelry
10 RealEsta real estate
11 Furnitur furniture
12 Superann superannuation
13 Liabilit liability
14 Vehicle vehicle

Enter the sequence number for the object: **6**
Building new SEC, filename **Inherita.stn**
Enter the SEC description :**Chart to resolve Common Pool Status for Inheritance.**

The Start of all Sequences in this SEC

This decision is 'C'oncluding, 'R'esolving, 'J'oining,
<Return> to continue the path or 'Q'uit:
Enter before :**The inheritance**
Attach a function to this question (Y/N): **N**
Enter words that define a choice : **was**

Another a choice (X - end choices): **was not**
Another a choice (X - end choices): **X**
Enter the words to complete choice:**intended to benefit both parties.**

This completes one sequence of a Chart. The program then continues, recurses, with the continuing sequences for each choice of the current sequence until each choice reaches a conclusion.

For the choice :The inheritance was intended to benefit both parties.
This decision is 'C'oncluding, 'R'esolving, 'J'oining,
<Return> to continue the path or 'Q'uit:

Editing a SEC

At the opening screen the user enters 'E', moves to the required sequence, selects the part of the sequence to be changed and enters the new detail.

Edit a previous SEC, or a New SEC, or Quit (E / N / Q): **E**
Enter Add a choice , Delete a choice , Edit a Sequence or Quit:
Enter (A / D / E / Q): **E**
Enter id or Browse (B):**B**
The business 1 `does' 2 `does not' 3 `not sure' involve parties to the marriage and no-one else.
Enter 1, 2 .. the sequence or Stop : **S**
The buses 1 `does' 2 `does not' 3 `not sure' involve parties to the marriage and no-one else.
For this Sequence do you wish to :
edit the B efore, A fter or C hoice
the J oin (reset a JOIN or make a Seqn into a JOIN)
or Q uit: **B**
Enter the new BEFORE :**The business**

Running a Chart

Run the file RUNSTN.EXE.

The user is presented with a list of the DAT file in the current directory and asked to

The main menu for all Sessions is the SEC "SESSIONR.SEC"
The program is Menu/Choice driven
Some control is affected by the following keys
ESC to return to MAIN MENU.
'~' to go back one SEQUENCE.
'x' to end a SESSION.
'x' to end a SESSION.
The previous Sessions are:

1 GIFT.DAT Example of gift.
2 FURNITUR.DAT Example of furniture.
'N'ew, 'L'oad or 'V'iew prevoius session or 'Q'uit (N/L/V/Q):

Running a New Session

The use enters 'N' at the main menu and continues to enter appropriate menu selections to step through the appropriate Chart, in the following example this is the BUSINESS SEC.

'N'ew, 'L'oad or 'V'iew previous session or 'Q'uit (N/L/V/Q): **N**

MAIN MENU : 1 `Assets Common Pool Determination' 2 `Marriage details' 3 `Personal details' .
Enter 1, 2, .. the sequence of your choice **1**
ASSETS MENU : 1 `Financial' 2 `Gift' 3 `none of these' .
Enter 1, 2, .. the sequence of your choice **3**
ASSETS SECOND MENU : 1 `Business' 2 `Furniture' 3 `Jewelry' 4 `Real estate' .
Enter 1, 2, .. the sequence of your choice **1**
The business 1 `does' 2 `does not' 3 `not sure' involve parties to the marriage and no-one else.
Enter 1, 2, .. the sequence of your choice **2**
The business 1 `was' 2 `was not' acquired during the marriage.
Enter 1, 2, .. the sequence of your choice **1**
The parties 1 `do' 2 `do not' have a controlling interest in the business
Enter 1, 2, .. the sequence of your choice **1**
There 1 `is' 2 `is not' evidence of a sham.
Enter 1, 2, .. the sequence of your choice **2**

The concluding sequence is :

Add to Pool

The user is shown the conclusion for this asset. The user enters **X** at the Main Menu and is prompted for a description for this session. This session is saved in the DAT file.

Viewing a Previous Session

The use enters 'V at the main menu and select the previous session from the list and then enters the name of the asset they wish to see. The sequence of steps taken in the session for this asset is 'played back' and the conclusion displayed.

'N'ew, 'L'oad or 'V'iew previous session or 'Q'uit (N/L/V/Q): **V**

1 BUSINESS.DAT Business example.
2 FURNITUR.DAT Example of furniture.
3 GIFT.DAT Example of gift.
Choose the number of the Session **1**

From the Net Objects run

business

Enter the name of the ChartObjectect to view trace (the name is the left column above) :**business**

The business does not involve parties to the marriage and no-one
The business was acquired during the marriage.
The parties do have a controlling interest in the business
There is not evidence of a sham.
The business does involve parties to the marriage and no-one else

The conclusion is :Add to Pool.
Enter to view another or Q uit:

STN File Structure

The ChartObjects for a chart are held in SEC files. The following example is from BUSINESS.SEC.

Determine Common Pool Status for a business — *Chart description*
~ — *delimiter*
0 — *object id*
0 — *join into*
0 — *not sure*
The business — *before text*
involve parties to the marriage and no-one else. — *after text*
3 — *number of choices*
not sure — *choice*
does not — *choice*
does — *choice*
~ — *delimiter*
01 — *object id*
etc

The C++ code for this structure is defined in the files THEID.H, SEQUENCE.H.

DAT File Structure

The DAT files contains a list of all the available ChartObjects. The BUSINESS.DAT file for the above session is as follows:

Business example. — *description*
~ — *delimiter*

```
1                                          chartobject identifier
ChartObject                                chartobject name
0                                          path through chart
~
2
          :
          :
8                                          chartobject identifier
business                                chartobject name
01001                                      path through chart
~
9
jewelry
0
~
```

Appendix 6 - C++ Code (on disk)

Building a Chart

Run the file BUILDSTN.EXE.

The user is presented with a list of the SEC file in the current directory and asked to

Edit a previous SEC, or a New SEC, or Quit (E / N / Q):

Building a new SEC

At the opening screen the user enters 'N'. The program then lists the ChartObjects available. If a SEC Chart exists for the object then this will be displayed.

Edit a previous SEC, or a New SEC, or Quit (E / N / Q): **N**
The ChartObjects (Containers) available to you are:
1 ChartObject ChartObject
2 Person person
3 marriage Marriage
4 Asset asset
5 Gift gift
6 Inherita inheritance
7 InDisput in dispute
8 Business business
9 Jewelry jewelry
10 RealEsta real estate
11 Furnitur furniture
12 Superann superannuation
13 Liabilit liability
14 Vehicle vehicle

Enter the sequence number for the object: **6**
Building new SEC, filename **Inherita.stn**
Enter the SEC description :**Chart to resolve Common Pool Status for Inheritance.**

The Start of all Sequences in this SEC

This decision is 'C'oncluding, 'R'esolving, 'J'oining,
<Return> to continue the path or 'Q'uit:
Enter before :**The inheritance**
Attach a function to this question (Y/N): N
Enter words that define a choice : **was**

Another a choice (X - end choices): **was not**
Another a choice (X - end choices): **X**
Enter the words to complete choice:**intended to benefit both parties.**

This completes one sequence of a Chart. The program then continues, recurses, with the continuing sequences for each choice of the current sequence until each choice reaches a conclusion.

For the choice :The inheritance was intended to benefit both parties.
This decision is 'C'oncluding, 'R'esolving, 'J'oining,
<Return> to continue the path or 'Q'uit:

Editing a SEC

At the opening screen the user enters 'E', moves to the required sequence, selects the part of the sequence to be changed and enters the new detail.

Edit a previous SEC, or a New SEC, or Quit (E / N / Q): **E**
Enter Add a choice , Delete a choice , Edit a Sequencé or Quit:
Enter (A / D / E / Q): **E**
Enter id or Browse (B):**B**
The business 1 `does' 2 `does not' 3 `not sure' involve parties to the marriage and no-one else.
Enter 1, 2 .. the sequence or Stop : **S**
The buses 1 `does' 2 `does not' 3 `not sure' involve parties to the marriage and no-one else.
For this Sequence do you wish to :
edit the B efore, A fter or C hoice
the J oin (reset a JOIN or make a Seqn into a JOIN)
or Q uit: **B**
Enter the new BEFORE :**The business**

Running a Chart

Run the file RUNSTN.EXE.

The user is presented with a list of the DAT file in the current directory and asked to

The main menu for all Sessions is the SEC "SESSIONR.SEC"
The program is Menu/Choice driven
Some control is affected by the following keys
ESC to return to MAIN MENU.
'~' to go back one SEQUENCE.
'x' to end a SESSION.
'x' to end a SESSION.
The previous Sessions are:

1 GIFT.DAT Example of gift.
2 FURNITUR.DAT Example of furniture.
'N'ew, 'L'oad or 'V'iew prevoius session or 'Q'uit (N/L/V/Q):

Running a New Session

The use enters 'N' at the main menu and continues to enter appropriate menu selections to step through the appropriate Chart, in the following example this is the BUSINESS SEC.

'N'ew, 'L'oad or 'V'iew previous session or 'Q'uit (N/L/V/Q): **N**

MAIN MENU : 1 `Assets Common Pool Determination' 2 `Marriage details' 3 `Personal details' .
Enter 1, 2, .. the sequence of your choice **1**
ASSETS MENU : 1 `Financial' 2 `Gift' 3 `none of these' .
Enter 1, 2, .. the sequence of your choice **3**
ASSETS SECOND MENU : 1 `Business' 2 `Furniture' 3 `Jewelry' 4 `Real estate' .
Enter 1, 2, .. the sequence of your choice **1**
The business 1 `does' 2 `does not' 3 `not sure' involve parties to the marriage and no-one else.
Enter 1, 2, .. the sequence of your choice **2**
The business 1 `was' 2 `was not' acquired during the marriage.
Enter 1, 2, .. the sequence of your choice **1**
The parties 1 `do' 2 `do not' have a controlling interest in the business
Enter 1, 2, .. the sequence of your choice **1**
There 1 `is' 2 `is not' evidence of a sham.
Enter 1, 2, .. the sequence of your choice **2**

The concluding sequence is :

Add to Pool

The user is shown the conclusion for this asset. The user enters **X** at the Main Menu and is prompted for a description for this session. This session is saved in the DAT file.

Viewing a Previous Session

The use enters 'V at the main menu and select the previous session from the list and then enters the name of the asset they wish to see. The sequence of steps taken in the session for this asset is 'played back' and the conclusion displayed.

'N'ew, 'L'oad or 'V'iew previous session or 'Q'uit (N/L/V/Q): **V**

1 BUSINESS.DAT Business example.
2 FURNITUR.DAT Example of furniture.
3 GIFT.DAT Example of gift.
Choose the number of the Session **1**

From the Net Objects run

business

Enter the name of the ChartObjectect to view trace (the name is the left column above) :**business**

The business does not involve parties to the marriage and no-one
The business was acquired during the marriage.
The parties do have a controlling interest in the business
There is not evidence of a sham.
The business does involve parties to the marriage and no-one else

The conclusion is :Add to Pool.
Enter to view another or Q uit:

STN File Structure

The ChartObjects for a chart are held in SEC files. The following example is from BUSINESS.SEC.

Determine Common Pool Status for a business	*Chart description*
~	*delimiter*
0	*object id*
0	*join into*
0	*not sure*
The business	*before text*
involve parties to the marriage and no-one else.	*after text*
3	*number of choices*
not sure	*choice*
does not	*choice*
does	*choice*
~	*delimiter*
01	*object id*
etc	

The C++ code for this structure is defined in the files THEID.H, SEQUENCE.H.

DAT File Structure

The DAT files contains a list of all the available ChartObjects. The BUSINESS.DAT file for the above session is as follows:

Business example.	*description*
~	*delimiter*

```
1                                          chartobject identifier
ChartObject                                chartobject name
0                                          path through chart
~
2
        :
        :
8                                          chartobject identifier
business                                chartobject name
01001                                      path through chart
~
9
jewelry
0
~
```