University of Southampton

Event Calculus to Support Temporal Reasoning in a Clinical Domain

by

Geetha Kalyani Abeysinghe

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Engineering and Applied Science
Electronics and Computer Science

September 23, 1993

University of Southampton

ABSTRACT

Faculty of Engineering and Applied Science

Electronics and Computer Science

Doctor of Philosophy

Event Calculus to Support Temporal Reasoning
in a
Clinical Domain

by Geetha Kalyani Abeysinghe

This work concerns temporal aspects of a knowledge based system which holds information on patients as they progress through their treatment in a vascular surgery department.

Representing and using knowledge about temporal relationships so as to provide decision support to a historical knowledge base of patient data is investigated. Event Calculus, in first order classical logic augmented with negation by failure, provides an effective framework for reasoning about time. From Kowalski and Sergot's original Event Calculus we arrive at a simple and flexible framework which can be used as a temporal support in a medical knowledge based system.

We show how Event Calculus can be used to describe a simple model of the clinical pathway in vascular surgery. Patient information in the medical record is formalised in a structural framework to suit the Event Calculus. Medical knowledge about investigation and treatment options is added to the model so that the resulting system can recommend the options which are appropriate at any particular time. It is shown how these recommendations provide decision support by recommending what should be done next, and when to re-evaluate measurements that become unreliable.

It is argued that there are advantages to be gained by adopting a general temporal reasoning framework because it can be extended to support various medical and administrative tasks. The extensions available to the Event Calculus, further its suitability as a temporal reasoning framework in the medical domain.

A prototype system, essentially a research workbench over a realistic domain, is built using Prolog to illustrate the temporal reasoning capabilities provided by the Event Calculus framework. Using case studies it is demonstrated how the prototype system fulfils the decision support abilities we aimed to achieve in the knowledge base.

An alternative framework for describing processes, Process Modelling implemented in the language PML, has been compared with the Event Calculus framework. On the basis of formulating examples in both frameworks as executable programs, we conclude that PML enforces normative behaviour in a process particularly well, whereas Event Calculus is more suited to decision support.

It is concluded that the Event Calculus framework allows many useful features of medical information systems to be rationally reconstructed in a very clear way. This makes it easier to build systems which can be modified and enhanced. Thus the framework provides not only ease of representation, but also ease of maintenance and further development.

Acknowledgments

My humble gratitude to my parents, especially my father who supported, and encouraged me throughout my education.

I wish to thank Dr. Wendy Hall and Dr. Gillian Lovegrove for their support and encouragement during the initial stages of my studentship. Thanks to my colleagues at Southampton, especially Andy King and Andy Verden who were immensely helpful when I was struggling to get adjusted to a new country and a new culture. Thanks to Xiaoying Fu and Salleh Syed-Mustaffa, their friendship and humour lightened my gruelling days.

Thanks are due to Dr. Dave De Roure who took over my supervision for a short period, for his assistance, guidance, and encouragement and for always having time to listen to my problems.

My immense gratitude to Charles Ranaboldo for his valuable time spent with us: his help in introducing us to the intricacies of vascular disease and surgery; helping with the knowledge assimilation; and testing the prototype system using patient data. Thanks are also due to Tony Chant at the Royal South Hampshire hospital, who first instigated this research and imparted his expertise through out our work.

I am indebted to Paul Soper who supervised me from the very beginning. For introducing me to logic programming, for his guidance and supervision through out my research. I thank him for his tremendous patience and immeasurable understanding of my shortcomings, especially during my initial days. For any research student the initial days are difficult, more so when one also has to get adjusted to a different country and a different way of life. Thank you, Paul, for smoothing the hard edges of my early days, for your invaluable support and advice in all matters throughout my stay in Southampton.

My thanks to the Committee of Vice-Chancellors and Principals of the Universities of the U.K. for giving me financial support for the academic period 1989-1991. I would also like to thank Savithri Kariyawasam for her support during my second year.

My thanks to the IOPT board for their financial support for the work on process modelling. Thanks to professor Peter Henderson for directing us towards that research, and to Bob Snowden for his help and useful comments. I would like to thank Mark

Greenwood, for installing the PSS system, helping me to understand PML, and for his time and patience throughout the IOPT work. I thank him especially for reading my work regarding process modelling, and his valuable comments on it.

Finally, I wish to thank my husband Vijay who was an invaluable support; his immense patience and understanding especially during the last days of my write up. Thank you for taking care of me and encouraging me through my difficult days.

Contents

List of Figures

1 Introduction

> **Where there is action, there is time, and where there is time, there is potential need for reasoning about time (Perlis, Elgot-Drupkin and Miller, 1991).**

Representing time in data bases has become a research topic in many areas (McKenzie, 1986; Bolour, Anderson, Dekeyser and Wong, 1982; Snodgrass, 1990; Maiocchi and Pernici, 1991). Whatever method may be used in knowledge representation and retrieval – a relational data base, a deductive data base – it is necessary to represent the changing world. Information stored in the data base changes continuously by updating, correction and revision. When representing an item of information, let it be a person's personal record, a company's financial status or a patient's diseased condition, it is not sufficient just to represent the current status of affairs, but the past too has to be represented appropriately. The validity of a storage or retrieval operation is partly determined by the information being placed in the right temporal context as events in the outside world impinge upon it.

Even for data base applications the need to represent time is evident. For example, in representing a person's personal record, changes to name, address, job history, marital status and so on, have to be independent of one another. It is not very desirable to store a copy of the record each time one or the other gets updated. Especially in Artificial Intelligence (AI), where the aim is to duplicate, model, or simulate human intelligence and actions, a proper treatment of time is necessary. In recent years, many works on AI concerning methods of representing time and reasoning about it have appeared (Hajnicz, 1990; Perlis, Elgot-Drupkin and Miller, 1991; Koomen, 1991; Freksa, 1992; Wiederhold, Jajodia and Litwin, 1991; Allen, 1991). The subfield of artificial intelligence that acknowledges a central role to time is usually referred to as temporal reasoning (Nardi and Tucci, 1989).

Medical information systems have to deal with the problem of out-dated data and

new data, and the need arises to represent temporal information. In applications such as keeping medical records and patient management, the relative time happenings of events plays a critical part and thus requires a problem solving model which can capture effectively change of information/knowledge with time. Often, the patient history is the only available means for establishing the diagnosis (Kohane, 1987). To be able to effectively use such a patient history, it is necessary that the automated systems be capable of sophisticated temporal reasoning.

The general aim of this thesis is to apply knowledge based system techniques to the problem of keeping medical records and the support of patient clinical management. A basic data base approach to keeping clinical records can be extended in two broad ways: introducing a representation of time to the data base; and extending the data base to a knowledge base which stores implicit as well as explicit information. We make both these extensions. What is new about our approach however, is that a general temporal reasoning framework is built into the basic structure of the knowledge based system, rather than being tackled in an ad hoc way. Our aim in doing this is two fold: firstly to show that general temporal reasoning techniques, in particular the Event Calculus, can be scaled up to a domain considered in realistic detail; and secondly to lay the groundwork for a new generation of medical information systems built on this more general framework.

1.1 Time in Data Bases

There are three concepts of time identified when dealing with temporal information, event time (or valid time), transaction time and user-defined time (Snodgrass and Ahn, 1985). **Event time** is the real time at which events occur in the real world. **Transaction time** denotes the time at which information was received by the data base or the knowledge base. **User-defined time** is necessary when neither of the former defined two times can handle the available temporal information. As an example of user-defined time we can consider the well known 'promotion' example (Snodgrass and Ahn, 1985; Snodgrass and Ahn, 1986; Sripada, 1988). Consider the event of promoting John to assistant sales manager. The event description will have three attributes associated with it; Name, Rank, Approved-Date. The Approved-Date, the date when

the promotion was approved by the board of directors, is the user-defined time. The date when the promotion takes effect is the event time. The date the information about the event is received by the data base is the transaction time. We are primarily interested in the event time. Most explicitly, in this thesis general temporal reasoning will be applied to event time only. We believe that this alone suffices to represent time effectively in the domain of clinical practice since what affects the patient's condition is not the decision but the action.

The other main candidate for temporal reasoning is transaction time. It is simpler than event time in several ways and might be included in a more complete system. One feature of transaction times is that they are not subject to revision. The transaction time associated to event information is usually automatically added by a built-in clock in the system and hence not only are the transaction times in chronological order but also once added there is no reason to change them again. The event time contrarily may be corrected or revised. The event of correction or revision itself will be associated with a new transaction time. Further reading on transaction time can be found in (Sripada, 1988).

How we represent the event time, whether simply by an integer or by a calendar date, depends on the domain modelled, and the user requirements. The representation of transaction time is totally independent of the application domain.

1.2 Types of Data Bases and Knowledge Bases

Based on event time and transaction time, four types of data bases can be defined (a similar classification applies to knowledge bases). Each differ in their ability to support the two time concepts mentioned in Section 1.1.

Snapshot Data Bases The data base at any instant represents a snapshot of the world modelled at that instant. Snapshot data bases do not deal with time in any special way i.e. they do not support any concept of time. As information stored in the data base changes or get updated, the old information gets replaced by the new information. Thus, the old information or the past states of the data base are not remembered.

Rollback Data Bases The data base supports transaction time, the time at which information is received by the data base. All past states of the data base are stored, indexed by time. As a result rather than the history of the world modelled, the history of the data base activities are recorded. By taking a snapshot of the data base at any past instant, a snapshot of any relation at that instant can be obtained. The action of selecting a past state of the data base is known as *rollback*. Data bases which supports the action of rollback are known as *rollback data bases*.

Changes to a rollback data base can be made to the most current state or snapshot. There is no way of correcting errors of past. Hence, past errors cannot be forgotten.

Historical Data Bases The data base supports event time, thus the time interval for which a relation is valid in the real world is represented. A historical record is kept per relation. As errors occur they are corrected by modifying the data base accordingly. No record is kept of the errors corrected. Since transaction time is not supported at all, past states of the data base are lost, i.e. the data base cannot be viewed as at any past instant of time.

Temporal Data Bases Temporal data bases support both transaction time and event time. Therefore, they capture completely the retroactive/proactive changes.

In this thesis we talk about a knowledge base rather than a data base. The knowledge base is a more general concept than a data base represented in our formulation as a deductive data base. Deductive data bases represent the convergence of data bases and logic programming (Grant and Minker, 1992).

A deductive data base is a finite set of deductive rules of the form,

$$A \longleftarrow B_1, B_2, \ldots, B_n \qquad \text{where } n \geq 0$$

where A is an atom, and the B_is are literals (that is, atoms or negated atoms), and all the variables are assumed to be universally quantified in front of the formula in which they occur. A is called the *conclusion* of the rule and the B_is the *conditions*. When n=0, the deductive rule is called a fact. A deductive data base provides a declarative, logic-based language for expressing queries, reasoning, and complex applications on data bases (Minker,1988; Ullman, 1990). The logic underlying deductive data bases

is the Horn Clause subset of first-order predicate calculus extended with a meta-level inference rule, negation by failure, to implement negation (Gallaire, Minker and Nicolas, 1984). Negation by failure is an efficient way to express the completeness of the positive information stored in the knowledge base (Clark, 1978).

The typology of data bases as regards time, given above, carries over to knowledge bases. In this thesis we develop a historical model of a deductive data base which captures the history of the events which occur in the domain as is best known at any given instant of time.

1.3 Temporal Information Management

There is much ongoing research on temporal information management (Maiocchi and Pernici, 1991; Snodgrass and Ahn, 1986; Dean and McDermott, 1987). Work has been published regarding the taxonomy (Snodgrass and Ahn, 1985) and concepts (Jensen, Clifford, Gadia, Segev and Snodgrass, 1992) of temporal information.

In temporal data management, two main directions of research can be identified (Maiocchi and Pernici, 1991):

- extending data models to incorporate time dimension,
- providing temporal reasoning capabilities to knowledge based systems.

In this thesis we are interested in the latter aspect of research. Within this category, our interest lies with Artificial Intelligence systems which are concerned with interpreting temporal data and their relationships.

A significant amount of common-sense reasoning involves time in one aspect or another. Many of the facts we are accustomed to deal with change over time, and hence many of the inferences we make depend crucially upon whether or not a fact or conjunction of facts is true at a point, or throughout an interval. We use Event Calculus, a representation within a logic-programming framework introduced by Kowalski and Sergot (Kowalski and Sergot, 1986), to reason about time in a knowledge base. Event Calculus takes events as the basic notion of the ontology while being neutral as to the duration of the event itself.

The Event Calculus formalism is based on first order predicate logic. Unlike conventional temporal logics, which represent time implicitly using modal operators

such as *past* and *future* to reason about time, the Event Calculus represents time explicitly. We now discuss briefly two other temporal reasoning formalisms based on predicate logic with an explicit representation of time.

Allen (Allen, 1983) deals with the problem of temporal knowledge being relative and therefore not necessarily being representable by a date. The basis of Allen's approach consists of an interval-based temporal logic together with a computationally effective reasoning algorithm based on constraint propagation. In Allen's theory, to specify a particular occurrence of an event a unique time interval over which the event takes place has to be specified. Events are described by the predicate *occur(E, P)*, denoting that event *E* occurred over the time interval *P*. Likewise a predicate *holds(U, P)* is used to denote that the property/relation *U* holds throughout the interval *P*. The characteristic feature of Allen's formalism is his ontology of time expressed in terms of the time intervals *P*. He uses 13 mutually exclusive relations to relate two time intervals. For any given two intervals X and Y these time relations are as given in Figure 1.1 together with the inverses of the first six relations. Allen also introduces a

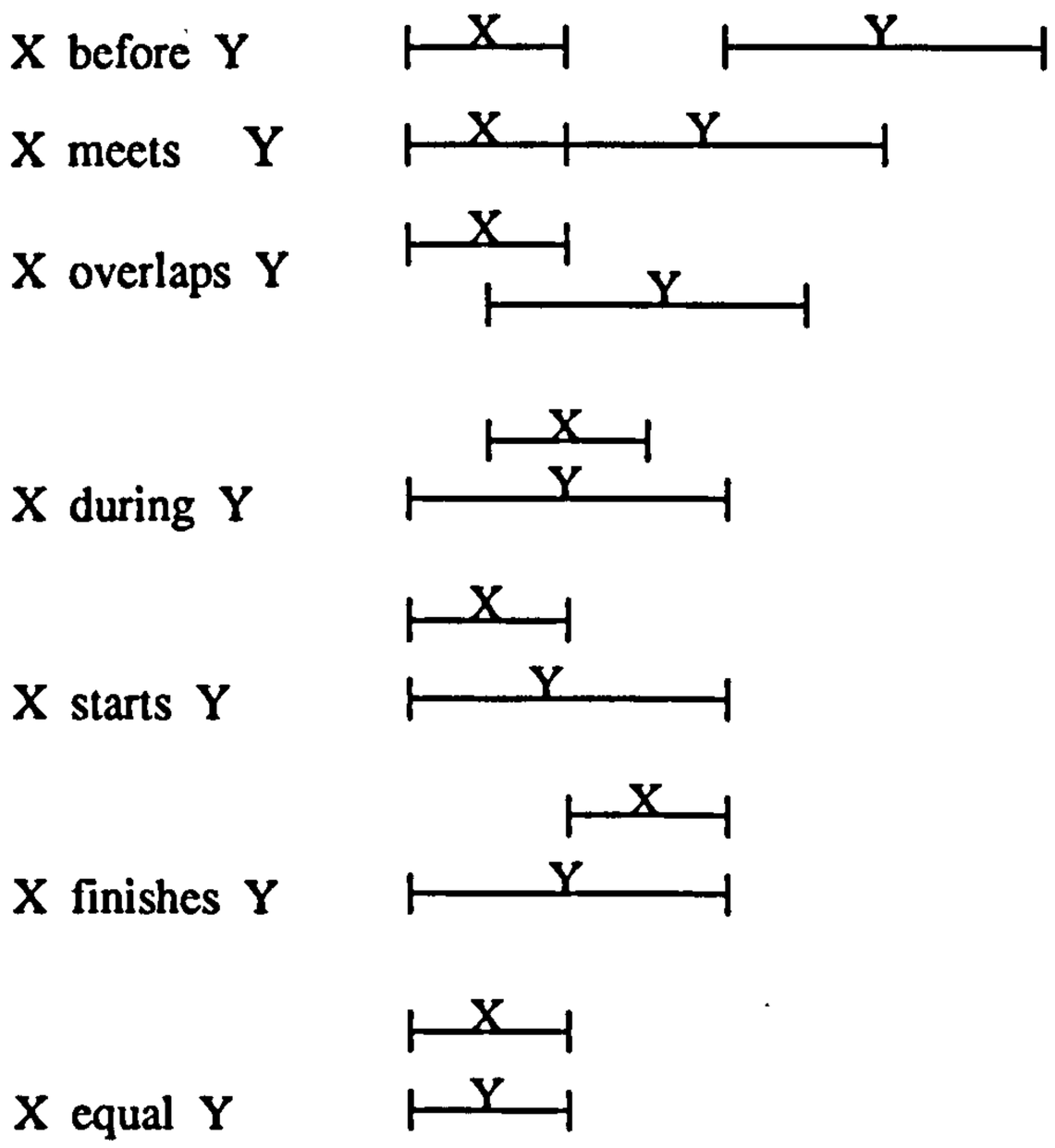


Figure 1.1: The mutually exclusive time relations of Allen's (Allen, 1983)

notion of reference interval which can capture the temporal hierarchy implicit in many domains (Allen, 1983; Maiocchi and Pernici, 1991).

The detailed relationship between Allen's logic and the Event Calculus has been studied by Sadri (Sadri, 1986). She argues that the two formalisms, insofar as they reason about events, time intervals and time-dependent properties, are closely related, and that the Event Calculus can be extended or specialised to provide the features of Allen's logic. For example, Allen has not given a logical formulation for default persistence although he suggests it is desirable in a temporal system. The Event Calculus does support default reasoning through negation as failure and in this sense has added temporal reasoning power. But the two formalisms can be made similar either by suppressing the negation as failure in the Event Calculus or by extending Allen's logic. On the other hand Allen has used his logic for a number of purposes which go beyond reasoning about events, such as the analysis of intention and belief, and in this aspect extends the Event Calculus. However, for our purposes, for reasoning about events, the two formalisms differ largely in style.

Another formalism based on predicate logic has been presented by Lee, Coelho and Cotta (Lee, Coelho and Cotta, 1985). They have aimed to develop a temporal system for representing and reasoning about time dependent information and events specifically for business data applications. Since Lee et al consider a specific domain, business application, it is convenient for them to take calendar dates as their basic ontology of time, dates represented in days, months and years. Events are associated with a definite time (an exact calendar date) or an indefinite time (by indicating that the event occurred sometime during an imprecisely specified date). Also events are identified with either time points or time intervals, the most common time points used in the domain being the end points of the time intervals. Time points and intervals are related by a predicate *during(T, P)* which expresses that time point *T* is during the time interval *P*.

Sadri (Sadri, 1986) also made a detailed comparison of Lee et al's formalism with the Event Calculus. Again she argues that the Event Calculus can be extended or specialised to give the features of Lee et al's formalism, but in this case the latter tends to be more specialised because it was set up for a particular application. We mention some of the similarities and differences between the formalisms which have been noted by Sadri. For Lee et al properties are assumed to persist unless it can be shown that they have terminated. Thus they include default reasoning in a manner similar to the

Event Calculus. Lee et al do not discuss the possibility of event descriptions being incomplete. For example, it is not clear how to add new knowledge about a more definite time of occurrence for an event which is already specified with an indefinite time. Completion of partial events is by inference and not by default. Thus, there are no withdrawals of the completions by default by any new information, unlike in the Event Calculus for which event descriptions may be completed by default using the information available. Lee et al's formalism requires events to be entered into the system in the order that they occur in the real world, whereas Event Calculus does not impose any restrictions on the order in which event occurrences are entered. In this respect Lee et al's approach is close to the special case of the Event Calculus where events are assumed to be entered into the data base in the order that they take place. Our general conclusion about Lee et al's formalism, following Sadri, is that it differs in style to Event Calculus and is a more specialised formalisation.

All the three formalisms, Allen's, Lee et al's and the Event Calculus, are similar in that, they analyse the concept of events, and events and time are represented explicitly. Events may occur simultaneously and imply the start or/and end points of time periods (or time dependent relations/properties). If the formalisms were extended to have similar temporal reasoning power, the differences would be largely ones of style of representation. One feature of the Event Calculus representation is that event occurrences are given unique names. In Allen's and Lee et al's formalisms an event occurrence is distinguished by the type of the event and the time of its occurrence. Having unique event names make adding new information about events which are already described easy. Describing simultaneous events of the same type is also made easier. The main reason we have chosen to develop our system with the Event Calculus, however, is because it is explicitly formalised in a logic programming style.

A somewhat different view of temporal data management emphasises the concept of process. Event Calculus can also be considered to be a process description framework. A process is a set of partially ordered steps intended to reach a goal (Feiler and Humphrey, 1993). An abstract representation of the process is called a process description. Process description languages can be evaluated by the extent to which they provide constructs useful for representing and reasoning about the various aspects of the process. The perspectives that a process description is able to present are

bounded by the constructs of the language used (Curtis, Kellner and Over, 1992). A more procedural alternative to the Event Calculus as a process description language is process modelling in the language PML. Process modelling languages provide a means to define the process so that it could be enacted by a machine. The enactment of the model can be completely represented only if the representing language is capable of modelling the history of the enactment. PML does not have the capability of representing such historical information, but Event Calculus, formulated in classical logic with an explicit representation of time, maintains a full historical knowledge base.

1.4 Aim of the Thesis

The general aim of our research was to design and prototype a medical information management system with the Event Calculus framework as its temporal back bone. The application domain, vascular surgery, was to be considered in sufficient detail to provide: (a) a significant test of the scalability of the Event Calculus to a realistic domain, and (b) an appreciation of the usefulness of the framework for further development in the domain.

In respect of this aim it is important to attempt to characterise the class of systems of which our application domain of vascular surgery is an exemplar; both so as to be clear what aspects of the Event Calculus are being tested; and also so as to indicate the range of applications which might benefit from a similar approach. We can distinguish two modes of use of the Event Calculus. On the one hand a *descriptive* mode in which it is assumed that the system has no influence on what events are added to the system. An example of this mode would be story understanding; the task of the system is to use its reasoning capability to make maximum sense of the incoming events with little regard for any possibility of actively seeking further information. On the other hand a *semi-prescriptive* mode in which the Event Calculus is used as a temporal information management system. In this mode the intention is that the knowledge stored in the system should both model the state of the application domain and guide future actions. This means that completeness of information is relatively important or, put another way, that it is important to distinguish between what is unknown and what

is false. As regards part (a) of the aim we believe that our application exercises the features of Event Calculus which are important in the semi-prescriptive mode rather than the descriptive mode. This will turn out to mean that we de-emphasise the use of backwards persistence and emphasise the interactive use of new events to prompt the search for further information. As regards part (b) of the aim we believe that our medical application is representative of knowledge based system applications which model the development of a process over time and provide some kind of semi-prescriptive function in the domain. Clearly decision support systems belong to this class.

To be clear our intention in this research was not to build a fully operational system to be used by the clinicians, but to build a prototype in which we can show that, given patient data as event instances, Event Calculus provides a good supporting basis for temporal reasoning for the domain of vascular surgery considered in realistic detail. The prototype is intended to be a research workbench, which can be used to explore further the benefits of the approach.

As components of our general aim, we had the following goals:

1. Arrive at a simple Event Calculus framework based on the more general framework introduced by Kowlaski and Sergot (Kowlaski and Sergot, 1986) to be used for medical information systems.

2. Find a representation for the domain dependent patient information in a framework suitable for Event Calculus.

3. Provide a support tool for patient management which,

   (a) provides recommendations,

   (b) supports better clinical practice,

   based on the 'expert approach to the problem.'

4. Compare and contrast the Event Calculus framework with an imperative process description formalism, process modelling in the language PML.

5. Based on the prototype to suggest possible extensions which allow further support to the clinical pathway by way of research, audit, and other useful

support functions.

1.5 Contributions of the Thesis

Our work is concerned with the temporal aspects of a knowledge based system which holds information about hospital patients as they progress through their treatments. The necessity to provide temporal support to knowledge based systems in the domain is established. Some relevant medical information systems are reviewed, especially those which have taken time into consideration.

Among the many theories available for reasoning with time we have chosen Event Calculus which provides an efficient way of dealing with time in a deductive data base. Event Calculus allows us to reconstruct the state of the world at any specific instant without having to store it explicitly. Recent work in extending the Event Calculus to include transaction time (Sripada, 1988) and macro events (Evans, 1989; Evans, 1990) enhance its suitability as a temporal support. In the thesis we arrive at a simple framework, based on the more general framework introduced by Kowalski and Sergot, which can be used as the temporal reasoning mechanism of a knowledge base in our domain.

The detailed domain dependent patient information for vascular surgery is represented in a framework suitable for Event Calculus. Ramifications are used as a simple and elegant way of knowledge representation. Although ramifications provide a very clear knowledge representation method, why it cannot serve as the only representation mechanism, especially in a clinical domain, is presented.

A prototype system, a research workbench, is developed based on our framework using the logic programming language Prolog. In the prototype system, we have combined the deductive capabilities of logic programming and the time reasoning capabilities of Event Calculus to develop a historical knowledge base which provides decision support in the domain of vascular surgery. How this system provides a decision support tool for patient management, providing recommendations for better clinical practice is demonstrated using case studies.

Process modelling refers to the problem of developing a formal representation of the tasks and division of labour in a particular setting (Maresh and Wastell, 1990).

Process programming is the construction of detailed, executable process descriptions, or programs. PML is a language in which process descriptions can be expressed. PML is investigated, and compared with Event Calculus as a process description framework.

Finally, possible further extensions to the work done so far are suggested.

1.6 Organisation of the Thesis

In Chapter 2, we describe the domain of Vascular surgery, the current practice, and the problems perceived by the clinicians. The clinical pathway which we model in the system is explained in detail identifying the different stages a patient goes through after being admitted to the hospital. The goals of the knowledge based system we designed are identified and the approach taken is described.

In Chapter 3, we describe a number of existing medical information systems, which handle time. They are described under the two broad categories: data bases and expert systems. The importance of representing patient data in its proper time context is established by presenting an overview of the medical record. The essential importance of time in the medical domain is re-affirmed by the overview. The variety in the application domains shows the necessity of representing time in medical information systems. Most of the systems have used traditional methods of knowledge representation and therefore entail having elaborate schemes for reasoning, in particular about time, and retrieval of information.

The Event Calculus framework used in the application domain is described in Chapter 4. The general framework originally introduced by Kowalski and Sergot is first described, then a general formalism is developed to suit the application domain requirements. Some other applications which have used the Event Calculus framework, and various extensions that have been proposed to Event Calculus, are discussed.

How the information recorded in the patient record is represented in a form appropriate for Event Calculus is described in Chapter 5. The case frames that can be identified in the domain are described under the different stages in the clinical pathway: interview, investigation and treatment. The time relative relations are identified and the initiating and terminating rules are given for each.

How a set of recommendations built on top of the knowledge base can assist in patient management is described in Chapter 6. These recommendations support patient management decisions not only by recommending what action should be taken next, but also by specifying when should the action be carried out. Although ramifications provide a simple and clear way of representing the domain knowledge in some cases, if it is used as the only representation method it can cause complex representation problems in other cases. This dichotomy of behaviour is illustrated in this chapter.

The implemented prototype system, its desirable characteristics and its drawbacks, its user interface, the query facility, and its answer justification capabilities, are described in Chapter 7. Considering two case studies, the query facility provided is described in detail, highlighting the decision support features described in Chapter 6.

We compare the capabilities of Event Calculus with that of Process Modelling provided by the language PML, in Chapter 8. The outcome is compared by modelling two examples in both formalisms: a simple interactive bank account, and the more complex clinical pathway in vascular surgery.

Chapter 9 gives our conclusions and suggestions for future extensions.

A considerable amount of detailed information on the prototype system is supplied in appendices. This is summarised here for completeness. Appendix A displays the current patient record used at the vascular surgery department at Royal South Hampshire Hospital. Appendix B–D illustrate the user interface facilities with sample displays. Appendix B describes the main menu options (and the sub-menu options available for each, if there are any) provided in the user interface: selecting a patient, data input, data display, and the help facility. For each option sample dialogue displays are given. Appendix C describes the query facility available to the user, that is finding what properties are true about a patient at any specific time. The answer justification facility is illustrated in Appendix D. Taking a case study how the system provides *why* and *whynot* explanations is illustrated with accompanying sample dialogue displays.

2 Identification of Knowledge Based System Goals for Vascular Surgery and Overview of Event Calculus Approach

The vascular surgery department at the Royal South Hampshire Hospital (RSH), our collaborators in this research, is a large organisation dealing with around a thousand patients each year. Vascular surgery is a rapidly expanding area of medical practice attempting to reduce the suffering and death associated with arterial diseases. Arterial disease is an epidemic of unparallelled proportions, contributing to the deaths of a comparatively large percentage of all people.

The development of computer support in vascular surgery is strongly motivated by current trends. The demand for patient care has increased very rapidly in the recent years but the available funding for hospitals has been very limited. Thus, the efficient utilisation of available resources by improved patient management has become essential in order to optimise patient care.

One aspect of the response to increased demand for medical services has been the accelerating introduction of computers throughout the health services, usually for administrative purposes and employing conventional data base technology.

Clinicians at the RSH Vascular Surgery Department are concerned to influence the introduction of computerisation into their field. About three years ago they approached computing researchers at the University of Southampton and the present research was initiated with the aim of applying knowledge based system technology to provide decision support for clinical management in vascular surgery.

Our attempt to develop a formalism for a support environment to assist the management of patients in vascular surgery is complemented by the research conducted in parallel at the RSH on arriving at an effective, structured patient record for data

collection (Appendix A).

In this chapter we identify the goals of the knowledge based system. In order to do so, we will first describe the clinical pathway that is modelled by the knowledge base, introducing the terminology used in this field. We then discuss the current problems as perceived by the clinicians and from these problems we identify the goals to be satisfied by our knowledge based system. The chapter ends with a brief over view of our proposed solution to the problem of building such a system. It is based on incorporating temporal reasoning, using the Event Calculus approach, into the basic structure of the knowledge based system. The details of our approach are developed in Chapter 4, but meanwhile this overview provides a context for our discussion in Chapter 3 of other approaches to including time into a knowledge based system.

2.1 The Clinical Pathway

A major aspect of this thesis is to show how the Event Calculus can represent a model of the clinical pathway in which specialists review patients at variable intervals depending on how the results of investigations and consultations are progressing. In this section we describe the clinical pathway (the part under our focus) for the arterial side of the vascular surgery.

Initially, the patients are directed to the specialists in the department by their General Practitioners (GP). First the patient is examined by a consultant. If the problem the patient is suffering from is assumed to be of vascular nature then the patient is recommended for admission to the hospital for further investigation of the disease and later for treatment if necessary.

The progression of stages through which a patient passes while undergoing treatment is referred to as the *clinical pathway*. We commence the clinical pathway from the point where the patient is admitted to the hospital. Before describing the pathway we will define some of the terminology used in the thesis.

interview A meeting between the clinician and the patient is termed as an interview. The patient may be examined, symptoms and signs observed, and these may be compared with test results or expected results of a treatment. Based on the evaluation done, a conclusion or a hypothesis may be made, or a previous

hypothesis may be eliminated.

The first interview the patient has when he or she enters the pathway is termed as the **initial interview.**

diagnosis A conclusion or a hypothesis made at any point in the pathway. We describe the diagnosis made at the initial interview as an 'initial diagnosis' and a diagnosis which is sufficiently refined so as to decide a treatment plan as a 'final diagnosis'. These two terms are only used to distinguish one from the other in our literary description of the system. However, in the actual representation just the term 'diagnosis' is used to represent both; they are distinguished by their structures.

general tests These are the tests carried out in order to evaluate and make hypotheses about a patient's diseased condition. They include,

- preliminary tests which are done on all patients (unless in exceptional cases such as an emergency) irrespective of the patient's particular problem (e.g. blood test, blood pressure). Some of these may not be helpful in the diagnostic process but will be used in the treatment plan or for anaesthetic purposes.
- tests carried out based on patient's responses to the clinician's questions (e.g. a patient who reports to be suffering from diabetes will be tested for blood sugar level, a patient who complains about chest pain will undergo a chest X-ray).

specific tests Tests done in order to support a particular diagnosis, i.e. a belief or a hypothesis, or to disprove it. For example,

- the test, ultrasound will be carried out if it was suspected that the patient suffers from dilatory arterial disease;
- an arteriogram will be carried out if it was suspected that the patient suffers from occlusive arterial disease or both dilatory and occlusive arterial diseases.

At the initial interview the patient is examined by a clinician, usually a junior doctor. The results of the preliminary general tests (e.g. blood test, blood pressure)

may be available at this interview to the clinician. An initial diagnosis will be made based on the symptoms and signs observed, general test results available, and the results of the examinations (such as pulse) carried out. Depending on the diagnosis, specific tests may be performed. The results of the specific tests will confirm/support the diagnosis or eliminate it. The accumulation of all information gathered up to this point (the family, social and medical history of the patient, the risk factors involved, loss of independence to the patient, the observations made at the interview, and the results of the tests carried out) will lead to a treatment plan.

A particular patient admitted for vascular surgery might progress through the following typical stages:

interview	The patient's history and examination is recorded by (usually) a junior doctor and an initial diagnosis is produced (e.g. occlusive arterial disease).
investigation	Further tests are carried out by a specialist (e.g. arteriogram by radiologist) and a more specific diagnosis is produced (e.g. embolism, at(left, common_femoral)).
treatment	Recommended treatment is carried out (e.g. surgeon carries out the bypass operation)
followup (of treatment)	This essentially consists of a combination of the above three stages.

The investigation stage normally continues until one of the following applies:

- a final diagnosis is made which determines a treatment,
- sufficient information is gathered for a treatment decision to be made although no specific final diagnosis has been made
- unable to collect more information — a decision will be made on available data.

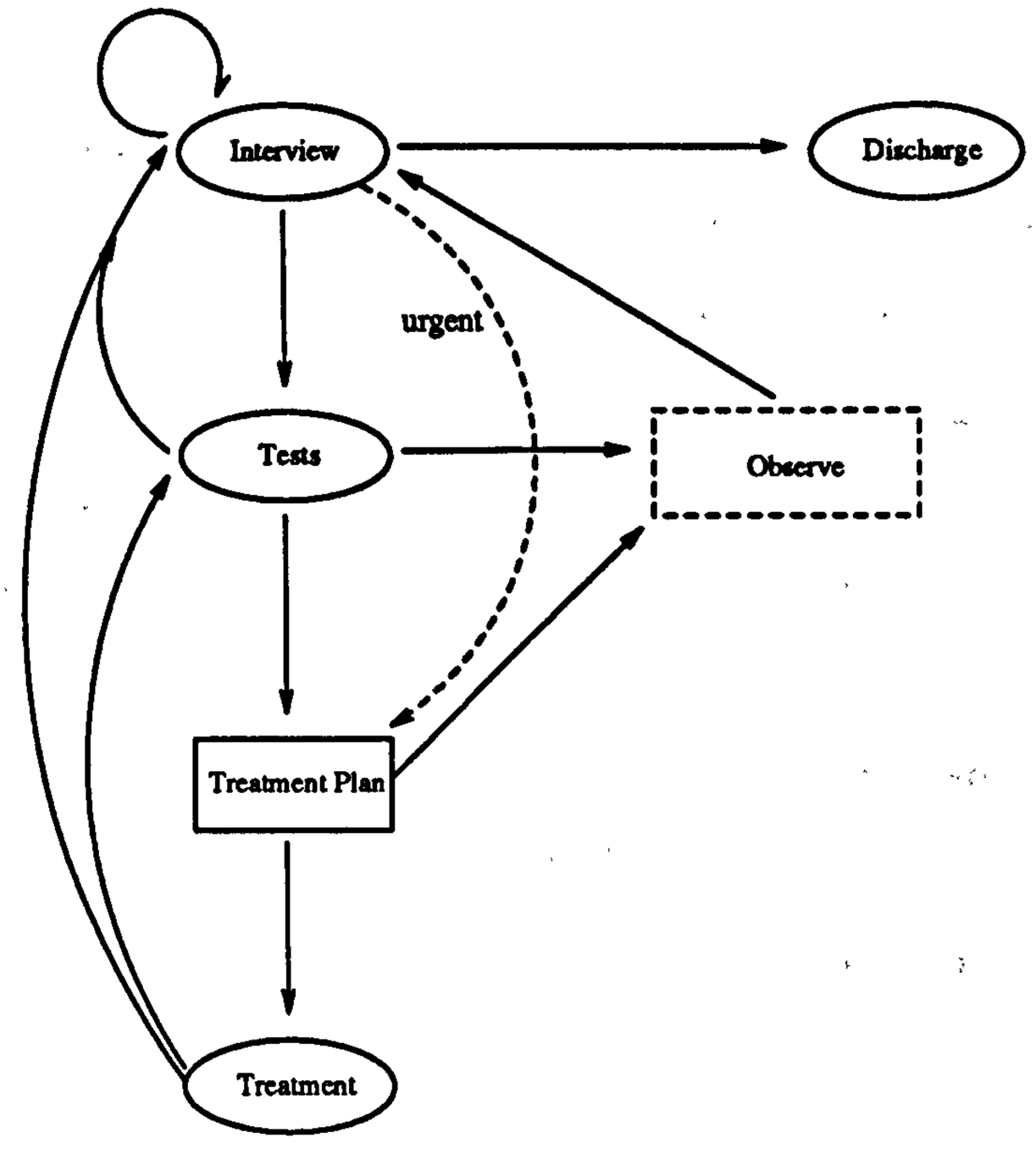
In between the investigation stage and the treatment stage there will be the treatment plan where all experts concerned with a particular case get together and decide on an optimal plan for the patient concerned. We have not included this planning stage in the pathway for simplicity, since the patient has no relation to it. Although this is also not represented in the task hierarchy (given in Figure 5.1 in Chapter 5) the treatment plan has been given much consideration when building our inference system.

The followup stage may extend over a number of years. A treatment, in most cases, is followed by an interview at which the patient's condition is evaluated depending on the progression of symptoms. If the symptoms still persist, the patient may be subjected to further investigation. Specific tests may be carried out, which may lead to further treatment. If the symptoms no longer can be observed, the patient may be discharged from the hospital after a certain period of observation. Specific tests are carried out to confirm the patient's condition during this period. A discharged patient does not leave the pathway completely unless it is confirmed that the patient has been cured from the arterial condition, the expert believes that no help could be provided to reduce the patient's symptoms, or the patient meets death operative or otherwise. It is the nature of the disease that for many patients it is a case of reducing the suffering rather than complete cure. After leaving the hospital, the patient is monitored at regular intervals until leaving the clinical pathway.

One may question the need to distinguish a followup stage if it only consists of the other three stages. We have made this distinction because the sequence of events occurring before the first treatment is often different to that occurring after, thus differentiating the followup stage facilitates better understanding of the representation.

The clinical pathway is represented diagrammatically in Figure 2.1. The solid arrows indicate the next step that might take place in the pathway. The dotted arrow indicates a step that may occur in an urgent case.

Once in the pathway a patient may be directed back to an earlier stage (for example, for further tests at the investigation stage, or for interview and tests at the treatment stage). Since the stages represent increasing degrees of commitment, intermediate stages will only be skipped in unusual circumstances (for example, in an emergency). Communication between stages takes place, at present, by passing along the patient's



represents an event that would occur in the pathway

represents a stage where decisions regarding the treatments of patients are taken but, not represented as an event in the pathway

represents a period where the progression of the patient along the pathway is temporarily suspended.

Figure 2.1: An overview of the clinical pathway

paper-based notes. One of the aims of the support environment is to enhance the communicative function of the notes at the point when each clinician assesses the requests of the previous stage and reports the results/recommendations of the current stage.

2.2 Current Problems

From the point of view of a clinician the overall aim of introducing a computerised system is to improve the quality of patient care. Problems with the current system of record keeping as perceived by clinicians included the following:

1. Incomplete and/or incorrect data can be recorded on the paper-based notes. This can lead to varying degrees of non-optimal decisions, when the notes are used to communicate between stages of the evaluation and between clinicians in different fields (for example surgeon and radiologist);

2. A lack of control over the consistency in the quality of assessment, for example before and after treatment or between teams;

3. A failure to adequately take into account up to date risk factors which bear on a clinical decision;

4. An inability to properly respond to increasing demands to justify resource consumption.

5. Unavailability of the information when it is required by more than one person involved in the patient's health care.

2.3 Knowledge Based System Goals

Our general aim is to develop a framework based on Event Calculus that could provide general temporal reasoning capabilities to a knowledge based system. We have chosen a relatively complete sub-domain of vascular surgery, the treatment of arterial disease in the lower renal area[1], as our application domain. Consequently we aim to show how Event Calculus can afford a clear and flexible method for modelling the interdependencies of properties which change with time, and for accessing historical information in the application domain of vascular surgery.

Within this general aim, we identify the following specific goals which will enable the knowledge based system to satisfy the current problems identified by clinicians.

[1]The arterial tree below the infra renal

1. To incorporate an historical knowledge base of patient records.

2. To provide a framework for flexible interactive retrieval and assimilation of information which can be tailored to:

   (a) retrieve relevant information in the patient records for decision support;

   (b) support completeness of data collection and consistent practice in particular situations.

3. Provide support for patient management by generating system recommendations.

2.4 Overview of Our Approach

Kohane in (Kohane, 1987), writes that,

> "Expert systems performance and knowledge acquisition is at best incomplete and at worst fatally flawed if the system does not posses a systematic and principled "understanding" of temporal knowledge. Therefore, even though temporal reasoning is not in and of itself sufficient to produce satisfactory behaviour in expert systems, it is a necessary and significant step in that direction."

Temporal reasoning is important in many application areas, especially in medicine. In health care, the patient history is a widely used diagnostic tool. Sometimes the patient history may be the only available information to arrive at a diagnosis. Therefore any computer-based system which deals with representing patient information should have the capability of representing and reasoning with historical data. Temporal information is an integral part of medical knowledge. In some areas of medicine it is more explicitly acknowledged. In particular in vascular surgery relating the timing of events to a patient's clinical presentation is essential.

Our approach is to build a general temporal reasoning framework, into the knowledge base. Some typical features which are expected of a general temporal reasoning system are as follows. The ability to handle incomplete information about the past and, correspondingly, assimilate information in any order. The ability to

accept partially-specified information, for example that an event occurs before another rather than the explicit times of both events. The ability to reason about transaction time as well as event time. The ability to reason about persistence as a characteristically temporal form of default reasoning. The Event Calculus has the potential to supply these functionalities, and it is in this sense that we refer to it as a general temporal reasoning framework. It is important to state, however, that in our application we have not exercised all these functionalities equally. In particular we have not made explicit use of under-specified events; that is we do not reason by default about absent properties of events nor do we reason transitively about partially ordered events. We have included transaction time in our formalism, so we cannot explain erroneous conclusions drawn in the past. Our application does take into account incomplete past information and reasons about persistence. It is important to add that we believe the other temporal reasoning functionalities of the Event Calculus may turn out to be of use for future developments of the system. We argue that this potential for development is a key reason for using a general framework, in particular the Event Calculus.

Event Calculus is formalised within a logic programming framework. It seeks to combine the expressive power of case semantics with the deductive power of logic programming (Kowalski, 1992; Kowalski and Sergot, 1986). Formalised in a Horn Clause subset of Classical logic augmented with negation as failure, the formalisation is executable as a logic program. Therefore, applications based on the formalism can be easily implemented in Prolog.

We will show that Event Calculus can represent a simplified model of clinical practice in vascular surgery in which specialists review patients at variable intervals depending on how the results of investigations and consultations are progressing. A system of recommendations is constructed on top of the historical knowledge base, so that recommendation of appropriate options at any time can be generated, thus providing a powerful framework for support of patient management decisions.

We have implemented a system with the above characteristics. We refer to the resulting system as a research workbench. By this we mean that the system forms a suitable basis for the investigation and development of further tools in the application domain. In order to fulfil the function of a research workbench our software has been implemented in a declarative style which can be easily understood and modified.

We have also refrained from optimising with respect to the application, prefering generality at the expense of efficiency. The development of our final system divides into two phases: the development of the Event Calculus based historical knowledge base for the domain - this forms the research workbench itself; and the development of the patient management system which provides a tool for use with the workbench. The development of the patient management system is to some extent a test of the effectiveness of the workbwnch. It also provides one of the core tools for use in this domain.

By developing a research workbench, a prototype system, we will demonstrate how we achieve the knowledge based system goals set out in the previous section. Our approach is to apply query-the-user techniques developed for logic databases (Sergot, 1982; Wolstenholme, 1992b) to control user interaction and the generation of explanations. The user can then ask for an explanation of a recommendation. A response is produced by applying the techniques for generating explanations in logic databases, but because of our Event Calculus formalism we are able to produce, in addition to rule traces, the sequence of related events which lead up to a conclusion (Chapter 7, Section 7.1.5). As well as being informative this sequence of events provides access to the detailed information on the patient record which is stored in the associated case frames. The system can also operate in 'why not' mode in response to user queries, explaining why a particular recommendation is not given or why a property of the domain which seems to be true is not so at a specified time.

2.5 Conclusion

The uptake of knowledge based system technology in medical practice began in the 1970s and well known knowledge based systems such as MYCIN (see Section 3.2) emerged. These knowledge based systems used different methods and formalisms for representing and managing knowledge (Ramoni, Stefanelli, Magnani and Barosi, 1992). Most of the existing systems have used the production rule software approach (MYCIN (Buchanan and Shortliffe, 1984), VM (Fagan, 1980), ONCOCIN (Kahn, 1988; Kahn, Fagan and Tu, 1991a), PUFF (Aikins, Kunz, Shortliffe and Fallat, 1984)), the frame based approach (ESPRE (Connelly, Sielaff and Scott, 1990), HELP (Evans, 1991))

or a combination of both (GALEN (Rector, Nowlan and Kay, 1992; Rector, Nowlan and kay, 1993), (FOETOS (Alonzo-Betanzos, Moret-Bonillo and Hernádez-Sande, 1991)). Contrastingly we use a logic based approach using the deductive capabilities of the logic programming language, Prolog.

The novel feature of our system is that a general temporal reasoning framework is built into the basic design of the knowledge base. Our justification for taking this approach is that, by its greater expressiveness, such a system can satisfy the goals given in the above section and thereby may be more attractive to medical practice. Temporal reasoning is important in many application areas and in particular in vascular surgery to relate the timing of events to a patient's clinical presentation. Computerised vascular registries (see van den Akker, van Bockel, Brand and van Schilfgaarde,1991 for a recent example) do not have this ability. From our research (Soper, Abeysinghe and Ranaboldo,1990; Soper, Abeysinghe and Ranaboldo,1991; Soper, Abeysinghe and Ranaboldo,1992) we believe that the general temporal approach may provide considerable advantages in terms both of construction methodology and performance of the system.

In this chapter first and foremost we have established the goals of our knowledge based system. Before doing so, we described the clinical pathway that is modelled by the knowledge base and introduced the terminology used in the field. The current problems in the practice of vascular surgery as perceived by the clinicians were then discussed. Based on these problems we identified the goals to be satisfied by our knowledge based system. Finally an overview of the Event Calculus approach we have taken was described.

In the next chapter we describe some other computer-based medical information systems that involve time. It will be seen that most of the systems have used traditional methods of knowledge representation and this entails having elaborate schemes for reasoning, in particular about time, and retrieval of information. The next chapter will give a flavour of what the current medical information systems capabilities are and how they have been achieved, and will create an appreciation of what our system could further provide to enhance and add to these capabilities.

3 Other Computer-Based Medical Information Systems Involving Time

In the previous chapter we gave a brief overview of the Event Calculus approach we have used in our system. The aim of this chapter is to survey some other medical information systems, and in so doing to establish the necessity to develop a general temporal reasoning formalism that could be used in such systems. We are selective in our discussion concentrating on information systems which take into account temporal aspects of the data. These systems are described under the two broad categories: data bases (systems that aim to store, retrieve and transmit patient related information); and knowledge bases (advice systems that can apply stored information to the solution of clinical problems, providing decision support, and/or making decisions). Before our discussion the functions and increasing importance of the medical record is outlined.

3.1 Functions of the Medical Record

The medical record (also referred to as the patient record) is the main source of information in patient care. It summarises the patient's history and documents observations, diagnostic conclusions, and treatment plans. The medical record also provides a means of communication between those who are employed in health care; for example, between the various specialist clinicians who deal with the patient, between these specialists, junior doctor and nurses and so on. Typically clinicians initiate a therapeutic action by recording orders on the medical record. The people who complete the order (the radiologist, the junior doctors, the nurses in care) in turn record their actions, observations, and results of the actions (if any) on the records. The medical record also ensures continuity of care during a patient's hospitalisation and ensures continuity across outpatient visits (Shortliffe and Perreault, 1990). Most sophisticated medical information systems which have aimed at automating the

medical record have attempted to preserve the format of the paper-based medical record in use.

With the evolution of various computer related technologies there has been an explosion of information. The amount of information available for the clinician has become so vast that, when it came to access and analyse the information, the manually kept medical record became a hindrance (Collen, 1991). Inaccessibility is a common drawback of paper records, especially in large hospitals where the retrieval of a particular record may take considerable time; for example, the record may be in the administration office for financial transactions, or it may be in the laboratory awaiting a transaction from the blood transfusion personnel, when needed elsewhere. Since it is vital that the information recorded in the patient record be available to the clinician quickly and easily, the introduction of computer technology to handle information has become essential. Computerised medical record systems and medical data bases are being introduced at many health care institutions. At present, standards for Electronic Medical Records are being mooted among many standards bodies (e.g. Institute of Medicine, European Workshop on Open Systems) (Rector, Nowlan and Kay, 1991).

3.2 Medical Information Systems

With the introduction of Artificial Intelligence (AI) techniques to the medical field, it became apparent that computers can be used for more than just data storage and retrieval. One of the most important developments of AI research, the knowledge based system, is now being used in many medical applications.

Medical information systems have tackled many different areas in the medical field starting from attempting to computerise the medical record to various monitoring and advice systems. We mention some of the better known systems so as to give an idea of the range of application in the medical field. One of the first and best-known systems for decision support in the medical field, MYCIN, which supports diagnosis of bacterial infections, was introduced by Shortliffe (Buchanan and Shortliffe, 1984). Wieding et al (Wieding, Kretschmar and Schönle, 1990) have aimed to develop a reference system for textbook knowledge, giving a complete and rapid availability of information from the medical literature. SESAM-DIABETE is an interactive educational expert system that

provides personalised advice and therapeutic recommendations for insulin-requiring diabetic patients (Levy, Ferrand and Chirat, 1989). The OASYS data base of Stoodley and Sikorski is used for auditing purposes in orthopaedic surgery (Stoodley and Sikorski, 1991). GALEN aims to develop a model of coding notation for medical terminology (Rector, Nowlan and Kay, 1993). This variety in application area of different information systems shows that considerable potential benefits could follow from building automated, sophisticated information systems if the resulting systems perform adequately in their application domain. In the rest of this chapter we focus on a number of medical systems which consider the time-related aspect of the patient information. These systems differ not only in their application areas, but also in the methodologies used for handling the time dimension.

3.2.1 Medical Data Base Systems

Martin in (Martin, 1976) defines a data base as: a collection of interrelated data stored together with controlled redundancy to serve one or more applications in an optimal fashion; the data are stored so that they are independent of programs which use the data; a common and controlled approach is used in adding new data and modifying and retrieving existing data within the data base.

The main objective of a medical database is to provide fast access to patient related data necessary for efficient patient care. This data will be used by many application programs which are designed to help a variety of functions such as, administration, patient care, and research. We now describe three well known data bases of this type which have considered time-oriented data; the COSTAR system, the HELP system, and the TOD system. A number of comments are made on how these systems compare with our Event Calculus approach; these comments will become clearer after reading Chapter 4.

3.2.1.1 COSTAR

COSTAR (COmputer-STored Ambulatory Record) system was designed to perform the data management functions needed in the care of ambulatory patients. The system uses a date stamp and a coded status to represent the patient's current clinical condition. The purpose of the system was to replace the traditional document-based medical

record with a comprehensive, centralised, and integrated information system (Barnett, Justice, Somand, Adams, Waxman, Beaman, Parent, Van Deusen and Greenlie, 1979).

All information recorded in a COSTAR patient record is linked to a *code*[1]; the collection of codes is called the *directory* (Payne, Goroll, Morgan and Barnett, 1990). The COSTAR directory provides the capability to process and store data which are very non-uniform in nature.

A reference to a specific COSTAR code at a patient visit is referred to as an *event*. The representation of events for the code NSAID ibuprofen is given in Figure 3.1. Each

CODE: ibuprofen

SUMMARY STATUS: inactive

DATE	STATUS	TEXT
3/2/84	null	"600 po tid, #84"
3/28/84	null	"1 yr supply, 2 refills"
4/2/86	null	"600 po tid, 1 yr supply"
6/24/86	inactive	""

Figure 3.1: Representation of the information contained in a medication section of a COSTAR patient record. The text field associated with a visit represents the medication dose, route, amount dispensed, and number of refills. Associated with each event is also a status which indicates whether the medication has been made active or inactive at the visit (Payne, Goroll, Morgan and Barnett, 1990).

code whose status is changed by a visit is represented as an event of that code. Thus an event of a code represents the change in status of that code. In this perspective an event in COSTAR can be compared with the notion of event in Event Calculus. An event in Event Calculus starts or ends one or more time periods in which a property of the domain (such as a diagnosis, a test result – represented as a relation) is true.

Each clinical code is associated with a *summary status* which represents the *current* status of the code. By searching the *summary status* of each code it is easy to determine whether at the present time the code is active or inactive (Payne, Goroll, Morgan and Barnett, 1990). For example the *summary status* for diagnostic codes take values *major,*

[1]Codes are concepts such as diagnoses, medications, laboratory tests, or physical examination findings

minor, inactive, or *presumptive.* The *summary status* is used to suppress routine display when a clinical entity is no longer active or medically significant (Barnett, Justice, Somand, Adams, Waxman, Beaman, Parent, Van Deusen and Greenlie, 1979). In the Event Calculus approach we achieve the same temporal reasoning effects within a more general system.

Although COSTAR has no decision making capability in itself, the vast number of programs in the software package enable a variety of activities to be carried out on patient information including production of condition- or diagnostic-related flowcharts.

Many enhancements have been added to the system. Osburn et al in (Osburn, Neches, Shissler and Kittredge, 1984) describe one such enhancement to COSTAR, the inclusion of a Problem Oriented Record Structure and decision support capability.

Payne et al in (Payne, Goroll, Morgan and Barnett, 1990) describe the techniques used for a historical cohort study of the effects of non-steroidal anti-inflammatory drugs (NSAID) on blood pressure control using COSTAR. Here they use an algorithm to determine whether a medication is considered in COSTAR to be active at an arbitrary date. When using this algorithm they assume a medication or a diagnosis to be active for that patient between the time it was made active in the past (if ever) until it is explicitly discontinued in the patient record. This is very similar to the forward persistence of relations in the Event Calculus (described in Chapter 4, Section 4.2.1), where a relation once initiated by a known event is assumed to persist forwards in time until it is known to be terminated by another event.

An alternative method for determining the duration of a medication has also been described. By reviewing the text field of the most recent event of the medication code (Figure 3.1) and determining the date the medication would be exhausted from the amount prescribed; if this duration is less than the duration given by the algorithm, then the medication may be considered inactive when that date is reached. We find a similarity in the latter method to the way we terminate a relation in the Event Calculus when its `life_time` has expired (see Chapter 4, Section 4.2.2). The `life_time` of a relation is considered to be the duration of time, starting with the initiation of a relation, during which it is reasonable to assume that the relation remains true if it is not changed by a subsequent event.

The Medical Query Language (MQL) can be used to extract and analyse information from COSTAR. Payne et al claim that by the use of MQL, it is possible to determine (within the limitations of the data entered into COSTAR) the medication regimen and active diagnoses at any desired time or interval within a patient's COSTAR record (Payne, Goroll, Morgan and Barnett, 1990).

3.2.1.2 The HELP System

HELP (Health Evaluation through Logical Processing) was designed to cater for both administrative and clinical research needs of the hospital and to provide decision support (Burke, Classes, Pestotnik, Evans and Stevens, 1991). The system contains an integrated, time-oriented, computerized medical record containing patient information from most clinical areas (Evans, 1991). The drug monitoring component of the HELP system falls into the category of knowledge based systems (Dasta, Greer and Speedie, 1992). Thus, HELP can be thought of as a hybrid, between the two categories data base and knowledge base.

Data is input to the system during hospital visits of the patient, in the emergency room, and at the out-patient clinics, automatically through digital or analogue interfaces (ECG, clinical laboratory) or manually by professionals. There are three types of data entry screens: multiple choice (items are selected from menus), set of fields where values are entered, and time screens (time is tagged automatically to entered data). Thus the time attached to data input through the time screen is the transaction time.

The data base structure makes use of two data models: the relational model and the hierarchical model. There are two relational files for the recording of historical patient data: **Master Patient Index** (MPI) and the **abstract file**. MPI contains a record for each patient consisting of demographic data. The **abstract file** is indexed by patient admission number and the date of admission. It contains a record for each admission. All data stored in the patient record has the date and the time appended to them.

The HELP system's knowledge base consists of knowledge frames. The information processing is data-driven and time-driven. Specific frames are activated when some key information is entered to the data base – this is called data-driven. Certain modules can be activated at a certain time of the day, which is called time-driven.

HELP decisions are made using predefined decision frames. Results of the HELP

decision process are also stored as part of the patient record along with an indication whether the decision is current or the date and time when the decision became inactive. Decision support is provided in three ways:

1. a report representing data and the generated interpretation of the data,

2. an alert message on data entry,

3. a care person interacting with the system through terminal access.

HELP has advantages for information gathering since it is built into the structure of the hospital along with the essential services. Therefore, information becomes available from patients undergoing monitoring or lab investigations rather than relying on another person to key in data (Ellis, 1987).

HELP is used in many other clinical applications such as, computer monitoring and clinician education in antibiotic therapy, in-vitro[2] susceptibility data, and drug utilisation (Burke, Classes, Pestotnik, Evans and Stevens, 1991; Evans, 1991). HELP is also used to assist clinicians and nurses in blood ordering and review of orders for appropriateness (Gardner, Golubjatnikov, Laub, Jacobson and Evans, 1990).

HELP meets the description of an interactive hospital diagnostic data base, with a core of medical knowledge and an automated reasoning process that evaluates all incoming patient information against its own medical knowledge base, and responds with suggestions and alerts. HELP system supports transaction time as opposed to our system which deals with event times. HELP uses a model based representation of data and a frame based representation of knowledge. We have used a logical representation of the knowledge. The historical deductive approach used by us allows the knowledge base to be viewed as it was at any given instant. This removes the necessity to store the conclusions made by the system. HELP's decision support through alert messages has similarities to the support we provide through recommendations.

While HELP is an impressive practical system which has identified the importance of representing patient information within the correct time window, it does not have the general temporal reasoning capability provided by Event Calculus. Event Calculus allows a flexible and a more general representation of temporal information. The logic

[2]outside the living body and in an artificial environment (Webster's Medical Desk Dictionary, Merriam-Webster Inc, 1986)

programming framework and the deductive approach of our system provides a natural way of querying the data base enhancing the decision support capabilities.

3.2.1.3 The TOD Software System and ARAMIS

The Time-Oriented Data base (TOD), is a data base management system. It is the most widely used data model for electronic patient data base systems (Kahn, Fagan and Tu, 1991a). In addition to entering, updating, storing and retrieving clinical information, TOD provides built-in statistical programs for data analysis, and supports the creation of data files for analysis using standard statistical packages.

TOD data bases are structured to store patient information that has been collected over the course of multiple clinic visits (Shortliffe and Perreault, 1990). Within the TOD data base two levels of hierarchy can be recognised: the patients, and the visits of the patients to the clinic. The two levels are used by TOD in an identical manner. The data base is *schema*[3] driven.

There are two data files. The *header* file contains non-chronological patient information. Header elements are those whose change through time does not need to be recorded. The *parameter* file contains visit information. Parameter elements are those recorded at each visit (McShane, Harlow, Kraines and Fries, 1979).

Each value in the data base has an associated time stamp that denotes when that value was observed and that distinguishes multiple recordings of the same attribute (Kahn, Fagan and Tu, 1991a). Hence data bases designed using the TOD model are historical data bases (Kahn, 1988). An example of a standard TOD-based patient record is shown in Figure 3.2.

All clinical records are kept in a large tabular format where columns represent each clinical visit and rows represent relevant clinical parameters that are being followed over time. Time oriented data on all patients are kept current by transferring each column of new data to the computer data bank by transcriptionists (Shortliffe, Buchanan and Feigenbaum, 1984). On-line data checking occurs as part of the time-oriented entry programs.

---

[3] A schema is a structure which defines the specific content of the individual databank (Wiederhold, Fries and Weyl, 1975). For each position in the databank, each element's units, data type, qualifiers, and all processing characteristics are defined by the schema. Therefore, the position of an element in the databank file determines its meaning (McShane, Harlow, Kraines and Fries, 1979).

WBC X 1000	8.8	8.6	7.6	8.0	4.0
% polys	57	36		54	24
% lymphs	31	30		20	30
PCV	38.2	35.6	32.8	33.9	27.2
Haemoglobin	13.1	12.2	11.1	11.4	9.1
Platelets	344	347	300	244	296
BSA (m^2)		2.3			
Arm assignment	B				
Chemo Name		POCC	POCC	VAM	POCC
Cycle #		1	1	2	2
Subcycle		A	B		A
Visit type	LAB	TX	TX	TX	TX
Procarbazine		200	200		200
Vincristine		1.5	2.0		1.5
Cytoxan		1300	1300		1300
CCNU		130			130
VP16				170	
Adriamycin				110	
Methotrexate				65	
Cum. Adriamycin (mg/m^2)				48	
Day	22	30	06	27	20
Month	Jan	Jan	Feb	Feb	Mar
Year	86	86	86	86	86

Figure 3.2: A sample TOD-based patient record. The names of clinical, laboratory, and therapeutic attributes are listed in the left column. Dates associated with measurements are listed across the bottom. Patient-specific attribute values on a specific date appear in the centre of the Table. Blank boxes indicate dates during which the attribute was not measured or observed. Time advances from left to right (from Kahn, Fagan and Tu, 1991).

Among the outputs that can be generated from information in the data bases are: a summary of the patient's current status, graphical display of the change of a specific parameter over time for a specific patient, and statistical analysis for clinical investigations.

The American Rheumatism Association Medical Information System (ARAMIS) is a nationwide data base system which originally used TOD for data management and analysis (later ARAMIS was transferred to MEDLOG, the microcomputer version of TOD). ARAMIS provides decision support by giving therapy recommendations (McShane and Fries, 1988).

ARAMIS data bases contain information about the long term clinical courses of patients who have arthritis and rheumatic disease and have been observed in the clinical setting. ARAMIS offers a prognostic analysis for a new patient when a management decision is to be made. A practitioner may select clinical indices for a patient. Using these descriptors the computer locates relevant patients. Therapy

recommendations are made on the basis of a response index calculated for the matched patients. A prose case analysis also can be made for the patient. This summarises the relevant data from the data bank and explains the basis for the recommendation (Shortliffe, Buchanan and Feigenbaum, 1984). The prose case analysis provided by ARAMIS is closer to the why explanations given in our system, explaining why a particular recommendation has been made, or why a particular property is true at a specific time.

The widely demonstrated prototype system, developed by Robert Blum, for automating the discovery and confirmation of hypotheses from large clinical data bases, the RX Project, uses a subset of the ARAMIS/TOD data base (Clancey and Shortliffe, 1984).

The expert system ONCOCIN used the TOD model as its initial patient data base. In Section 3.2.2.2 we describe the ONCOCIN system and the temporal networks used by it.

3.2.1.4 Discussion

The hospital medical record ensures continuity across outpatient visits. The main purpose of the computer-stored medical record is to overcome the disadvantages of the traditional paper-based medical record rather than to introduce new functionality.

In this section we have discussed a few such attempts at achieving this purpose. Medical data bases which provide simultaneous, multiple, and fast access to the stored data have helped to provide better patient care in numerous ways. The data captured is used by application programs designed for a variety of purposes. Stored patient data can be linked to other data bases such as those in laboratory, pharmacy, and at other institutions.

Medical record is also a source of new medical knowledge (Shortliffe and Perreault, 1990). By applying inference mechanisms to the information stored, expert consultation, support and guidance for various tasks could be provided. We describe some systems which try to do this in the following section.

3.2.2 Medical Knowledge Based Systems

Harman and King in (Harman and King, 1985) quote Professor Edward Feigenbaum of Stanford University, the originator of the first acknowledged knowledge based system, DENDRAL, to have defined a knowledge based system as: ...'an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. Knowledge necessary to perform at such a level, plus the inference procedures used, can be thought of as a model of the expertise of the best practitioners of the field'.

The knowledge of the system consists of facts and heuristics. The *facts* constitute a body of information that is widely shared, publicly available, and generally agreed upon by experts in the field. The *heuristics* are mostly private, little-discussed rules of good judgement (rules of plausible reasoning, rules of good reasoning) that characterise expert-level decision making in the field. The performance level of a knowledge based system mostly depends on the quality of its knowledge base, that is, how much of the expert's knowledge the knowledge base contains.

Knowledge based systems could be of help in a variety of areas of patient management and facilitate more effective and sophisticated access to patient record and other information facilities (Glowinski, O'Neil and Fox, 1989b). The use of knowledge based systems in the medical field has increased in the past decade and new systems have been emerging in a variety of areas related to medicine (Alonzo-Betanzos, Moret-Bonilli and Hernández-Sande, 1991; Shortliffe and Perreault, 1990). In spite of this fact there are relatively few systems which have included the time dimension or temporal reasoning facilities in the design of the system. Some of the systems which have taken time into consideration are discussed in this section.

3.2.2.1 ESPRE

ESPRE, Expert System for Platelet Request Evaluation, is designed as a decision support tool. It provides automated decision support for blood bank personnel in assessing requests for platelets (Sielaff, Scott, and Connelly 1991).

The system evaluates a patient's case, based on the guidelines followed by laboratory technologists when evaluating a case, and decides whether platelet transfusion is appropriate or not for that case. The conclusions of the system are reviewed by

blood bank technologists. The time taken to review an ESPRE conclusion is far less than the time taken to review original case data.

Assessing whether a given case is appropriate for platelet transfusion involves time-related considerations such as the rate at which the platelet count is falling or recovering, time between the last two transfusions, and expected post-transfusion platelet count. These are achieved by attaching the date and the time for each transfusion and platelet count in the data base.

ESPRE knowledge base is hierarchical and uses a frame structure for data organisation as well as for knowledge base organisation, with production rules embedded within the knowledge frames (Sielaff, Connelly and Scott, 1989).

The initial frame that is to be processed for a given case is determined by the most recent platelet count (that is, a platelet count should have been obtained within the last twenty four hours after the most recent transfusion). These initial frames (at the top level of the hierarchy) are called *platelet count frames*. The diagnoses or patient conditions are represented by lower level frames.

ESPRE is directly linked to the laboratory computers from which a patient's transfusion history (blood component transfused, time of initiation and time of transfusion, volume of transfusion, character of platelet component transfused), blood chemistry test data, hematology[4] can be retrieved as and when required.

Platelet evaluation is done by backward chaining. First, a request is classified based on the most recent platelet count. Then, the presence of a diagnosis or clinical conditions that justify a transfusion for that platelet count is determined. If at least one indication is present the system concludes that the transfusion is appropriate and recommends the number of units to be transfused. If not, it provides a summary of all relevant laboratory and clinical values to facilitate overview by blood bank clinicians.

During the consultation ESPRE generates an explanation of its conclusions. This explanation includes a list of the indications that were applicable in connection with the patient's most recent platelet count, a summary of the patient specific data used, the conclusions drawn by the system, and its recommendations. This information is printed in the form of a report at the end of the consultation. A listing of all the rules in

[4]The science of blood and blood forming tissues including disorders thereof (Churchill's Illustrated Medical Dictionary, 1989)

the knowledge base that were considered, in the order in which they were considered, together with the outcome of each rule, can be generated if needed.

If any one of the applicable indications is determined to be present, the system concludes that the transfusion is appropriate. If none of the applicable indications can be confirmed, the report will also contain a list of the indications that could not be evaluated for the lack of data, along with which data were missing.

ESPRE needs only the current and the three most recent snapshots of the domain. For example, the platelet count immediately before and after the most recent transfusion, and the time between the two most recent transfusions. The rates such as: the decrease or recovery of the platelet count; falling rate of haemoglobin; falling fibrinogen level, are calculated by the change between the last two data points of the variable in question.

Discussion

The method of processing a platelet request in ESPRE has a similar result to the processing of a query in a deductive data base, but in the latter case this processing takes place as part of a more general theory. The explanations provided by the system in the printed reports are very similar to the `why` and `whynot` explanations provided by our system, explaining the system generated recommendations. An important difference is that, along with the explanations of the deductive rule that lead to the conclusion, we also provide the history of events which lead to the conclusion. ESPRE provides all relevant laboratory and clinical values, and patient specific data which supported the conclusion.

ESPRE needs only the three most recent snapshots of the domain and therefore the complete past history of the patient is not needed to be represented.

3.2.2.2 ONCOCIN

ONCOCIN is a decision support expert system that aids clinicians in the clinical management of patients who are receiving complex experimental cancer treatments under the guidance of a research protocol[5].

[5] A protocol contains carefully written instructions to the clinician detailing how a patient is to receive a new cancer treatment.

The system consists of: a knowledge base in which the skeletal plans and refinement heuristics are represented; a time-oriented data base, that represents patient data and therapy plans; and two run-time processes called the *Interviewer* and the *Reasoner* that manage the data entry and consultation session (Tu, Kahn, Musen, Ferguson, Shortliffe and Fagan, 1989).

ONCOCIN's knowledge base is frame based and is composed of a collection of cancer protocols and accepted medical heuristics. Knowledge of the temporal relationships among clinical events[6] occurring during the course of treatment is embodied in production rules. In the following it is important not to confuse the use of the term 'event' with its use in the Event Calculus. Roughly speaking an Event Calculus event corresponds to the starting or ending of an ONCOCIN event, so the latter approximates to an Event Calculus relation.

The ONCOCIN Temporal Model

ONCOCIN initially used the Time-Oriented Databank (TOD) model (described in Section 3.2.1.3) in its patient data base. Later, a new data base structure, called the *temporal network* (TNET) was developed to address the difficulties uncovered in using the TOD-based data base. TNET was further extended to a second data base structure called the *extended temporal network* (ETNET). In this section we describe the temporal network TNET and the additional features introduced in ETNET.

ONCOCIN temporal model assumes that decision making occurs only during outpatient clinic visits. Therefore the basic unit of time is a discrete patient visit indexed by a calendar date.

The temporal network TNET is a structural collection of data structures, called TNODEs. Each patient has a unique TNET that represents the critical clinical events in that patient's past and present clinical history (Kahn, Fagan and Tu, 1991a).

Each TNODE represents an interval of time during which an important clinical event was happening to the patient. A TNODE has two exclusive states:

1. *open*, which represents an interval that is currently happening,

2. *closed*, which represents an interval that has concluded.

---

[6]Examples of key clinical events are recent clinic visits, treatment courses, and past hospital admissions (Kahn, 1988)

The set of all open TNODEs defines the current clinical context for the patient.

At the beginning of a clinical event a TNODE is created, and the current calendar date is asserted as the event's *start* date; when an event concludes, the relevant TNODE is closed and the current calendar date is asserted as that event's *stop* date. These dates support the queries which refer to absolute calendar times, for example, ten days after an event (Kahn, Fagan and Tu, 1991a).

The temporal network functions both as the patient data base and as a representational structure for temporal queries about the patient. For example, TNET supports temporally dependent queries such as,

- what was the last (or first) recorded white blood cell count (WBC) ?
- what was the first (last, highest, or lowest) platelet count during the last three cycles of chemotherapy ?
- what was the lowest platelet count ever recorded during the administration of VAM[7] ?
- was there any haemoglobin value less than 8.0mg/dl during the previous (any, current) course of radiation therapy ?

These queries refer not to specific dates but rather to relative temporal order (Kahn, Ferguson, Shortliffe and Fagan, 1985; Kahn, 1988; Kahn, Tu and Fagan, 1991b).

TNET deals exclusively with events for which the starting and the stopping times are known without ambiguity (Kahn, 1988). Event Calculus allows partial description of events. This gives much flexibility when describing information (yet) unknown.

The data query language TQUERY is used to store and retrieve data from the temporal network TNET. TQUERY specifies data storage and retrieval requests in terms of clinical contexts rather than calendar dates (Kahn, Tu and Fagan, 1991b).

In order to get over the limitations of TNET, an extended interval-based data base management system, ETNET was developed. ETNET has the added ability of supporting context-dependent temporal reasoning.

[7]VAM is a chemotherapy regimen consisting of three drugs: Vincristine, Adriamycin and Methotrexate.

ETNET is a knowledge representation technique, a temporal mechanism, and a data base management system. As a knowledge representation scheme, ETNET encodes temporal abstractions as intervals of time during which some event is occuring. As a temporal reasoning system, ETNET adds and removes rule sets as intervals are created, and terminated. As a data base management system, ETNET intervals are access paths for context-dependent data storage and retrieval.

Context-dependent reasoning is implemented by ETNET, by associating rules with data structures called ETNODEs. Each ETNODE represents an interval of time during which a specific clinical context is present. Thus an ETNODE can be compared to the notion in the Event Calculus of a relation holding between its initiating and terminating events, but there are some important differences such as the treatment of non-monotonicity.

The presence of a clinical context may define a set of descriptive features that are of particular interest when that context exists, or may trigger a search for closely associated events (Kahn, Fagan and Tu, 1991a).

For all ETNET objects, the starting (creation) and stopping (termination) dates must be known calendar dates. In Event Calculus , we are only concerned with the ordering (or the partial order) of events but not the actual event times. This provides much flexibility and allows the system to accept partially described events.

There are two dates which are exceptions to the above rule: when future is defined as something after the latest calendar date in the current network; and past is defined as sometimes before the earliest calendar date in the current network. ETNET cannot encode dates that are imprecise (e.g. last month), uncertain (e.g. possibly last year), or disjunctive (e.g. either today or tomorrow). ETNET does support changing an interval's starting or stopping time to a new date if additional information causes the current interval times to be incorrect; but in all cases however, the dates must be specified completely.(Kahn,1988)

ETNET can be used in either a data-driven (forward chaining) or goal-driven (backward chaining) manner. ONCOCIN uses ETNET only as a data directed system. When data driven, ETNET uses all available rules to derive conclusions, to instantiate new ETNET intervals, and to close existing intervals. When intervals are opened, new rules are added to the rule set; when intervals are closed, a set of rules are removed

from the rule set. ETNET continues to execute rules until no more conclusions can be drawn and no more intervals can be opened or closed.

Rules that instantiate new ETNODEs can create two classes of temporal intervals: minimum spanning intervals and maximum spanning intervals. A minimum spanning interval is the smallest interval over which a rule's predicate is always true. A maximum spanning interval is the largest interval in which the rule's predicate is only true at the interval end points. The maximal interval over which a property holds continuously, as opposed to a minimum spanning interval, has been introduced by Evans in Event Calculus (Evans, 1989; Evans, 1990; Evans and Shanahan, 1989) using the relation `Mhold_for`.

ETNET has features similar to a frame-based system – a frame denotes the class of a clinical event (for example, the protocol class, the drug class) and each frame instance denotes an interval of time during which that event class is present. From this perspective ETNET is similar to our system, where each instance of a case-frame denotes the occurrence of an event which may start a time period in which a relation/s holds true.

The ONCOCIN Subsystems Interviewer and Reasoner

The overview of the ONCOCIN system is shown in Figure 3.3. *Interviewer* is the data acquisition subsystem. Data is input via a flow sheet designed to have the same format as that used at the oncology clinic at Stanford. This allows the clinician to view the patient history and input current data.

The inferencing subsystem *Reasoner* asserts new data items into the temporal network (Kahn, 1988), updates the TNET, and constructs therapy recommendations. The recommendations consist of a set of proposed therapy actions, laboratory tests, and an appropriate date for the patient's next visit to the clinic. ONCOCIN's recommendations are very similar to the system recommendations produced by our system where we recommend the appropriate tests and treatments to be done, the next suitable task to be carried out and when it should be carried out. The clinician either approves the advice or overrides it, and has the option of asking for explanations before doing so (Kahn, Fagan and Tu, 1991a). This again is similar to our system in that, although the next recommended action is suggested, any action can be accepted at any time. Thus

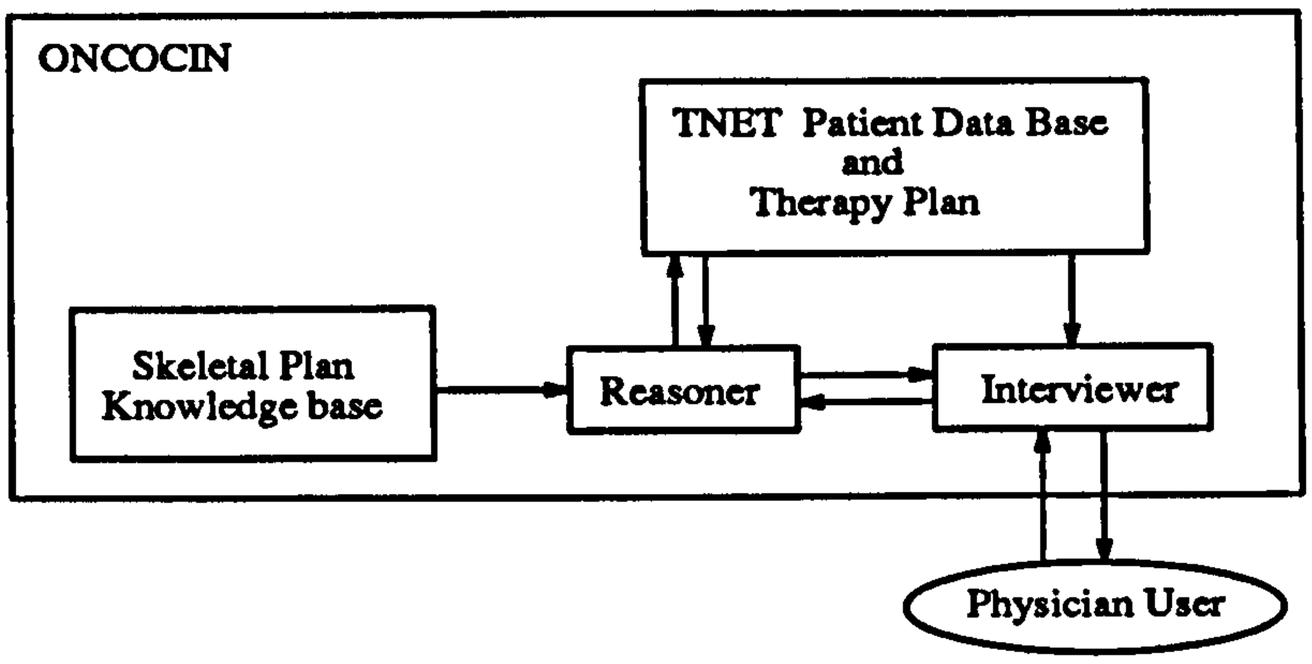


Figure 3.3: The components of ONCOCIN. The Reasoner uses the knowledge base and patient data to construct therapy recommendations. The Interviewer gathers new patient data and displays past data plus the proposed therapy plan to clinician treating a patient who has cancer. It sends the user-supplied current data to the Reasoner for storage in the data base. The arrows show the flow of information among the various parts of the system (from Tu, Kahn, Musen, Ferguson, Shortliffe and Fagan, 1991).

---

the clinician can override the system recommendations at any given time and ask for explanations via *why* and/or *whynot* queries.

Each interaction with ONCOCIN is time-stamped with a *current time*. Actions whose scheduled stopping times are later than the *current time* are represented by *open* intervals. Actions that have terminated before *current time* are represented by *closed* intervals (Kahn, Fagan and Tu, 1991a).

As long as the interval representing an action is *open*, the attribute values of the interval remain *current*. Determining that an action has terminated, and thus closing the interval associated with it, has the effect of making a collection of assertions about the interval as no longer true in the current state. This is similar to a procedural view of what we do declaratively in the Event Calculus approach. The open-ended time intervals are similar to our relations persisting forwards in time. The result of closing a time interval, making assertions about that interval no longer true in the current state, resembles the termination of our relations by an event which ends the time period in which those relations were true.

Discussion

For each patient, ONCOCIN's data base contains a patient node that holds static patient data, as well as a time line of discrete visits that store the time-varying data. The static patient node and the sequence of visits are also modelled as time intervals. The static patient interval starts when the patient data are first recorded in the system, and remains open as long as there is the possibility of more data being entered for the patient. The visit nodes are modelled as zero-duration intervals. Each visit is indexed by a calendar date, so all the time-varying data are time stamped. Thus, the same attribute may have values differentiated by the time at which each of those values was valid.

The ability to associate data with concurrent events is exploited extensively in ONCOCIN. TNET and TQUERY are designed to support context-dependent queries required by a rule-based expert system that reasons about therapy plans for a specific patient. Context-dependent queries are used to ensure that only data that are clinically relevant to the current decision making task are examined.

TQUERY is a procedural language hence, in contrast to using declarative languages, the writer of the query must know of the underlying structure of the data base (in this case of TNET).

A query request always returns only the most recently asserted value for an attribute. Therefore, the patient data base can be reconstructed as it was when ONCOCIN constructed a therapy recommendation. Erroneous data can be corrected, and corrected values can be used in later consultations (Fagan, 1980).

The system is not non-monotonic, that is, although correction of data is allowed at any time, the planner has no means of retracting only those conclusions that depend on the changed data. It has to back up to the top level and the planning process must be restarted. This contrasts with the non-monotonic reasoning in the Event Calculus approach.

3.2.2.3 VM

The Ventilator Manager (VM) was designed to interpret on-line physiological data (used to manage post surgical patients receiving mechanical assistance in breathing)

in the intensive care unit (ICU) (Fagan, 1980; Fagan, Kunz, Feigenbaum and Osborn, 1984).

The knowledge in VM is based on relationships between various parameters in the domain, such as respiration rate, sex of the patient, and hyper-ventilation. Information represented as parameters has been classified as: constant, continuous, volunteered, and deduced. In our Event Calculus approach we have counterparts to all these classes. In particular we represent the volunteered information by the case `report` in the case frame of the event of type interview in our representation, which represents the symptoms reported to be present by the patient at the interview. The deduced information is represented by the case `result` of an interview, which indicates a diagnosis made by the clinician.

A parameter has a list of properties associated with it, two of which are: the last time any conclusion was made about the parameter (property UPDATED-AT), and the length of time that a measurement can be assumed valid (the property GOOD-FOR) (Fagan, 1980). Associated with each value the parameter can assume, is a list of intervals when that conclusion was made. Each interval is calculated in terms of elapsed time since patient data first became available.

In the Event Calculus approach the property UPDATED-AT can be thought of as the time of the most recent event which assigned that value to that parameter. The property GOOD-FOR is similar to the case `rely_time` of an event in our representation. The `rely_time` is a clinician determined property which indicates for how long the values related to that event can be relied on. It is mainly used by the clinician to indicate when that particular event should be repeated.

The two properties, UPDATED-AT and GOOD-FOR, are used to determine the status of the parameter over time, when it was last given a value, and the time period that a conclusion made about this parameter can reasonably be used for making future conclusions. Figure 3.4 shows a snapshot of the parameter HAEMODYNAMICS. In Figure 3.4, STABLE is a list of intervals when that conclusion was made. Each interval is calculated in terms of elapsed time since patient data first became available. Thus in Figure 3.4, the haemodynamics were stable from 2-8 minutes into the program, momentarily at 81 minutes, and in the interval 99-110 minutes elapsed time (Fagan, 1980).

HAEMODYNAMICS

DEFINITION: (HAEMODYNAMICS)

CONCLUDED-IN: (STATUS.STABLE-HEMODYN,A,CMV)

USED-IN: (THERAPY.CMV-A THERAPY.A-T THERAPY.T-PIECE-TO-EXTUBATE)

GOOD-FOR: NIL [this is derived parameter so reliability is based on other parameters]

UPDATED-AT: 110 [last updated at 110 minutes after start]

STABLE: ((99.110) (82.82) (2.8))

Figure 3.4: A snapshot of the parameter HEMODYNAMICS in VM (Fagan, 1980)

Knowledge is represented as production rules. When a rule succeeds the conclusions made (in the form of a parameter assuming a value) are asserted to exist at the current time. If the same conclusion was asserted the last time that data was available, then the new conclusion is considered a continuation of the same situation. The time interval is extended to include the current time. It is presumed that the time period between successive conclusions is short enough to assume continuity. Since VM is concerned mainly with the measurements which are very small distance a part this assumption may be sufficient. But for the domain of vascular surgery this will be too presumptuous.

We implement the continuity of a relation between two time instants by applying the constraint `not interrupted(T1, U, T)` (described in Chapter 4, Section 4.2.1) in our formalism. `interrupted(T1, U, T)` is true if it can be shown that the relation `U` initiated at time `T1` was interrupted before or at the same time as `T`. Thus we assume the continuation of a relation between two time instants only when it cannot be shown that the relation was interrupted during the period between the two instants.

Suggestions are made as a part of the ACTION part of the production rules. Given a particular case what therapeutic maneuver to take is suggested. This is some what similar to the recommendations we make in our system, that is given the patient condition what tests to be done, or what treatment plan is suitable to follow. The suggestions of VM also include additional factors that cannot be verified by the

program, e.g the alertness of the patient.

Discussion

VM was developed as an experimental prototype designed to interpret quantitative data collected in the ICU and to aid in managing the care of postoperative patients who were receiving mechanical ventilator assistance. It applied AI techniques to detect possible errors in data measurement and to suggest adjustments to therapy based on the patient's status over time and on long term therapeutic goals.

Although the inference mechanism in VM takes into account the relationship between VM's suggestions and actual therapy changes, when there are discrepancies VM cannot assume that the patient is actually in the stage the system has determined to be optimal. Such discrepancies require resetting the patient context and reevaluation of the therapy rules. In contrast the Event Calculus approach gives flexibility to the system allowing any event to be accepted at any given time. Although the next suitable task is recommended by our system, the clinician can counter-command the system generated recommendation and the system is capable of accepting the unrecommended event.

VM does not exhibit non-monotonicity. It is unable to re-evaluate past conclusions, especially when measurements are taken but are not reported until some time later (Fagan, Kunz, Feigenbaum and Osborn, 1984). Contrastingly, when the system receives new information about past events, Event Calculus automatically and non-monotonically withdraws any contradictory conclusions made in the past.

3.2.2.4 TUP and the Prototype System THRIPHT

The Temporal Utility Package (TUP) is a utility that is designed to perform domain and system independent temporal reasoning (Kohane, 1986; Kohane, 1987).

In TUP knowledge is represented by Range Relations (RREL). A RREL is an object which specifies the upper and lower bounds on a temporal distance between two points in time. RRELs successfully represent time points, intervals, qualitative and quantitative temporal relations, groups of points, common temporal "yardsticks", and alternate temporal contexts (Shortliffe and Perreault, 1990).

TUP denotes the present by the time point NOW. A new instance of NOW is generated upon every assertion or query about the present. A unique NOW event name is generated and the current date and time on the host-computer real-time clock is tagged to it.

Roughly speaking TUP uses the term 'event' as corresponding to the concept of a 'relation' in the Event Calculus. For example, pallor, loss of weight, anaemia are considered as events in TUP (Kohane, 1987) whereas, they will be represented as relations in the Event Calculus approach. An event in TUP is associated with a time interval in a manner similar to Event Calculus associating a time period to each relation. TUP associates persistence to every event the same way Event Calculus associates persistence to relations. In TUP, the persistence of an event is bounded by the interval associated with it. In the case where persistence of an event is unknown TUP assumes as does the Event Calculus, that persistence extends to infinity until further knowledge is gained about the persistence of that event.

The Temporal Hypothesis Reasoning In Patient History-Taking (THRIPHT) is a medical expert system prototype developed to investigate the kinds of temporal reasoning required for the task of medical diagnosis and to demonstrate TUP's application. Together they illustrate why temporal reasoning is necessary for successful record generation medical expert systems and how to provide this capability.

Discussion

The method of temporal inferencing in TUP is by constraint propagation. The temporal data base is clustered[8] into reference sets. This prevents the computational burden from becoming onerous. Also the clustering of events make the retrieval of information much quicker by focusing on a small subset of the total knowledge base. TUP requires that events within the same cluster be more closely related to each other than to events in other clusters (Kohane, 1987). The Event Calculus does not enforce such restrictions on the ordering of events.

For each TUP event in the knowledge base a corresponding interval is asserted automatically (with default bounds $0, +\infty$ between the onset and the end of the

[8]Grouping of events according to some particular criteria is known as clustering.

interval). The Event Calculus Approach provides more flexibility in that what is important is the temporal order of events.

3.2.2.5 RÉSUMÉ

Temporal abstraction plays a central role in the management of patients who are being treated according to experimental clinical protocols. RÉSUMÉ is a system that performs temporal abstraction of time-stamped data (Shahar and Musen, 1993a). It is composed of a temporal reasoning module, a static domain knowledge base, and a dynamic temporal fact base. The temporal fact base is loosely coupled to an external data base, where patient data and clinical events are stored and updated. The input data for the RÉSUMÉ system and its output conclusions are stored in the temporal fact base (Shahar, Samson, Tu and Musen, 1992).

Temporal abstraction mechanisms are implemented as domain independent rules and functions in CLIPS (an object-oriented shell for knowledge based systems). The rules are triggered in a data-driven fashion. Insertion into the temporal fact base of an event[9], a new data point, or a clinician asserted abstraction can trigger one or more temporal abstraction mechanisms either adding or deleting abstraction intervals[10] to or from the temporal fact base. Abstracted intervals are also stored in the external data base allowing temporal queries (Shahar and Musen, 1993b).

The effect of data which arrive in the present but pertain to an earlier time point is catered for. The non-monotonicity of the knowledge base is achieved by partial revision of the current abstracted history. The data that is affected by the newly added data is re-evaluated, modifying the interpretation of data. This effect is mediated through a truth maintenance system, limiting the propagation of changes throughout the temporal fact data base to only those intervals that might be affected by the change. Event Calculus, utilising negation by failure, deals with such non-monotonicity automatically.

The primary interest of RÉSUMÉ is to generate interval-based abstractions when given time-stamped raw clinical data.

---

[9] An event is defined as an action or a process.

[10] An interval has a start and an end point. Both points can have a temporal granularity (e.g. a day or an hour)

3.3 General Discussion

With the technological advances in the last decade the volume of information available for the health care experts has increased at an astronomical rate. The demand to access the information combined with the drawbacks faced by paper based medical records, and limitations of the human capability to remember and analyse the available information, has made the use of computer technology in health care essential. With the introduction of artificial intelligence techniques to medicine the interest in this area has increased. By late 1970s knowledge based systems in different areas in medicine had started to emerge.

It has become apparent that further progress in the area of artificial intelligence in medicine requires sophisticated means to represent and reason about time (Kohane, 1986). As early as the 1960s the importance of the time dimension in representing the patient information was recognised. Historically, data bases have dealt with time by treating it as another attribute or by attaching a time tag to every record. The time-oriented medical data base ARAMIS is an example of this approach. ARAMIS used the Time-Oriented Data model, TOD. The development of TOD was started in 1968 at Stanford University School of Medicine. The system's representation and syntax explicitly recognised the importance of the time dimension in clinical observations, a feature lacking in traditional statistical-support systems (Shortliffe and Perreault, 1990).

In late 1980s Michael G. Kahn (Kahn,1988) developed an object-oriented temporal network, TNET (a logical extension of the TOD model) to assist the ONCOCIN system in the retrieval of context dependent knowledge[11]. This was further extended to ETNET which supports both storage and retrieval of context-dependent data and context-dependent reasoning (Kahn, Fagan and Tu, 1991a). Both TNET and ETNET perform only a limited number of temporal tasks. They cannot be used as general temporal reasoning models due to their domain dependent temporal assumptions (for example orderings of specific events). But in the support of ONCOCIN they have been successful because of the central importance of adhering to protocols in this field of

[11] All the factual, heuristic,and procedural knowledge is grouped by the context in which it applies. Given a collection of contexts, the system can retrieve only the knowledge that applicable in those contexts.

medicine.

With the objective of applying information and communication technologies to health care, the Advanced Informatics in Medicine (AIM) program (an EC research and development program) has launched a number of projects. The DILEMMA project addresses the shared care principle between primary and specialist care in oncology, including the integration of new knowledge based systems assisted software addressing cancer care management (Jones, 1993). DILEMMA is concerned with sharing information and expertise amongst health care professionals. It aims to provide decision support and communications via knowledge based systems and telematics techniques. The domain of oncology requires a temporal formalism that can express intervals and events. Therefore a combined interval-based and an event-based temporal reasoning approach is used (Reeves, 1993). The system is implemented in DiProlog.

Several query languages incorporating time have also been designed over the last decade to manage the temporal information stored in the temporal data bases. The procedural query language TQuery was developed at Stanford to support construction of temporal rules in ONCOCIN. The popular query languages SQL and Quel have been considered for extensions to support time-oriented facilities. TSQL, which is a superset of SQL, has been proposed to retrieve information from a historical data base (Navathe and Ahmed, 1988). TQUEL (Temporal QUEry Language) was developed to query a temporal data base (Snodgrass, 1987). TQUEL is a derivative of Quel, the query language for the Ingres relational data base management system. The type of queries that could be processed by the data base is limited by the capabilities of the query language used by it.

3.4 Conclusion

We have described a few of the widely used computer-based medical information systems which have considered the representation of time-oriented data and reasoning about time dependent relationships.

At the present time there appears to be a consensus in the research community on the need for rich temporal structures in medical knowledge bases (Dojat and Sayettat,

1993). This consensus and the increasing interest in the research community regarding the provision of temporal capabilities to medical knowledge based systems tends to confirm the necessity of it. But it is also apparent from our discussion how including time is important in medical information systems. Each of the systems discussed has introduced temporal reasoning as an essential part of achieving the desired medical application. The key conclusion we wish to draw from the discussion however, is that each system has explicitly programmed its own temporal reasoning system and that therefore it is worthwhile to investigate whether these special-purpose routines can be derived within a more general temporal framework.

In this thesis we describe a general temporal reasoning formalism that can be used as the backbone to a knowledge based system. Event Calculus allows many features of the other systems to be rationally reconstructed in a very clear way. This makes it easier to modify the system, to develop enhancements and new functionalities without fundamentally changing the temporal reasoning. In the next chapter we describe this formalism, the Event Calculus framework, which we propose for use in the clinical domain.

4 An Event Calculus Framework for Medical Information Systems

The special role that time plays in information processing is an interesting topic in many disciplines especially in medicine. In patient care, the domain to which we apply our formalism, the clinician is not only interested in the present condition of the patient but also in the evolution of the patient's disease, and the patient's medical, social and family history. If the world we represent is to be modelled as it is then we need to model a changing world where the information available changes and gets revised or corrected continuously.

Among the considerable related literature concerned with the formalisation of time (Allen, 1983; Lee, Coelho and Cotta, 1985), the Event Calculus provides an effective framework for reasoning about time using first order logic. It uses the horn clause subset of classical logic augmented with negation by failure, taking the notion of an event as its basic element.

In this chapter we describe a framework based on Event Calculus which can be used for medical information systems. The key point is that the Event Calculus approach integrates time into the basic framework of the knowledge based system, and this distinguishes our approach from the systems described in Chapter 3 which generally introduces time in a more ad hoc way.

The chapter starts with an introduction to the Event Calculus in Section 4.1. We review the original formalisation of the Event Calculus which is more general than the framework we have applied in the medical domain. In Section 4.2 we describe our formalisation of the Event Calculus for medical information systems. Section 4.3 describes extensions that have already been introduced to Event Calculus.

4.1 Introduction to Event Calculus: A Review

Event Calculus is based on general axioms concerning notions of events, relations and the periods of time for which the relations hold. A typical event starts and/or ends zero or more time periods in which a relation holds. As events occur in the domain, the general axioms imply the new relations which hold true in the new state of the world modelled, and infer the termination of the relations which no longer hold true from the previous state.

We use examples, over simplified for clarity, to illustrate the general ideas introduced in the Event Calculus. The Event Calculus rules which we incorporate in our formalism have been represented in Prolog. Identifiers starting with lowercase letters are used to represent predicate symbols and constants, and identifiers starting with uppercase letters are used to represent variables.

Example 1.

Consider the following two events `e1` and `e2` occurring at time instants t_1 and t_2 respectively. Event `e2` happens after the event `e1`.

```
e1      Mary gives bookA to John.

e2      John gives bookA to Bob.
```

The event `e1` of Mary giving the bookA to John starts a period of time (say `p1`) in which the property *John has bookA* is true. Similarly, event `e2` of John giving bookA to Bob starts a period of time (say `p2`) in which the property, *Bob has bookA* is true. The two properties: *John has bookA*, and *Bob has bookA* can be represented in Prolog by the two relations,

```
has(john, bookA) and
has(bob, bookA),
```

respectively. This sequence of events could be illustrated as in Figure 4.1.

The two events `e1` and `e2` can be described in Event Calculus using Prolog clauses as given in Figure 4.2.

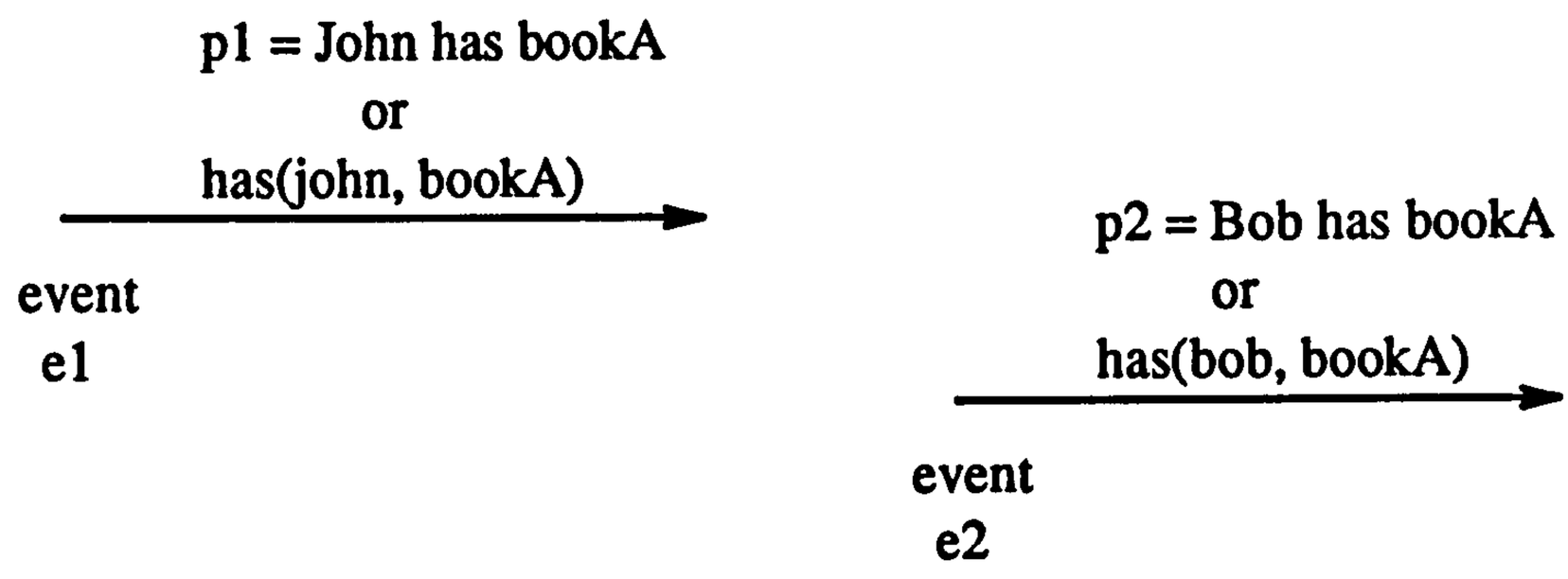


Figure 4.1: The event sequence of e1 and e2 and the properties holding true after the events

```
action(e1, give).                    action(e2, give).
time(e1, t1).                        time(e2, t2).
object(e1, bookA).                   object(e2, bookA).
giver(e1, mary).                     giver(e2, john).
recipient(e1, john).                 recipient(e2, bob).
```

Figure 4.2: Description of the events e1 and e2 in Event Calculus

We can say that after the event `e1` the relation `has(john, bookA)` holds true or in other words the event `e1` initiates the relation `has(john, bookA)`. We also can say that event `e1` ends the time period in which the property *Mary has bookA* was true or in other words, the event `e1` terminates the relation `has(mary, bookA)`. Similarly, we can say that event `e2` terminates the relation `has(john, bookA)`.

The initiating and terminating rules are defined using the domain dependent knowledge. Basically the initiate and terminate rules define the relations started and ended by each event that can occur in the domain. The Event Calculus axioms use this knowledge to infer the state of the world at any given instant.

For example we can write using Example 1 given above in Figure 4.1, and the representation given in Figure 4.2, as:

```
An event E initiates the relation has(X,Y) if
        the event E is of type give and,
        the object given is Y and,
```

the recipient is X.

An event E terminates the relation has(X,Y) if
the event E is of type give and,
the object given is Y and,
the giver is X.

These rules can be represented in Prolog as in Figure 4.3.

```
initiates(E, has(X, Y)) :-
                action(E, give),
                object(E, Y),
                recipient(E, X).

terminates(E, has(X, Y)) :-
                action(E, give),
                object(E, Y),
                giver(E, X).
```

Figure 4.3: Prolog rules representing the initiation and the terminating of the relation `has(X,Y)`

The first rule in Figure 4.3 expresses that if an event `E` represents the action of giving an object `Y` to the recipient `X` then, the event `E` initiates the relation `has(X, Y)`. The second rule denotes that, if there is an event `E` which represents the action of `X` giving an object `Y` then, that event terminates the relation `has(X, Y)`. We call the relations initiated and terminated by events *base relations*. Figure 4.4 illustrates the relations initiated and terminated by events `e1` and `e2`.

Event Calculus associates time periods with relations through the predicate holds (Kowalski and Sergot, 1986). The predicate `holds(after(E, U))` denotes that the relation `U` holds true for the period `after(E, U)`. Similarly that the relation `U` holds true for the period `before(E, U)` is denoted by the predicate `holds(before(E,`

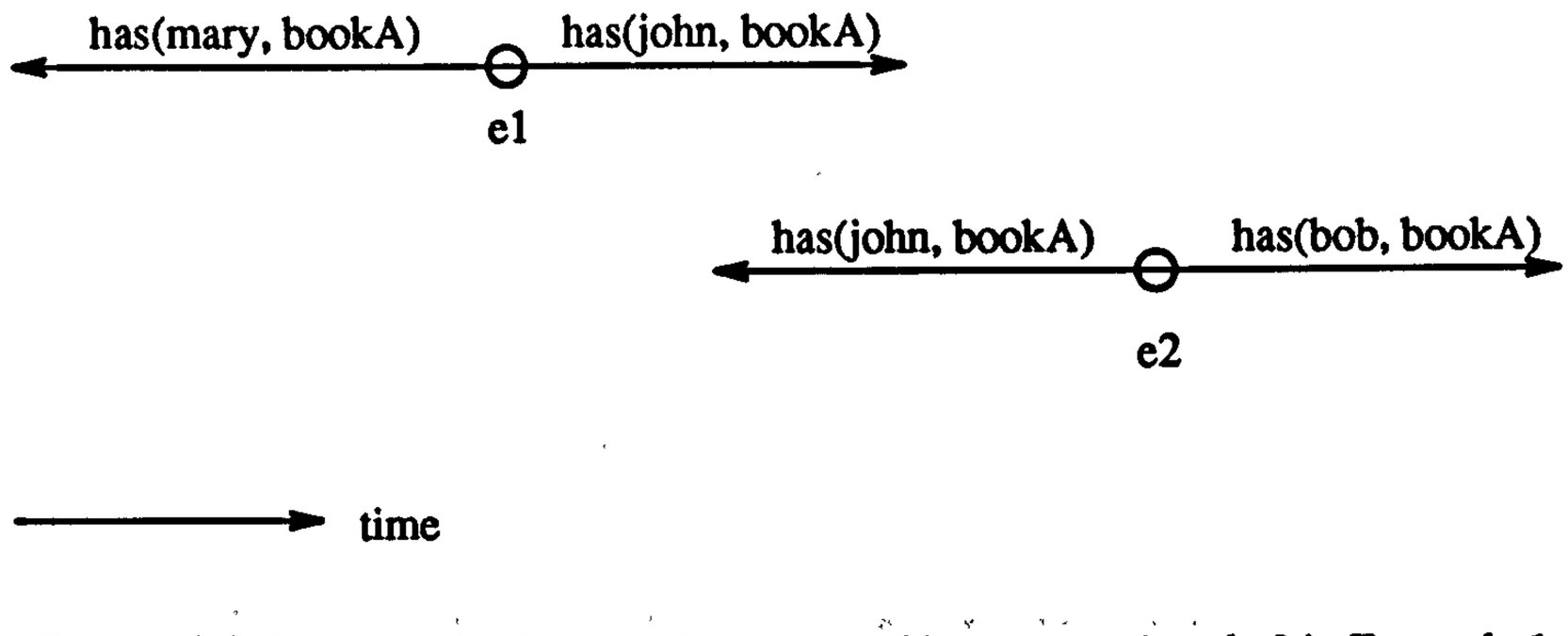


Figure 4.4: Relations initiated and terminated by events e1 and e2 in Example 1

U)). For example, that the relation has(john, bookA) holds true after the event e1 is represented as,

```
holds(after(e1, has(john, bookA))),
```

and the fact that the relation has(bob, bookA) is true after the event e2 is represented by,

```
holds(after(e2, has(bob, bookA))).
```

These facts can be deduced from the following rules:

```
holds(after(E, has(X,Y))) if initiates(E, has(X,Y)).

holds(before(E, has(X,Y))) if terminates(E, has(X,Y)).
```

The first rule states that the relation has(X,Y) holds true in the time period after(E, has(X,Y)) if the event E initiates the relation has(X,Y). The second rule states that the relation has(X,Y) holds true in the time period before(E, has(X,Y)) if the event E terminates the relation has(X,Y). More generally, for an arbitrary relation U these rules can be expressed as given in Figure 4.5.

In Figure 4.5 the rule EC1 can be paraphrased as, a relation U holds true for the time period after(E,U) if the event E initiates the relation U. The rule EC2 can be paraphrased as, a relation U holds true for the time period before(E,U) if the event E terminates the relation U.

```
holds(after(E,U)) :-
                    initiates(E,U).                    EC1

holds(before(E,U)) :-
                    terminates(E, U).                  EC2
```

Figure 4.5: The Event Calculus rules for the predicate `holds`

These time periods do not include the events which start and/or end them. For the period `after(E,U)`, the starting event is known to be `E`. If the terminating event is not known to exist then the period is believed to extend to the future indefinitely. Otherwise the period is believed to extend to the future until such time as it is known that an event must have terminated it. There are several ways in which this information can be deduced (see below the discussion about Figure 4.7). Thus the determination of the end point of `after(E, U)` assumes forwards persistence. Similarly, for the time period `before(E,U)`, the terminating event is known to be `E` and the initiating condition is deduced (assuming backwards persistence).

Knowing the time period for which a relation holds true, given any time instant `T` it can be determined whether a relation holds at that time instant or not. A relation `U` holds true at any instant `T` if it is known that `U` holds true for a time period `P` and `T` occurs during the period `P` (Kowalski, 1992). The period `P` can be defined as `after(E, U)` or `before(E, U)`. This is represented by the rules given in Figure 4.6. The axioms given in Figure 4.6 can be paraphrased as,

A relation U holds true at time instant T if,

the relation U holds true in the time period after(E, U), and

the time instant T is in after(E, U).

A relation U holds true at time instant T if,

the relation U holds true in the time period before(E, U), and

the time instant T is in before(E, U).

A time instant T is in the time period P if,

the event which starts the period P occurs at T1, and

```
holds_at(U, T) :-
                    holds(after(E, U)),
                    T in after(E, U).                    EC3.1

holds_at(U, T) :-
                    holds(before(E, U)),
                    T in before(E, U).                   EC3.2

T in P :-
        start(P, E1),
        time(E1, T1),
        end(P, E2),
        time(E2, T2),
        T1 < T < T2.                                     EC4
```

Figure 4.6: Event Calculus axioms for the predicate `holds_at`

---

the event which ends the period P occurs at T2, and
the instant T occurs after T1 and before T2.

We use the operators $'>', '<', '\leq', '\geq'$ to represent the ordering of time (and hence the events). `time(E, T)` denotes that the event `E` occurred at time instant `T`. Here `'in'` is used as an infixed predicate symbol. `start(P, E)` and `end(P, E)` represent the start and the end events of the time period `P` respectively. `start(P, E)` expresses that the event `E` starts the time period `P` and `end(P, E)` expresses that the event `E` ends the time period `P`. These are described by the Event Calculus axioms given in Figure 4.7.

The rules EC5 and EC7 express that the event `E` starts the time period `after(E, U)` and the event `E` ends the time period `before(E, U)` respectively. The operator '=' denotes that the time period to its left is the same as that appearing to the its right. Rules EC6 and EC8 express that if the time periods `after(`E'`, U)` and `before(E,U)` are one and the same then, the event E' starts the period `before(E, U)` and the event E

```
start(after(E, U), E).                                    EC5

start(before(E, U) ,E') :-
                    after(E', U) = before(E, U).          EC6

end(before(E, U), E).                                     EC7

end(after(E', U),E) :-
                    after(E', U) = before(E, U).          EC8
```

Figure 4.7: Event calculus axioms for the relations `start` and `end`

ends the period `after(`E'`, U)` respectively. The conditions under which two periods can be identified as the same (i.e. the definition of the '=' predicate) is an important, but still a problematic, area of the Event Calculus. Considering the period `after(E, U)` there are a number of ways in which we may deduce an 'end' event E'. The simplest case is when we know of a later event E' which terminates `U` (and hence is the end point of a period `before(`E'`, U)`) and there is no other way by which we can deduce that `after(E, U)` ended before E'. The other cases occur when we know of an event E'' (later than `E`) which initiates or terminates a relation incompatible with `U`. We can then deduce that there must be an event E' earlier than E'' and later than `E` which ended `after(E, U)`. There are also a similar set of cases for deducing the start point of `before(E, U)` which assume persistence backwards in time. All these different cases were fully enumerated in the original Event Calculus paper (Kowalski and Sergot, 1986). In our application of the Event Calculus to the medical domain, described in the next section, we avoid the problematic cases for deducing the end (start) points to time periods.

Using the `holds_at` predicate we can query whether a particular relation holds true at any given time. Going back to the example in Figure 4.2, say it is known that time instant t_a occurs after t_1 and before t_2 and time instant t_b occurs after t_2. This is illustrated in Figure 4.8.

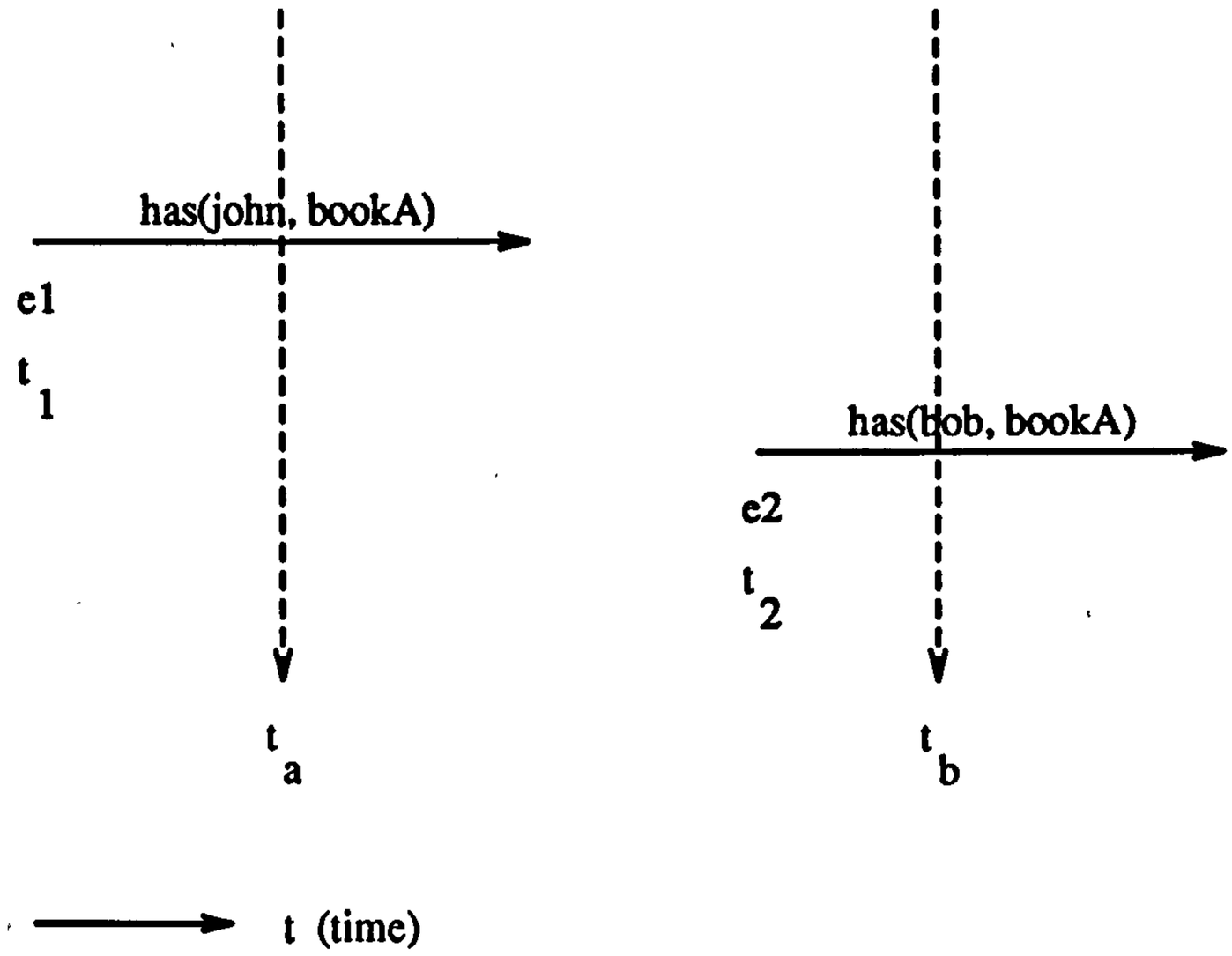


Figure 4.8: Illustrating the relations holding at time instants t_a and t_b in the event sequence of e1 and e2

Now if the query,

```
?  holds_at(U, t_a)
```

is posed the answer would be,

```
U= has(john, bookA).
```

Whereas, the query,

```
?  holds_at(U, t_b)
```

would be answered as,

```
U= has(bob, bookA)
```

since `has(john, bookA)` no longer holds true after e2.

The Event Calculus allows us to treat past and future symmetrically. Event descriptions can be assimilated in any order irrespective of the order in which they actually occur. This also facilitates the handling of incomplete information. Incomplete event descriptions are dealt with by using default reasoning. When an update

description conflicts with information derived by default reasoning the update is accepted and the conflicting information earlier derived by default is automatically and non-monotonically withdrawn. This is illustrated by the following example.

Example 2. (Extension of Example 1)

```
e4      bookA is given to Ann.
```

Let us assume that the event e4 occurs at time instant t_4 after t_2. Then we can describe event e4 as in Figure 4.9.

```
action(e4, give).
time(e4, t4).
object(e4, bookA).
recipient(e4, ann).
```

Figure 4.9: Representation of the event of Ann receiving bookA in Event Calculus

Let us consider two time instants t_c and t_d. The time instant t_c lies in between the time instants t_b and t_4 and the time instant t_d occurs after t_4 as shown in Figure 4.10.

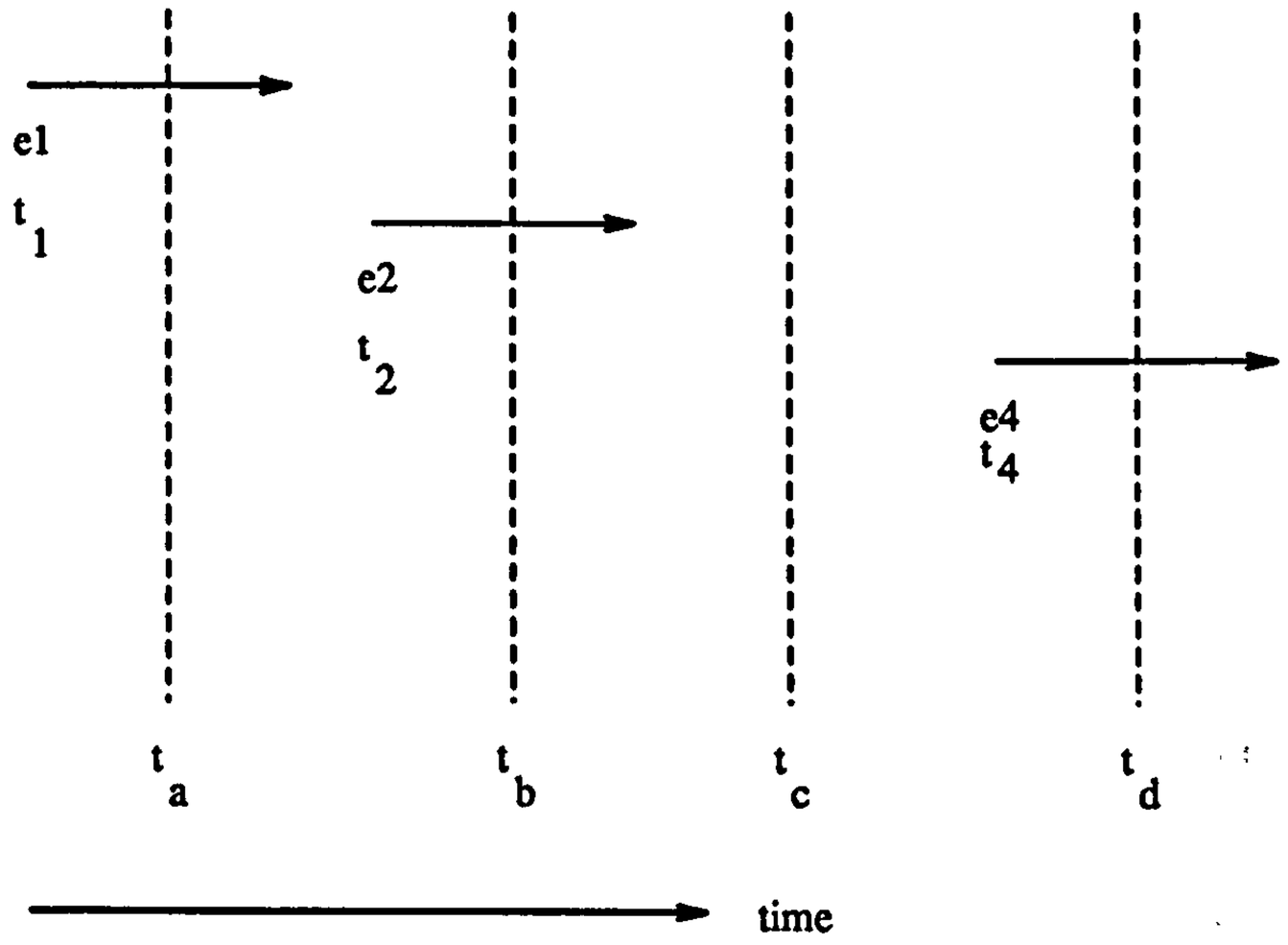


Figure 4.10: Illustrating the time points t_1, t_2, t_4 and t_a-t_d in the event sequence of e1 to e4

In this situation if the query,

```
?  holds_at(U, t_c)
```

is posed, the answer received will be,

```
U = has(bob, bookA).
```

Assume the following information was received at this stage:

```
e3      Bob returns bookA back to Mary
```

The event e3 occurred at time instant t_3, after t_2 and before t_4. The event sequence after the receival of information about event e3 is illustrated in Figure 4.11.

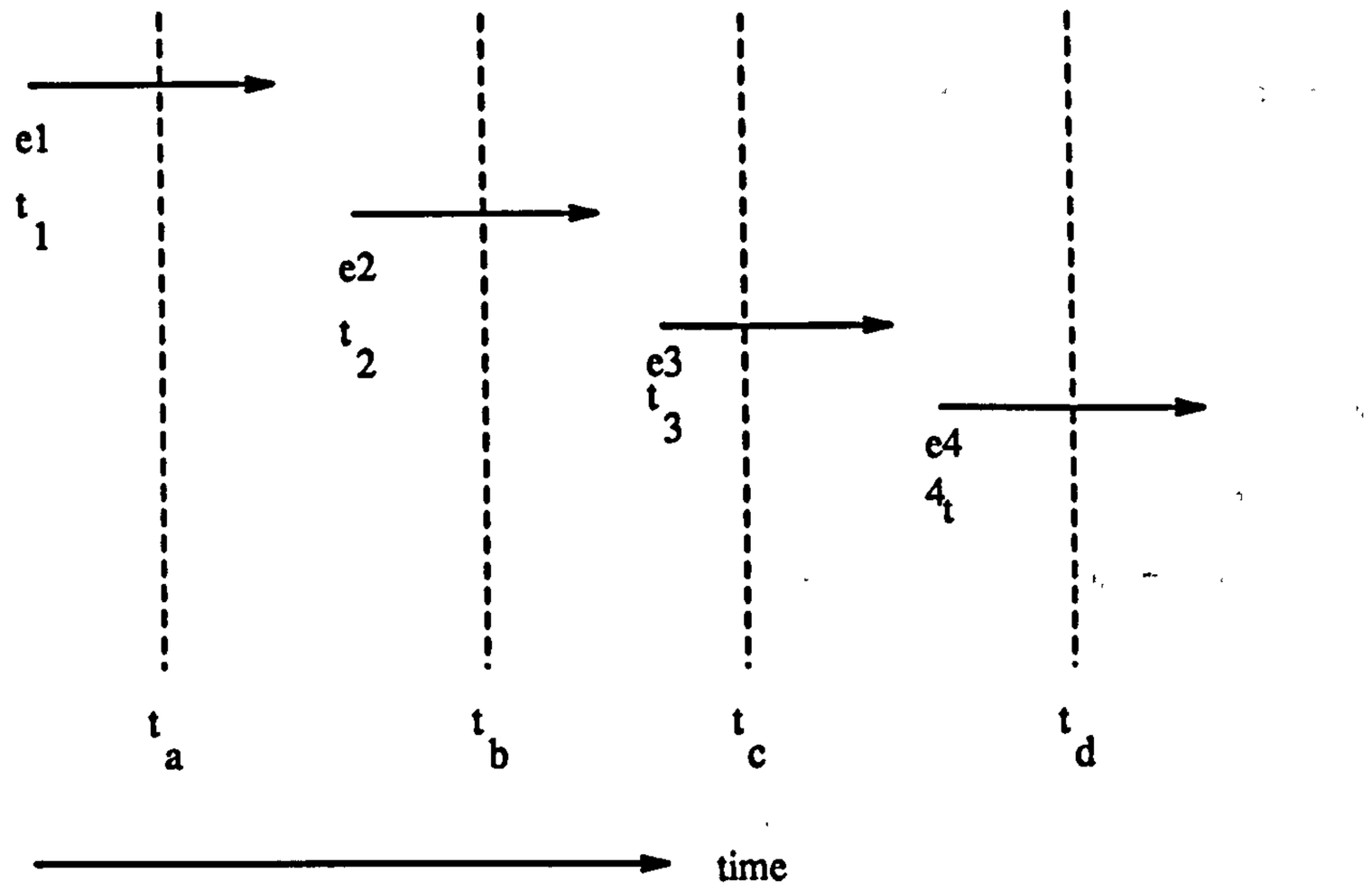


Figure 4.11: Illustrating the time points t_1-t_4 and t_a-t_d in the event sequence of e1 to e4

If the same query, `holds_at(U, t_c)` is posed now, the answer will be,

```
U = has(mary, bookA).
```

The former conclusion that, *'Bob has bookA at time t_c'* is non-monotonically withdrawn and a new conclusion that, *'Mary has bookA at time t_c'* is made.

As it is clear from the given examples all updates are additive in Event Calculus. There is no explicit deletion of information as in conventional data bases. The termination of a relation or that a relation no longer holds true is denoted by,

1. adding information about an event which terminates the relation thus ending the time period for which the relation held true, or

2. adding an event which initiates a new relation which implies the end of the former, or

3. adding an event which revises the former belief of the state of the world thus starting a new belief period.

We do not make use of the third method mentioned above, since we have not considered the transaction time of events which leads to belief periods. Work on belief periods could be read in (Sripada, 1988).

Events are given unique names which makes it easy to add new information about events already described earlier. In the example given in Figure 4.10 the event `e4` has incomplete information. It is known that Ann was given a book, but no information is given as to by whom. If later it was known that the giver was Mary, this fact could be just added to the event descriptions of `e4` as `giver(e4, mary)`.

Events need not be associated with absolute time, they can act as temporal references for example, the information that `e1` occurred before `e2`. We can associate a numerical time to each event but this is not essential as what is important is the ordering of the events.

Events, time periods and time instants are clearly distinct from one another. Therefore, it is possible to have simultaneous events. For example, there can be two simultaneous events one of interview and another of taking pulse. They will have same event times but different event names will distinguish them from one another.

Events can be considered to be taking place at instants of time. It should be noted that the time periods do not contain the events which start or end them. This leaves the possibility of events having durations. Work has already been done extending the Event Calculus to deal with events with duration (Sadri, 1986; Evans, 1989; Evans, 1990).

4.2 Event Calculus Used in the Application Domain

4.2.1 Adapting the holds_at Predicate

We believe that a simplified form of the Event Calculus is sufficient for our domain. The major adaptation we make to the original Event Calculus formulation presented in the previous section, is to only consider the relations persisting forwards in time. The original Event Calculus treats the past and the future in a symmetric fashion. This is not necessarily an advantage since it ignores the 'arrow of time', the manifest difference between the past and future. The significance of making the distinction when describing change in a logic programming framework, and in particular in the Event Calculus, has been discussed by Shanahan (Shanahan, 1989). In the type of applications we are considering, semi-prescriptive knowledge based system applications, it is important to recognise incompleteness in past information. Clearly this is so for medical applications. In such applications the uncontroled inclusion of backwards persistense is harmful because it hides the need to seek further information. In contrast an approach based on forwards persistence can be used to encourage good practice, in our case good practice in the medical field. We illustrate our viewpoint with a simple example.

Example 3. Suppose the event `e1` denotes 'Johnny is admitted to hospital.' In the Event Calculus we can conclude,

(1) `holds_at(registered(johnny), T)` `T` after `e1`

(2) `holds_at(unregistered(johnny), T)` `T` before `e1`

But the significance of (1) and (2) is very different. (1) expresses a *post-condition* of `e1`, that is a prediction that Johnny is registered. This prediction can be achieved by implementing forwards persistence based on causality. On the other hand (2) expresses a *pre-condition* for `e1`, that is we are required to explain how `e1` satisfied its pre-condition. Such an explanation can be achieved by abduction, hypothesising any additional events needed to make the pre-condition of `e1` true.

To continue the example, suppose `e1` occurred on Saturday, but also on the previous Monday Johnny had been admitted to hospital and registered. According to the knowledge base, using forwards persistence, Johnny would be registered on

Saturday and therefore the pre-condition for `e1`, that he be unregistered, would not be true. A natural abduction would now assume an event of discharge of Johnny between Monday and Saturday, thus satisfying the pre-condition by forwards persistence. We note in passing that the assumption of complete knowledge about the past would imply that pre-conditions are always satisfied.

We wish to make two points about the above discussion. First that by distinguishing past and future in this manner, we have no need of backwards persistence. Post-conditions are predicted on the basis of forwards persistence and explanations are generated on the same basis. The second point is that this viewpoint is well suited to our application domain. While we do not assume complete knowledge of the past, the aim of a practical knowledge based system for clinical records is that it should accurately reflect the current state of affairs. To a greater extent, therefore, new events will satisfy their pre-conditions in the knowledge base and when they do not we would like to draw attention to this fact. This is the approach we adopt in our system. Forwards persistence is built into the Event Calculus axioms, but, instead of backwards persistence, our system enters an interactive explanation (or abduction) mode which allows the pre-conditions of a new event to be explored. We will describe how the `holds_at` predicate in `EC3.1` of Figure 4.6 introduced in Section 4.1 was adapted to a simpler form on this basis.

Since we include only forwards persistence, we are interested only in the first part of the `holdsat` predicate defined in Figure 4.6, that is in EC3.1. Because the only periods involved are of the form `after(E, U)` we can simplify the definition of the `in` predicate. Only EC5 and EC8 in Figure 4.7 are needed. Making these transformations, the axiom EC3.1 can be written,

```
holds_at(U, T) :-
        holds(after(E1, U)) and
    time(E1,T1),
    T1 < T,
    not (∃E' at T' such that
        T1< T' ≤ T and
        terminates(E', U)).                    (EC3')
```

We do not give the details of this transformation since Kowalski (Kowalski, 1992) has already given $EC3'$ and named it the persistence axiom. The form of the Event Calculus we use is adapted from this axiom with various notational changes and a more detailed specification of the ways in which a relation can be terminated.

It is now notionally simpler to eliminate the time periods altogether by using EC1 and EC2 in Figure 4.5 to replace the `holds(Period)` predicate with `initiates` and `terminates` predicates. The time period `after(E1, U)` in $EC3'$ can be eliminated altogether by introducing the predicate `initiates`. The predicate `initiates(E,U)` denotes that the relation `U` is started (or initiated) by the event `E`, and implies that `U` is true after the event `E`. The introduction of the predicate `initiates` into the axiom $EC3'$ is shown in Figure 4.12.

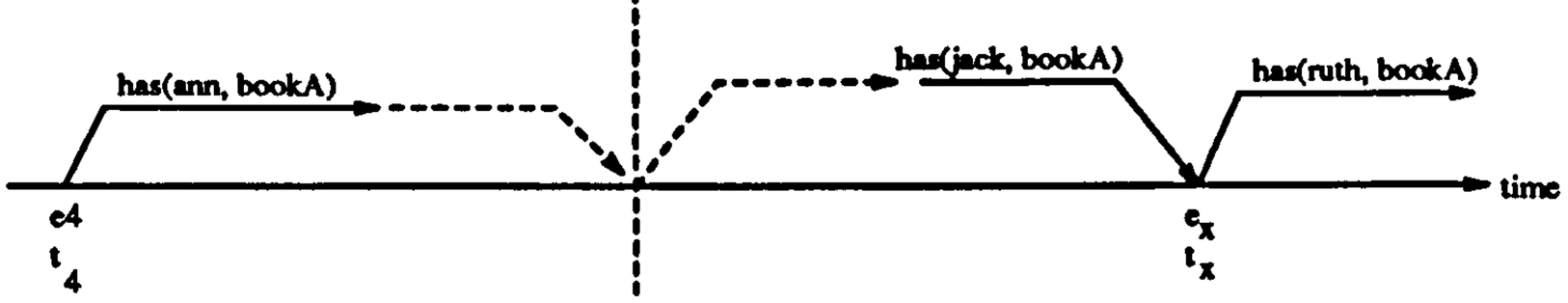
```
holds_at(U, T) :-
        initiates(E1, U),
        time(E1, T1),
        T1 < T,
        not (terminates(E', U),
        time(E', T'),
        T1 < T' ≤ T ).                                  EC9
```

Figure 4.12: Introduction of the relation `initiates` to the `holds_at` axiom

The axiom `EC9` denotes that, a relation `U` holds true at time instant `T` if it is initiated by an event `E1` occurring before `T` and, there is no intervening event E' occurring between the event `E1` and time `T` which terminates `U`.

A relation may also cease to hold true by the initiation or termination of an identical or an incompatible relation (a relation which cannot be true simultaneously). For example, taking Example 2 given in Section 4.1, let us consider the event `e4` occuring at t_4 of giving bookA to Ann. It can be inferred that the relation `has(ann, bookA)` holds true after `e4`. Suppose it becomes known that event e_x occurred at t_x after t_4, where bookA is given to Ruth by Jack. By default we can reason that there should have occurred some event after t_4 and before t_x which initiated the relation

has(jack, bookA). If the domain specifies that a book cannot be possessed by two people, then we can safely say that the two relations has(ann, bookA) and has(jack, bookA) are incompatible. Thus, the initiation of has(jack, bookA) terminates the relation has(ann, bookA). This is illustrated in Figure 4.13.



The solid lines denote what is known
The broken lines indicate what is assumed by default

Figure 4.13: Terminating of a relation by the initiation of an incompatible relation

The above described phenomenon can be incorporated by using a predicate exclusive. exclusive(U, U') is true when the two relationships U and U' are identical, or they are incompatible, as shown in the following rules:

```
exclusive(U, U).
exclusive(U, U') :-
        incompatible(U, U').
exclusive(U, U') :-
        incompatible(U', U).
```

The predicate incompatible is domain dependent. For example considering the example given above, we can write,

```
incompatible(has(ann, bookA), has(jack, bookA)).
```

or more generally,

```
incompatible(has(Person1, Object),
     has(Person2, Object)) :-
          not(Person1=Person2).
```

In the above rule the operator '=' defines unification (not to be confused with '=' of Figure 4.7).

Terminating of a relation by the initiation or termination of an exclusive relation can be represented as:

```
terminates(E, U) :-
            initiates(E, U'),
            exclusive(U, U').

terminates(E, U) :-
            terminates(E, U'),
            exclusive(U, U').
```

The representation of the axiom EC9 can be improved by introducing a predicate `interrupted` to arrive at rule `EC9.1` given in Figure 4.14. `interrupted(T1,U,T)` denotes that predicate `U` is interrupted between the time instants `T1` and `T`. A relation `U` is interrupted between the time instants `T1` and `T` if there is an intervening event between `T1` and `T` which terminates the relation `U`.

```
holds_at(U, T) :-
            initiates(E1, U),
            time(E1, T1),
            T1 < T,
            not interrupted(T1,U,T).                    EC9.1

interrupted(T1,U,T) :-
            terminates(E', U),
            time(E',T'),
            T' > T1,
            T' ≤ T.                                     EC10
```

Figure 4.14: Introduction of the relation `interrupted` in the `holds_at` relation

Kowalski in (Kowalski, 1992) also suggests the possibility of adding an extra condition to limit the persistence to a period of time reasonable for the relation. We introduce this idea through the predicate `too_long_after`, which will be described in the next section. Subject to this modification EC9.1 and EC10 are the domain independent axioms of the Event Calculus used in our approach.

4.2.2 Introducing Autonomous Change

Sometimes a relation may change without the presence of any known new event. For example, say an event of taking the pulse happens on 12th July 1990 and a pulse count of X was noted. Three months after the event, on 15th October 1990, it is unreasonable to assume that the same pulse count holds. What was true on 12th July 1990 has ceased to be true some where between the 12th of July 1990 and the 15th of October 1990, although no known specific event has occurred to change it. This kind of 'autonomous' change is common in our domain. Knowledge about a patient's condition can only be relied on for a certain length of time, after which it should be re-assessed.

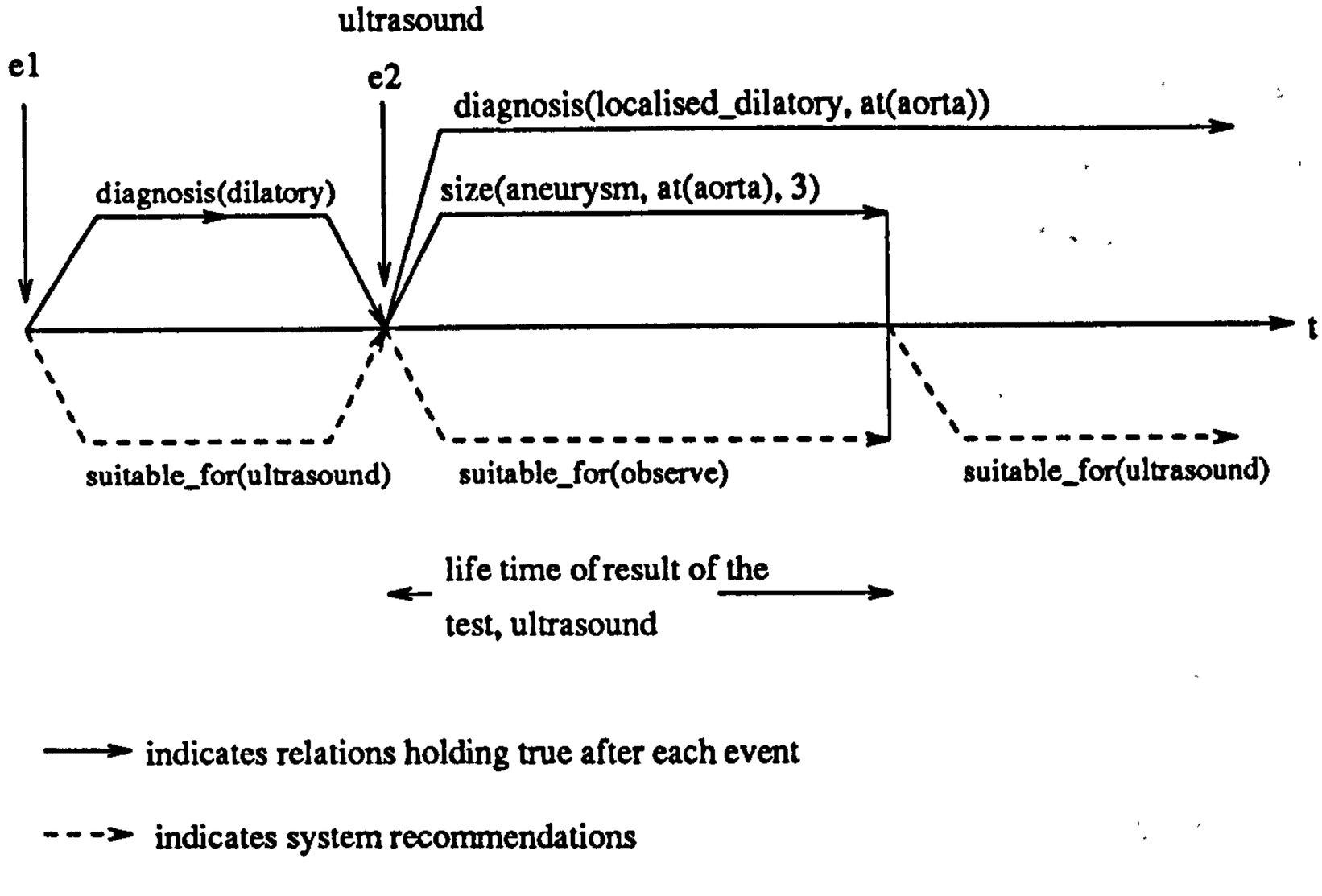


Figure 4.15: Autonomous termination of the relation `size(aneurysm, at(Location), Size)`

For example, an aneurysm[1] is considered to be a risk to the patient only if its

[1]We use the term `localised_dilatory` to refer to the dilated condition of the vessel. The dilation

size exceeds the 'critical size'. 'Critical size' represents the boundary where the 'safe' condition changes to 'unsafe'. When the size of the aneurysm is less than the 'critical size', the risk caused by the aneurysm is less than that caused by subjecting the patient to surgery. If the size is less than critical then the patient is put under observation where tests will be carried out from time to time to assess the condition of the diseased vessel (i.e. the condition of the aneurysm). These repeated tests are done because of the possible changing condition of the vessels. The results of the tests (the aneurysm size) become unreliable after a certain length of time. We call this time period during which a relation's correctness can be relied on as the `life_time` of that relation. In the vascular surgery domain size of the aneurysm (represented by the relation `size(aneurysm, at(Location), Size)` in our formalism) initiated by the test ultrasound, no longer holds true after a certain period of time (the life time of the relation initiated), thus, causing the test to be repeated. This is represented by Figure 4.15.

The Event Calculus axioms are extended to include this feature of autonomous change. This is modelled by the predicate `too_long_after`, which is represented as a general feature of the Event Calculus. The axiom `EC9.1` of Figure 4.14 described in Section 4.2.1 will be now changed to rule `EC9.2` shown in Figure 4.16. The rule `EC9.2` expresses that a relation `U` holds true at time `T` if it is initiated by some event `E1` at time `T1` before `T` and it cannot be shown that `U` was interrupted between the event `E1` and the time instant `T` and, the time point `T` is not too long after the event `E1` which initiated the relation `U`. A time instant `T` is considered to be too long after the initiation of a relation `U` by an event `E1` if the time interval `I` between the event `E1` and `T` is longer than the life time I' of the relation `U`.

The relation `life_time( U,` I'`)` is a domain dependent relation and applies to only those relations for which it is unrealistic to believe that once initiated they persist into the future, for any length of time however long, until they are known to be changed by a later event. The axiom EC9.2 in Figure 4.16 is our form for the domain independent axioms of the Event Calculus.

---

itself is commonly known as the 'aneurysm'.

```
holds_at(U, T)   :-
                    initiates(E1, U),
                    time(E1, T1),
                    T1 < T,
                    not interrupted(T1, U, T),
                    not too_long_after(T1, U, T).          EC9.2

interrupted(T1,U,T) :-
                    terminates(E', U),
                    time(E',T'),
                    T' > T1,
                    T' ≤ T.

too_long_after(T1, U, T)  :-
                    separation(T1, T, I),
                    life_time( U, I'),
                    I' < I.
```

Figure 4.16: Introduction of the notion of autonomous change in the relation `holds_at`

4.3 Extensions That Can be Made to the Event Calculus

In this section we consider a number of ways in which the Event Calculus has been extended. Event Calculus has been extended to include: events in space; events with duration; persistence of relations backwards as well as forwards; and both event time as well as transaction time; and object oriented features. We believe the ability to extend the Event Calculus enhances its suitability as a temporal framework for medical applications. We use predicate names similar to ours in the following descriptions.

4.3.1 Consistency Maintenance

Pimentel and Cuadrado in (Pimentel and Cuadrado, 1990) describe a system used for understanding textual narratives. Event Calculus is formulated within the framework

of stratified[2] logic programs which can be used for general purpose non-monotonic reasoning. The system employs a caching mechanism which stores conclusions drawn from the axioms along with justifications. Maintaining the consistency of the cache after updates is handled by the Consistency Maintenance System (CMS), a logic programming implementation of a justification-based truth maintenance system.

The formalism described by Pimentel and Cuadrado is related to the formalism described in this thesis. Events are considered to occur at time points. Facts about domain objects are recorded in a frame-like fashion using the predicate `holds(Object, Slot, Value)`. For example an event of 'the gunman loaded the gun' is described as below:

```
holds(e1, isa, load).
holds(e1, agent, gunman).
holds(e1, patient, gun).
```

Temporally scoped facts are dealt with using a four argument version of the predicate, `holds(Object, Slot, Value, Period)` which indicates that the object, `Object` has the value, `Value` for slot, `Slot` for a period of time `Period`. The period denotes a term of type `before(E)`, or `after(E)`, for some event `E` which corresponds to the relation dependent periods in the original Event Calculus.

As an example, with respect to the event `e1` given above, it could be inferred (Pimentel and Cuadrado, 1990),

```
holds(gun, load_status, unloaded, before(e1)).
holds(gun, load_status, loaded, after(e1)).
```

Temporally scoped facts such as those given above are derived from instances of case frames using domain dependent rules.

The default assumption of persistence of facts forwards in time is captured by the `holds_at` predicate. The predicate `holds_at(Object, Slot, Value, E)` indicates that a fact holds at the time of an arbitrary event `E`. `holds_at` is defined as,

```
holds_at(Object, Slot, Value, E) :-
```

[2]A logic program is stratified when it is free of recursion through negation

```
holds(Object, Slot, Value, after(E1)),
E1 < E,
not interrupted(Object, Slot, Value, E1, E).
```

This axiom denotes that it is assumed by default that a fact persists from earlier to a later point in time unless it is known to have terminated explicitly by a recorded event. The predicate `interrupted` is true when a fact is known not to hold continuously between two events and is defined by the following rules:

```
interrupted(O, S, V1, E1, E2) :-
    holds(O, S, V2, after(E)),
    exclusive(V1, V2),
    E1 < E,
    E < E2.

interrupted(O, S, V1, E1, E2) :-
    holds(O, S, V2, before(E)),
    exclusive(V1, V2),
    E1 < E,
    E < E2.
```

`interrupted` is defined without negation, so the axioms satisfy the stratification property. Two values (for a given slot in a given instance) are considered to be `exclusive` if they are either identical or incompatible.

Problem solving takes place by querying the stratified logic program, which exhaustively computes answer substitutions for the query predicate by backtracking. Each resulting ground predicate is cached in the CMS with a justification. The CMS maintains the consistency of the cache in the face of non-monotonic updates. The main interest of this work from our point of view is that it shows how consistency maintenance and an object oriented style can be introduced into the Event Calculus.

4.3.2 Representing the History of a Software Project

Nardi and Tucci in (Nardi and Tucci, 1989) use Event Calculus to represent and query the history of the evolution of a software project. The history of a system developed

under System for Access and Version Control (SCAV) is formalised in terms of Event Calculus. SCAV is a tool for the management of a software project, which considers a project as a data base.

Every SCAV command causes a state transition. Thus, there is a direct correspondence between the commands and the events. The format of a SCAV command is,

```
operation/op-name/op-actor/op-argnum/op-arglist
```

The commands are converted to Prolog clauses as:

```
act(operation, op_name).
act(operation, op_actor).
act(operation, op_date).
act(operation, op_argnum).
act(operation, op_arglist).
```

The representation of the history is obtained through the specification of initiates and terminates rules. These rules corresponds to the elements which describe the state. For example, the rules about how the project developers describe their assignment to the project and the subprojects: `inproject(User)`, `notinproject(User)`, `insubp(User, Subproject)`.

Both persistence of relations forwards as well as backwards is assumed. The implementation of the `holds_at` predicate is similar to axiom `EC9.1` described in Section 4.2.1. However the two rules differ in the way the predicate `interrupted` is defined, in particular, a relation can only be interrupted between two events `E` and `E1`, by the initiation or the termination of an exclusive relation inbetween the two events.

```
holds_at(E, R) :-
    holds(after(E1, R)),
    E1 < E,
    not interrupted(E1,R, E).

holds_at(E, R) :-
    holds(before(E1, R)),
```

```
        E < E1,
        not interrupted(E ,R, E1).
```

where, holds is defined as,

```
holds(after(E, R)) :-
    initiates(E, R).

holds(before(E, R)) :-
    terminates(E, R).
```

The predicate interrupted is defined as,

```
interrupted(E, R, E1) :-
    exclusive(R, R1),
    holds(after(E2, R1)),
    E < E2,
    E2 < E1.

interrupted(E, R, E1) :-
    exclusive(R, R1),
    holds(before(E2, R1)),
    E < E2,
    E2 < E1.
```

A relation is always exclusive to itself, and it is exclusive to any incompatible relation. The main interest for us of this work is that it applies the Event Calculus to a very different domain. Also it uses persistence backwards in time rather than our approach.

As regards domain of application, Event Calculus has also been used in air traffic control (Bedford, Rosser and Kowalski, 1990). To the best of our knowledge using Event Calculus as the temporal reasoning mechanism is a novel feature for a medical information system.

4.3.3 Events in Space

Borillo and Gaume (Borillo and Gaume, 1991) extend Event Calculus by adding a spatial component. Their work is based on the belief that temporal components are usually associated with spatial descriptions to picture the world. They assume that the world under study is composed of persons who move in a universe of spatial entities. The universe is composed of places, and each place is a connected part of the space. Relations are defined between individuals and spatial entities, and between event structures and states.

Their attempt is to make the expressive power and the deductive capability of the system when reasoning about time closer to that of humans. To achieve this, the `holds_at` predicate described in Section 4.1 has been extended to a three arguments predicate `holds_3_at`. The additional argument (to that described in Section 4.1) is an answer *yes*, *no*, or *maybe*. If it is known that `U` holds for the period `after(E, U)` (or `before(E, U)`) and `T` occurs during that period then the answer *yes* is given. If a relation `U` holds true for a time period of which either the start or the end is not known then the answer `maybe` is given. For example a relation `U1` holds for the period `after(E1, U1)` and a relation `U2` holds for the period `before(E2, U2)`. The two relations `U1` and `U2` are incompatible and it is known that `E1` occurs before `E2`. When the end of the period `after(E1, U1)` or the start of the period `before(E2, U2)` is not known, it cannot be known exactly at what instant `U1` was terminated or `U2` was initiated between `E1` and `E2`. In this case the value of the third argument (`Answer`) for the query `holds_3_at(U2,T,Answer)`, where `E1> T > E2` , will be *maybe*. The queries to which either *yes* or *maybe* is not true, the answer will be *no*.

Medical decision making often involves uncertainty for the clinician. The main interest in this work is that the choice of giving 'maybe' to a query could be applied to represent uncertainty.

4.3.4 Events with Duration

Evans (Evans, 1989; Evans, 1990) has shown how the Event Calculus axioms can be extended to deal with temporal granularity by introducing the concept of a macro event. He defines a *macro event* as an event whose duration is non zero.

Kowalski and Sergot (Kowalski and Sergot, 1986) have not committed to temporal

granularity. They have taken into consideration time intervals started and/or ended by events, not including the events themselves in the time intervals. This leaves the possibility of having events of duration.

Any event when viewed from different levels of abstraction (having different time granularities) will have different durations of time. For some events it will be unnatural to view them with different durations whereas, for some it may be necessary under certain conditions. For example in a clinical domain let us consider two treatments one of giving a medication (e.g. a pill) and another of surgery (e.g. bypass). It is extremely unlikely that the event of taking a pill may need to be divided into sub-events under any level of abstraction. Contrastingly, the treatment bypass may be viewed with different time durations when viewed from different user levels. It may be an instantaneous event when viewed from the level of clinicians other than those concerned with the operation. Symptoms present before the bypass may be absent after the event. From the level of the anaesthetist the bypass may be an event of duration which can be divided into a number of sub events, during which s/he may have to change the level of anaesthesia.

Evans extends the Event Calculus axioms to deal with time granularity and event/sub-event interaction. The time ordering relation *after* is extended to include time values at different levels of granularity.

Sripada counter argues against taking temporal granularity into consideration (Sripada, 1990). He argues that events are instantaneous when they are viewed as abstract concepts and that one can deal with different levels of abstraction independently of the granularity of time.

4.3.5 Considering both Event Time and Transaction Time

Sripada (Sripada, 1988) has extended the Event Calculus axioms to deal with transaction time and event time separately. He introduces two predicates `basis(Tr, E)` and `revises(E1, E2)` to deal with transaction time. The predicate `basis(Tr, E)` indicates that `Tr` is the transaction which is the basis for believing the information regarding event `E`. `revises(E1, E2)`, denotes that event `E1` is a correction of event `E2`.

With the introduction of the transaction time, the term *belief period* comes into

consideration. The time periods are now represented as, `after(Tr, E)` and `before(Tr, E)`, denoting a time period started by a transaction `Tr` in which the information regarding the event `E` is believed, and a time period ended by the transaction `Tr` in which the information regarding the event `E` was believed, respectively. The predicate `holds` is changed to `bholds`. For example, the predicate `bholds(E, P)` expresses that the event `E` is believed to be true in the period `P`.

Unlike event time the transaction time only moves forward in time in a chronologically ascending order. Still, the events may be entered in any order with respect to the time they occur in the real world. At any given time instant, the data base may not have a complete record of all events that has occurred in the domain but, it will have a complete record of all transactions that had occurred in the data base.

4.4 Conclusion

Event Calculus provides an approach to reasoning about events and time. Because the concepts introduced in Event Calculus can be expressed in Horn clauses augmented with negation by failure, they can be expressed in Prolog thus making the formalism suitable as a implementation framework. The basic notion of the ontology is the event. Event Calculus has gained expressive power by treating events and time explicitly.

The main purpose of this chapter was to describe a frame work based on Event Calculus which can be used as the basis of a medical information system. First we have introduced the Event Calculus as described by Kowalski and Sergot (Kowalski and Sergot, 1986): giving simplified examples we have introduced the basic notion of Event Calculus, relations initiated by events holding true in given time intervals; how the explicit treatment of events allow for updates that provide new information about the past; how default reasoning is obtained on the basis of incomplete information.

By assuming persistence of relations forwards in time alone we arrive at a simplified form of the original Event Calculus given by Kowalski and Sergot. The idea that it may be unreasonable to assume that some relations persist infinitely into future, if it was not known of any event terminating that relation, has been adopted by allocating a life time to such relations. The introduction of this extension completes the simple and flexible framework used by us for temporal support of a medical

knowledge base.

We also reviewed some of the ways in which the Event Calculus has been extended. We believe the range of these extensions confirm the suitability of using Event Calculus as a temporal basis for developing medical information systems.

In the following chapters we describe how this formalism provides a simple but flexible back bone in the representation of a time related knowledge base, especially in the domain of vascular surgery and how this system is used as a decision support tool in the domain.

5 Formalisation of the Vascular Surgery Domain

In this chapter we represent the information recorded in a patient's clinical record in an appropriate form for the Event Calculus. We aim to show how Event Calculus provides a flexible framework for the assimilation of patient information. We model our representation on the patient record currently in use at the Royal South Hampshire Hospital (RHS) (a copy of the record is given in Appendix A). There is ongoing research done at the RHS on improving and standardising the paper-based patient record currently in use.

Clinical data are collected with variable degrees of inaccuracy. The way patients define themselves as 'sick', the answers given by the patients to a clinician's questions, also the questions asked by the clinician etc. depend on the individual's ability to express him/herself and the questioner's way of interpreting the answers. Even in the decision making process the same test result can be interpreted in two different ways by two experts in the field. There is much ambiguity in even defining a diagnosis among different persons. This variability of the clinical domain makes knowledge representation a complex problem.

In Section 5.1 we describe the framework in which we enter the events into the system and give the general event descriptions which are common to every event instance that occur in the domain. Each event type is considered in a separate section starting with the event type interview in Section 5.2. For each event type, the case frame, and the relations initiated and terminated from the case instances are given. In Section 5.3 we describe the modelling of the investigations under the two broad categories: specific tests, and general tests. In Section 5.4 we describe the treatments under the three categories: surgical, non-surgical, and non-active (as shown in the task hierarchy in Section 5.1, Figure 5.1).

5.1 General Event Descriptions

We enter information into the system in the form of event instances. Each instance of an event belongs to an event type. We give the generic name *task* to the types of events which can occur in the clinical pathway. Basically, a task is any event that can be recorded in the patient record. For each task that can exist in the domain we define a case frame.

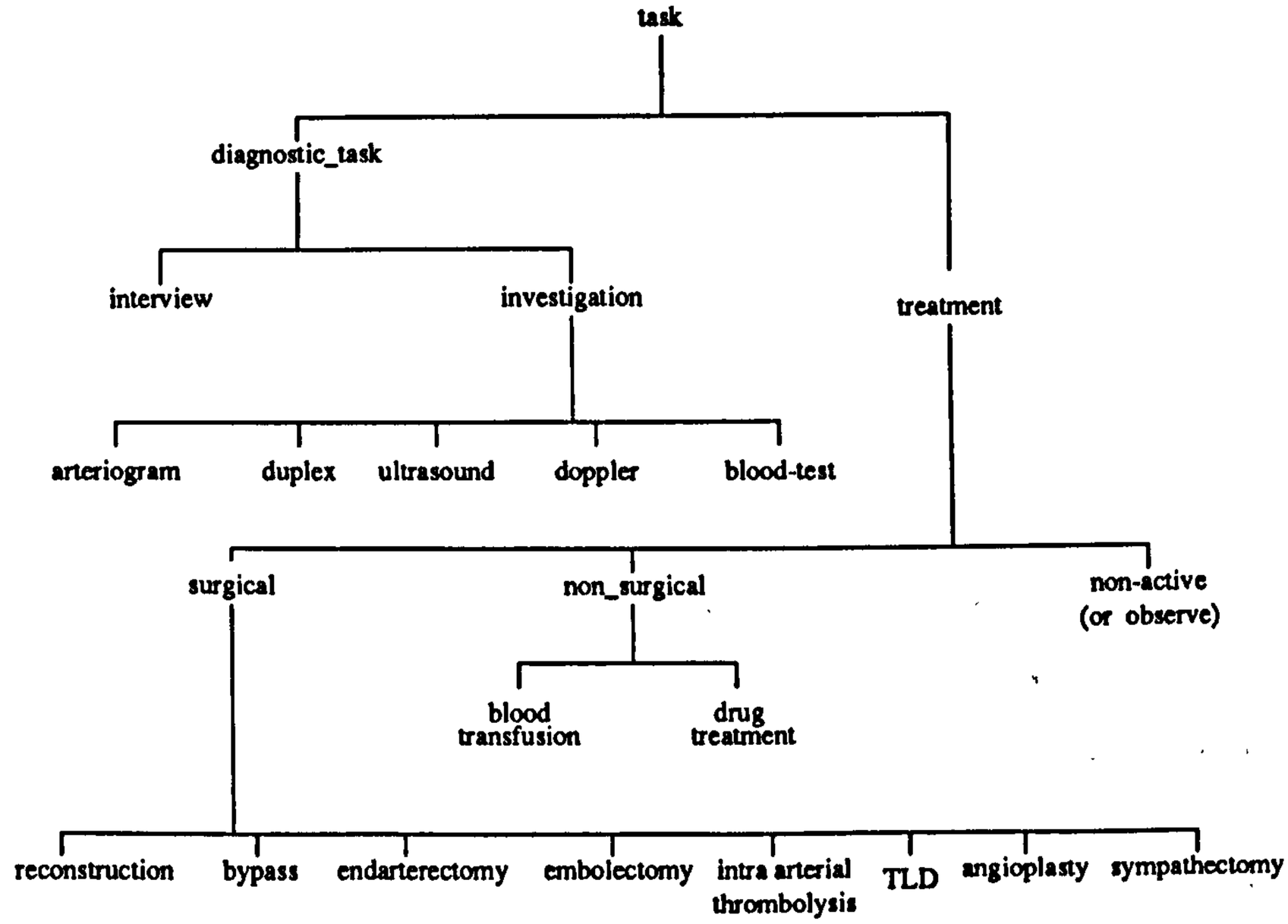


Figure 5.1: The task hierarchy.

Figure 5.1 shows the hierarchy of tasks. This is based on the stages through which a patient admitted for vascular surgery progresses. In order to simplify the clinical pathway, we have broken the pathway into four main stages, interview, investigation, treatment and followup (of a treatment). These four stages have been described in more detail in Chapter 2, Section 2.1. The last stage followup can be called a null stage in the sense that it consists of different combinations of the other three stages. Thus, it is not represented in the task hierarchy in figure 5.1 and will not be discussed in this chapter. Its function is to facilitate system recommendations and give the user a better understanding of the representation.

In principle event instances can be entered into the system in any order and correspondingly there is no ordering implied amongst the event types. Nevertheless there is a notion of the clinical pathway, a sequence of evaluation and treatment stages, through which a patient progresses during their stay in hospital. We have described this pathway in Chapter 2, Section 2.1. The broad stages of the pathway are shown in Figure 5.2.

```
interview      ⟶ leads to ⟶ initial diagnosis

investigation  ⟶ leads to ⟶ final diagnosis

treatment      ⟶ leads to ⟶ followup(of vascular treatment)
```

Figure 5.2: The broad stages of the clinical pathway

In order to give a coherent account of the domain model we follow the order shown in Figure 5.2, picking typical event types and base relations from each stage of the pathway. With each event in the task hierarchy we associate semantic cases (a case frame) which are inherited by the tasks beneath them.

The only events that can occur in reality are instances of leaf nodes in the task hierarchy, for example an instance of interview, arteriogram, angioplasty. Such an event can be recorded in the knowledge base as an instance of a non-leaf task if there is lack of information for some reason. Event instances are formalised in our system asserting facts of the form,

```
inst(E, Event_type)
```

which indicates that E is an instance of an event of type Event_type[1]. Each event is associated with its event time, the time at which the event occurred. This is represented by the event description,

```
etime(E, Event_time)
```

[1]In the following we try to indicate entities formalised in our system in this special font. Also here we have use the variable Event_type in place of Task for clarity

denoting that the event time of the event E is Event_time. Since every event must have the semantic cases inst and etime, even if their values are not known, these 'general' cases are inherited by all case frames (of all tasks) and are suppressed in the discussion below.

5.2 Interview Stage

There is just one event type in this stage, *interview*, the case frame of which is shown in Figure 5.3. The form of the interview case frame is designed to be exactly the same as the form of the current version of the Vascular Record (see Appendix A) currently in use at the RHS.

Each of the cases medical_history, surgical_history, social_history, and family_history represents the patient's past and present details. All non-vascular diseases the patient has suffered in the past such as, diabetes, hypertension, rheumatic fever, chronic bronchitis, renal failure, and asthma which may have an effect on the future treatment plan is represented in the medical_history. surgical_history represents the past vascular surgeries, their dates, the reasons, the procedures (e.g. reconstruction, angioplasty), the vascular locations at which the surgeries were carried out, the outcome of each surgery, and other details pertaining to each treatment, and all non-vascular surgeries which might affect the treatment plan. social_history includes the patient's habits, hobbies, way of living, and marital status. family_history indicates any blood relations who have suffered from vascular or any other associated diseases, and if there is any, the pertaining details.

The cases drugs_taken and allergies are self explanatory. drugs_taken describes the drugs being taken by the patient (if any), and allergies if the patient is allergic to any substance.

The systematic_enquiry represents the detailed enquiry carried out by the clinician about general health and condition of the patient such as, whether the patient has angina, ankle swelling, cough, wheeze, whether can climb stairs, and whether the weight is steady.

The results of the examination carried out by the clinician to determine whether

event type	case	value
interview	medical_history	| the vascular
	surgical_history	| patient
	social_history	| record
	family_history	|
	drugs_taken	|
	allergies	|
	systematic_enquiry	|
	examination	|
	report	complaint/s present
	absent	complaint/s absent
	result	diagnosis (a hypothesis or a conclusion)
	eliminates	diagnosis

Figure 5.3: The case frame of the event `interview`.

the patient has any other clinical problems is represented by the case `examination`. For example whether the patient is suffering from jaundice, cyanosis, or clubbing.

The case `report` represents the complaints present in the patient. An initial interview will contain only complaint/s present. An interview may contain more than one `report`, one for each manifestation. A subsequent interview can have either or both complaints present or/and complaints absent. For example at a subsequent interview a patient may inform that s/he no longer suffers from pain in the left calf. Such symptoms which are reported to be absent are represented by the case `absent`.

In the case labelled `result` the interviewing clinician is expected to identify the diseased problem/s, those of vascular nature (*primary problems*) and those which may have relevance to the decision process (*secondary problems*). The `result` essentially represents the clinician's assumption or conjecture made regarding the patient's disease state. An event of an *interview* may have more than one `result`, each representing a different problem. We term each of these a *diagnosis*. Thus, we are completely avoiding automation of the diagnostic problem frequently addressed in medical knowledge based systems.

After the initial interview, at which much of the information on the record is collected, subsequent interviews will generally change only selected parts of the record. For example, in a subsequent visit the patient might report that the progression of symptoms is better, or that the symptoms are completely absent, or a patient who said s/he is a smoker may in a later visit say that s/he has stopped smoking.

Depending on the observations made at a subsequent interview the clinician can change the conclusion/s made at an earlier interview. This is represented by the case `eliminates`. For example, at the initial interview based on the observations the clinician makes an initial diagnosis of `occlusive_arterial`. At a second interview happening later, the clinician suspects the patient is anaemic and recommends a blood test to be carried out. The results of the test reveals that the patient is indeed anaemic and hence a treatment of blood transfusion is recommended. After the treatment the clinician observes that the manifestations thought to have been caused by the occlusive arterial disease have disappeared and thus concludes that the symptoms had been caused by the anaemic condition. This eliminates the previous conclusion and will be represented by the case,

```
eliminates(Event, occlusive_arterial).
```

Each event occurring is represented by a corresponding instance of a case frame in the knowledge base. Each instance of a case in a case frame is entered into the system as an event description. Each instance of a case frame gives rise to zero or more time-dependent relations which are identified as significant for the management of surgical patients. These relations are termed *base relations*. For example, from an event of type interview the base relations shown in Figure 5.4 are identified.

`symptom(Symptom, Body_side)`

meaning a vascular symptom, Symptom is present on the side of the body, Body_side. The second argument will not be represented in the case where the symptom is not associated to a particular side of the body, e.g. symptom, back pain.

`diagnosis(Initial_Diagnosis)`

meaning that there is a possibility that the patient has disease Initial_Diagnosis.

Figure 5.4: The base relations initiated by the event `interview`.

Using the event descriptions we initiate (and/or terminate) base relations. For every base relation the initiating and terminating conditions are specified. In the rest of this section we describe in detail the initiating and the terminating rules for the base relations given in Figure 5.4.

The symptoms that are identified as caused by vascular diseases are, `claudication`, `rest_pain` on lower limb (also known as pain at rest), `ulceration`, `gangrene`, `back_pain`, and `abdominal_pain`. All symptoms manifest in association with one or both sides (that is left, right or both) of the body. The exception to this are `back_pain` and `abdominal_pain` which are not associated with a particular side of the body. This necessitates having two formats for the relation `symptom`.

The relation `symptom` can be initiated only by an event of an interview. If the symptoms can be associated to a particular side of the body (e.g. pain in left thigh, pain in right calf) then the relation,

`symptom(Symptom, Body_side)`

is initiated. If the symptoms cannot be associated to a particular side (e.g. abdominal pain, back pain) then the relation,

`symptom(Symptom)`

is initiated. The initiating rules for the base relation `symptom` are given in Figure 5.5. The first rule `I1.1` shown in Figure 5.5 can be paraphrased as: an event `E` initiates the

```
initiates(E, symptom(Symptom, Body_side)):-
                inst(E, interview),
                report(E, Symptom, Body_side).          (I1.1)

initiates(E, symptom(Symptom)):-
                inst(E, interview),
                report(E, Symptom).                     (I1.2)
```

Figure 5.5: Initiating rules for the relation `symptom`

relation, `symptom(Symptom, Body_side)` if the event `E` is an `interview` and the case instance `report` reports the fact that the patient has complained that he suffers from `Symptom` on side, `Body_side` of the body. Paraphrased the second rule `I1.2` can be read as: an event `E` initiates the relation, `symptom(Symptom)` if the event `E` is an `interview` and the case instance `report` reports the fact that the patient has complained that he suffers from `Symptom`.

In the interview, the case `report` represents the complaints present as reported by the patients. The relation `symptom` initiated using an instance of the case `report` denoting complaints present is terminated if the patient informs the absence of those complaints at an interview. The terminating rules are given in Figure 5.6.

```
terminates(E, symptom(Symptom, Body_side)):-
                inst(E, interview),
                absent(E, Symptom, Body_side).          (T1.1)

terminates(E, symptom(Symptom)):-
                inst(E, interview),
                absent(E, Symptom).                     (T1.2)
```

Figure 5.6: The terminating rules for the relation `symptom`.

The rule T1.1 states that, if at an event of an interview the absence of the symptoms Symptom on Body_side of the body is reported, then that event terminates the relation symptom(Symptom, Body_side). The rule T1.2 can be interpreted in the same manner, with the difference that the symptoms are not associated to a particular side of the body.

```
initiates(E, diagnosis(D)  :-
inst(E, Etype),
kind_of(Etype, diagnostic_task),
result(E, D).                                    (I2)
```

Figure 5.7: Initiating rules for the relation diagnosis(D)

The case result represents the disease problem/s identified by the clinician at the interview. The second base relation diagnosis, given in Figure 5.4, identified as being initiated by an event of an interview represents these problems. The initiating rule for this base relation is given by I2 in Figure 5.7. The rule I2 states that if any event E which is a diagnostic task has results that the problem D is present, then that event initiates the relation diagnosis(D). The events of type interview and test fall into the category diagnostic_task as represented in the task hierarchy diagram given in Figure 5.1. An initial diagnosis can only be initiated by an event of type interview. But, we have used the term 'diagnostic_task' in the rule in Figure 5.7 since the same rule is used to initiate a final diagnosis in some cases. For example, diagnoses of non-vascular nature like anaemia and diagnoses of vascular nature which do not have any location associated with them like general dilatory.

An initial diagnosis is terminated when a final diagnosis which is a specialisation of the former is initiated, or a diagnostic task eliminates the initial diagnosis. This is represented in Figure 5.8. The rule T2.1 in Figure 5.8 denotes that if an event E initiates a diagnosis NewD which is a specialisation of the diagnosis D then, event E terminates the diagnosis D. A final diagnosis is considered to be a specialisation of its initial diagnosis. For example, anaemia is a specialisation of clinical_anaemia, and localised_atherogenic_arterial is a specialisation of occlusive_arterial.

```
terminates(E, diagnosis(D)):-
            initiates(E, diagnosis(NewD)),
            specialisation(NewD, D).                    (T2.1)

terminates(E, diagnosis(D)):-
            inst(E, Etype),
            kind_of(Etype, diagnostic_task),
            eliminates(E, D).                           (T2.2)
```

Figure 5.8: The terminating rules of the relation `diagnosis(D)`.

The rule `T2.2` denotes that if there is an event `E` of type `diagnostic_task` which eliminates a previously formed conclusion `D` then, that event terminates the relation, `diagnosis(D)`. For example, at an initial interview the clinician concludes the patient has both occlusive and dilatory diseases and an arteriogram is carried out. The arteriogram reveals the clinician's conclusion to be incorrect, and that only occlusive condition is present in the patient. The test eliminates the hypothesis of dilatory and thus, terminates the relation `diagnosis(dilatory)`.

5.3 Investigation Stage

In general tests are instigated by clinicians for three broad reasons: following a treatment protocol (common in oncology); based on the questions posed at an interview; or to confirm the results of a previous test. Whatever the originating reason for a test we have classified them into two broad types:

- specific tests
- general tests

Tests carried out in order to prove or disprove existing beliefs are known as *specific tests* and usually require experts (e.g. radiologist) to perform them. Tests carried out in order to evaluate and make hypothesis about a patient's diseased condition are known as *general tests*.

5.3.1 Specific Tests

The specific tests can be one of arteriogram, duplex, doppler or ultrasound and are usually carried out on a particular location of the arterial tree except in some cases of the arteriogram which may cover the whole body.

The case frame of a specific test is given as in Figure 5.9. `Investigation` denotes

event type	case
`Investigation`	`result`
	`eliminates`

Figure 5.9: The case frame of a specific test.

the specific test carried out. The case `result` represents the observations made from the results of the test about existing disease conditions and is given as,

```
result(Event, Location, Result)
```

Information contained in the argument `Result` of the case `result`, varies according to the test carried out. For example, in an arteriogram this will be the vascular status at a particular location of the arterial tree, in ultrasound it will be the diameter of the vessel at the given location. This will be discussed further under each specific test later in this section.

The second argument `Location` in `result`, represents the disease site in the arterial tree. The arterial tree starting from the infra renal area is divided into 13 sites (shown in Figure 5.10) to facilitate ease of representation. Each site is bounded by an upper and a lower node.

A location in the arterial tree is represented by the pair `(Side, Location)` where `Side` represents left or right side of the body and the `Location` is a site in the arterial tree. The exception to this representation is when the diseased location is one of aorta or aortic bifurcation which is not associated to a side of the body.

The case `eliminates` represents the confirmation that a particular vascular disease does not exist. It will essentially disprove an initial diagnosis made at an earlier

Nodes	Site/Disease extent
Infra Renal	
	Aorta
Low Aorta	
	Aortic Bifurcation
Upper Internal Iliac	
	Internal Iliac
Iliac Bifurcation	
	External Iliac
Upper Common Femoral	
	Common Femoral
Femoral Bifurcation	
	Upper Superficial Femoral
	Profunda Femoral
Middle Superficial Femoral	
	Lower Superficial Femoral
Adductor Hiatus	
	Above Knee Popliteal
Knee Joint Line	
	Below Knee Popliteal
Popliteal Trifurcation	
	Run Off - Peroneal
	- Anterior Tibial
	- Posterior Tibial
Distal Vessel	

Figure 5.10: The 13 sites of the arterial tree (starting from the infra renal area) and the nodes which bound the sites

interview. `eliminates` causes the removal of a previous diagnosis/hypothesis and is represented as,

```
eliminates(Event, Diagnosis)
```

We will consider each specific test separately below, identifying the base relations that are initiated and terminated by each test.

5.3.1.1 The Arteriogram

The arteriogram is a test carried out for many cases of vascular disease since it gives an overall picture of the arterial system. When first done the test is usually carried out on the total relevant area of the human body. For example, we are interested on the part of the arterial tree below the infra renal area and thus the initial arteriogram (if not otherwise specified by the clinician) is carried out for the whole lower trunk of the body. If performed for a second time for the same patient (depending on the patient's condition, e.g. the change in the status of the vascular system has been minimal over the period concerned judged by the progression of symptoms) the arteriogram may be done for a specific section of the body only.

The case `result` of the arteriogram represents the observations made on the X-ray about existing disease conditions and is given as,

```
result(Event, Location, Vascular_Condition)
```

Each instance of the case `result`, represents a diseased location except in the case where multiple diseases may occur at the same arterial site. For example, if the arteriogram shows that both vascular conditions thrombosis and localised atherogenic conditions are present at the arterial site left common femoral then, this will be represented as,

```
result(Event, (left, common_femoral), thrombosis)
```

```
result(Event, (left, common_femoral),
                    localised_atherogenic_arterial)
```

This will be further discussed later in Section 5.3.1.5. So there can be many such instances of the case `result` for one particular event. Usually a segment[2] of the

[2]A segment consists of two or more consecutive sites in the arterial tree

arterial tree will be diseased. We represent a segment as a list of arterial sites. One instance of the case `result` will represent the diseased segment. For example,

```
result(Event, [aorta, aortic_bifurcation], embolism)
```

Sometimes the diseased locations may not be consecutive to one another. In our representation this will give a list of result instances for each location in the arterial tree.

Currently information on the arteriogram is provided by the radiologist on an X-ray report form. The present form is undergoing development, but its contents can be analysed to give the patient's vascular state, that is the condition of the vessels at each of 13 locations on the arterial tree (Figure 5.10). We are interested only on those locations which appear to be not normal (deciding the threshold between the normality and the abnormality is left to the clinician). We represent the abnormality observed as a `Vascular_Condition` in the case `result` which takes the following values:

atherogenic_arterial a thickening of the inner lining of the arterial wall ,

dilatory enlargement or widening of the vessel ,

embolism blocking of the circulation in small vessels by air bubbles or small or part of a blood clot carried by the blood stream ,

thrombosis the blockage of a vessel by the formation of blood clots.

The `Vascular_Condition`, determines the final diagnoses of vascular nature (arterial part) for the patient. The final diagnoses (of vascular nature) which are initiated by their respective initial diagnoses are given in Figure 5.11. The base relations identified in the case of an arteriogram are:

`diagnosis(Diagnosis, Location)`
meaning that the vascular disease, Diagnosis is present at the arterial site, Location.

`suspected_diagnosis(Diagnosis, Location)`
meaning that there is a possibility that the vascular condition Diagnosis may exist at the arterial site, Location.

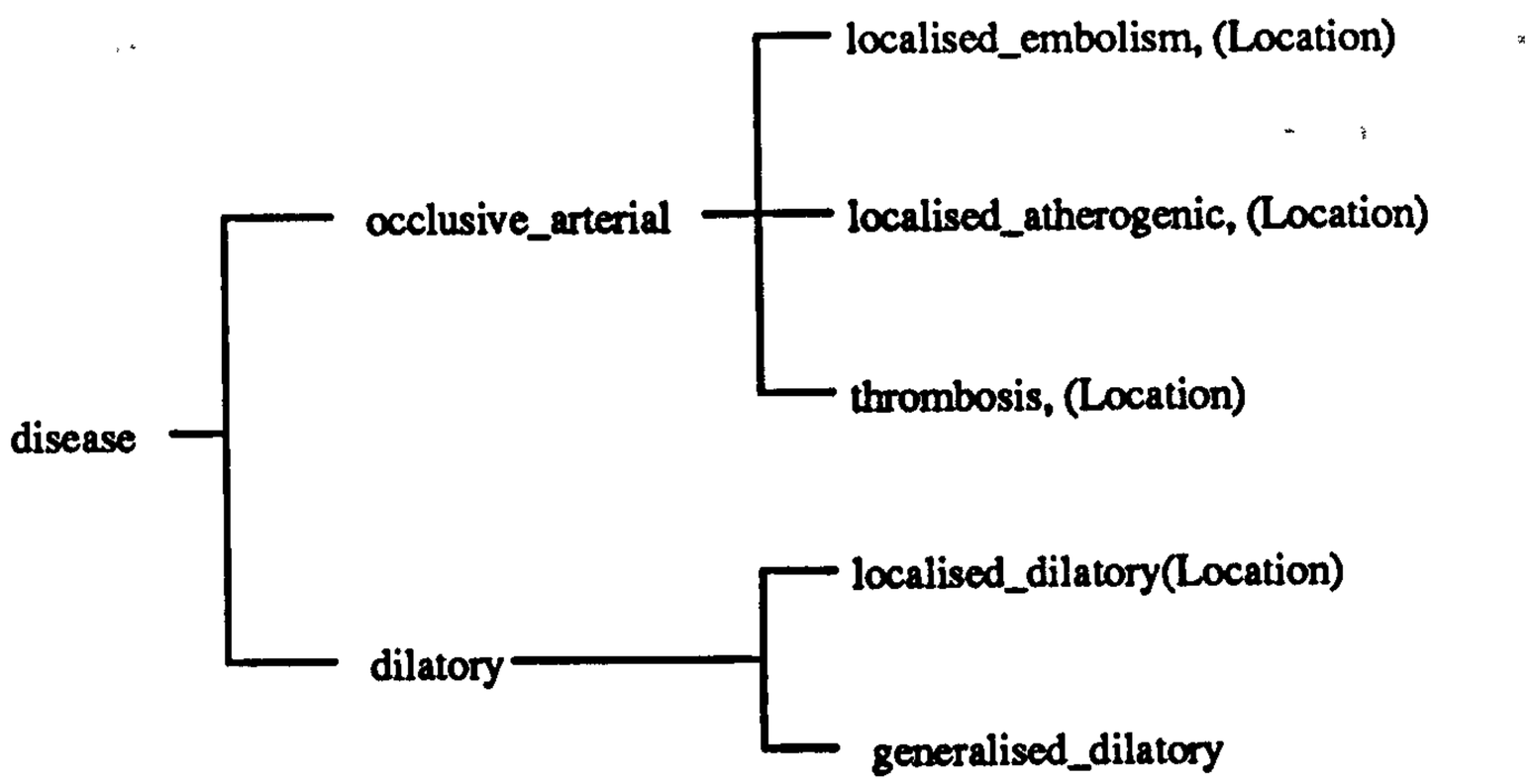


Figure 5.11: Disease hierarchy

Each case `result` initiates a final diagnosis of occlusive nature which is the confirmation of the initial diagnosis made at an earlier interview, or a new diagnosis which was not suspected at the interview stage. The final diagnosis can be considered as a specialisation of the initial diagnosis and will be significant for choosing treatment options for that patient. The initiating rule for a final diagnosis in the case of an arteriogram is given in Figure 5.12.

```
initiates(E, diagnosis(D, Location)) :-
                    inst(E, arteriogram),
                    result(E, Location, D),
                    not (D = localised_dilatory).          (I3)
```

Figure 5.12: The initiation rule for the relation `diagnosis(D, Location)`.

The rule `I3` states that the relation `diagnosis(D, Location)` will be initiated by an event `arteriogram`, if the event description `result` of the arteriogram specifies that `D` exists at `Location`, and the disease condition `D` is not localised_dilatory. The constraint `not (D = localised_dilatory)` in rule `I3` is to avoid a diagnosis being initiated by an arteriogram result in the case when the disease condition is dilatory. The arteriogram result is considered to be a confirmation only for diseases of occlusive nature. For the dilatory condition the `result` of an arteriogram is not sufficient

to initiate a confirmed final diagnosis. For example, in the case it was observed that location `Location` may have dilatory condition, the `result` of the arteriogram initiates a relation `suspected_diagnosis` denoting uncertainty and hence the need for further investigation before a final diagnosis is initiated. The rule for the initiation of the relation `suspected_diagnosis` is shown in Figure 5.13.

```
initiates(E, suspected_diagnosis(localised_dilatory,
                                 Location)) :-
    inst(E, arteriogram),
    result(E, Location, localised_dilatory).          (I4)
```

Figure 5.13: The initiation rule for `suspected_diagnosis(localised_dilatory, Location)`.

The relation `suspected_diagnosis(localised_dilatory, Location)` will be terminated by the results of the test ultrasound, that is by the initiation of a confirmed final diagnosis of localised dilatory, or by the elimination of the suspected diagnosis. The terminating rules for the relation `suspected_diagnosis` will be described in the next section (Subsection 5.3.1.2) under the description of the test, ultrasound.

A final diagnosis is terminated by the event of a treatment carried out at the same location, or if the top and bottom incision points of the treatment, encloses the location of the diagnosis. This is represented by the rules in Figure 5.14. The necessity for having two terminating rules for the relation `diagnosis(D, Location)` will be clear after the discussion of the event of type treatment. The events of type bypass and reconstruction have two locations associated with them, the top and the bottom incision points, whereas other surgical treatments are only associated with one location.

For major vessels the arteriogram result alone is sufficient in most cases to decide on the treatment plan. But, for smaller vessels additional investigation such as a duplex scan may be needed.

```
terminates(E, diagnosis(D, Location)):-
        inst(E, Etype),
        kind_of(Etype, surgical_treatment),
        location(E, Location).                                    (T3.1)

terminates(E, diagnosis(D, Location)):-
        inst(E, Etype),
        kind_of(Etype, surgical_treatment),
        location(E, incision(at_top(Top),
                             at_bottom(Bottom))),
        encloses(Top and Bottom, Location).                       (T3.2)
```

Figure 5.14: The terminating rules for the relation `diagnosis(D, Location)`.

Note: an `and` operator has been defined for readability

5.3.1.2 The Ultrasound

The ultrasound test is carried out to confirm the vascular condition, dilatory. The result is a digital picture which shows a cross sectional view of the vessel. The expert analyses and gives the result of the test as the diameter of the vessel in milli-metres.

When carried out for the first time the ultrasound may be performed at more than one location, that is, on a number of places in the suspected disease area and thus will not be given a particular location. When carried out after an arteriogram the exact location that may be diseased is known and therefore the ultrasound will be carried out at that particular location. The case `result` of the ultrasound is given as,

```
result(Event, Location, Diameter)
```

which indicates that the diameter of the vessel at location, `Location` is `Diameter` (in milli-metres). The ultrasound result suffices for the decision of the treatment plan in the case of dilatory disease except when the disease location is aorta. In that case, an arteriogram is usually done to clearly define the disease area.

The base relation identified in the case of an ultrasound is,

`diagnosis(localised_dilatory, Location)`

meaning that the vascular condition localised dilatory is present at the arterial site, Location.

The base relation `diagnosis(localised_dilatory, Location)` is initiated by the event of an ultrasound if the result of the test denotes that the diameter of the vessel at the location is above normal size. This is represented by the rule given in Figure 5.15. According to the experts the size of the artery can be generalised. We take

```
initiates(E, diagnosis(localised_dilatory, Location)):-
        inst(E, ultrasound),
        result(E, Location, Diameter),
        accepted_normal_diameter(Normal),
        Diameter > Normal.                                    (I5)
```

Figure 5.15: The initiation rule for the relation `diagnosis(localised_dilatory, Location)`

the `accepted_normal_diameter` to be one milli-meter as per the advice from the clinicians at the RHS. This can be avoided by making this a clinician dependent piece of data, in which case it will be made into an event description. The prevailing view is that the `accepted_normal_diameter` can be made into a fixed value.

The relation `diagnosis(localised_dilatory, Location)` will be terminated by the event of a surgical treatment as given by the rules in Figure 5.14.

The relation `suspected_diagnosis(localised_dilatory, Location)` (described in Section 5.3.1.1) will be terminated by the results of an ultrasound. The initiation rule for this relation was given in Figure 5.13. The terminating rules are given in Figure 5.16. The first rule `T4.1` of Figure 5.16 states that the relation,

`suspected_diagnosis(localised_dilatory, Location)`

will be terminated if the ultrasound eliminates the suspicion of dilated disease. The second rule `T4.2` states that the relation will be terminated if the event initiates the relation `diagnosis(localised_dilatory, Location)` confirming the suspicion.

UNIVERSITY OF SOUTHAMPTON LIBRARY

```
terminates(E, suspected_diagnosis(localised_dilatory,
                                                    Location)) :-
        inst(E, ultrasound),
        eliminates(E, (localised_dilatory, Location)).    (T4.1)

terminates(E, suspected_diagnosis(localised_dilatory,
                                                    Location)) :-
        inst(E, ultrasound),
        initiates(E, diagnosis(localised_dilatory, Location)).
                                                          (T4.2)
```

Figure 5.16: The terminating rules for the relation `suspected_diagnosis(localised_dilatory, Location)`

5.3.1.3 The Duplex Scan

The duplex scan is carried out to find out the degree of stenosis of a vessel. Persons may differ when describing an image (e.g. X-ray) of a vessel as being not normal. The distinction between normal and not normal may be clear in some cases and not so in others. When the judgement is difficult the clinician may wish to do further investigation in order to confirm his/her judgement. In the case of arterial occlusion, the duplex scan is carried out for this purpose. It may also be performed when there are multiple occurances of occlusion to find out the location which is most acutely occluded.

The result of the duplex scan is a percentage of reduction in the diameter of the vessel and indicates the altered blood flow. The reduction in the flow can be analysed as *not significant*, *significant*, or *very significant*. Usually a reduction of 50-75 percent is considered to be *significant*, and any percentage above that amount as *very significant*. The case `result` of a duplex scan is represented as,

```
result(Event, Location, Percentage)
```

The test is always carried out at a particular location. The base relation identified in the case of the event duplex is,

`stenosed_site(Location)`

meaning that the vascular site Location is stenosed significantly so as to warrant a treatment

The initiating rule for the relation is given in Figure 5.17. The rule `I6` states that the

```
initiates(E, stenosed_site(Location)):-
                    inst(E, duplex),
                    result(E, Location, Percentage),
                    Percentage > 50.                    (I6)
```

Figure 5.17: The initiation rule for the relation `stenosed_site(Location)`

relation `stenosed_site(Location)` will be initiated by an event of duplex, if the result of the event carried out at the location, `Location` denotes that the percentage stenosed at that location is greater than fifty.

The relation `stenosed_site(Location)` will be terminated by the same criteria as that applied for a final diagnosis at a particular location. Therefore, we can correctly say that the relation is terminated by an event which terminates the diagnosis at that location. This is represented by the rule given in Figure 5.18.

```
terminates(E, stenosed_site(Location)):-
                    inst(E, Etype),
                    kind_of(Etype, surgical_treatment),
                    terminates(E, diagnosis(D, Location)),
                    specialisation((D, Location),
                              occlusive_arterial).          (T6)
```

Figure 5.18: The terminating rules for the relation `significant_site(Location)`.

5.3.1.4 The Doppler Probe

Changes in the blood flow caused by the hardening of the arterial walls can be directly observed by the changes in the speed of blood flow which is essentially the change

in blood pressure. The arterial pressure is measured in the arm and the leg using a doppler probe. A ratio of 1.5-0 is considered as normal. A reduced ratio denotes arterial disease. For patients who suffer from diabetes, the critical value of the ratio may change.

The event description `result` for the test doppler is given as,

```
result(Event, Location, Ratio)
```

which indicates that the doppler index at the location `Location` is `Ratio`. The doppler index is much used in the followup stage to evaluate how successful the treatment has been.

5.3.1.5 Representing Multiple Diseases

There can be more than one `Vascular_Condition` present at any given time. Both dilatory and occlusive conditions can be present simultaneously at different locations, or at the same location of the arterial tree. When both conditions are present at the same location we term the vascular problem as `mixed_arterial_disease`.

In the case of mixed arterial disease, the event descriptions of the arteriogram will contain an instance of the case `result` for each disease condition at that location. For example if the event e3 of an arteriogram reveals that there is dilatory condition at the aorta, and both dilatory and atherogenic arterial conditions at left internal iliac, then the event description of e3 will contain the case instances,

```
result(e3, aorta, localised_dilatory).
result(e3, (left, internal_iliac), localised_dilatory).
result(e3, (left, internal_iliac),
                    localised_atherogenic_arterial).
```

If the dilatory condition is more prominent than the occlusive condition at the site diseased with mixed arterial disease, then the site will be treated as if for the case of localised dilatory and vice versa if the occlusive condition is dominant. The treatment, in any case, will terminate both diagnoses at that location as given by the rules `T3.1` and `T3.2` in Figure 5.14.

When the multiple conditions occur at different sites of the arterial tree, they are considered as multiple occurrences of the localised condition. This situation is dealt

with as described in the above sections, giving multiple instances of the case `result`, each representing one particular location.

5.3.2 General Tests

The function of the general test result can be classified as,

- to confirm the primary problem
- to identify secondary problems if any present
- to monitor existing conditions
- for anaesthetic purposes

General tests are carried out routinely or at the request of a clinician. Some of the routine tests are usually done before the patient is admitted to the hospital (e.g. haemoglobin, blood group, electrolytes) and therefore the results of these tests are available for the clinician at the time of the initial interview. Some routine tests help in establishing the problem, for example,

the volume of the pulse indicates the nature of disease,

bruit[3] indicates roughening of the arterial wall.

Tests carried out at a request depend on the nature of the patient (e.g. chest X-ray, ECG). Some tests depend on patient's response to questions posed at the initial interview (e.g. blood sugar level).

The general tests which are relevant for the decision process of patient management are blood test and pulse. All other tests help in the risk benefit analysis, but we do not deal with this and therefore they will not be discussed in this thesis.

5.3.2.1 Blood Test

The blood test is carried out for two specific reasons in the investigation stage,

- When the patient is suspected to be anaemic, that is, if `diagnosis(clinical_anaemia)` holds true, and

[3]The sound that turbulent blood flow produces in a diseased blood vessel

- When the patient is known to be suffering from hypothyroid, to test the level of thyroxine in the blood stream.

The case frame of a blood test is given in Figure 5.19.

event type	case	value
blood_test	result	| anaemia
		| thyroxine(Level:mmol/litre)
	eliminates	| clinical_anaemia

Figure 5.19: The case frame of a blood test.

In the first case, that is if the patient is suspected to be anaemic, the result of the test initiates the relation `diagnosis(anaemia)` or eliminates the hypothesis `diagnosis(clinical_anaemia)`. In the latter case, that is if the patient is known to be suffering from hypothyroid, then if the test result reveals that the level of thyroxine in the blood stream is lower than that considered to be safe, the test result initiates the relation `below_required_level(thyroxine)`. The initiating and terminating rules for the above described relations will be described below.

The relation `diagnosis(anaemia)` is initiated by the rule `I2` as given in Figure 5.7 in Section 5.2 and is terminated by the rule `T2.1` of Figure 5.8 of the same Section.

The initiating rule for the relation, `below_required_level(thyroxine)` is given in Figure 5.20. The rule `I7` of Figure 5.20 states that the relation,

`below_required_level(thyroxine)`

will be initiated by an event of a blood test if the test result revels that the level of thyroxine in blood is lower than six mmol per litre which is considered by the experts to be the safe level. The relation `below_required_level(thyroxine)` will be terminated if an event of a blood test result reveals the level of thyroxine in blood to

```
initiates(E, below_required_level(thyroxine)):-
                inst(E, blood_test),
                result(E, thyroxine(Level:mmol/litre)),
                Level < 6.                                    (I7)
```

Figure 5.20: The initiation rule for the relation `below_required_level(thyroxine)`

be more or equal to the safe level of six mmol per litre. This is represented by the rule `T7` given in Figure 5.21.

```
terminates(E, below_required_level(thyroxine)):-
                inst(E, blood_test),
                result(E, thyroxine(Level:mmol/litre)),
                Level >= 6.                                   (T7)
```

Figure 5.21: The terminating rule for the relation`below_required_level(thyroxine)`

5.3.2.2 Pulse

The volume of the pulse gives an indication of the vascular condition of the vessels and helps in forming a hypothesis about the patient's condition. The case frame for the event pulse is given in Figure 5.22. The case frame (as shown in Figure 5.22)

event type	case	value
pulse	result	| absent(-)
		| reduced(+)
		| normal(++)
		| increased(+++)

Figure 5.22: The case frame of the event `pulse`.

contains only the case `result` which takes values `absent, reduced, normal` or `increased`. Deciding the volume of the pulse is left to the clinician.

In the current patient record the pulse is represented by the symbols -, +, ++ and +++ representing absent, reduced, normal and increased respectively. The user may input either in symbolic form or in the respective text format.

The event description `result` of the event pulse will initiate the base relation `pulse(Volume)` using the rule given in Figure 5.23. The terminating rule is given

```
initiates(E, pulse(Volume)):-
                    inst(E, pulse),
                    result(E, Volume).                    (I8)
```

Figure 5.23: The initiation rule for the relation `pulse(Volume)`

```
terminates(E, pulse(Volume)):-
                    inst(E, pulse),
                    initiates(E, pulse(NewVolume)),
                    not Volume = NewVolume.               (T8)
```

Figure 5.24: The terminating rule for the relation `pulse(Volume)`

in Figure 5.24. The base rule initiated will be terminated by a different result of a subsequent event `pulse`, that is by the initiation of another relation giving a different volume of pulse.

5.4 Treatment Stage

The treatment event type can be surgical, non-surgical or non-active as shown in the task hierarchy in Figure 5.1. It is convenient to have a treatment type non-active although this introduces no change in the patient. It is sometimes referred to as *observation* by clinicians. The other two treatment types are discussed below.

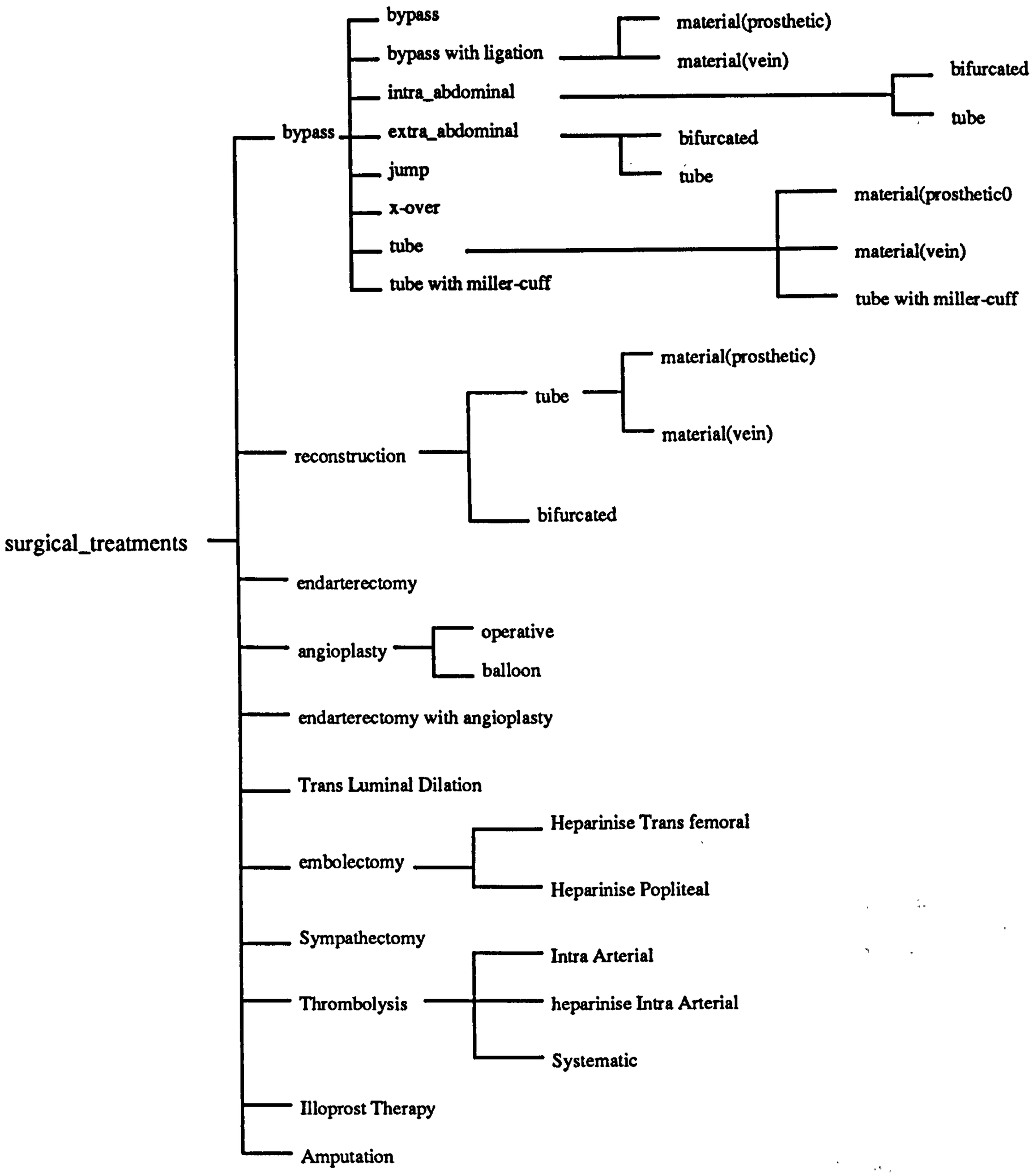


Figure 5.25: The surgical treatments

5.4.1 Surgical Treatments

Our main focus here is on surgical treatments. There are 11 main types of surgical treatments which are further broken down to 29 treatments based on the material used and the type of the graft. The complete list is shown in Figure 5.25.

All surgical treatments inherit the case instances of `location` and the `outcome` of the treatment. The case frame of the surgical treatments of the type bypass and reconstruction is shown in Figure 5.26. `Treatment` denotes the specific treatment

event type	case	value
Treatment	location	| top incision
		| point and the
		| bottom incision
		| point
	outcome	| patent
		| occluded
	material_used	| prosthetic
		| vein
		| dacron ...

Figure 5.26: The case frame of the events of type `bypass` and `reconstruction`

represented. `outcome` expresses the condition of the new arterial segment introduced to the vascular tree by the surgery. The new segment replaces the diseased vessel and the latter becomes irrelevant. The new tube is now referred to by the corresponding locations of the vessel it replaced. The surgical history denotes whether a particular location is a graft or the original native vessel. The case frame for all other surgical treatments is given in Figure 5.27.

For a given diagnosis at a particular location there can be several appropriate treatment options. The treatment options for a given diagnosis-location pair is constructed into a list in the order of the most suitable first.

A surgical treatment terminates the diagnosis for that location (this is given by the

event type	case	value
Treatment	Location	| any of the | 13 nodes | in the | arterial tree | (Figure 5.10)
	outcome	| patent | occluded
	material_used	| prosthetic | vein ...

Figure 5.27: The case frame of the surgical event other than those of type bypass and reconstruction

rules `T3.1` and `T3.2` of Figure 5.14) even though the evaluation of the performance of the surgery takes place only in the subsequent interview and investigation events which is called the followup stage. Due to the same reason an event of type surgical treatment does not initiate any base relations.

5.4.2 Non-surgical Treatments

The non-surgical treatments relevant to the vascular surgery domain are blood transfusion and drug treatment. These two are discussed below.

Blood Transfusion:

The treatment of blood transfusion does not have any case instances pertaining to it except the general cases described in Section 5.1. The treatment is followed by a blood test which determines the outcome of the transfusion.

Drug Treatment:

The case frame of an event of type drug treatment is shown in Figure 5.28.

event type	case	value
drug_treatment	drug_given	| thyroxine
	dosage	| Value:Unit e.g. 4:mmol
	rely_time	| number of | days

Figure 5.28: The case frame of the event of type drug treatment

The case `rely_time` specifies how long the drug should continue, after which the condition of the patient is re-evaluated (by investigation or at an interview). The event of a drug treatment initiates the base relation `on_drugs(Drug)`. This will have a life time equal to that of the `rely_time` of the event. The relation will autonomously terminate after the specified `rely_time`. We illustrate the autonomous termination of relations after the clinician specified `rely_time`, by a case study (Case Study 2) in Chapter 7, Section 7.2.

We do not have a sophisticated system to input drug treatment. At present we only deal with the drug thyroxine. Event Calculus provides the raw basics which can deal with any type of event. Although the basic system is present if we were to deal fully with drug treatment further tools have to be developed.

5.5 Conclusion

We have described in this chapter a representation for the domain dependent patient information in a framework suitable for Event Calculus. For each event that can occur in the domain, we have given case frames and identified the time relevant relations, the *base relations*. Initiating and terminating rules are written for all base relations using the case instances. A case frame for a particular event represents all the time relative patient information relative to that event. Thus the case frames also support

completeness of data collection.

Event Calculus provides a flexible framework for assimilation of information contained in the patient record. Time dependent properties of the domain are identified as base relations. These base relations provide relevant information needed for decision support. In the next chapter we describe how the base relations can be used to generate system recommendations. It is important to remember, however, that the user can override the system recommendations which hold at any given time.

6 Decision Support for Patient Management

Up to this point in the thesis we have been concerned primarily with the development of the research workbench. We now turn to the application of this workbench to decision support in the vascular surgery domain. Decision support systems can be generally categorised into: those which assist the clinician in determining *what is true* about the patient (that is assist in deciding the correct diagnosis); those which assist the clinician in determining *what to do* for the patient (that is what investigations to be done, what treatment plan to be followed). Our system falls into the latter category. We have left the diagnostic problem to the clinician and have concerned ourselves only with providing support in deciding *what to do*.

The patient is seen by the clinician and other personnel in health care at variable intervals of time during his/her progress through the clinical pathway. Each such a meeting between the patient and the health expert is called an interview. The interval between interviews may vary from a number of hours to a number of weeks. If the patient is acutely ill, the frequency of reviewing the patients' conditions may be high and at other times, where the patient is seen for routine follow-up, the frequency may be low .

At each interview the patient's condition is evaluated and observations are recorded (at present) in the paper-based patient notes. It is also usual practice to indicate in the patient notes what actions are to be carried out next. For example, any tests that ought to be done, changes in drug dose or changes in the drug itself, what treatment is appropriate, or what progress of symptoms in the patient should be observed for a certain period of time before any further action is taken. We call these recommendations.

We have constructed on top of the knowledge base containing the domain know-

ledge described in Chapter 5 a set of system generated recommendations which apply at any given time. These recommendations are based in a general sense on medical expertise. Broadly speaking this consists of non-controversial medical knowledge, a notion of good clinical practice and rules of thumb used in medical inferencing. In the rest of this chapter we show how through these recommendations we provide the clinician with the decision support necessary for better patient management.

6.1 System Recommendations

We express system recommendations mainly as ramifications through the `holds_at` relation (described in Chapter 4) and sometimes as base relations initiated and terminated by events. The base relations have a direct bearing to the events that happen in the domain. The ramifications do not directly depend on the events but are expressed in terms of base relations and/or other ramifications. The main purpose of this section is to illustrate how and when these recommendations may be of use to the clinician.

There are conflicting advantages in representing a property of the domain as a ramification. Ramifications provide a very simple and elegant way of representing knowledge. But if we were to strictly adhere to a policy of representing recommendations only by ramifications we have to be prepared to sacrifice other aspects of knowledge representation such as simplicity. This is clear in the case of recommending the next task to be done (discussed in Section 6.1.3). Whether we represent a particular property by a ramification or by a base relation is a practical issue. We take a pragmatic approach to this problem and when the situation warrants use both methods to represent a property.

6.1.1 Recommendation of Tests

From the symptoms and signs observed at the interview, the clinician makes an initial diagnosis as to what may cause them, which we call an initial diagnosis. Each initial diagnosis is associated with one or more tests which may help to confirm or eliminate it.

For example, if it was supposed that the patient suffers from clinical anaemia then it is recommended that a blood test should be carried out to investigate this

supposition. This is represented by the ramification (R1) in Figure 6.1. If an initial

```
holds_at(recommended_test(blood_test), T) :-
        holds_at(diagnosis(clinical_anaemia), T).          (R1)
```

Figure 6.1: The ramification for the recommendation of a blood_test in the case of clinical anaemia

```
holds_at(recommended_test(ultrasound), T) :-
        holds_at(diagnosis(dilatory), T).                  (R2)
```

Figure 6.2: The ramification for the recommendation of ultrasound in the case of dilatory disease

diagnosis of dilatory disease was made at the interview then an ultrasound test is recommended for further investigation. This is represented by (R2) as shown in Figure 6.2.

If more than one test is recommended at the same time then usually all tests will be carried out eventually. The order in which they are carried out is immaterial unless specifically stated. The ability to be neutral about the ordering of some events is a feature of the Event Calculus since this formalism is based on a partial ordering of events.

6.1.2 Recommendation of Treatments

From the results of the tests carried out to confirm the initial diagnosis a final diagnosis which is a specialisation of the initial diagnosis is made. Each final diagnosis is associated with one or more recommended treatments.

For example, if a final diagnosis of anaemia is made (by the results of a blood test carried out), then a treatment of blood transfusion is recommended. This is represented by the ramification (R3) given in Figure 6.3.

If a final diagnosis (of vascular nature) `Diagnosis` is made at the vascular location, `Location` then a recommendation of the treatment `Treatment` is made,

```
holds_at(recommended_treatment(blood_transfusion,
                                for(anaemia)), T):-
        holds_at(diagnosis(anaemia), T).                    (R3)
```

Figure 6.3: The ramification for recommendation of the treatment blood transfusion in the case of anaemia

where `Treatment` is the first in the list of treatment options suitable for `Diagnosis` at `Location`. This is represented by `(R4)` in Figure 6.4.

```
holds_at(recommended_treatment(Treatment,
                    for(Diagnosis), Location), T) :-
        holds_at(treatment_options_for( Diagnosis,
                    Location, [Treatment|_], T).            (R4)
```

Figure 6.4: The ramification for recommending a treatment for a final diagnoses of vascular nature

Treatments are ranked in the order of their suitability. Only one treatment option is selected at any given time. Alternative appropriate treatments are given by the relation `recommended_alternative_treatments` where a list of treatments are given in the order of their suitability as mentioned above. This is represented by `(R5)` in Figure 6.5. One of the treatments in this list may become active in case the recommended treatment fails or the health expert disagrees with the system recommendation.

Unlike in the case of recommended tests, only one treatment will be recommended at any given instant of time.

6.1.3 Recommendation of the Next Task to be Done

It is usual practice for the clinician to write down on the paper-based patient notes what action to be taken next. The system can propose the action to be carried out next with the use of recommendations. This is modelled by two relations:

`awaiting(Task)` represented as a ramification

```
holds_at(recommended_alternative_treatments([Second|Treatmentlist],
                    for(Diagnosis), Location), T) :-
        holds_at(treatment_options_for( Diagnosis,
                    Location, [First,Second|Treatmentlist], T).
                                                                    (R5)
```

Figure 6.5: The rule for recommending alternative treatments for final diagnoses of vascular nature

`next_task(Task)` represented as a base relation

Although any event can occur at any point in the clinical pathway there is an ordered sequence in which events occur. The order in which events usually occur up to the treatment stage changes in the followup stage. This dichotomy of behaviour entails that we represent the next task to be done by two different ways.

Up to the treatment plan what action takes place next depends on the results of the most recently occurring event. For example: what test/s should be performed in the investigation stage that follows the initial interview, depends on the diagnoses made at the interview; clinicians decide on the treatment plan based on the results of the investigation. Therefore the next task to be done in this case is modelled by a ramification. But in the followup stage what action should take place next depends on the treatment performed. For example: the treatment endarterectomy is usually followed by the test duplex scan, which in turn if necessary is followed by an arteriogram; the treatment, endarterectomy with angioplasty, is followed by a doppler test and then if necessary by a duplex scan; treatments of type reconstruction are usually followed by an interview and then if necessary by an arteriogram. Therefore in the followup stage the next action to be taken is more suitably represented by a base relation. Thus there is a need to represent this phenomenon both by a ramification and a base relation.

There can be more than one task recommended at any given time (for example, `recommended_test, recommended_treatment`). If not explicitly indicated any task among those recommended may be carried out first. For example, it was

diagnosed at the initial interview that the patient is suffering from both clinical anaemia and dilatory disease. In this case a blood test will be recommended to investigate the anaemic condition and an ultrasound will be recommended for the investigation of the dilatory condition (indicated by the relations `recommended_test(bloodtest)` and `recommended_test(ultrasound)` respectively). Either of these tests may be carried out first. If the ultrasound was carried out first, and the results indicate that the diseased arterial location needs to be treated, then a suitable treatment will be recommended. At this point in time, a recommendation for the treatment of the dilatory condition, and a recommendation for the blood test will hold true. It is good clinical practice to carry out all investigations before any treatment is undertaken. Thus the recommended test takes precedence over the recommended treatment. This is implemented by the relation `awaiting(investigation)` holding true at that instant. In this section we consider rules which impose some ordering structure on the recommendations.

An example of such an ordering is provided by an anaemic patient. If blood transfusion is recommended as a treatment for anaemia then the blood transfusion is done before before any other event. This is represented by the ramification (R6) given in Figure 6.6. This rule will cause the clinician to be informed that the patient is

```
holds_at(awaiting(treatment), T):-
        holds_at(recommended_treatment(blood_transfusion,
                                       for(anaemia)), T).
                                                              (R6)
```

Figure 6.6: The ramification denoting that blood transfusion whenever recommended should be the first event to be carried out

awaiting treatment and thus infer that of all the recommendations holding (e.g. other tests) it is the treatment, that is the blood transfusion, which should be carried out next.

The above example is really the exception. It is better medical practice (as explained later in Section 6.2.2) to complete all relevant investigations before carrying

out any treatment plans. This is represented in two ways. We represent this feature implicitly by saying that if there is any test recommended the next task to be done is the investigation. The same is represented explicitly by saying that any recommended treatment should be delayed until all investigations are carried out. The former is given by the rule (R7) in Figure 6.7 and the latter is represented by the rule (R8) given in Figure 6.8. As mentioned before the only exception for this is the case where a blood transfusion has been recommended as a treatment for anaemia which is represented by the ramification (R6) given in Figure 6.6. The constraint `not holds_at(recommended_treatment(blood_transfusion, for(anaemia)), T)` in (R7) accommodates the case represented in Figure 6.6.

```
holds_at(awaiting(investigation), T):-
        holds_at(recommended_test(Test), T),
        not holds_at(recommended_treatment(blood_transfusion
                                for(anaemia)), T).          (R7)
```

Figure 6.7: The ramification implying that all investigations should be carried out first (except in the presence of a recommendation for blood transfusion)

```
holds_at(awaiting(treatment), T):-
        holds_at(recommended_treatment(_, _, _), T),
        not holds_at(awaiting(investigation), T).          (R8)
```

Figure 6.8: The ramification implying that treatment should be delayed until all investigations are completed

Recommendation of the next step to be carried out in the clinical pathway under certain circumstances can be more easily generated as a base relation rather than as a ramification. This dichotomy of representation, base relation or ramification, is an aspect of the so called ramification problem (Kowalski, 1992). The need for this diversity of representation was discussed at the beginning of this section. We can further illustrate our approach by an example from the rules governing follow up of a

surgical treatment.

The outcome of a surgical treatment is determined by carrying out a specific test. Which specific test is to be carried out is based on the treatment performed. Thus in the case of a surgical treatment the recommended task is initiated by the event of the treatment and is indicated by the base relation,

```
next_task(Task)
```

The surgical treatment, endarterectomy with angioplsty, is usually followed by a doppler test and thus initiates the relation `next_task(doppler)` as given in Figure 6.9.

```
initiates(E, next_task(doppler)):-
        inst(E, endarterectomy with angioplsty).
```

Figure 6.9: The base relation denoting that an event of endarterectomy with angioplsty should be followed by a doppler probe. Note: the operator 'with' has been defined for readability

The treatment Trans Luminal Dilation (commonly known as TLD) initiates the relation `next_task(duplex)` thus indicating that it should be followed by the test duplex scan. This is represented in Figure 6.10.

```
initiates(E, next_task(duplex)):-
        inst(E, trans_luminal_dilation).
```

Figure 6.10: The base relation denoting that an event of TLD should be followed by a duplex scan

In the case of the non-surgical treatment, blood transfusion, the outcome can be confirmed only by carrying out a blood test. Thus an event of a blood transfusion is usually followed by a blood test. The event blood transfusion thus initiates the relation, `next_task(blood_test)`. This is represented by the rules given in Figure 6.11.

Although in most cases a test is a direct consequence of a diagnosis, in the follow-up stage the test that follows the treatment depends on the treatment itself. This is

```
initiates(E, next_task(blood_test):-
        inst(E, blood_transfusion).                                    (B1)
```

Figure 6.11: The initiation of the base relation recommending a blood test by the event of a blood transfusion

mainly because the purpose of this test is to evaluate the performance of the treatment done, and to decide on further action to be taken if necessary. We believe this may be a good heuristic to apply in advance of modelling the domain, as a guide to knowledge representation.

In this section we have shown how certain properties need to be modelled by a ramification as well as a base relation. If we have been over persuasive the reader may wonder whether there is a necessity for the ramification `awaiting(Task)`. Our position is: the circumstances under which an event is recommended may directly depend on another event, or it may depend on a base relation/ramification. In the first case the recommendation of the next task is better represented by a base relation. In the latter case it is better represented by a ramification.

The difference between the uses of the two types of relation, as can be seen, is that the ramification `awaiting(Task)` denotes the next stage at which the patient should be in the clinical pathway, that is Task is interview, investigation or treatment, etc., whereas, the base relation `next_task(Task)` specifically denotes the specific (instantiated) Task to be carried out next.

6.2 Modelling medical practice

Medical practice is medical decision making (Shortliffe and Perreault, 1990). Modelling medical practice involves modelling the decision making process. Deciding what questions are to be asked from the patient at the interview, what tests need to be carried out to confirm the suppositions made regarding the condition of the patient, what treatment procedures should be taken to relieve the patient's current condition, etc. are decisions which a clinician has to make during the progress of a patient along the clinical pathway. In this decision making process not only s/he needs accurate data

and relevant knowledge but, mostly s/he needs good problem solving skills which the experts of the field possess. How we support the clinician in the challenging task of decision making, by providing rules modelling various medical decision procedures written in the form of ramifications, is described in this section.

6.2.1 Interference of Diagnoses

A patient may suffer from multiple diseases simultaneously (as discussed earlier in Chapter 5, Section 5.3.1.5) in which case there can be interference between two different diseases. In vascular diseases when such interference occurs usually, though not always, one problem takes higher priority over the other. We represent such cases by the relation,

```
higher_priority(D, over(D1))
```

which denotes that disease `D` takes a priority over disease `D1`. The function symbol `over` is only for better understanding of the relation and itself does not contribute anything to the representation.

For example, if the patient suffers from anaemia and any kind of occlusive arterial disease then, the anaemic condition takes priority and is treated first. This is good clinical practice since, it may be the case that the symptoms thought to have been caused by occlusive disease may have actually be due to the anaemic condition. This is represented by the ramification (R8) given in Figure 6.12.

```
holds_at(higher_priority(anaemia, over(D)), T) :-
        holds_at(diagnosis(anaemia), T),
        holds_at(diagnosis(D, Location), T),
        specialisation(D, occlusive_arterial).                    (R8)
```

Figure 6.12: The rule depicting the priority of anaemia over diseases of occlusive nature

When the occlusive arterial condition and the dilatory condition occur at the same time in a patient, the dilatory condition takes priority over the occlusiveness due to its high risk nature unless it is the case that the dilatory condition is very mild so that

there is no possibility of a clinical event happening. This is represented by (R9) in Figure 6.13.

```
holds_at(higher_priority(localised_dilatory,
        over(localised_atherogenic_arterial)), T) :-
        holds_at(diagnosis(localised_atherogenic_arterial,
                                        Location), T),
        holds_at(causes_risk_of_event(localised_dilatory,
                                        Location1), T).
                                                    (R9)
```

Figure 6.13: The ramification indicating the priority of localised dilatory condition over the localised atherogenic arterial condition

The vascular condition `localised_dilatory` is thought to be causing a risk of a clinical event happening if the size of the dilation (that is the size of the *aneurysm*) is greater than the *critical size*. The critical size varies according to the location. The ramification for the relation `causes_risk_of_event` is shown in Figure 6.14.

```
holds_at(causes_risk_of_event(localised_dilatory, Location), T):-
        holds_at(size(aneurysm, Location, Diameter), T)
        critical_size(Location, CriticalSize),
        Diameter > CriticalSize.
```

Figure 6.14: The ramification indicating when the dilatory condition can cause a risk of an event

The occlusive condition itself can be of different types (as shown in Figure 5.11 of Chapter 5). When the occlusive condition thrombosis is present with any other occlusive condition, the thrombosis condition takes priority over the others. This is represented by the ramification (R10) given in Figure 6.15. In (R10) both `Location` and `Location1` may represent the same vascular site.

```
holds_at(higher_priority(thrombosis, over(D)), T):-
        holds_at(diagnosis(thrombosis, Location)), T),
        holds_at(diagnosis(D, Location1)), T),
        not D=thrombosis,
        specialisation(D, occlusive_arterial).                    (R10)
```

Figure 6.15: The ramification indicating the priority of thrombosis condition over other occlusive conditions

If a patient suffers from hypothyroid then the level of thyroxine in blood should be controlled before any surgical treatment should be done. In such a case if it is known that the thyroxine level in blood is lower than the normal level (6mmol/litre) then, the treatment is postponed and the patient is treated with thyroxine until its level in blood reaches the normal level. This is represented by the ramification given (R11) in Figure 6.16. In this case the recommendations will not only help better patient

```
holds_at(higher_priority(hypothyroid, over(D)), T):-
        Holds_at(diagnosis(hypothyroid), T),
        holds_at(diagnosis(D, Location)), T),
        holds_at(below_required_level(thyroxine), T).             (R11)
```

Figure 6.16: The ramification indicating the priority of hypothyroid

management but also resource management by perhaps avoiding unnecessary delays in treatment.

6.2.2 Better Clinical Procedures

Unlike other specific tests the arteriogram carries a certain degree of risk to the patient. The risk factor becomes high if the patient also suffers from anaemic conditions. Therefore, the arteriogram should only be performed provided no kind of clinical anaemia is present. This is represented by (R12) in Figure 6.17.

It is good clinical practice (if time allows) that the treatments be delayed until all

```
holds_at(recommended_test(arteriogram), T):-
        holds_at(diagnosis(occlusive_arterial), T),
        not ( holds_at(diagnosis(D), T),
            kind_of(D, clinical_anaemia)).                    (R12)
```

Figure 6.17: The rule depicting that arteriogram should only be done in the absence of anaemic conditions

the investigations have been completed. This is implicitly implied in the rule given in Figure 6.7. The relation `awaiting(Task)` denotes the next task to be carried out, as described in Section 6.1.3 earlier in this chapter.

The system recommendations need not be followed necessarily. At any given time the clinician can take his/her autonomous decision over riding the system suggestions. Thus the decision making control is given to the clinician. This is illustrated with a case study in Chapter 7 Section 7.2.

6.3 Autonomous Change

Another aspect of the patient management system is that, as well as proposing recommended actions, it can also propose when these actions should be carried out.

Typically knowledge about a patient's condition can only be relied upon for a certain length of time, after which the patient's condition should be re-assessed. This behaviour is modelled by introducing a simple form of autonomous change into Event Calculus using the relation `too_long_after`. In this context we mean by autonomous that the change is not caused by known events. This was described in Chapter 4 Section 4.2.2.

```
too_long_after(T1, U, T) :-
        separation(T1, T, I),
        life_time( U, I'),
        I' < I.
```

As represented in Figure 4.16 (of Chapter 4, Section 4.2.2) the relation `too_long_after`

is given above. In the above relation the life-time of a relation is represented as a property independent of time. But, as we studied the nature of the domain it was apparent that our previous assumption was incorrect and, the life-time of a relation may vary with the context and therefore with time. For example, the life-time of the relation `pulse_count(Count)` is not the same throughout a patient's life, but varies according to the patient's condition at any given time. By life-time of the `pulse_count(Count)` we mean the time interval (starting from the time at which the pulse was taken) in which the results can be relied on (assumed to be true). Life-time can also be looked upon as how often the test needs to be repeated.

The life-time is either calculated (by pre-defined rules in the knowledge base) or it is allowed to be a clinician determined piece of data. In the latter case it is made to be an event description represented by `rely_time`. For example, if an event of a blood test is represented as,

```
inst(e, blood_test).

etime(e, 1988/3/12).

result(e, thyroxine(8:mmol/litre)).

rely_time(e, 5).
```

then this implies that the results of the event `e` can be relied on only for five days. Thus the relation/s initiated by `e` will have a life-time of five days and the test will be repeated after that duration. The calculated life-time is considered as the default value which is replaced by the clinician determined value whenever the latter is present.

The `rely_time` is related to the initiating event. Also as explained in Chapter 4 Section 4.2.2, the life-time of the relation, `size(aneurysm, Location, Size)` at any given time depends on the size of the aneurysm and the rate of change of the size at that time. To take into account this time dependence we modify the above given rule to include the initiating event of the relation. We choose the event rather than only the time of the event so that other contextual information can be invoked. The modified rule is represented in Figure 6.18. The relation `life_time(E, U, I')` indicates that the life time of the relation `U` initiated by the event `E` is I'.

Any event can be associated with a reliability period `rely_time` by which a clinician can override any system generated life-time (considered as a default) of

```
too_long_after(E, U, T):-
        separation(E, T, I),
        life_time(E, U, I'),
        I' < I.
```

Figure 6.18: The relation `too_long_after` after modification

an associated relation. For example, the life-time of the relation `size(aneurysm, Location, Size)` which is generated by the system can be overridden by the clinician specifying a `rely_time` for the event ultrasound which initiates the relation. This is shown below.

```
inst(e, ultrasound).
etime(e, 1989/2/12).
result(e, aorta, 4.5).
rely_time(e, 21).
```

We can represent the life-time of a relation as given in Figure 6.19 where E is the event initiating the relation U. If there is no `rely_time` specified for an event then we can

```
life_time(E, U, I):-
        rely_time(E, I).
```

Figure 6.19: The relation representing the life time of a relation

explicitly define the life-time of each relation. For example, the life-time of the relation `size(aneurysm, Location, Size)` is represented as given in Figure 6.20.

As a consequence of this modification including the event dependent life-time, we have been able to implement in a straightforward way that, when the life-time of a relation expires, the system will recommend that the event which initiated that relation should be carried out. For example, if `localised_dilatory` condition was confirmed by the results of an ultrasound which notifies that the size of the aneurysm is

```
life_time(E, size(aneurysm, Location, Size), 183):-
        rate_of_change(E, aneurysm, Location, 0),
        Size < 4.

life_time(E, size(aneurysm, Location, Size), 91):-
        rate_of_change(E, aneurysm, Location, 0),
        Size ≤ 4.

life_time(E, size(aneurysm, Location, Size), 91):-
        rate_of_change(E, aneurysm, Location, Y),
        Y > 0.3.

life_time(E, size(aneurysm, Location, Size), Days):-
        rate_of_change(E, aneurysm, Location, Change),
        Change < 0.3,
        etime(E, Y/M/D),
        days_in(Y, Days).
```

Figure 6.20: The relations representing the life time of `size(aneurysm, at(Location), Size)`

3.5 then the rate of change of the aneurysm will be zero. If so, according to the first rule in Figure 6.20 the relation will have a life time of 183 days. After that time the diagnosis will still hold true, but the relation `size(aneurysm, at(Location), Size)` will no longer hold true and thus an ultrasound will be automatically recommended. This is represented in Figure 6.21.

As illustrated in this section, as a consequence of introducing autonomous change, notification of due and overdue events (or actions) can be generated. This provides a further means of controlling consistency in the patient management procedures and in turn in the quality of care.

```
holds_at(recommended_test(ultrasound), T):-
        holds_at(diagnosis(localised_dilatory, at(Location)), T),
        not holds_at(size(aneurysm, Location, Size), T).
```

Figure 6.21: The ramification representing the autonomous recommendation of the test ultrasound

6.4 Conclusion

We have shown in this chapter how a system of recommendations has been built on top of the knowledge base. This provides a support tool for patient management which provides recommendations and supports better clinical procedures.

We have described how ramifications provide a simple knowledge representation form in most cases, but can also prove to be cumbersome in others. With examples we have shown why a pragmatic solution to this problem is necessary. The ramifications also provide a means of accessing relevant information stored in the knowledge base.

We have shown how the system recommendations generate the most suitable treatment and also a list of alternative treatments that could be undertaken in the case of failure of the former. It is at this point, that is at the point of deciding the best treatment procedure, that the risk/benefit analysis should be undertaken. The module of risk/benefit analysis has been omitted in our implementation but it is clear that such a module could be included in a independent and straightforward way. All the information concerning a particular patient and other relevant aspects of the context are accessible from our system and would allow risk/benefit analysis to be specialised to the particular case. The result would be a revised and annotated ranking of the treatment options. Another reason for omitting risk/benefit analysis at this stage is that it is a sizable research area. Although the medical experts use such inference mechanisms when taking a treatment decision, still no hard and fast rules have been created to mechanise such procedures. Currently there is much ongoing research in arriving at a suitable method for such a module (Crane, 1988).

With examples we have illustrated the ease of building a comprehensive model of medical practice in the domain of vascular surgery. Whatever the application area

is, medical practice changes continuously and needs to be extended constantly. Event Calculus provides a flexible framework for such a changing domain of application.

In the next chapter we describe the implemented system: the user interface; the query facility; and the answer justification capability. By example case studies we illustrate how the implemented system successfully demonstrates the decision support capabilities we have described in this chapter.

7 Prototype System: Structure and Case Studies

In Chapter 2 Section 2.1 we described the clinical pathway. In Chapter 4 we have shown how a simplified form of the Event Calculus, based on the more general framework introduced by Kowalski and Sergot, can provide a temporal framework for describing this pathway. The domain model of the arterial side of vascular surgery was represented in a form suitable for Event Calculus in Chapter 5. How a set of ramifications built on top of the knowledge base can be used to support better patient management was shown in Chapter 6.

This chapter describes how the system described so far has been implemented in a prototype system. The structure of the prototype system is described in Section 7.1 and the success of the implementation is demonstrated using example case studies and sample input and output in Section 7.2.

7.1 The System Description

7.1.1 System Implementation

Prolog is a natural language for building expert systems. The first to argue explicitly for the use of Prolog to build expert systems were Clark and McCabe (Clark and McCabe, 1982). In our case the choice of Prolog is indicated even more strongly because our formalisation of the whole problem has been in augmented Horn clause logic which has a direct mapping into Prolog.

Our system has been implemented on a Macintosh SE, using the Macintosh Operating System 6.0. The program was initially written in MacPROLOG version 3.0. When the version was updated desirable features of newer versions were introduced. The current version of our system is in MacPROLOG version 3.5 under Operating

System 6.

The prototype system consists of a patient data base, knowledge base consisting of medical knowledge, the general rules of Event Calculus, a meta interpreter, and a user interface.

The data base consists of patient data in the form of a collection of event descriptions. Each patient has his/her own unique data base.

When the user selects a patient by choosing the patient's name (see Appendix B.2: Selecting a Patient), patient data of that patient will be loaded into the system.

Each patient has two data files. One is a Prolog source file which contains the event descriptions as Prolog clauses. The other is a text file containing all information gathered during interviews. Arguably the best format for displaying data in our system is as natural language text, since it is easy to understand without any need to have prior knowledge of the underlying format of data storage. Storing information in a text file will naturally duplicate some of the data which is already in the Prolog source file and will increase storage cost to some extent. But, it also facilitates fast access to information and removes the additional processing cost involved in transforming the Prolog clauses to text format before being displayed. Note that we do not provide natural language processing capabilities.

At present, we store as event descriptions, in Prolog clauses, only data which are relevant for time dependence and which are required for the inference mechanism. All other information, such as the social history of the patient, specialists' comments, information required for risk analysis, since it is not required for logical processing is stored in text files.

Data entry is through an interactive process, using menus, dialogue formats and free text. MacPROLOG Graphic Description Language (GDL) (Johns and Spenser, 1989) is used in creating some of the input forms. We make significant use of the MacPROLOG's menu management system which supports the use of popup and hierarchical menus. Hierarchical menus are created by including sub-menus in a menu on the menubar (these sub-menus in turn may have sub-menus). Popup menus do not appear as part of the menu bar. They may be "popped up" at any point on the screen associated with it. We employ menu handling in a number of different ways. The simplest allows a single choice from a single menu. A rapid selection of

combination of choices can be made from a hierarchically arranged pull-down menu (In Appendix B.3 Figure B.5 illustrates the appearance of a hierarchical menu, and Figure B.6 a popup menu).

We make extensive use of MacPROLOG dialogue windows both to accept input and to display information. Both the predefined dialogues and the primitives available to create our own specialised dialogues (with buttons, text fields, edit fields and scrolling menus) are used. Appendix B.2 Figures B.2– B.4, Appendix B.3 Figures B.7–B.10, and Appendix B.4 Figure B.11 illustrate some of the dialogue windows used in our system.

As new events occur in the domain, relevant information about them is entered into the system as event descriptions. The new status of the patient is given by querying as to what relations are holding true after the entry of data by choosing the option *Query* from the main menu. The user can also query the past status of the patient at any instant in the past, in the same manner.

Patient information can be displayed by choosing the option *Display Data* from the menu. The option has the sub-menus *Event Information* and *Patient Information*. The option *Event Information* displays a dialogue window where the user is allowed to select the events to be displayed, and the format to be displayed. Events can be displayed in a brief format giving just the event names, types and the event times, or in a detailed format where all selected events will be displayed in detail (as prolog relations). The option *Patient Information* displays all information gathered at past interviews in text format.

At the present state of the prototype system only one patient can be processed at any given time. This could be easily remedied by adding an extra argument to the Prolog terms containg the patient information, whereby each patient could be uniquely identified.

7.1.2 Representation of Time

Allen (Allen, 1991) has states in his review of methods of representing time,

> 'A good representation of time for instantaneous events, if it is possible, is using an absolute dating system.'

We have found that such a representation, which supports reasoning about time points, is practical for our system. This is basically because, at some suitable level of granularity, our events can be considered to be instantaneous.

Event times are mapped to calendar dates in the form Year/Month/Day. We do not go to the fine grains of minutes and seconds nor do we deal with the management of time granularities, that is specification of times at different levels of abstraction (Evans, 1989; Evans, 1990).

We are aware that our coarse granularity of time may need to be extended for the implementation of a fully operational system. But, to achieve our goal, demonstrating the suitability of Event Calculus as the back bone of temporal reasoning to a historical knowledge base in a realistic context, and for our prototype which is intended as a research workbench, this representation of time is quite sufficient. A more sophisticated handling of time could be introduced without major changes to the structure of the present system.

7.1.3 User Interface

The user interface is mainly menu driven. First the user should select the patient. If the patient information is already in the data base then, the name of the patient is selected from a menu (Appendix B.1 Figure B.1, Appendix B.2 Figures B.2 and B.3). If a new patient is to be entered the patient's family name and initials are asked to be input (Appendix B.2 Figure B.4). The name input is used to create the two patient data files, the Prolog source file and the text file (described in Section 7.1).

The user can chose one of, *Data Input*, *Display Data*, or *Query* from the main menu which allows the user to input data, view existing patient data, or query the knowledge base respectively (Appendix B.1, Figure B.1). The menu option *How to use the menu* of the main menu is essentially a help facility which describes the functionality of each menu option provided (this option is further described in Appendix B.5).

7.1.4 Query Facility

Queries are a means of retrieving information from a logic program. The user can query as to:

1. whether a particular relation is true at a specified time,

2. which relations are true at a specified time,

3. why a particular relation is true at a specified time, or

4. why a particular relation is not true at a specified time.

For the first type of query, if the goal could be deduced from the knowledge base, then the answer *yes* is given (see Appendix C.1, Figure C.3). If the goal cannot be deduced then the answer *no* (see Appendix C.1, Figure C.4) is given. For the second type of query, all solutions to the goal are found and displayed. The other two queries are discussed in the next section.

We should emphasize that the prototype we have developed is intended to be a *research workbench* and not a system to be tested in a clinical situation. For this reason the interface/displays have been left in a 'raw' form, for example the `holds_at` predicate has not been stripped out, so that researchers can use and modify the system easily.

7.1.5 Answer Justification

Answer justification refers to the ability of an expert system to explain why certain conclusions have been arrived at. There is a great deal of interest today in developing new tools for answer justification (Wolstenholme, 1992a; Wolstenholme, 1992b; Reggia, Perricone, Nau and Peng, 1985a; Reggia, Perricone, Nau and Peng, 1985b). Such explanations are considered as an essential element of any decision support system. The user may desire to know why a recommendation is given, or if the recommendation differs with his/her personal view, then s/he may want to know why a particular recommendation was not given. We have provided the 'why' and 'whynot' justification to system conclusions. Because of the Event Calculus formalism, we are able to produce in addition to the rule traces, the sequence of related events which lead up to a particular conclusion. As well as being informative this sequence of events provides access to the detailed information stored in the associated case frames.

Using meta-interpreters[1] as a basis for explanation facilities in expert systems is discussed by Sterling and Shapiro (Sterling and Shapiro, 1986). Writing a meta-

[1] A *meta-interpreter* for a language is an interpreter for the language written in the language itself. The concept of a meta-interpreter is due to Sussman and Steele (Sterling and Shapiro, 1986) who were the first to propose using the ability of the language to specify itself as a fundamental criteria for language design.

program is easy in Prolog since both the data and the program are the same, i.e. Prolog terms.

The response to *why* queries is produced by applying the techniques for generating explanations in logic data bases. In the framework of logic-based knowledge representation formalisms, derivation trees built by the query evaluation process form the raw material from which explanations are generated (Brauffaerts and Henin, 1988). We have adopted a meta-interpreter given by Sterling and Shapiro (Sterling and Shapiro, 1986) with slight modifications. This program constructs a proof tree containing the rules fired when a goal is proved to be true. This proof tree is used to explain how that particular solution was found. Explanation is given in natural language with instantiated Prolog predicates. The basic program is as shown in Figure 7.1.

```
why(Goal) :-
        solve(Goal, ProofTree),
        interpret(ProofTree).

solve(true, true):-!.
solve((A,B),(ProofA, ProofB)):-
        solve(A, ProofA),
        solve(B, ProofB).
solve(A, (A :- Proof)):-
        clause(A,B),
        solve(B, Proof).

interpret(ProofTree) displays the ProofTree in a suitable
manner.
```

Figure 7.1: The basic structure of the meta Interpreter

The interpreter presents one branch of the proof tree at a time (on request from the user). For example consider the following program:

```
g :- a, b, c.

a :- d, e, f.

b :- x, y, z.

d :- k, m.

c, e, f, x, y, z, k, and m are given facts.
```

If the explanation for g is asked by posing the query `why(g)`, then the explanation "because `a`, `b`, and `c` are true" is given. After explaining the definition for `g`, if the user asks for further explanations, then proof tree for `a` will be displayed depth wise. After the full explanation for `a` is given then, the explanation for `b` will be displayed and so on. The user does not have any control over the method of display. But posing different queries, the user is able to access the knowledge embedded. This is further illustrated by example dialogues in Appendix D.1. The explanation for the negation by failure (that is in the case of the query, `why(not g)` is catered in a manner similar to the whynot explanation discussed below.

The *whynot* explanations justify failures. We have extended the basic program given in Figure 7.1 so that in the case of failure of a goal, a proof tree is built exhaustively, for all the definitions in the knowledge base for that goal. For each definition, the proof tree is built as far as the failed clause is met. The failed clause is asserted as a node, with the clause as its head and the value *false* as the body. The main reason for asking *whynot* explanations is that the user does not agree with a system recommendation or because the user expected the failed goal to succeed. The answer to the query `whynot` is displayed starting from the first explanation in the knowledge base (for the failed goal). On request from the user the explanations for the successive definitions are presented. For each definition the proof tree is displayed starting from the first clause up to the one which failed (that is the node with *false* as its body). For example, let us suppose that the predicate `p` has three definitions in the knowledge base,

```
p :- q1, q2.

p :- r1, r2, r3.

p :- s1.
```

Say for example the clauses `q1, r1` and `r2` evaluate to true. Then, to the query `whynot(p)`, first the reason for failure of the first definition will be explained, that is "because `q1` is true, but `q2` is not true". If more explanations are asked for, then why the second definition, `p :- r1, r2, r3`, failed will be explained, in the same manner.

There is much debate as to the best way a `whynot` explanation should be given. Wolstenholme (Wolstenholme, 1992a) suggests better ways of displaying failures in logic-based advice systems. Brauffaerts and Henin (Brauffaerts and Henin, 1988) present a meta-interpreter which produces proof trees justifying both success and failure of query evaluation.

7.2 Case Studies

In Chapter 6 we have described how our system assists through recommendations the decision making process of managing patients in vascular surgery. These recommendations can be overridden by the clinician's autonomous decision at any time. Event Calculus provides a flexible basis for the system where any event can be accepted at any time, and this gives the flexibility required for the clinical domain. The basic Event Calculus structure provides a means of accessing patient oriented historical data. By means of the `holds_at` predicate the state of the world at any specific instant can be reconstructed without having to explicitly store it. In this section, by means of case studies, we illustrate how the prototype system successfully caters for all these functions.

The outputs shown in each case study is the result of selecting the *Query* option of the main menu (see Appendix C: Query Facility), and then selecting the sub-menu option *inquires*. This displays a dialogue window from which the option *Relations holding at a given time* is chosen (Appendix C, Figure C.1). Time points at which the patient status is required is typed in and the output obtained is displayed.

7.2.1 Case Study 1: Interference between two vascular diseases

When both vascular disease conditions, occlusive and dilatory, are present simultaneously in a patient the dilatory condition usually takes priority over the occlusive

condition due to its high risk nature. If the investigations reveal that there is no risk caused by the dilatory condition, the occlusive condition is investigated. Whichever condition is of higher risk to the patient is treated first. In this case study we show how through the relation `higher_priority` the occlusive condition is suppressed by the dilatory condition when the latter is thought to be or known to be causing a greater risk to the patient.

Scenario

At the initial interview the patient complains of pain in the left leg (thigh) and rest pain on the left side. An initial diagnosis of occlusive arterial is made and an arteriogram is carried out to investigate the diagnosis. The arteriogram reveals that localised atherogenic arterial condition is present at the arterial location, left upper superficial femoral. The arteriogram also reveals that a dilated condition, not detected at the initial interview, may be present at the location left common femoral. The dilatory condition is further investigated at this stage by an ultrasound. The results of the ultrasound confirms the presence of the dilatory condition and determines that it causes a risk. The dilated condition takes a higher priority over the atherogenic condition and is treated first. At the interview following the treatment it was noted that the symptoms have reduced and the atherogenic arterial condition is observed for further developments.

Illustration

The event descriptions for the above explained scenario are shown in Figure 7.2. The relations holding true after the initial interview are shown in Figure 7.3. The relation `awaiting(investigation)` which holds after the initial interview (Figure 7.3) indicates that the next task to be carried out should be the investigation. The state of the patient after the system recommended investigation, the arteriogram, is performed is shown in Figure 7.4. As can be seen in Figure 7.4 not only the occlusive condition is confirmed by the presence of the localised atherogenic arterial condition, but it is revealed that it is also possible that a dilatory condition exists at another location. As explained in Chapter 5, Section 5.3.1.1 the arteriogram result alone is not sufficient to initiate a final diagnosis of the vascular condition, dilatory. As shown in the rule `I3` of

```
inst(e1, interview).
etime(e1, 1989/4/10).
report(e1, leg_pain, left, thigh).
report(e1, rest_pain, left).
result(e1, occlusive_arterial).

inst(e2, arteriogram).
etime(e2, 1989/4/25).
result(e2, (left, common_femoral), localised_dilatory).
result(e2, (left, upper_superficial_femoral,
                    localised_atherogenic_arterial).

inst(e3, ultrasound).
etime(e3, 1989/5/10).
result(e3, (left, common_femoral), 5.5).

inst(e4, (reconstruction:tube, material(prosthetic))).
etime(e4, 1989/5/30).
location(e4, (left, common_femoral)).
material_used(e4, prosthetic).
vessel_outcome(e4, patent).

inst(e5,interview).
etime(e5, 1989/6/5).
absent(e5, rest_pain, left).
absent(e5, leg_pain, left, thigh).
```

Figure 7.2: The event descriptions of the Case Study 1

File Edit Search Windows Fonts Eval **Program Options**

Answers to Query

Answers to the query are :

holds_at(diagnosis(occlusive_arterial), 1989/4/11)

holds_at(symptoms(rest_pain, left), 1989/4/11)

holds_at(symptoms(leg_pain, left, thigh), 1989/4/11)

holds_at(awaiting(investigation), 1989/4/11)

holds_at(recommended_test(arteriogram), 1989/4/11)

Figure 7.3: The relations holding true after the initial interview in Case Study 1

File Edit Search Windows Fonts Eval **Program Options**

Answers to Query

Answers to the query are :

holds_at(diagnosis(localised_atherogenic_arterial, at((left,upper_superficial_femoral))), 1989/4/26)

holds_at(suspected_condition(localised_dilatory, at((left,common_femoral))), 1989/4/26)

holds_at(symptoms(rest_pain, left), 1989/4/26)

holds_at(symptoms(leg_pain, left, thigh), 1989/4/26)

holds_at(causes_symptoms(localised_atherogenic_arterial, at((left, upper_superficial_femoral))), 1989/4/26)

holds_at(awaiting(investigation), 1989/4/26)

holds_at(recommended_test(ultrasound), 1989/4/26)

holds_at(higher_priority(localised_dilatory, over(localised_atherogenic_arterial, at((left, upper_superficial_femoral)))), 1989/4/26)

Figure 7.4: The relations holding true after the arteriogram in Case Study 1

Chapter 5, Figure 5.12 the arteriogram does not initiate a final diagnosis if the vascular condition is dilatory. Instead the arteriogram initiates the relation,

```
suspected_condition(localised_dilatory,
                                    at((left, common_femoral)))
```

as given by the rule I4 of Figure 5.13, implying there is need for further investigation. The symptoms are still believed to be caused by the localised atherogenic arterial condition which is the only final diagnosis holding true at that time. The suspected dilatory condition takes a higher priority which is indicated by the relation,

```
higher_priority(localised_dilatory,
                                over(localised_atherogenic_arterial,
                                at(left, upper_supperficial_femoral)))
```

holding true. Therefore no treatment is recommended for the localised atherogenic arterial condition. An ultrasound is recommended to investigate the dilatory condition. The relation (awaiting(investigation) implicitly implies that treatment should be delayed until the investigations are completed.

The relations holding true after the ultrasound is carried out is shown in Figure 7.5. As shown in Figure 7.5 the dilatory condition is confirmed by the ultrasound. The symptoms present may be caused by both arterial conditions present as indicated by the relations,

```
causes_symptoms(localised_dilatory, at((left, common_femoral)))

causes_symptoms(localised_atherogenic_arterial,
                                at(left, upper_supperficial_femoral))).
```

The localised dilatory condition takes a priority over the localised atherogenic arterial condition and treatment is recommended for the former. The state of the patient after the treatment is carried out is given in Figure 7.6. The event of treatment terminates the diagnosis localised dilatory. The localised atherogenic condition still persists. The persisting localised atherogenic condition needs to be investigated before deciding on a treatment plan since the surgical treatment may have changed the vascular condition of the vessel. An arteriogram is recommended for this investigation. There can be more

than one recommendation at any given time. If not specifically stated any one of these recommendations may be carried out next. After some surgical treatments usually an interview is recommended. The magnitude of the symptoms and signs observed (for example, pulse), at the interview give the clinician a clear indication of the outcome of the treatment. A clinician with much experience may take management decisions totally depending on the results of the interview. If the results of the interview do not give a clear indication a specific test is carried out to determine the success of the treatment. Thus an interview is recommended after the treatment *reconstruction:tube, material(prosthetic))*. Whether the localised atherogenic condition will be treated or not depend on the outcome of the interview. A further treatment will be done only if symptoms continue to persist since they should be caused by the localised atherogenic arterial condition (provided the treatment has been a success). Therefore, although a test has been recommended, the next recommended task is the interview as given by the relation `next_task(interview)` in Figure 7.6. Note that the clinical practice (thus the sequence of events) changes in the followup stage, before the treatment the investigations are given a precedence.

The state of the patient after the event interview is shown in Figure 7.7. At the interview, it is observed that the symptoms are reduced or absent. In Figure 7.2 the event description `absent` is used to denote the absence of the former severity of symptoms. Therefore, the symptoms should have been caused mainly by the dilatory condition. It is therefore recommended that the localised atherogenic arterial condition should not be immediately treated but only observed. Since no symptoms are caused by the localised atherogenic condition, the risk of surgery to the patient is more than that caused by the diseased condition. Thus the only treatment option that is suitable is *observe*. This is indicated by the relation,

```
treatment_options_for(localised_atherogenic_arterial,
    at((left, upper_superficial_femoral)), [observe]).
```

Answers to the query are :

```
holds_at(diagnosis(localised_dilatory,
            at((left,common_femoral))), 1989/5/11)

holds_at(diagnosis(localised_atherogenic_arterial,
            at((left,upper_superficial_femoral))), 1989/5/11)

holds_at(symptoms(rest_pain, left), 1989/5/11)

holds_at(symptoms(leg_pain, left, thigh), 1989/5/11)

holds_at(size(aneurysm, at((left,common_femoral)), 5.5), 1989/5/11)

holds_at(causes_symptoms(localised_dilatory,
            at((left,common_femoral))), 1989/5/11)

holds_at(causes_symptoms(localised_atherogenic_arterial,
            at((left,upper_superficial_femoral))), 1989/5/11)

holds_at(causes_risk_of_event(localised_dilatory,
            at((left,common_femoral))), 1989/5/11)

holds_at(awaiting(treatment), 1989/5/11)

holds_at(recommended_treatment(
            (reconstruction:tube,material(prosthetic)),
          for(localised_dilatory),
            at((left,common_femoral))), 1989/5/11)

holds_at(recommended_alternative_treatments(
            [(bypass with ligation,material(prosthetic))],
          for(localised_dilatory),
            at((left,common_femoral))), 1989/5/11)

holds_at(higher_priority((localised_dilatory,at((left,common_femoral))),
          over(localised_atherogenic_arterial,
            at((left,upper_superficial_femoral)))), 1989/5/11)

holds_at(treatment_options_for(localised_dilatory,
          at((left,common_femoral)),
          [(reconstruction:tube,material(prosthetic)),
           (bypass with ligation,material(prosthetic))]), 1989/5/11)
```

Figure 7.5: The relations holding true after the ultrasound in Case Study 1

File Edit Search Windows Fonts Eval Program Options

Answers to Query

Answers to the query are :

```
holds_at(diagnosis(localised_atherogenic_arterial,at((left,upper_superficial_femoral))),
1989/6/1)

holds_at(symptoms(rest_pain,left),1989/6/1)

holds_at(symptoms(leg_pain,left,thigh),1989/6/1)

holds_at(next_task(interview),1989/6/1)

holds_at(followup_of((reconstruction:tube,material(prosthetic))),1989/6/1)

holds_at(causes_symptoms(localised_atherogenic_arterial,at((left,
upper_superficial_femoral))),1989/6/1)

holds_at(recommended_test(arteriogram),1989/6/1)
```

Figure 7.6: The relations holding true after the treatment in Case Study 1

File Edit Search Windows Fonts Eval Program Options

Answers to Query

Answers to the query are :

```
holds_at(diagnosis(localised_atherogenic_arterial,at((left,upper_superficial_femoral))),
1989/6/6)

holds_at(followup_of((reconstruction:tube,material(prosthetic))),1989/6/6)

holds_at(recommended_treatment(observe,for(localised_atherogenic_arterial),at((left,
upper_superficial_femoral))),1989/6/6)

holds_at(treatment_options_for(localised_atherogenic_arterial,at((left,
upper_superficial_femoral)),[observe]),1989/6/6)
```

Figure 7.7: The relations holding true after the interview following the treatment in Case Study 1

7.2.2 Case Study 2: Exhibiting good clinical practice

Sometimes the first specific test of the investigation can be initiated by the GP. In this case the results of the test done will be available at the initial interview. Case Study 2 is one such example.

The phenomenon of one disease interfering with another can happen not only among vascular diseases but also among vascular and non-vascular diseases. The reason for such interference may vary from one case to another. Sometimes it is not safe to perform surgery on a patient under certain conditions. Under such conditions the disease which causes the danger takes precedence and will be dealt with first.

When there is more than one system recommendation at a particular time, one may take precedence over the other. Case Study 2 illustrates how good clinical practice such as performing all investigations before launching a treatment plan is modelled successfully by recommending the next task to be done.

The clinician can enforce the time period in which it is reasonable to believe information regarding an event by specifying a `rely_time` to that event. `rely_time` is also used to indicate how often a test has to be repeated. After the elapse of the `rely_time` specified, base relations initiated by that event will autonomously terminate causing the event to be recommended again. This behaviour is explained by example output displays below.

Scenario

The Ultrasound has already been done (by the request of the GP) when the patient comes for the initial interview. At the initial interview no symptoms were observed, but it was revealed that the patient suffers from hypothyroid. The condition hypothyroid was further investigated by a blood test before treating the localised dilatory condition. The test results indicated that the level of thyroxine in blood was lower than the expected. The patient is treated with the drug, thyroxine, until it is known that the thyroxine level in blood has reached a safe level. Only then the treatment for the localised dilatory condition is performed.

Illustration

Figure 7.8 shows the event descriptions for the Case Study 2.

```
inst(e0, ultrasound).
etime(e0, 1989/4/20).
result(e0, (right, internal_iliac), 6).

inst(e1, interview).
etime(e1, 1989/4/30).
result(e1, hypothyroid).

inst(e3, blood_test).
etime(e3, 1989/5/15).
result(e3, thyroxin(3:mmol/litre)).

inst(e4, drug_treatment).
etime(e4, 1989/5/17).
drug_given(e4, thyroxin).
rely_time(e4, 5).

inst(e5, blood_test).
etime(e5, 1989/5/28).
result(e5, thyroxin(8:mmol/litre)).

inst(e7, (reconstruction:tube, material(prosthetic))).
etime(e7, 1989/6/10).
location(e7, (right, internal_iliac)).
material_used(e7, prosthetic).
vessel_outcome(e7, patent).
```

Figure 7.8: The event descriptions of the Case Study 2

The relations holding true after the initial interview are displayed in Figure 7.9.

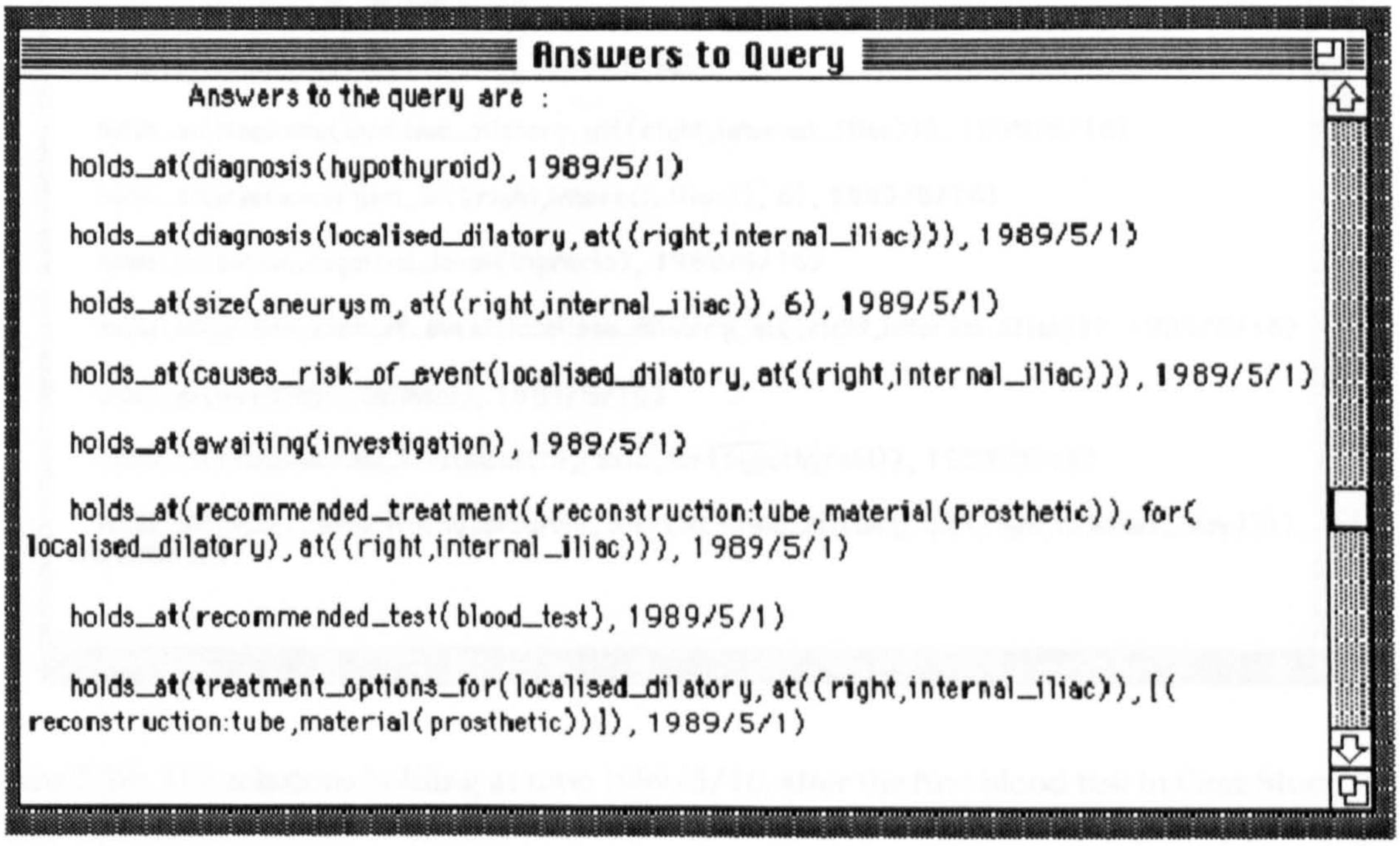


Figure 7.9: The relations holding true at time 1989/5/1, after the initial interview in Case Study 2

As can be seen in Figure 7.9 there are two system recommendation at this stage,

```
recommended_treatment((reconstruction:tube, material(prosthetic)),
        for(localised_dilatory), at((right, internal_iliac)))

recommended_test(blood_test)
```

The recommendation for the next task, `awaiting(investigation)` implies that the test should be carried out before the treatment.

A blood test is carried out next. The state of the patient after the blood test is given in Figure 7.10. The results of the blood test reveals that the thyroxine level in blood is lower than the normal level, indicated by the relation,

```
below_required_level(thyroxin)
```

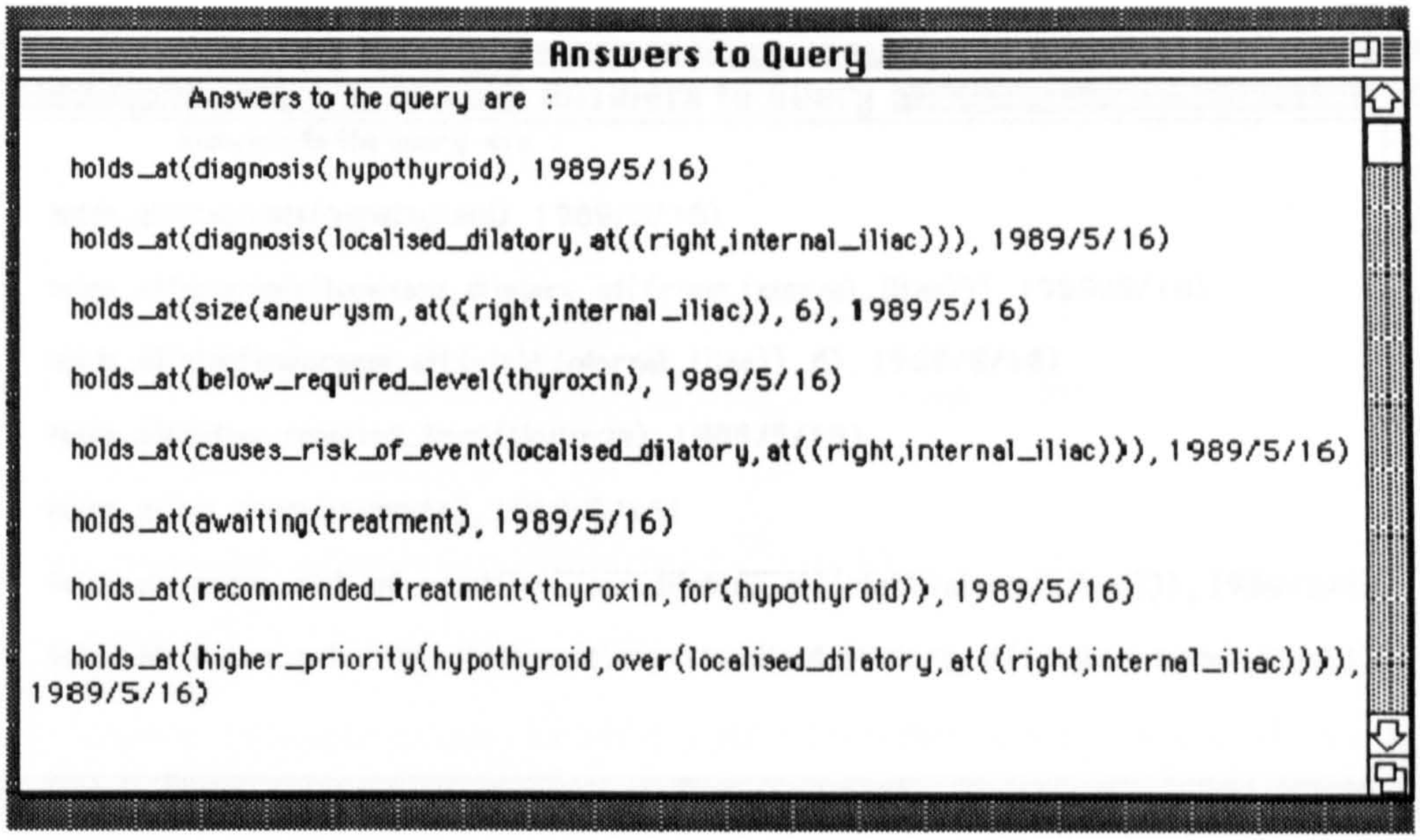


Figure 7.10: The relations holding at time 1989/5/16, after the first blood test in Case Study 2

and a treatment with the drug thyroxine is recommended. The lower thyroxine level in blood make the patient unsafe for surgery thus the hypothyroid condition takes priority over the localised dilatory condition.

The next task recommended is the drug treatment indicated by the relation,

```
awaiting(treatment)
```

The drug treatment starts at 1989/5/17 and is recommended to continue for 5 days, indicated by the event description `rely_time(e4, 5)` in Figure 7.8. The relations holding true at 1989/5/18 after the start of drug treatment are given in Figure 7.11. The relation `on_drugs(thyroxin)` is initiated by the event of drug_treatment and stands true after the event as shown in Figure 7.11. This relation will have a life time of 5 days after which, it will autonomously terminate as shown in Figure 7.12.

Answers to Query

Answers to the query are :

holds_at(diagnosis(hypothyroid), 1989/5/18)

holds_at(diagnosis(localised_dilatory, at((right,internal_iliac))), 1989/5/18)

holds_at(size(aneurysm, at((right,internal_iliac)), 6), 1989/5/18)

holds_at(below_required_level(thyroxin), 1989/5/18)

holds_at(on_drugs(thyroxin), 1989/5/18)

holds_at(causes_risk_of_event(localised_dilatory, at((right,internal_iliac))), 1989/5/18)

holds_at(higher_priority(hypothyroid, over(localised_dilatory, at((right,internal_iliac)))), 1989/5/18)

Figure 7.11: The relations holding true at time 1989/5/18, after the start of the drug treatment in Case Study 2

Answers to Query

holds_at(diagnosis(hypothyroid), 1989/5/25)

holds_at(diagnosis(localised_dilatory, at((right,internal_iliac))), 1989/5/25)

holds_at(size(aneurysm, at((right,internal_iliac)), 6), 1989/5/25)

holds_at(below_required_level(thyroxin), 1989/5/25)

holds_at(causes_risk_of_event(localised_dilatory, at((right,internal_iliac))), 1989/5/25)

holds_at(awaiting(investigation), 1989/5/25)

holds_at(recommended_treatment(thyroxin, for(hypothyroid)), 1989/5/25)

holds_at(recommended_test(blood_test), 1989/5/25)

holds_at(higher_priority(hypothyroid, over(localised_dilatory, at((right,internal_iliac)))), 1989/5/25)

Figure 7.12: The relations holding true at time 1989/5/25 in Case Study 2, after the life time of the relation `on_drugs(thyroxin)` expired.

The relation `below_required_level(thyroxin)` initiated by the results of a blood test is terminated only by the results of another blood test. Thus as long as it is known that the thyroxine level in blood is low either the patient should be under the treatment or awaiting it. Therefore at the termination of the relation `on_drugs(thyroxin)` indicating the patient is not under the drug treatment anymore, a treatment will be recommended. The recommendation will be withdrawn when it becomes known that the level of thyroxin is normal. Thus at 1989/5/25 two recommendations will stand true:

`recommended_treatment(thyroxin, for(hypothyroid))` and

`recommended_test(blood_test).`

But the relation `awaiting(investigation)` will indicate that the test should be carried out first.

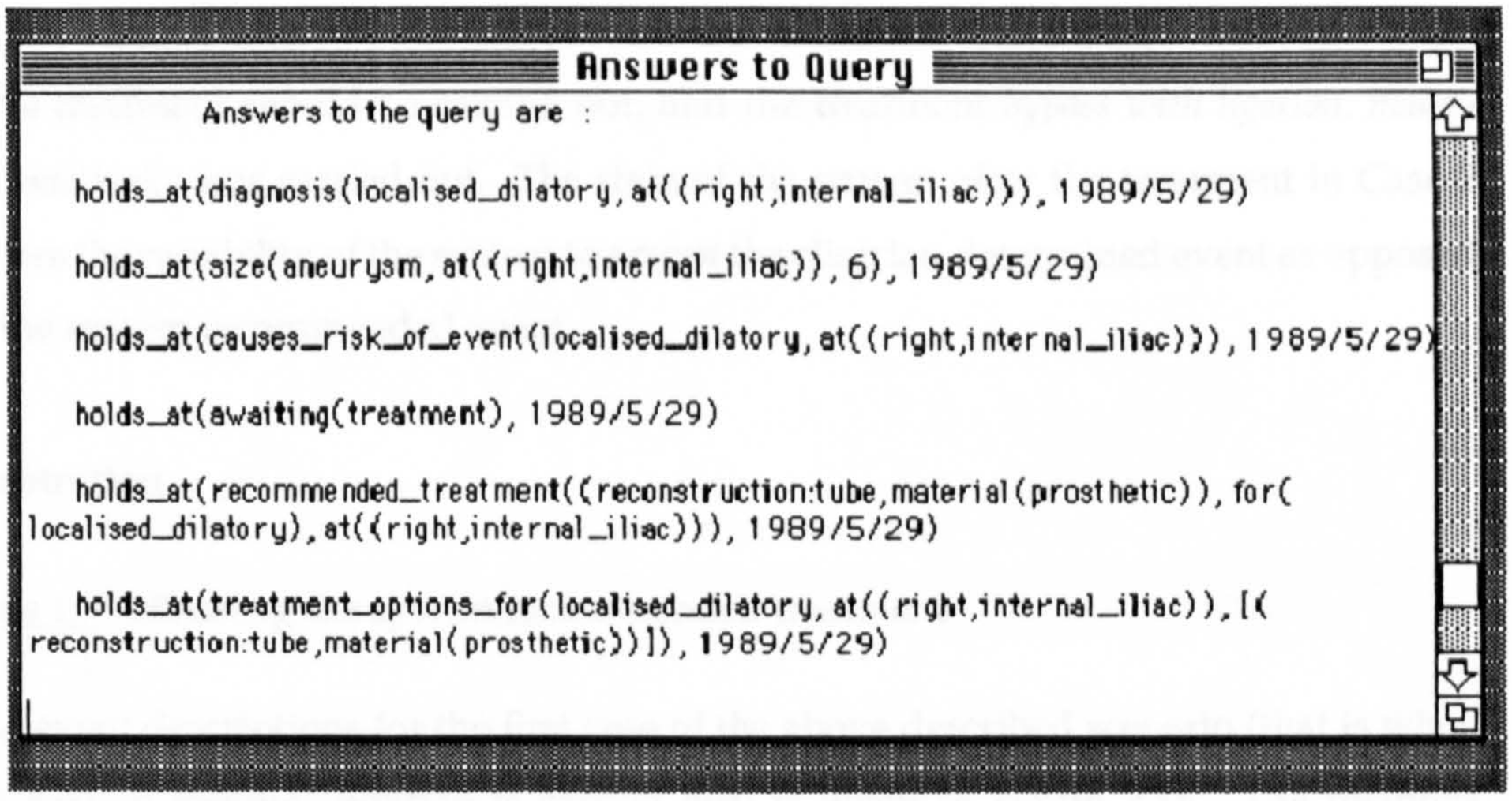


Figure 7.13: The relations holding true at time 1989/5/29, after the blood test which followed the drug treatment in Case Study 2

The state of the patient after the blood test is given in Figure 7.13. The hypothyroid condition no longer takes a priority, and the treatment of thyroxine no longer holds true. Treatment recommendations for the localised dilatory condition are made .The next task recommended is the treatment of the localised dilatory condition.

7.2.3 Case Study 3: Clinician overriding the system recommendation

The system recommendations need not be followed necessarily. At any given time the clinician can take his/her autonomous decision overriding the system recommendations. The following case study illustrates this. We first illustrate the case where the system recommendation is carried out. The case where the clinician countercommands the system recommendation is illustrated next.

Scenario

At the initial interview it was diagnosed that the patient is suffering from dilatory disease and an ultrasound is recommended for further investigation. The ultrasound confirms the dilatory condition and a treatment *reconstruction:tube, material(prosthetic)* is recommended. In the first case, Case1, the recommended treatment is carried out and the state of the patient is displayed after the treatment.

In Case 2, after the ultrasound test the clinician decides that a suitable alternative treatment should be carried out, and the treatment *bypass with ligation, material(prosthetic)* was carried out. The state of the patient after the treatment in Case 2 shows the capability of the system to accept the clinician determined event as opposed to the system recommended event.

Illustration

Case 1: Performing the system recommended treatment

The event descriptions for the first case of the above described scenario (that is when the system recommendation is carried out) is given in Figure 7.14. The relations holding true after the initial interview is shown in Figure 7.15

The recommended investigation, the ultrasound was carried out at `1990/2/24`. The state of the patient after the ultrasound test represented by the relations holding true at 1990/2/25, is given in Figure 7.16. The results of the ultrasound reveals that the condition `localised_dilatory` is present at the location left common femoral. A treatment of *reconstruction:tube, material(prosthetic)* is recommended.

```
inst(e1, interview).
etime(e1, 1990/2/12).
report(e1, leg_pain, left, thigh).
result(e1, dilatory).

inst(e2, ultrasound).
etime(e2, 1990/2/24).
result(e2, (left, common_femoral), 6).

inst(e3, (reconstruction:tube, material(prosthetic)))).
etime(e3, 1990/3/20).
location(e3, common_femoral).
material_used(e3, prosthetic).
vessel_outcome(e3, patent).
```

Figure 7.14: The event descriptions of Case Study 3:Case 1

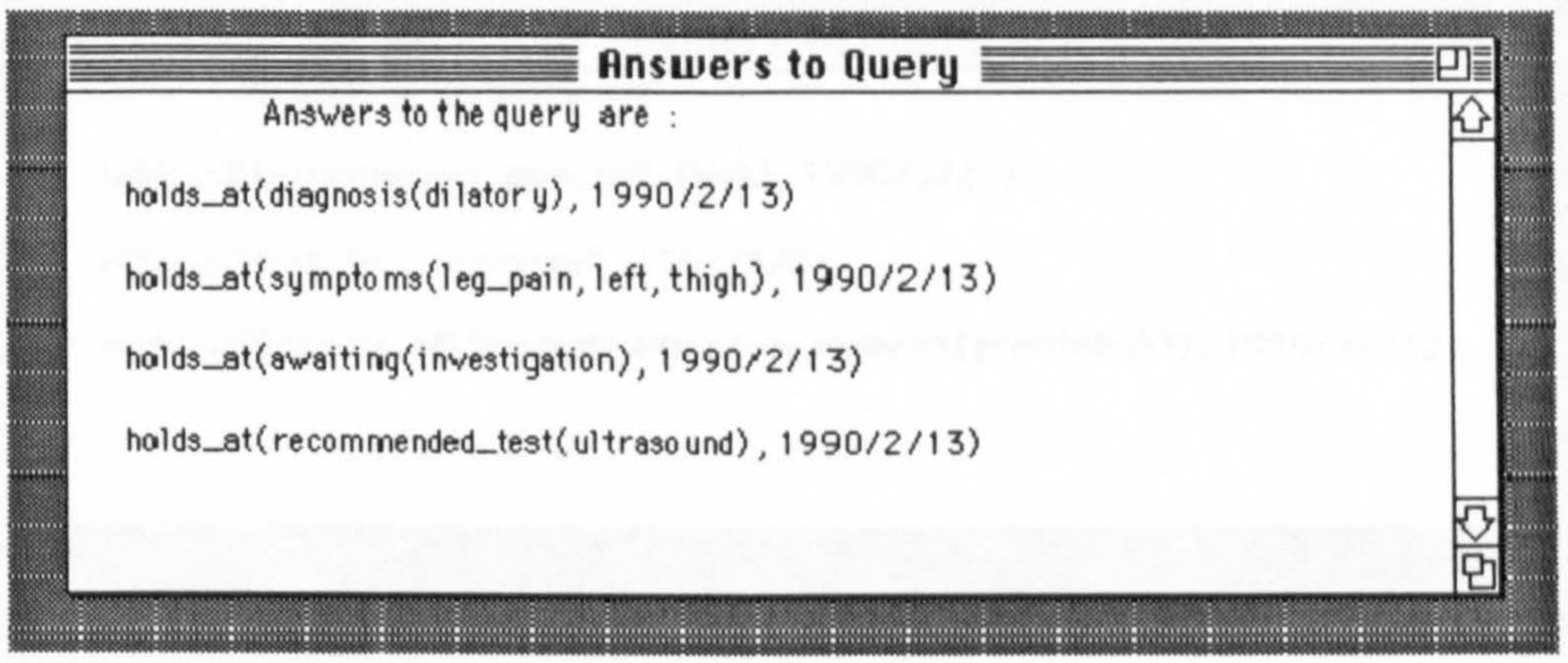


Figure 7.15: The relations holding true after the initial interview in Case Study3

Answers to Query

Answers to the query are :

holds_at(diagnosis(localised_dilatory, at((left,common_femoral))), 1990/2/25)

holds_at(symptoms(leg_pain, left, thigh), 1990/2/25)

holds_at(size(aneurysm, at((left,common_femoral)), 6), 1990/2/25)

holds_at(causes_symptoms(localised_dilatory, at((left,common_femoral))), 1990/2/25)

holds_at(causes_risk_of_event(localised_dilatory, at((left,common_femoral))),
1990/2/25)

holds_at(awaiting(treatment), 1990/2/25)

holds_at(recommended_treatment((reconstruction:tube,material(prosthetic)),
for(localised_dilatory), at((left,common_femoral))), 1990/2/25)

holds_at(recommended_alternative_treatments([(bypass with ligation,material(prosthetic))],
for(localised_dilatory), at((left,common_femoral))), 1990/2/25)

holds_at(treatment_options_for(localised_dilatory, at((left,common_femoral)),
[(reconstruction:tube,material(prosthetic)),
(bypass with ligation,material(prosthetic))]), 1990/2/25)

Figure 7.16: The relations holding true after the test ultrasound in Case Study 3

Answers to Query

Answers to the query are :

holds_at(symptoms(leg_pain, left, thigh), 1990/3/21)

holds_at(next_task(interview), 1990/3/21)

holds_at(followup_of((reconstruction:tube,material(prosthetic))), 1990/3/21)

Figure 7.17: The relations holding true after the treatment in Case Study 3:Case 1

The relation,

```
recommended_alternative_treatments([(bypass with ligation,
        material(prosthetic))], for(localised_dilatory,
        at((left, common_femoral))))
```

in Figure 7.16 describes the alternative suitable treatments for the localised dilatory condition at the vascular location, left common femoral. The state of the patient after the recommended treatment was carried out is given in Figure 7.17.

Case 2: Performing the clinician determined treatment

After the ultrasound, the clinician decides that the recommended alternative treatment is more suitable than the recommended treatment for the patient in this case. Therefore the treatment *bypass with ligation, material(prosthetic)* is performed. The event descriptions input in this case are given in Figure 7.18.

```
inst(e1, interview).
etime(e1, 1990/2/12).
report(e1, leg_pain, left, thigh).
result(e1, dilatory).

inst(e2, ultrasound).
etime(e2, 1990/2/24).
result(e2, (left, common_femoral), 6).

inst(e3, (bypass with ligation, material(prosthetic))).
etime(e3, 1990/3/20).
location(e3, common_femoral).
material_used(e3, prosthetic).
vessel_outcome(e3, patent).
```

Figure 7.18: The event descriptions of Case Study 3:Case 2

The state of the patient after the clinician recommended treatment is given in Figure 7.19. As illustrated the system will accept the change. The recommendation for

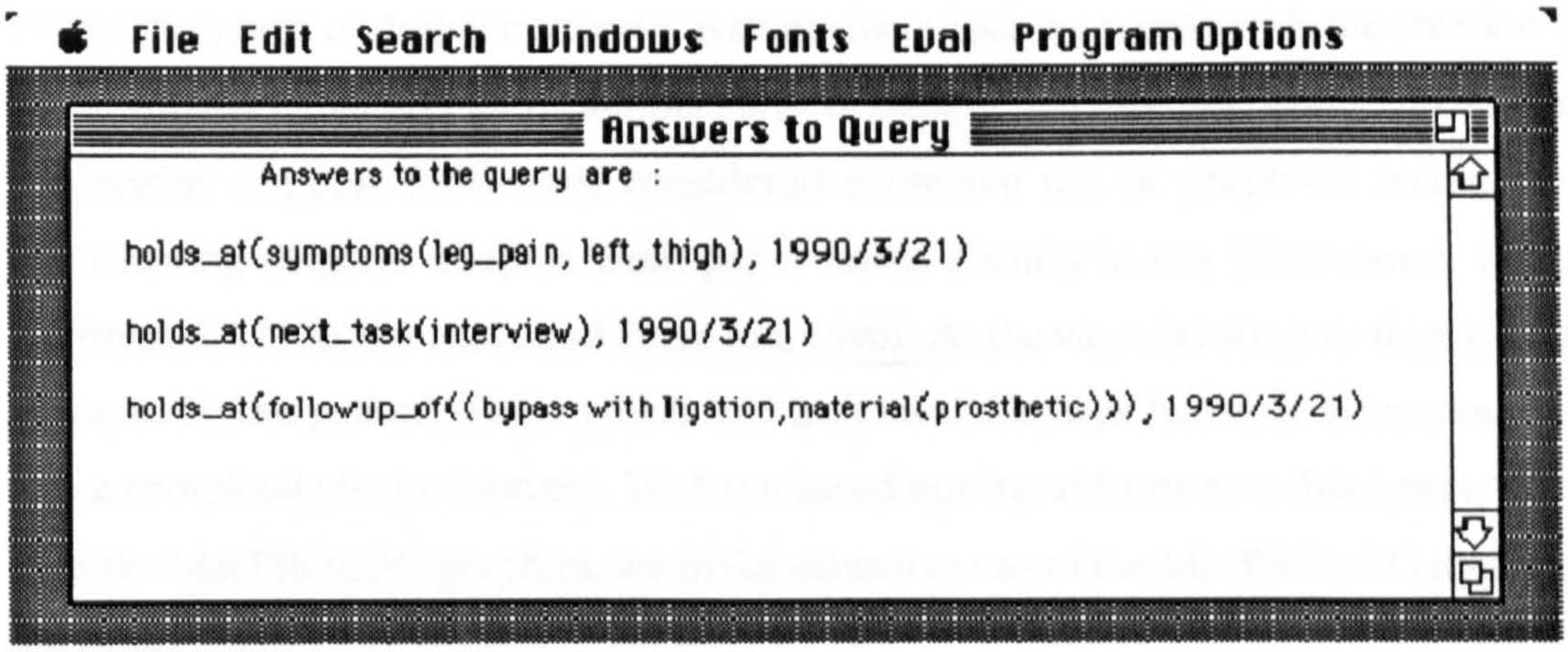


Figure 7.19: The relations holding true after the treatment in Case Study 3: Case2

reconstruction:tube, material(prosthetic) will no longer hold true.

The ability of the system to accept any event at any given instant of time enhances the flexibility of the system. In this connection it is important to note that the clinician is not compelled by the system to perform one of the recommended alternative treatments. The clinician has ultimate responsibility for the next event performed, and this may be a completely different treatment or some other action such as a test. Whatever this next event is it will be accepted by the system in a similar manner to the above. The only difference will be in the 'whynot' dialogue (see Section 7.1.5).

7.3 Conclusion

We have described in this chapter the structure of the prototype system: how patient data is stored, the user interface, the query system which provides the means of accessing the knowledge base and requesting explanations. Relevant information in the patient record can be retrieved for decision support through recommendations, and other relations holding true at a specific time. The recommendations also lead

to consistent practice in particular situations, for example, making sure the thyroxine level in blood is at a suitable level before surgery.

In recent years, many computer based decision-support tools have been developed to help medical personnel in clinical settings (Shortliffe, 1987; Dasta, Greer and Speedie, 1992). Designers of decision support systems have had to grapple with the creation of a user interface that will allow simple entry of patient information. Many developers of decision support tools have considered extensive use of graphical techniques, often having displays adapted from paper forms already in use (Hammond, 1993a; Hammond, 1993b, Shortliffe and Perreault, 1990). At the vascular surgery department at the RHS, the patient record which had graphical displays has now been modified into a completely textual format. We have based our input format on this hence, more than the MacPROLOG graphics, we make extensive use of the MacPROLOG dialogue windows.

As our aim is not at a fully operational system but rather a research workbench over a realistic domain, the display of information has been kept at the underlying raw level as Prolog clauses. We believe that this basic Prolog clause display will aid the researcher in comprehending and expanding the system more than a natural language display.

Through three case studies we have illustrated how the implemented system successfully exhibits the characteristics we have aimed for. By showing the displays of relations holding true at certain time points we have illustrated how tests and treatments are recommended. When more than one recommendation exists how the recommendation of the next task indicates which recommendation should be carried out is also shown. Not only what actions should be carried out next but, when these actions should be carried out is recommended. Any recommendation can be overridden by the clinician at any given time. This added flexibility enhances the suitability of the system.

The first two case studies illustrated how the system deals with the presence of multiple disease conditions: in case study one the presence of multiple vascular conditions, and in case study two simultaneous occurrences of vascular and non-vascular conditions. In both cases one disease interferes with the other by taking a higher priority. Case study two illustrates how methods of good clinical practice are

modelled. For example, it is good practice to make sure that the thyroxine in blood is at a normal level before embarking on any surgical treatment when the patient is known to have hypothyroid. Case study two also illustrates how relations can be autonomously terminated by associating the initiating event with a clinician determined `rely_time`.

The third case study illustrates how the system recommendations can be overridden by the clinician. Any event can be accepted at any time by the system. This flexibility, provided by the underlying Event Calculus, makes the system suitable for the clinical domain.

Providing informative explanations for improving the use of decision support systems has attracted increasing interest over the last years (Horacek, 1992). In our system, because of the Event Calculus formalism, we are able to produce in addition to the rule traces, the sequence of related events which lead up to a particular conclusion. As well as being informative this sequence of events provides access to the detailed information stored in the associated case frames.

The Event Calculus axioms are able to handle the domain requirements adequately. Data base update by addition alone preserves the history of the changing domain. The state of the domain as it was at any past instance can be rebuilt without any cost to additional memory.

The main feature of our system is that a flexible temporal reasoning framework is built into the structure of the knowledge based system. This flexible framework provides interactive assimilation and retrieval of information currently stored in the patient record.

We do not take uncertainty into consideration. This does not affect the accuracy of the system in anyway since the inference methods in the domain do not require such considerations.

Our intention in this research was not to build a fully operational system to be used by the clinicians, but to build a prototype in which we can show that, given patient data as event instances, Event Calculus provides a good supporting basis for temporal reasoning in the domain of vascular surgery considered in realistic detail. The prototype is intended to be a research workbench, which can be used to explore further the benefits of the approach. Some of the benefits have been illustrated by the case studies in this chapter.

8 Comparison of the Event Calculus Framework with a Process Modelling Approach

The precise description of processes as they develop through time is the basis for a wide range of computing applications for control and support of such activities as decision support and project management. Event Calculus can be considered to be a process description framework. In this chapter we compare and contrast the Event Calculus with another more procedural approach to process description, Process Modelling in the language PML[1] (Bruynooghe, 1991; Roberts, 1989). One aim of this study is to contribute to the methodology of the process modelling approach by viewing processes from a declarative (logical) stance as done in the Event Calculus. A second aim is to identify complementary aspects of the two approaches and, in particular, to identify any ways in which Process Modelling might complement Event Calculus in suitability for the vascular surgery domain.

In Section 8.1 we introduce Process Modelling and the language, PML, and propose suitable definitions for some of the terminology used relevant to Process Modelling concepts and PML. In order to investigate the relationship between Event Calculus and PML we formulated two examples in the two frameworks. The examples are: a simple interactive bank account; and part of the (more complex), clinical pathway in vascular surgery (described in Chapter 2. Section 2.1). The PML formulation of the latter and its comparison with the Event Calculus is not included in this chapter due to

[1]This part of the research was done under the Introduction of Process Technology (IOPT) Project. The IOPT project is a DTI sponsored collaborative project between ICL, Praxis, ISS and Manchester University. Its main goal is to develop and apply computer support for cooperative work in diverse industrial and government sectors, using process modelling and other forms of Process Technology. It also aims to enhance process modelling languages and methods.

lack of space (the complete formulation is included in (Abeysinghe and Soper, 1993)). In Section 8.2 we give the representation of the bank account in both frameworks, Event Calculus and PML, so that the reader can easily follow our comparison which follows in Section 8.3. In Section 8.2 first the notation for PML is specified and then the formalisations of the bank account example as executable programs are given for both PML and Event Calculus. The differences between the two formalisms as perceived by us are enumerated in Section 8.3. We discuss the results of our investigation in Section 8.4.

8.1 Process Modelling and the Language PML

Our first task is to define the terms that will be used in the course of our research. In this section we try to clarify some of the definitions relevant to Process Modelling and the language PML.

In spite of the considerable literature published (Conradi, Fernström, Fuggetta and Snowdon, 1992; Feiler and Humphrey, 1993) there is still no standard terminology in the field of Process Modelling. Users have defined Process Modelling and other associated terms according to their applications and implementations.

Galle (Galle, 1992) identifies a number of process views which together gives a complete description of a process:

- which are the data to be produced,
- what is to be done,
- who needs to carry out the activities,
- how and when to carry out the activities.

A process can be defined as: "a set of partially ordered steps intended to reach a goal" (Feiler and Humphrey, 1993).

A goal in many instances is achieved by many steps. Taking an example on making a cake, this goal could be achieved by the following steps:

- getting the ingredients together,
- mixing the dough,

- prepare the baking tray,
- cooking the cake.

The set of steps to reach the goal is called the process. The end result of each step is a subgoal. The set of steps which lead to a subgoal is defined as a subprocess. An actor who performs a subprocess, a human or a machine, is called an *agent*.

A Process Model is an abstract description of the process. It consists of subprocesses that can be enacted by agents. Each activity that an agent carries out at any given time is concerned with achieving a particular subgoal. The term *role* is used to describe the activity of an agent. Taking the example of making a cake given above: an agent may be given the role of managing the project, that is making the cake; another agent may be given the role of collecting the ingredients; and so on. In performing a role one agent may require to interact with another, or more precisely with a role that another agent is performing, in order to receive or impart information. The information is contained in entities called *resources* and is passed via *interactions* (Butler and Zientek, 1990).

Each role progresses by following the steps, the actions, prescribed by the process description to achieve the goal. There may be a choice of which action is to be taken. Thus a process model consists of: roles – that together achieve the goal of the process; interactions – that enable roles to communicate with one another; entities – that represent the information in the process; and actions – that describe the steps a role can take in achieving a subgoal (Butler and Zientek, 1990).

The process that is modelled as a whole may involve several agents working concurrently, and each may work at several tasks at any given time. This is supported by a network of roles. Each role has its own part to play in achieving a certain subgoal which is a part of the overall goal of the process.

Modelling a process begins with an analysis of the problem, the system, or a set of requirements . The data involved are identified and the relationships established (Roberts, 1989). The major goal of Process Modelling is to achieve a better understanding of the process. It also provides the ability to optimise processes and analyse the impact of proposed changes on the process prior to implementation.

PML is an Object Oriented Process Modelling language in which process de-

scriptions can be expressed (Snowdon, 1992). A process in PML, is considered to compromise of a network of roles which communicate through interactions. A key concept of the language is *class*. A class is a definition — a prescription of the behaviour of an instance of the class in the model. The characteristics of a class are called the *properties*. A class defines the properties for its members, objects which are instances of the class. An instance of a class is an occurrence of the class (Butler and Zientek, 1990). A class is similar to a case frame which we associate with each event type in the Event Calculus approach (see Chapter 5). In Event Calculus an instance of the case frame is the actual event.

The principal classes of PML are `Role`, `Interaction`, `Action`, and `Entity`[2] (Bruynooghe, 1991). Each principal class has a set of *property categories* associated with it. These are the main headings under which the information about the class is described. Taking the `Role` class as an example, the property category `resources` describes which data objects the role is manipulating, `assocs` describes the communications with other roles, `actions` describes the steps the role has to follow in order to achieve its goal, `termconds` describes the conditions which must be satisfied for the role to finish.

A `Role` exhibits behaviour. The behaviour of a `Role` is defined in terms of `actions` operating on its local entities (i.e. the `resources`) (Sa and Worboys, 1992). The actions allowed for a role at any given time is given by the 'action agenda' of that role.

The interaction between roles is defined by the class `Interaction`. The class `Actions` are the activities that can be carried out by the roles. PML provides a number of predefined subclasses of `Action` class. For example, the subclass `StartRole` creates a new role, and `BehaveAs` modifies existing roles. The `actions` of a role are defined in terms of these. They are also used to define the behaviour of user-defined subclasses of the `Action` class.

The class `Entity` provides a means of defining a data structure, representing the data manipulated by the process. This is usually a multifield item such as would be a record in other languages.

---

[2]Words starting with an uppercase refers to classes (or subclasses) in PML. This special font is used to distinguish words that are inherent to PML.

The `actions` of a PML role may be compared to the event types in Event Calculus. In the PML approach only certain actions associated with a role are allowed for that role at a given instant (actions that are displayed in the action agenda of the role). The actions on the action agenda are enforced by the PML system according to the conditions attached to actions. In the Event Calculus the events (actions) which are allowed at any given time can be explicitly modelled by introducing a relation `next_allowed_task`, defined in terms of other relations.

Process Models constructed using PML can evolve enabling arbitrary levels of refinement. Users interact with the model as and when user actions are triggered in the model, and so users may extend the model as they wish.

Basically Event Calculus separates the concepts of events and relations. Whereas PML encapsulates them within the concept of `Role`, specifically through the action agenda of the role.

8.2 Bank Account Example

To bring out the relationship between Process Modelling with PML and Event Calculus, this section describes how a simple example can be described in both formalisms. The example chosen concerns one client dealing with a bank account. A client is allowed to open an account at any time. Once an account is open, money can be deposited and the balance can be checked at all times. If funds are available money can be withdrawn. All actions cease to exist on the closure of the account.

8.2.1 Notation Used

Event Calculus: The Event Calculus is represented in Prolog. The same notation as used to describe the Event Calculus in Chapter 4 is used here. In particular identifiers starting with a lowercase letter are used to represent predicate symbols and constants, and those starting with an uppercase letter are used to represent variables.

PML: PML is upper and lower case sensitive. All class names start with an uppercase letter. The instance names of classes begin with a lowercase letter but can contain uppercase letters at any position other than the starting letter. For example we

can define a Role called Client, then we may declare two instances of that Role as clientNumber1 and clientNumber2.

PML gives each principal class a set of property categories. These are the headings under which the information about each class will be declared: the field names and their types (`resources` of a `Role`, and `parts` of an `Entity`); the activities (if any) that can be carried out (`actions` of a `Role`, and `parts` of an `Action`); the conditions which constrain the activities (`when` clause); and also the conditions which constraints the instances of a `Class` (the `termconds`); and the means of binding to external objects (`assocs` of a `Role` and an `Action`).

The property category headings are in lowercase. The names of the property categories follow the same rule as instances of classes: they begin with a lowercase letter and are followed by a colon. If the property is a resource then the colon is followed by the resource class name (which starts with an uppercase). If the property is an action then the colon is followed by one or more `Action` class names; if the action involves more than one activity, then all such calls are enclosed within brackets '{' and '}'. Every action ends by a boolean expression which starts with the word **when**.

Comments start with an exclamation mark and can be included anywhere in the program.

First we show the PML representation of the example and then the Event Calculus representation.

8.2.2 Representing a Bank Account in PML

The PML program for the bank account is shown in Figure 8.1. We use the entity `Account` with properties `name` and `balance` to represent a bank account. At the start only the action `openAccount` is available. When this action is selected, the name and the opening balance of the account is asked for. After the action `openAccount` is completed the actions `depositMoney`, `checkBalance`, and `closeAccount` will be available. The action property `withdrawMoney` appears on the action agenda only if there is a non-zero balance in the account. Money can be withdrawn only if the requested amount when withdrawn does not make the account overdrawn.

After the completion of the action `openAccount` the balance in account can be viewed at any instant by triggering the action `checkBalance`. Only the current state

of the account can be viewed.

```
!  This is the definition of a Role which represents an
!  interactive bank account.

classes
Account isa Entity with
parts
     name:String
     balance:Real
end with

resources
     amount:Real{}
     account:Account
     agreed:Bool{false}

actions
openAccount:GetNew(class=Account,
                        label='Supply Values',
                        object=account)
               when true
! the condition "when true" imply that an account can be open
! at any time

closeAccount:QueryAlert(answer=agreed,
                              question='Do you want to close the
                                        account?')
                    when nonnil openAccount
! the account is allowed to be closed only when the client
! confirms his wish to  close the account
```

```
withdrawMoney:{ Modify(agendaLabel='withdraw',
                       icon='UserAction',
                       label='Amount to be withdrawn',
                       object=amount);

                if amount > account.balance then
                    ViewObjectNow(label='Insufficient Funds
                                         in Account',
                                  object=account.balance)
                else
                    account.balance :=
                                  account.balance - amount
                end if;
                amount:=0.0
              }
                when
                       (nonnil openAccount
                        ~agreed &
                        account.balance > 0.0)
! the action withdrawMoney is allowed only after opening the
! account until it is closed (referred to by the value of the
! variable, agreed, to be false) and only if the balance is
! non zero.

depositMoney:{ Modify(agendaLabel='deposit',
                      icon='UserAction',
                      label='Amount to be deposited',
                      object=amount);
               account.balance := account.balance + amount;
               amount := 0.0
             }
```

```
                when
                    (nonnil openAccount &
                     ~agreed)
! depositing money can be done if an account has been open
! (nonnil openAccount) and it is not yet closed

checkBalance:ViewObject(object=account.balance,
                        agendaLabel='balance',
                        label='The current balance is')
                when (nonnil openAccount & ~agreed)
! Balance can be checked at any time after opening the account
!  and before closing it.
```

Figure 8.1: The bank example formulated in PML

In this example there is just one role so it does not appear explicitly in the program. The agent associated with this role can be considered to be the client, i.e. the agent that instigates the actions. In contrast to the Event Calculus, the PML process description only allows certain actions at a given point (time) in the process, represented by the action agenda of the role at that time.

8.2.3 Representing the Bank Problem in Event Calculus

The Event Calculus program for the bank account is given in Figure 8.2. In the Event Calculus representation the possible actions are the event types. They are: `openAccount`, `depositMoney`, `withdrawMoney` and `closeAccount`. Each event has the event cases,

```
inst(Event_Name, Event_Type),
etime(Event_Name, Event_Time),
account_name(Event_Name, Name_of_Client).
```

If the event is an `openAccount` or a `depositMoney` then it has additionally the case,

```
amount_deposited(Event_Name, Amount).
```

If the event is a withdrawMoney then it has additionally the case,

```
amount_withdrawn(Event_Name, Withdraw).
```

```
/** The Event Calculus formulation of the bank example
    is given below **/

/** The domain independent rules used **/
holds_at(U, T):-
    initiates(E1, U),
    etime(E1, T1),
    T1 < T,
    not interrupted(T1, U, T).

interrupted(T1, U, T):-
    terminates(E2, U),
    etime(E2, T2),
    T2 > T1,
    T2 ≤ T.

/** The base relations initiated and terminated **/

initiates(E, has_account(Name)):-
    inst(E, openAccount),
    account_name(E, Name).

initiates(E, balance(Name, B)):-
    inst(E, openAccount),
    account_name(E, Name),
    amount_deposited(E, B).
```

```
initiates(E, balance(Name, B)):-
    inst(E, depositMoney),
    amount_deposited(E, B),
    not (event_before(E, E2),
        initiates(E2, balance(Name,B2))).

initiates(E, balance(Name, B)):-
    inst(E, depositMoney),
    amount_deposited(E, B1),
    event_before(E, E2),
    initiates(E2, balance(Name,B2)),
    B is B2 + B1.

initiates(E, balance(Name,B)):-
    inst(E, withdrawMoney),
    amount_withdrawn(E, B1),
    event_before(E, E2),
    initiates(E2, balance(Name,B2)),
    B is B2 - B1.

terminates(E, has_account(Name)):-
    inst(E, closeAccount),
    account_name(E, Name).

terminates(E, balance(Name, B)):-
    initiates(E, balance(Name, NewB)),
    not NewB=B.

terminates(E, U):-
    inst(E, closeAccount).
```

```
/*** The ramifications **/

holds_at(has_money(Name), T):-
    holds_at(balance(Name, Balance), T),
    Balance > 0.

holds_at(over_drawn(Name), T):-
    holds_at(balance(Name, Balance), T),
    Balance < 0.

holds_at(next_allowed_tasks(Name,
        [depositMoney, withdrawMoney, closeAccount,
        openAccount]), T):-
    holds_at(has_money(Name), T).

holds_at(next_allowed_tasks(Name,
        [depositMoney, closeAccount, openAccount]), T):-
    holds_at(has_account(Name), T),
    not holds_at(has_money(Name), T).

/** Other time independent Prolog rules required for the
    domain **/

/* Given an event E, returns the event immediately before E */
event_before(E, E1):-
    etime(E, T),
    account_name(E, Name),
    etime(E1, T1),
    account_name(E1, Name),
    T1 < T,
    not (inst(E2, Etype),
        account_name(E2, Name),
```

```
        etime(E2, T2),
        T2 > T1,
        T2 < T).
```

Figure 8.2: Formulation of the bank account in Event Calculus

An event of an openAccount initiates the relation,

```
has_account(Name_of_Client).
```

If the event also has an amount of money deposited associated with it, then it also initiates the relation,

```
balance(Name_of_Client, Amount).
```

The relation has_account(Name_of_Client) is terminated by an event of a closeAccount. The relation balance(Name_of_Client, Amount) is terminated by the initiation of a new balance. A new balance is initiated by the succeeding event of a depositMoney or withdrawMoney. The event of closeAccount terminates all existing relations.

We introduce ramifications, that is relations which are time related but do not depend directly on any event. For example, for the bank account a person has overdrawn his/her account if at any instant T the balance in his/her account is less than zero. This is represented by the following ramification:

```
holds_at(over_drawn(Name), T) :-
        holds_at(balance(Name, Balance), T),
        Balance < 0.
```

Ramifications provide an easy way of accessing the knowledge base.

It is natural in Event Calculus to accept any event at any time, since it assumes an incomplete past. This contrasts with PML which restricts the allowed actions (events) which can happen at a given time. In the Event Calculus it is possible to restrict the events which can be accepted. This has been done in the bank example so that the next event is restricted to those events which comply with bank regulations by the introduction of the ramification, next_allowed_tasks. The Event Calculus also contrasts with PML in maintaining a historical knowledge base. Therefore the state of the account at any past instant of time can be viewed at any time.

8.3 Comparison of PML and Event Calculus

The different features of PML and Event Calculus as perceived by us are described in this section.

1. Event Calculus is based on events, relations and time periods in which a relation holds true. The state of the world at any given time is viewed by the relations holding true at that time.

   PML is based on the notion of concurrently executing agents all co-ordinating to achieve a common goal. It is based on roles, actions and interactions. Each role has a different view of the world. The view of the world modelled as seen by a role at any instant is represented by the actions available at that instant. The state of the world at any time is the collective view of all roles active at that time.

   The basic idea behind a role is that it denotes an active agent, be it a person or a machine. As such a role is the bearer of actions which can change the state of the system. Since active agents are the motor for a process this is a useful representation. While events in the Event Calculus do not have to be associated with roles, in a process modelling context they usually are, since they are usually associated with an agent.

2. In the usual case of the Event Calculus, the knowledge base can be viewed by any user.

   In PML, the view of the knowledge base depends on the user. A user corresponds to an instance of the class role. Roles may exchange information or provide information another role requires. For example, taking the example of making a cake given in Section 8.1, the agent who cooks the cake has to wait until all other subprocesses are completed. The agent who mixes the dough has to wait until ingredients initially required are collected and weighed. The agent who prepares the tray is independent of all other roles and can start at any time but should finish before or at the same time as the person who mixes the dough in order not to waste time idling.

   A role in PML is a local data store. A role can view only that part of the knowledge base relevant to it. If we want a particular role to view the world as viewed by

all roles in the process then it should be so modelled. We can limit what a role can see and what not. This provides a means of information hiding.

The modification of the Event Calculus to provide views of the knowledge base seems to be unproblematic. Also the timing of events can be influenced in the Event Calculus by using ramifications to define allowed actions.

3. In a logical database, the knowledge base should be logically consistent. New information received should be consistent with what is contained already. But there is no reason why there could not be several knowledge bases say $kb_1, kb_2, \ldots, kb_n$ (n > 1) relevant to different users $usr_1, usr_2, \ldots, usr_n$. In which case, the knowledge base kb_1 may be inconsistent with kb_2 since they are two views of two different individuals.

   In PML each role has a different view of the world. The way one role views the world need not be totally consistent with the way another role views the world at any given time.

4. Event Calculus has the ability to handle incomplete event information. This feature is implemented by means of the non-monotonic inference rule, negation by failure. If a past event is added to the system not only will any conclusions which follow from the event be derivable but also any conclusions which cease to be derivable because of the event will be non-monotonically withdrawn. Furthermore if the event would seem to be inconsistent with knowledge in the system the existence of an unknown event (or events) will be hypothesised to restore consistency. This viewpoint reflects the view that events in the environment are not strictly controlled by the system. It is opposite to that usually adopted in database systems where an event will only be accepted if it satisfies certain integrity constraints imposed by the system.

   By contrast PML implicitly assumes that there is complete knowledge of the current state and has no counterpart to this non-monotonicity.

5. In Event Calculus there is no order in which the events should happen. By way of relations holding at a given time the possible next event/s according to the knowledge base are known and this information could be used to constrain

future events, e.g. only allowing these events as acceptable. A basic feature of the Event Calculus is, however, the possibility of interpreting an 'impossible' event (according to the knowledge base) as being possible due to the existence of as yet unknown past events. This feature is closely related to the ability of the Event Calculus to handle incomplete information and default reasoning.

In PML only those actions that are active or allowed at a given time can be accepted by the process. Events that could happen at a given time are restricted and controlled by this means, e.g. in the example given in Section 8.2 the making of the client overdrawn is impossible.

8.4 Conclusion

We have compared the two process description frameworks, Process Modelling and Event Calculus, and have shown how both frameworks can be applied to the same example cases. The examples illustrated certain features of each framework.

The bank account example seemed more suited to the PML formalism. But it was also true that the example could be formalised easily in the Event Calculus. In the latter case, however, a relation `next_allowed_tasks`, holding at a given time, was introduced in order to restrict events to those permitted by the bank regulations. The basic effect of this relation is that events in the environment and transactional events are conflated. This is satisfactory for the bank example because events are actions directly on the state of the system. The bank example seems to illustrate how PML is well suited to this sort of process enactment and also that Event Calculus in a constrained form can be used for the same purpose. We can note that this is a prescriptive form of the Event Calculus using forwards persistence. A backwards persistence version would not be satisfactory; for example we could infer that an overdrawn account is in credit simply by adding an event of withdrawal. This further illustrates the point that backwards persistence can cause harm by hiding the need to seek more information before carrying out an action.

Once the basic PML correspondences had been uncovered, an approximate reformulation of a part of our previous Event Calculus representation of the clinical pathway in vascular surgery was fairly straightforward. In our formulation a patient

was represented as a PML role. Notionally we thought of the clinicians as acting in response to a patient's wishes. Another representation which might reasonably be studied would assign the clinicians to roles. Either way the events in this domain can be associated with active agents and therefore fit well into the PML structure of roles.

There is an important limitation in the PML framework which seems to make it unsuited for models such as the vascular surgery domain. Without specific extensions, PML does not keep all information on past states. In contrast a basic feature of the Event Calculus is that it maintains an historical knowledge base of all known past events. This last feature is very important for supporting decisions and explanations.

One thing that PML does particularly well is enforcing normative behaviour in a process. In a clinical domain there is a notion of good practice, for example certain information should always be gathered before undertaking a certain treatment, and this can be well regulated by the action agenda in PML. On the other hand the Event Calculus maintains a full historical knowledge base which can support a decision support system.

In domains such as vascular surgery it must be possible for non-sanctioned actions/events to occur (e.g. based on professional judgement at the time). Furthermore it must be possible for these non-sanctioned events to be entered into the system. This does not present any problem for the Event Calculus since the knowledge base will be non-monotonically adjusted, but for the PML representation it is not clear how to accommodate non-monotonicity. The difficulty and its solution seems to depend on two things: the maintenance or not of a distinction between events in the environment and transactional events and the interplay of this with the notion of normative behaviour.

An important characteristic of a process description framework is that it provides an understandable and maintainable representation of the process. We believe that both Event Calculus and PML have this characteristic. Another important feature of a process description language is the extent to which it supports changes to the model of the process, e.g. in vascular surgery this might be due to scientific developments in the field or changing medical practice. PML provides support for this kind of change whereas the Event Calculus, in the form we have presented, does not. An interesting line of further investigation would be to extend the Event Calculus in this direction.

9 Conclusion

In Artificial Intelligence, models of problem solving require models that can capture change. In planning the management of patients, one must capture the ever changing human physiology of the patient for the model to be effective. Historical knowledge about patient conditions is necessary so as to be able to answer queries about past treatments and also to plan future treatments. The domain of clinical decision support contains typical problems involving time that have to be solved by a temporal reasoning system. Problems such as,

- reasoning about histories of events in the past, to analyse the present condition of the patient,
- the planning of treatment procedures,
- the ability to deal with incomplete and imprecise data.

The main direction of the present research has been to provide temporal reasoning capabilities to a knowledge base. Reasoning about temporal data includes:

- to reason about the period of validity of information available,
- handling dependencies between different temporal data,
- handling incomplete temporal data.

In Chapter 1 we introduced three time concepts that are used in relation to data bases and knowledge bases. Using these concepts we described four types of data bases (knowledge bases), differentiated by their ability to support these time concepts. We defined a deductive data base, our notion of knowledge base, which represents the convergence of data bases and logic programming. We described the aims and the contributions of the thesis. The chapter ended with a brief summary of the organisation of the thesis.

In Chapter 2 we identified appropriate goals of a knowledge based system to support the management of patients in vascular surgery. In order to do so, we first described the clinical pathway that is modelled by the knowledge base, introducing the terminology used in the field. We then discussed the current problems as perceived by clinicians and from these problems we identified the goals to be satisfied by our knowledge based system. The chapter ended with a brief overview of our proposed solution to the problem of building such a system.

In Chapter 3 we have discussed some other related medical information systems concentrating on systems which take into account temporal aspects of the data. First, the functions and increasing importance of the medical record was outlined. The chapter was intended to give an idea of how varied the application domains of computer automated medical information systems are, and how the conventional methods of knowledge representation tends to make retrieval and reasoning about temporal information complex. We concluded that a general temporal framework that could be used in medical knowledge bases was likely to be advantageous.

The main purpose of Chapter 4 was to describe a framework based on the Event Calculus which can be used as the basis of a temporal medical information system. First we introduced the Event Calculus as described by Kowalski and Sergot (Kowalski and Sergot, 1986). Giving simplified examples we introduced the basic notions of Event Calculus: relations initiated by events holding true in given time intervals; how the explicit treatment of events allows for updates that provide new information about the past; how default reasoning is obtained on the basis of incomplete information. By assuming persistence of relations forwards in time alone we arrived at a simplified form of the original Event Calculus suitable for our domain. The idea that it may be unreasonable to assume that some relations persist infinitely into the future, if it is not known of any event terminating that relation, was accommodated by allocating a life-time to such relations. The introduction of this extension completed the simple and flexible framework used by us for temporal support of a medical knowledge base.

In Chapter 5 we have shown how the patient information can be structured to suit the Event Calculus framework in a natural way. In particular we have shown how this representation can be used to: input, as event instances, the information currently kept on the patient notes for a relatively complete sub-domain of vascular surgery (the

arterial part); access this information in a flexible way; and model some aspects of the timing of clinical tests and treatments.

We have left the diagnostic problem to the clinician and have concerned ourselves only with providing decision support. We have argued that some kind of temporal reasoning is needed to model clinical practice in this domain and that our approach, because it makes a clear separation between general temporal reasoning and the domain model, has advantages for construction and extension of the system. In particular in Chapter 6 we described how a modifiable and extensible system of rules for generating recommended options at any given time can be built on the basic historical knowledge base. We also indicated the way in which the system generated recommendations provide decision support to the clinician.

Chapter 6 also dealt with two specific problems: the variable periods of reliability of information typical of the medical domain; and the use of ramifications for knowledge representation. When data (or information) is too old to be reliable, then a new measurement is called for. Determining the optimal time to request new data is another important problem for time-based systems. We have dealt with this problem by assigning a default life-time to such time-dependent relations. The system assigned default value can be overridden by the clinician by assigning a `rely_time` to the event itself. As regards the second problem, we noted that knowledge representation with ramifications has advantages and disadvantages. Ramifications provide a very simple and elegant way of representing knowledge. But if we were to strictly adhere to a policy of representing recommendations only by ramifications we have to be prepared to sacrifice other aspects of knowledge representation such as simplicity. This is clear in the case of recommending the next task to be done. This dichotomy of behaviour make it unsuitable to be adopted as the only representation method. Whether we represent a particular property by a ramification or by a base relation is a practical issue. We take a pragmatic approach to this problem and when the situation warrants use both methods to represent a property.

We have implemented a temporal knowledge based system for the clinical pathway in vascular surgery in the Event Calculus framework. The prototype system is intended to be a research workbench rather than a fully operational system that could be used by the clinicians. Chapter 7 describes this system: how patient data is

stored, the user interface, the query system which provides the means of accessing the knowledge base and requesting explanations. We illustrated through case studies how the Event Calculus approach provides a good supporting basis for temporal reasoning for the domain of vascular surgery considered in realistic detail and how the system successfully caters for decision support functionalities described in Chapter 6. The results of this chapter underlines our main conclusion: that a system with general temporal reasoning provided by the Event Calculus has many of the promising characteristics we aiming for.

Event Calculus describes processes within a logic programming framework. In Chapter 8, we compared and contrasted the Event Calculus with another more procedural approach to process description, Process Modelling in the language PML. We introduced Process Modelling and the language PML, and proposed suitable definitions for some of the terminology used relevant to Process Modelling concepts and PML. In order to investigate the relationship between Event Calculus and PML, we considered two example processes formalising each as an executable program in each of the description languages. The first example was a very simple interactive bank account as described in Chapter 8. The second was a model of a part of the vascular surgery domain, the subject of this thesis; comparison of how PML describes this process has been reported elsewhere (Abeysinghe and Soper, 1993). We concluded that the strength of Process Modelling is twofold: to enforce process enactment; and to support the incorporation of a changing environment (e.g. due to technological change) into the model. On the other hand the strengths of Event Calculus rely on its generality: a rigorous distinction between system and environment can be maintained allowing for incomplete past knowledge; Process Modelling concepts such as role can be seen as useful special cases of Event Calculus concepts; and the existence of a historical knowledge base allows greater support for decisions and explanations. As regards the vascular surgery domain we concluded that the flexibility of the Event Calculus was needed.

In the remainder of our concluding chapter we indicate some of the future directions to progress our research. One area concerns negation by failure. On the positive side negation by failure plays an essential role in Event Calculus. As a consequence of negation by failure, default reasoning is obtained on the basis of

incomplete information. This characteristic is especially useful in the role of decision support. Clinicians often have to depend on incomplete data, and information about clinical events may not reach the decision maker in the same order as they occur in the real world. The flexibility of the Event Calculus framework, allowing information gathering to occur in any order irrespective of the order in which events actually happen, further enhances its suitability. If the default assumptions are made inconsistent on the arrival of new information, then they are automatically and non-monotonically withdrawn.

This is not to say that Event Calculus is problem free. The negation by failure rule as well as providing flexibility to the framework, also has its disadvantages. Pinto and Reiter (Pinto and Reiter, 1993) discuss some of the disadvantages caused by negation by failure. Negation by failure assumes every unknown fact to be false. Thus it is incapable of recognising incomplete data. There is ongoing research on how to address this point. One line of investigation is to use "constructive negation" (Chan, 1988). Another line (Denecker, Missiaen and Bruynooghe, 1992) combines the Event Calculus with abduction. The ways in which the Event Calculus might be enhanced with a more satisfactory treatment of defeasible reasoning is a promising area for future research.

In Event Calculus it is the temporal order of events that is important but not the exact event times. Thus events with unknown event times and partially ordered events can be represented. Although in our system we require the event time to be input it is not essential to do so. By the use of temporal relations such as 'after' and 'before' we can implement the partial order of events.

In the domain of vascular surgery there are no rigid protocols as in some other clinical domains such as oncology. There is a possibility of any event happening at any time. The flexibility of the Event Calculus framework suits very well such domains. This does not imply that Event Calculus is incapable of handling domains with rigid protocols. The introduction of integrity constraints to control the sequence of events received might be one way of achieving protocols. It will be an interesting area to pursue in future.

Whatever the application of a computer system is, the data available should be of high quality, especially when that data is used in *safety critical* systems such as hospital information systems (Krause, Fox, O'Neil and Glowinski, 1993) and medical

decision support systems. Usually, the storing of data involves many steps. In medical information systems the medical staff (nurses, clinicians, etc.) record information on forms, clerical staff input this data into the computer in whatever format necessary. This usually results in incorrect data being entered into the system. One way of reducing the error rate is by reducing the number of steps or the intermediate personnel involved. In our domain a major advance would be to persuade clinicians to enter data directly. This is highly desirable because much of the data is of highly specialised nature, and it would also cut down on inaccuracies. An easy method of input and friendly user interfaces will definitely add to positive persuasion. But perhaps the most important inducement is that the clinicians should perceive clear benefits in using the system.

There is increasing pressure to justify the efficiency of resource allocation in medical, and in particular hospital, practice. This is resulting in increased demand, from both clinicians and managers, for 'high quality' information concerning resource decisions (such as whether to change a treatment regime), for example for information on: outcomes of treatment; comparison between treatments; quality of results. Much of this 'high quality' information can only be collected by clinicians and others directly concerned with patient care, and at present procedures for doing this are ad hoc. We see the role of our system in this area as two fold. Firstly, as a Trojan horse: clinicians who wish to 'play' with it will (a) have to enter clinical data, and (b) have to do so in a disciplined way. Secondly, with the data obtained, the system clearly has potential to be tailored to perform various types of audit. Audit is a mechanism by way of which specific aspects of clinical work can be examined. It is a valuable tool for quality assurance, assessing the effectiveness of a particular treatment, and to review clinical work for further research. There is demand for routine clinical auditing which provides comprehensive analysis of a unit's activities (Campbell, Souter, Collin, Wood, Kidson and Morris, 1987; Castleden, Lawrence-Brown, Lam, McLoughlin, Thompson and Lopez 1988; Stoodley and Sikorski, 1991).

The primitive temporal notion in our formalism is the time point. This leaves the possibility of extending the formalisation to include different granularities of time (Evans, 1989; Evans, 1990; Wiederhold, Jajodia and Litwin, 1991). This leads naturally to the consideration of compound events which could provide a powerful knowledge

modelling framework. This is an interesting area for further research.

There is a great deal of interest today in developing new tools for answer justification (Wolstenholme, 1992a; Wolstenholme, 1992b; Reggia, Perricone, Nau and Peng, 1985a; Reggia, Perricone, Nau and Peng, 1985b). We have provided the 'why' and 'whynot' justification to system conclusions. The justification facility is not very sophisticated. Especially the 'whynot' justification can be made more complete by accommodating constructive negation.

There is a growing body of evidence to suggest that computer-aided decision making is superior to more traditional approaches (Hlatky, Califf, Harrell, Lee, Mark, Muhlbaier and Pryor, 1990). Clinicians will continue to face decisions about choice of therapy, but there will be better information in the future to guide such decisions. Decision support systems, such as knowledge based systems are under development in many areas of specialist medicine. Decision making such as diagnosis, has been the main focus of research in many medical information systems (Fox, Glowinski and O'Neil, 1989; Fox, 1989; Fox, 1992). Our aim has been to support the diagnosing process but not to take the control of the diagnosing task. This we have left to the human expert. We allow the clinician to make the decision, but aid him in providing the relevant information.

There are many other directions in which the system might be further developed, including: for clinical research tools; to provide information retrieval from a medical text base for educational purposes; and by linking to medical systems in other domains such as the Oxford System of Medicine for primary care (Glowinski, O'Neil and Fox, 1989a, Fox, 1989; Gordon, Fox, Glowinski and O'Neil, 1990; Fox, 1992).

We do not claim to have built a fully operational system suitable for installation in a hospital. But a substantial part of such a system has been built in order to prove that it is viable to build such a system in the domain of vascular surgery using Event Calculus.

A The Current Patient Record

VASCULAR PATIENT RECORD

PATIENT DETAILS

Surname:
Given Name:
Date of Birth:
Unit Number:

Address: (Two addresses if possible)

Date: Time: Consultant: Emergency Urgent Rout

Sex: M F Weight: Height:

PRESENTING COMPLAINT

Is the patient Asymptomatic for vascular symptoms? N Y ———

Does the patient complain of Intermittent Claudication? Y N ———

Which leg is affected: Right Left Both: R=L R>L L

Principle Site- Right: Foot Calf Thigh Buttock
Principle Site - Left: Foot Calf Thigh Buttock

Duration of symptoms:

Claudication distance: In home Shopping Social (eg. Gol

Associated sensory upset? None Tingling Numbness
Progression of symptoms? Better Same Worse

Additional information about claudication:

Is the patient suffering with Rest Pain? Y N ———

Which leg is affected: Right Left Both: R=L R>L L

Was the onset of pain: Gradual Sudden

How long affected? Hours Weeks Months

Associated sensory upset? None Tingling Numbness
Progression of symptoms? Better Same Worse

Is the patients affected by Ulceration or Gangrene? Y N ———

Which leg is affected: Right Left Both: R=L R>L L

Principle Site- Right: Toe Digits Forefoot Heal Ankle Shin BK
Principle Site - Left: Toe Digits Forefoot Heal Ankle Shin BK

How long a problem? (Months)

Progression of symptoms? Better Same Worse

Is sleep disturbed by limb pain? Y N
Is a limb ever hung out of bed? Y N
Does the patient sleep in chair at night? Y N

Has the patient suffered a Transient neurological event? Y N ———

How often: Once only Twice Multiple

Are the event: Unifocal Multifocal

Side affected: Right Left Both

Does amaurosis fugax occur: No Yes: Right Left Both

Does the patient suffer with any back or abdominal pain? Y N ———

Do you think that this is related to an Aneurysm? Y N

Duration pain a problem? Hours Days Weeks Month

Document any other details about the Presenting complaint:

YOU MAY HAVE TO REFER TO THE EXISTING NOTES FOR SOME OF THIS INFORMATION

PAST MEDICAL HISTORY

Previous MI	N	Y	Date most recent:		Total No:
Hypertension	N	Y			
Diabetes	N	Y	Diet	Oral	Insulin
			Year diagnosed:		
Rheumatic fever	N	Y			
TB	N	Y			
Jaundice	N	Y	Hep A	Hep B	Details:
Asthma	N	Y			
Chronic bronchitis	N	Y	How many years:		
DVT	N	Y	Left	Right	Both
Renal failure	N	Y			
CVA	N	Y	Residual deficit:	Right	Left Non

Any other major illness?

PAST SURGICAL HISTORY

Previous Vascular Surgery. Y N ———

Date	Procedure	Reason	TopVessel	Side	Lower-R	Lower-L	Graft	Outcome
EXAMPLE	RECON	ANEUR	AORTA		C-ILIAC	C-FEM	DACRON	PATENT

Procedures	Reason	Vessels	Grafts	Outcome
TLD ENDARTERECTOMY	OCCLUSIVE DIS	CAROTID AXILLARY AORTA SIF-AORTA C-ILIAC	DACRON EES	PATENT
RECONSTRUCTION ENDOLECTOMY	ANEURYSMAL DIS	Ext-ILIAC C-FEMORAL FEM-BIFURCATION SFA	PTFE INSITU-VEIN	OCCLUDED
ANGIOPLASTY(operative)	OTHER	AK-POP BK-POP TP-TRUNK AT PT PERONEAL	REVERSED-VEIN UMBILICAL	UNKNOWN

Other surgery

Cardiac	N	Y		Valvular	CABG	Other	Date:
Varicose Veins	N	Y		Right	Left	Both	
Sympathectomy	N	Y		Right	Left	Both	
Amputation	N	Y	R:	Toes	Foot	BK	AK
			L:	Toes	Foot	BK	AK

Op for Malignancy N Y Disease free period: Months
Details:

Any other surgery? Other details of PMH and PSH record below:

SYSTEMATIC ENQUIRY

Angina	N	Y	Exertional	At rest		
	Frequency:		Daily+	Daily	Weekly	Occasionally
Dyspnoea	N	Y	On stairs	On flat	At rest	
Palpitations	N	Y				
Orthopnoea	N	Y	Number of pillows at night:			
PND	N	Y				
Ankle swelling	N	Y				
Climb stairs	Y	N	Without stop	Stops on way up		
Cough	N	Y	No sputum	White	Coloured	
Wheeze	N	Y				
Haemoptysis	N	Y	Diagnosed cause		Undiagnosed	
Weight steady	Y	N	Increased	Decreased: Amount	Kg	
Dysphagia	N	Y				
Bowel Habit changed	N	Y	Diagnosed cause		Undiagnosed	
Rectal Bleeding	N	Y	Diagnosed cause		Undiagnosed	
Urinary symptoms	N	Y	Details:			
Nocturia	N	Y	Frequency:			
Haematuria	N	Y	Diagnosed cause		Undiagnosed	
TIA's	N	Y	Focal	Non Focal		
Amaurosis fugax	N	Y				

Record any other details from systemic enquiry below.

ALLERGIES & DRUGS

Allergies ~ N Y Details:

		DOSE	UNIT	FREQUENCY
Aspirin	N Y			
Nifederpine	N Y			
β-Blocker	N Y			
Steriods	N Y			

NAME

Others: 1.
2.
3.
4.

SOCIAL HISTORY

Has Pt ever Smoked	N	Y	< 10 day > 10 day
Pt stopped smoking	N	Y	Weeks Months Years (Ago)
Alcohol	N	Y	Unit week:
Hobbies	Y	N	Details:
Marital status			Single Married Widowed Separated
Children	N	Y	Local In UK Abroad
Accommodation:			House Flat/Bungalow Part 3 Rest-Ho Nursing Home Other
Lives alone	N	Y	
Has to climb stairs	N	Y	
Does own shopping	N	Y	
Employment. most recent:			
At present:			Working Off sick Unemployed Retired
sick/Unemployed, how long:			Weeks Months Years

FAMILY HISTORY

MI's	N	Y	Grand P	Parent	Sibling	Age:
CVA	N	Y	Grand P	Parent	Sibling	Age:
Arterial disease	N	Y	Grand P	Parent	Sibling	Age:
Diabetes	N	Y	Grand P	Parent	Sibling	Age:
Amputation	N	Y	Grand P	Parent	Sibling	Age:

EXAMINATION

Anaemic	N	Y	Clubbing	N	Y
Jaundice	N	Y	Cyanosis	N	Y
Lymphadenopathy	N	Y	Dyspnoea	N	Y
Breast:	N/A	NAD	Detail?		

CVS

Pulse rate B/min: Rythum: Regular Irregular Ectopics

BP Right arm: mm/Hg BP Left arm: mm/Hg

JVP	Not seen/NAD		Raised	
Apex beat displaced	N	Y	Not felt	
Thrill	N	Y		
Heart Sounds	I	II	I	(tick if present)
Added H sounds	N	Y		
Murmurs	N	Y	Detail:	

	LEFT		RIGHT		KEY	
	Pulse	Bruit	Pulse	Bruit		
Carotids					A	Amputated
Subclavian	-		-		o	No Bruit
Radials		-		-	I	Soft bruit
Renal	-		-		II	Harsh brui
Iliac	-		-			
Femoral					-	Absent
Popliteal					+	Reduced
Ant Tibial		-		-	++	Normal
Post Tibial		-		-	+++	Increased
PT/Brachial					Doppler Pressures	
DP/Brachial					Index	

Ulceration foot	N	Y	Right	Left	Both
Ulceration Ankle	N	Y	Right	Left	Both
Gangrene Toes	N	Y	Right	Left	Both
Gangrene Foot	N	Y	Right	Left	Both

CHEST

Chest clear	Y	N	Details:
Air entry equal	Y	N	
Breath sounds OK	Y	N	

ABDOMEN

Aneurysm	N	Y	Details:
Masses	N	Y	
Scars	N	Y	
Organomegally	N	Y	
Genitalia NAD	Y	N	

NEUROLOGICAL

			Right	Left	Details:
Vision Ok	Y	N			
Cranial N intact	Y	N			
Motor system	Arms Power:				
	Legs power:				
Sensory system OK	Arms				
	Trunk				
	legs				
Co-ordination					

	Radial	Brachial	Ulnar	Quads	Achill	Planta
Reflexes: Right						
Left						

Record any other information below:

Primary problem

Secondary problems

1. 5.
2. 6.
3. 7.
4. 8.

MANAGEMENT PLAN

INVESTIGATIONS ORDERED

Details:

FBC
U&E
CXR
ECG
G&S
X-match
Angio
Duplex
U/S

PRINT YOUR NAME & SIGN

DATE	TIME	DECISION

EVENTS & TREATMENTS

B User Interface

B.1 The Main Menu

The menu selection **Program Options** in menu bar provides a pull down menu containing the menu options available to the system user. The user can select the type of session by choosing a single or a combination of choices from this pull-down menu as shown in Figure B.1. The first choice has to be **Select Patient**.

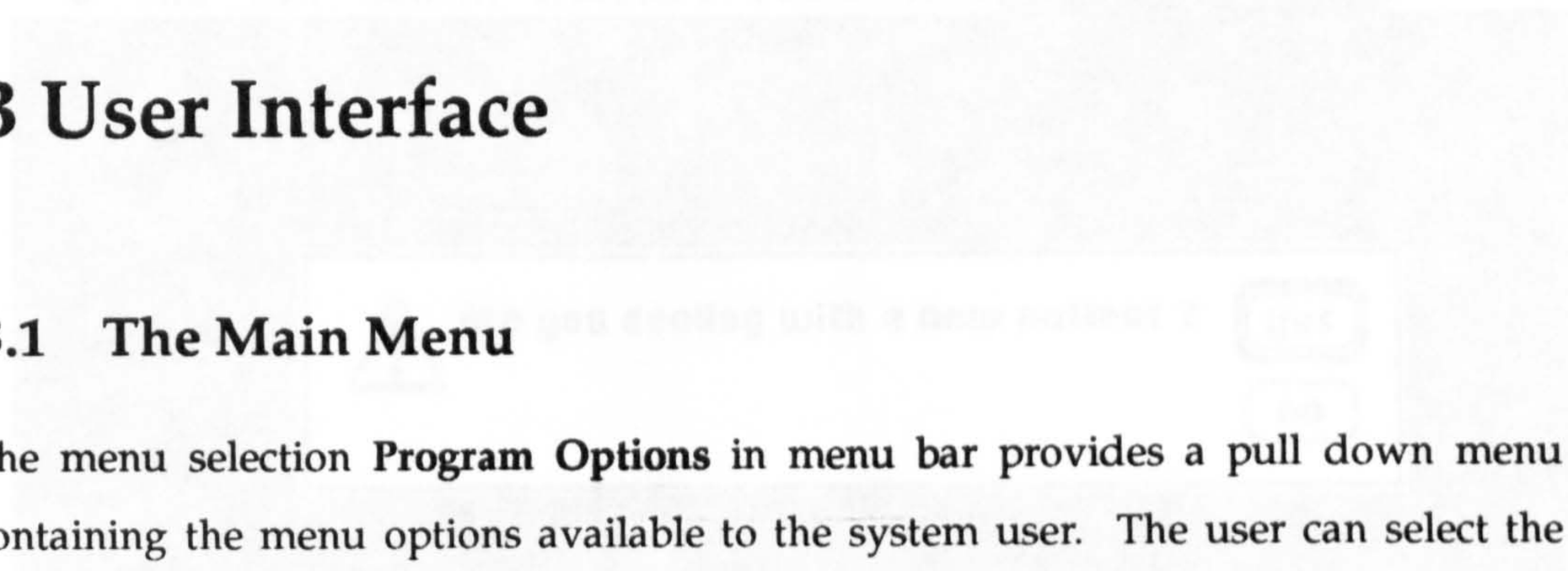


Figure B.1: The main pull-down menu of the user interface, which allows the user to select his/her action

B.2 Selecting a Patient

When the option **Select Patient** is chosen from the main menu, the user is asked whether s/he wants to process a newpatient or a patient whose information is already in the knowledge base (Figure B.2).

If the user answers **NO**, then a scroll-menu of names of all the patients whose data are on the knowledge base is displayed. The user can select the patient by clicking on the patient's name, and then clicking on the **OK** button or pressing the **Return** key (see

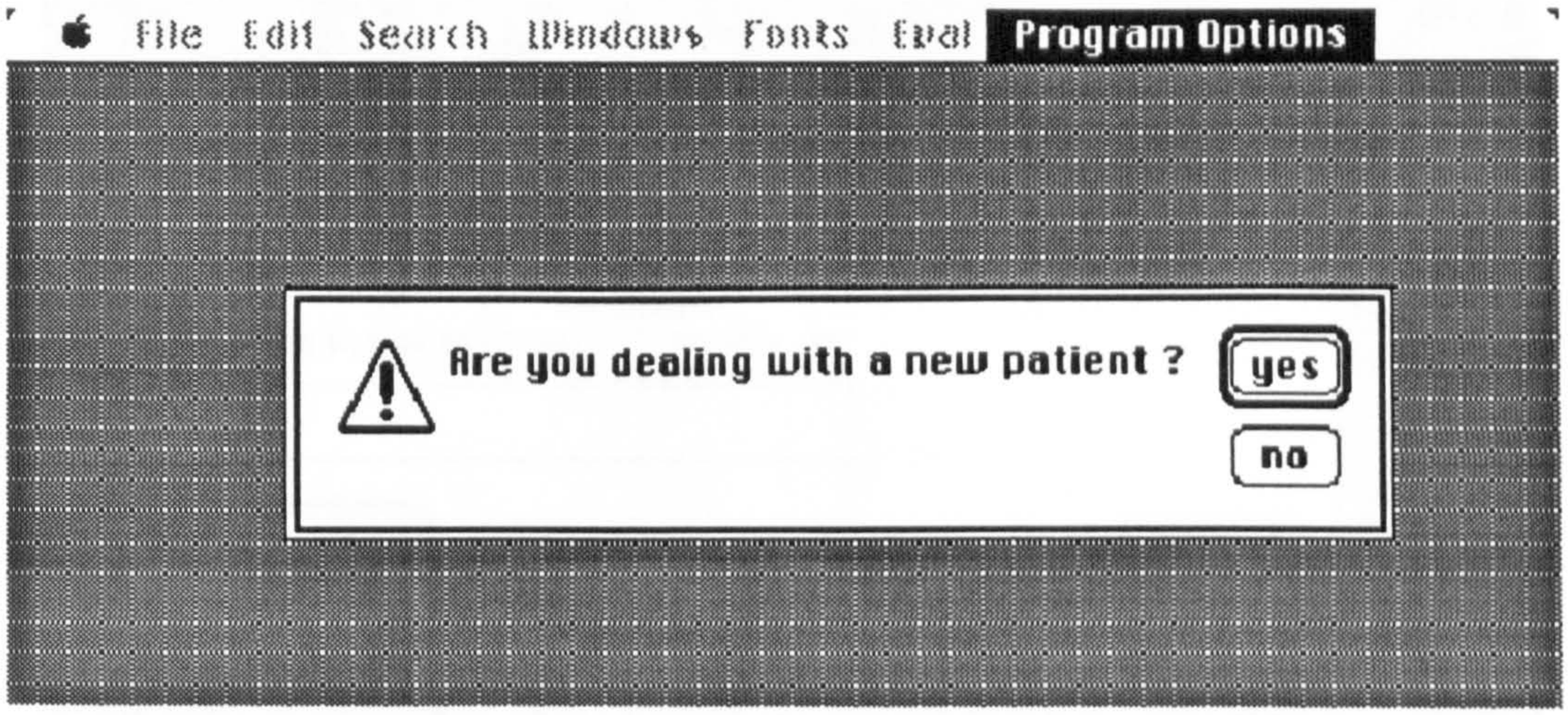


Figure B.2: Inquire as to whether the user wishes to process a newpatient or not

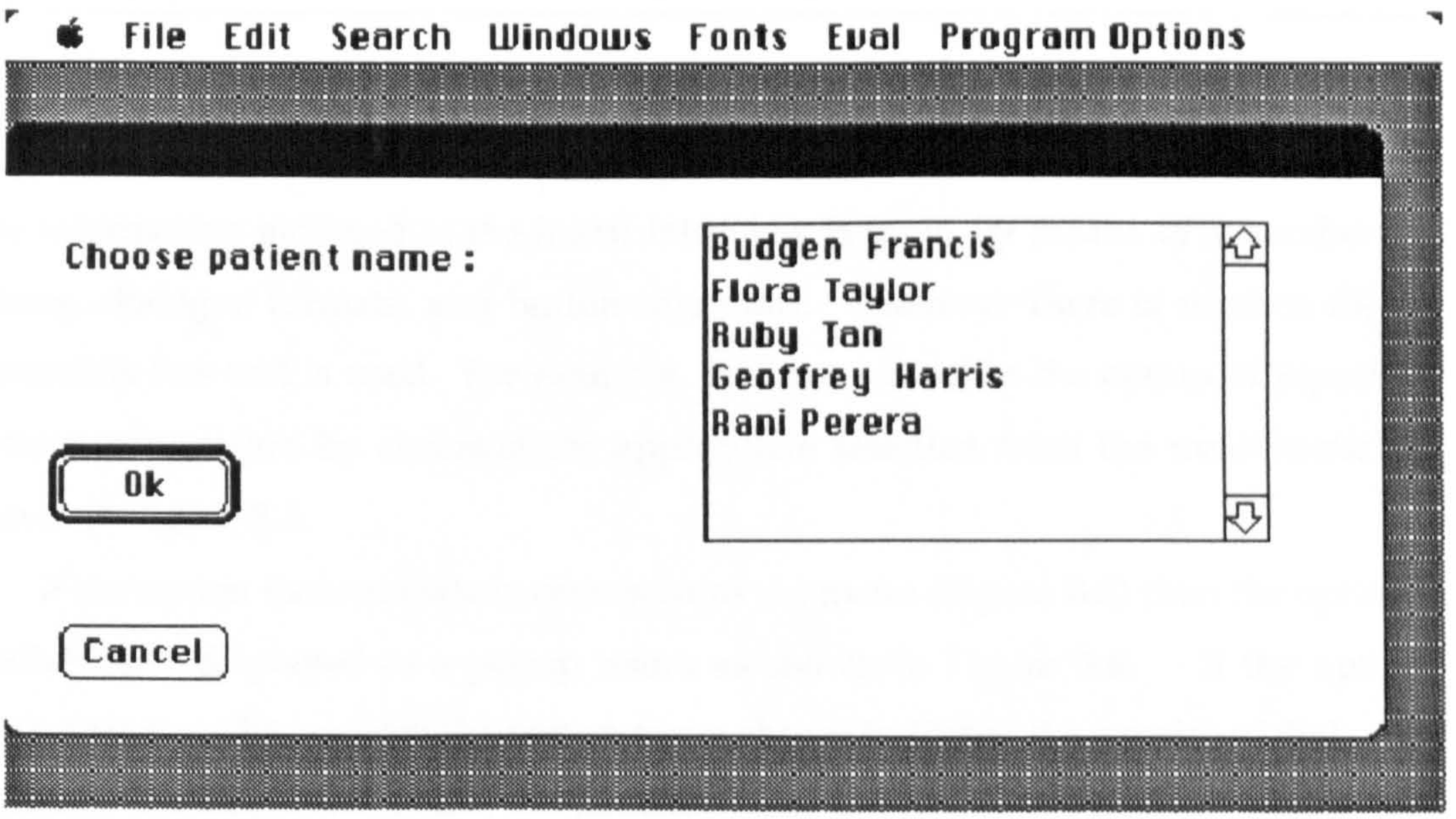


Figure B.3: A patient is selected by clicking on the patient's name from the scroll-menu

Figure B.3). If the answer to the question in Figure B.2 is **Yes**, then the user is asked to input the patient's name (Figure B.4). The name input here is used to create the data files relevant to that patient (described in Chapter 7 Section 7.1).

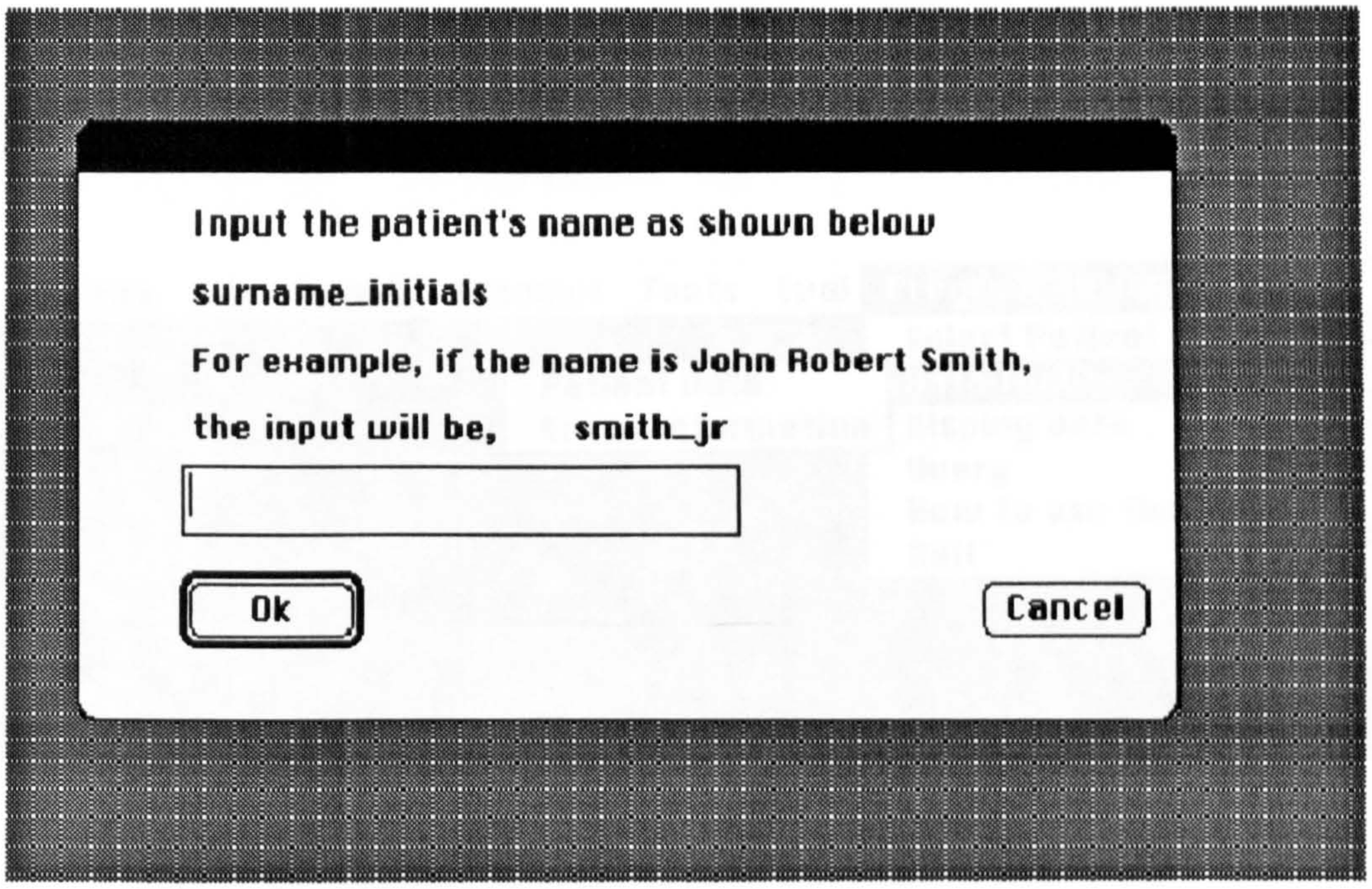


Figure B.4: The patient's name is input if the patient is new to the data base.

B.3 Input Data

The information gathered at the initial interview is input by means of a number of menus, dialogue formats, and button selections. Wherever there is need to fill in comments free text is used. For example, the user can select the option of inputing patient information by choosing the appropriate selection from the main menu as shown in Figure B.5.

If the option **Patient Data** is chosen from the menu (Figure B.5) then the options available are displayed as a popup menu as shown in Figure B.6. If the option **basic data** was chosen from the popup menu shown in Figure B.6 a series of dialogue windows are displayed where, the patient's basic information is read in. Figure B.7 shows one such window.

If instead of **basic data**, **ailments** is chosen from the popup menu (Figure B.6) then the dialogue window shown in Figure B.8 will be displayed. One or more complaints can be selected from the scroll menu displayed.

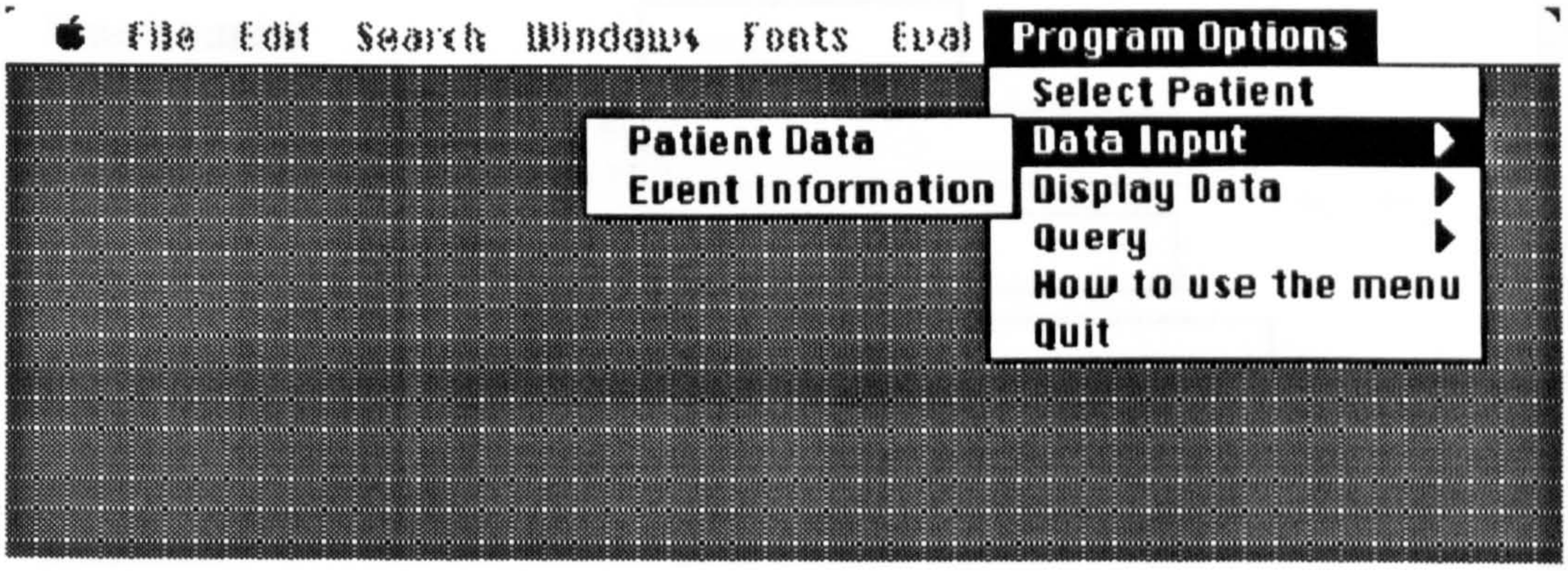


Figure B.5: Selecting the option to input data from the main menu

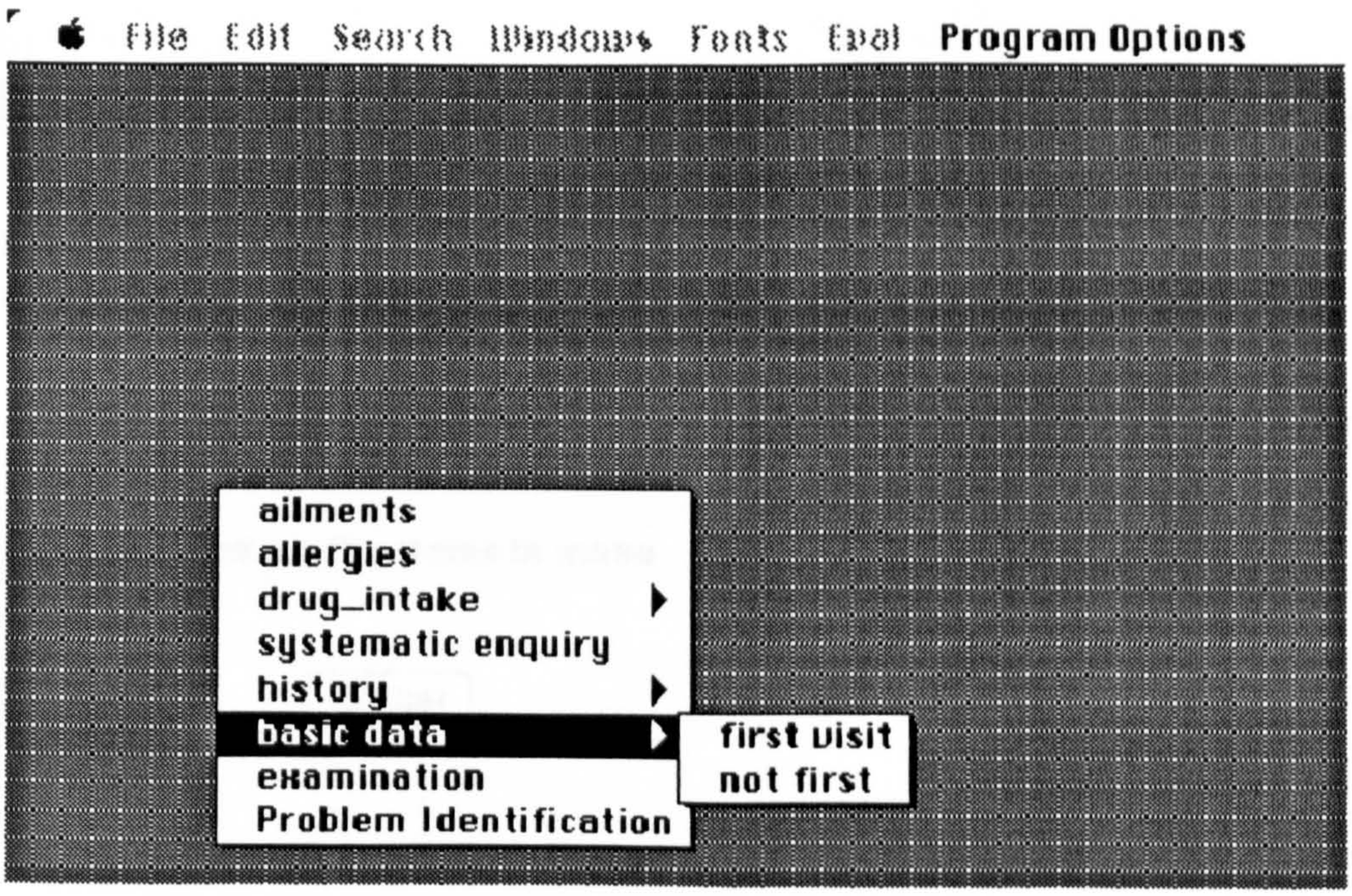


Figure B.6: Options for inputting patient information

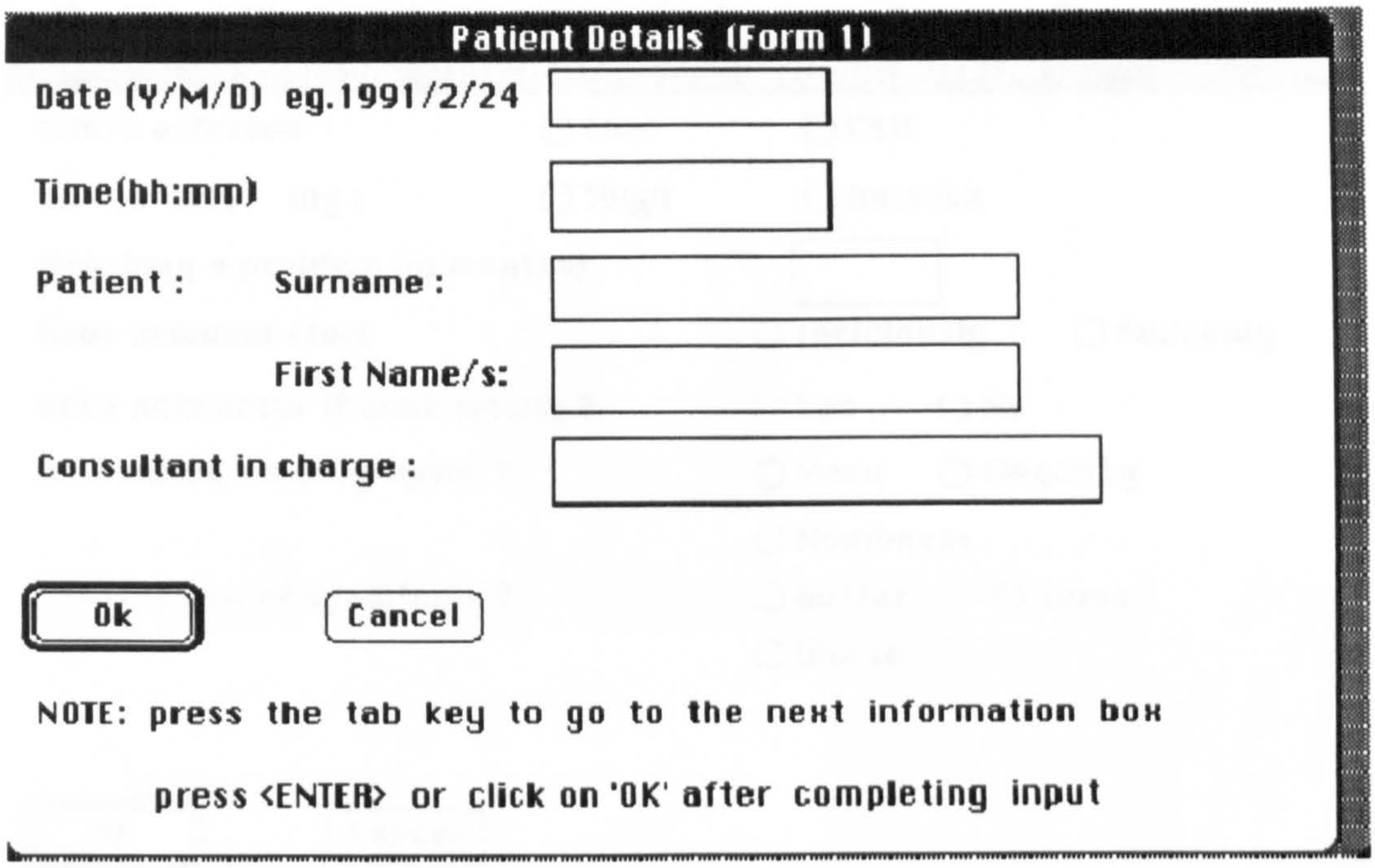


Figure B.7: The first dialogue window displayed for inputting patient's basic data

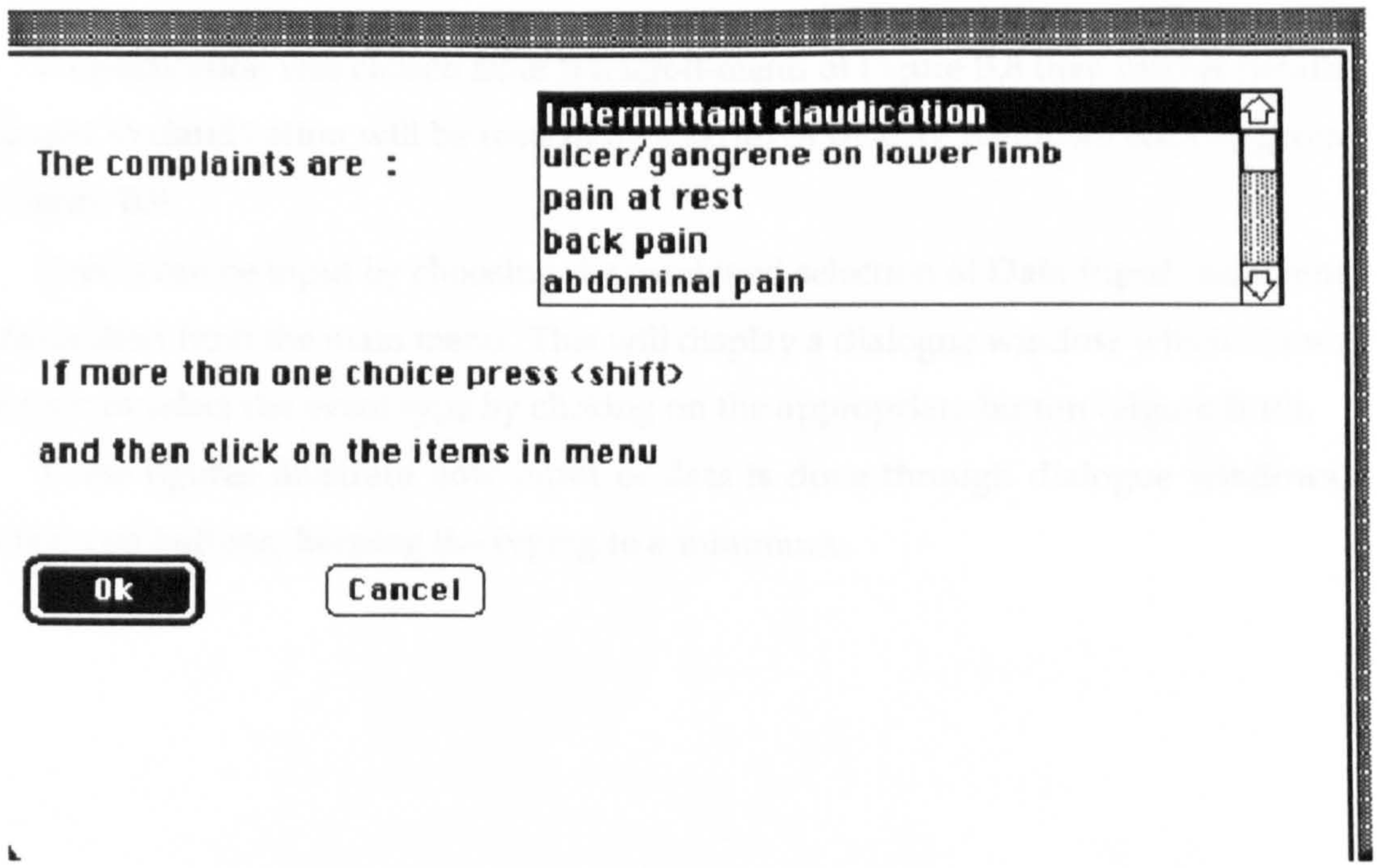


Figure B.8: The window for inputing symptoms of the patient

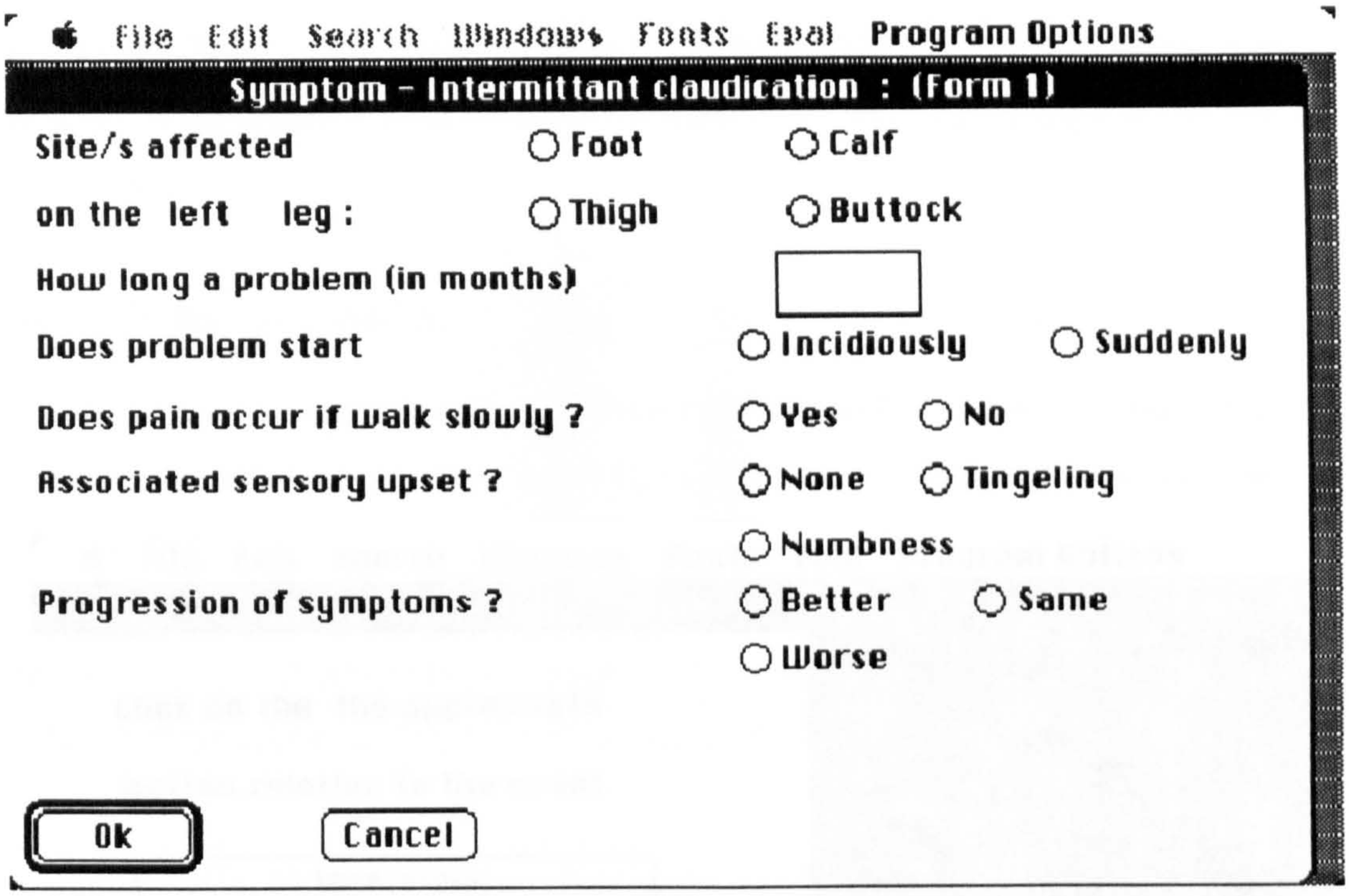


Figure B.9: Display window requesting further details of the ailment **intermittent claudication**

If claudication was chosen from the scroll-menu of Figure B.8 then further details relevant to claudication will be read in by a series of display windows such as given in Figure B.9.

Events can be input by choosing the combined selection of **Data Input** and **Event Information** from the main menu. This will display a dialogue window which allows the user to select the event type by clicking on the appropriate button (Figure B.10).

These figures illustrate how input of data is done through dialogue windows, menus and buttons, keeping the typing to a minimum.

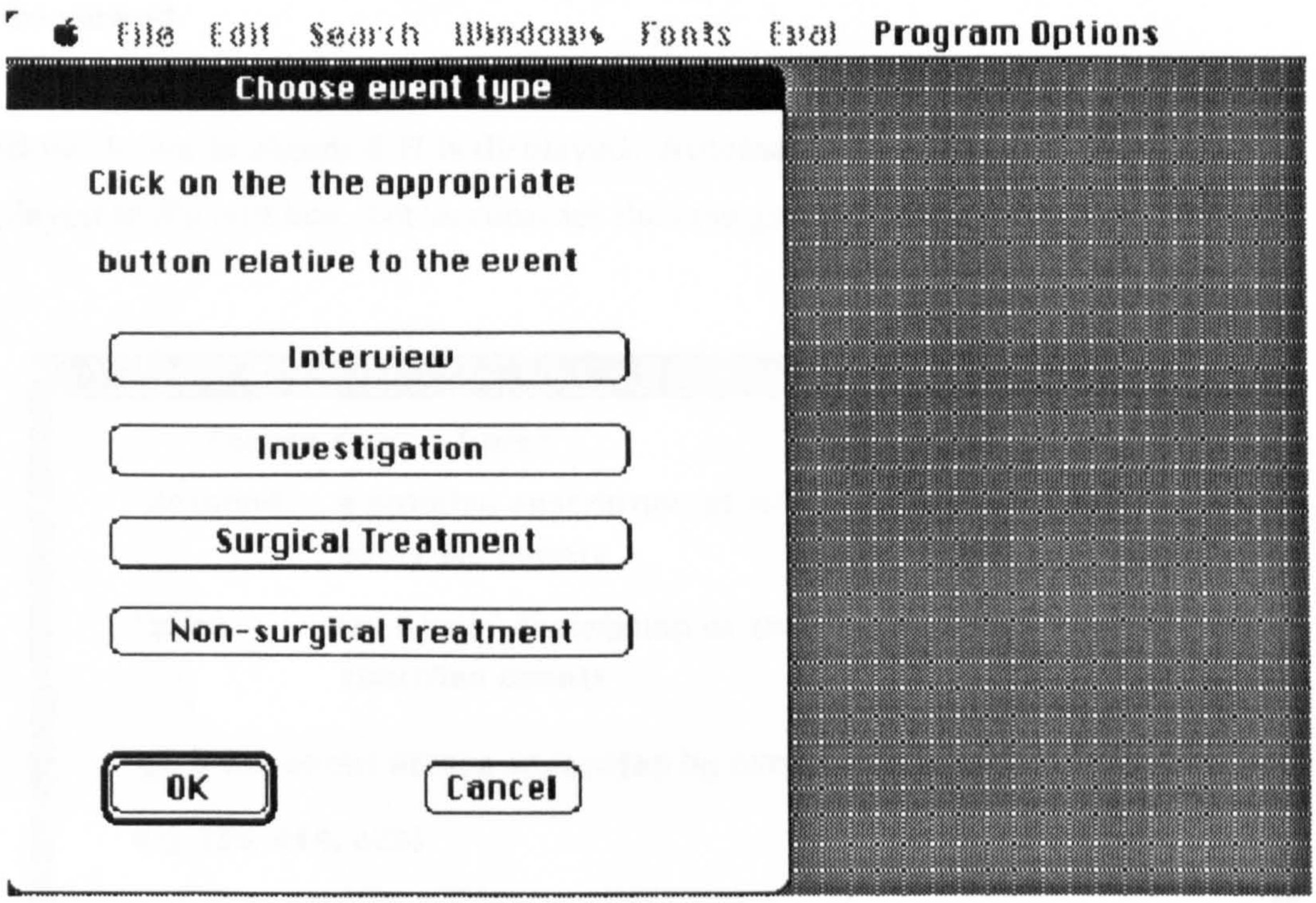


Figure B.10: Display window for selecting the type of the event to be input

B.4 Display Data

The **Display Data** option of the main menu (Figure B.1) has two options: **Event Information** and **Patient Information**. The output of these two options are shown in this section.

Display Event Information

This option allows the user to view information stored as event descriptions in the knowledge base. Either a brief or a detailed description of any set of selected events can be viewed.

On choosing the option **Event Information** from the main menu the dialogue window shown in Figure B.11 is displayed. Automatically a list of all events will be displayed in the edit box. Let us consider the case given in Case Study 1 of Chapter 7

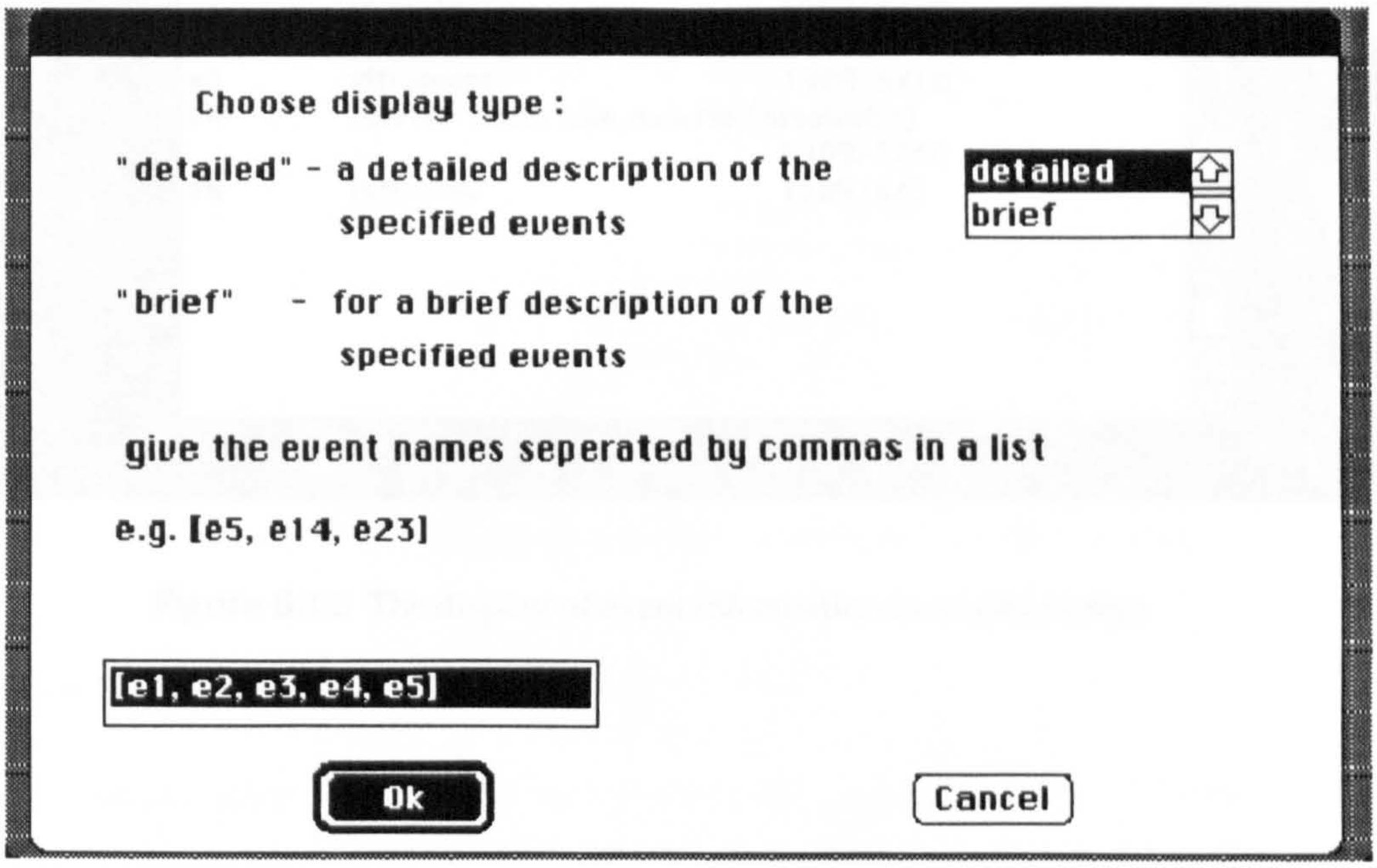


Figure B.11: The dialogue window to select the events desired to be viewed

Section 7.2. Event descriptions for five events are in the knowledge base. A list of the event names will be displayed in the edit box. If the user selects the option **brief** for the set containg all events then the display is as shown in Figure B.12.

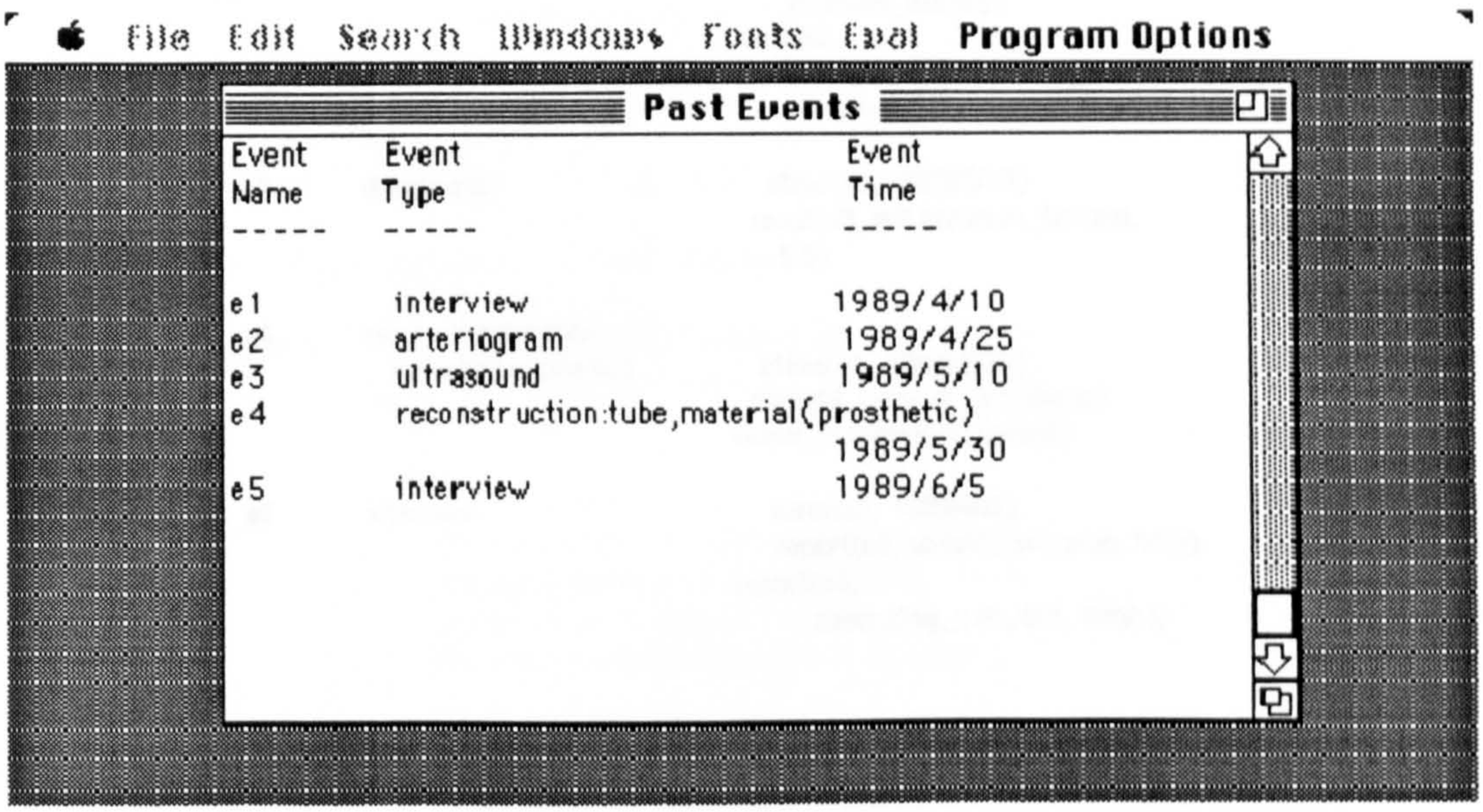


Figure B.12: The display of event information in a brief format

If the option **detailed** was selected from the display of Figure B.11 for all events in the knowledge base, then the display is shown in Figure B.13.

Past Events

Event Name	Event Type	Event Description
e1	interview	etime(e1, 1989/4/10) report(e1, leg_pain, left, thigh, 3) report(e1, rest_pain, left, 1) result(e1, occlusive_arterial)
e2	arteriogram	etime(e2, 1989/4/25) result(e2, left,common_femoral, localised_dilatory) result(e2, left,upper_superficial_femoral, localised_atherogenic_arterial)
e3	ultrasound	etime(e3, 1989/5/10) result(e3, left,common_femoral, 5.5)
e4	reconstruction:tube material(prosthetic)	etime(e4, 1989/5/30) material_used(e4, prosthetic) vessel_outcome(e4, patent)
e5	interview	etime(e5, 1989/6/5) report(e5, absent(rest_pain, left)) report(e5, absent(leg_pain, left, thigh))

Figure B.13: The display of event information in a detailed format

Display Patient Information

This option allows the user to view the text file containg patient information. For example if information was viewed after the initial interview in the Case Study 1 of Chapter 7 Section 7.2 then the display would be as shown in Figure B.14.

Patient Name: Flora Taylor
The date is: 1989/5/23 The Time: 10:13
Consultant in charge: T . C . Dodds

GP : H . P . Frost

Urgency of Consultation/Admission : Routien
Patient Status : In-Patient

Sex : female
Date of Birth: 1950/6/11
Occupation: house wife
Address: 34 Ramya Road ,
Southanpton S032 5 XY
Telephone Number: 782 562234

Weight of the patient: 7:st 3:lbs

Height of the patient: 5:feet 6:inches

Symptom : Intermittant claudication on left thigh
The problem has been 3 months long
The progression of symptoms is Same .

The problem starts Incidiously
The associated sensory upset is None and
The progression of symptoms is Same
Duration of the problem is 20 minutes
The claudication distance : Shopping

Sleep is not disturbed by limb pain
The limb is never hung out of bed
The patient does not sleeps in chair at night

Symptom : Pain at rest in left lower limb
The onset was Sudden and progression of symptoms Same
The associated sensory upset is None .

Sleep is not disturbed by limb pain
The limb is never hung out of bed
The patient does not sleeps in chair at night

Figure B.14: Part of the display of patient information after the initial interview in the Case Study 1 of Chapter 7 Section 7.2

B.5 Help Facility

The option **How to use the menu** of the main menu (Figure B.1) is a help facility. Here we make use of the predefined MacPROLOG predicate `browse` which allows the user to 'browse' through a structured text file. On choosing this option a dialogue with a scrolling text field on the left, and a scrolling menu on the right is displayed (Figure B.15). The scrolling menu contains all options available in the main pulldown menu **Program Options**, and the scrolling text contains a brief description of each option. This dialogue is shown in Figure B.15.

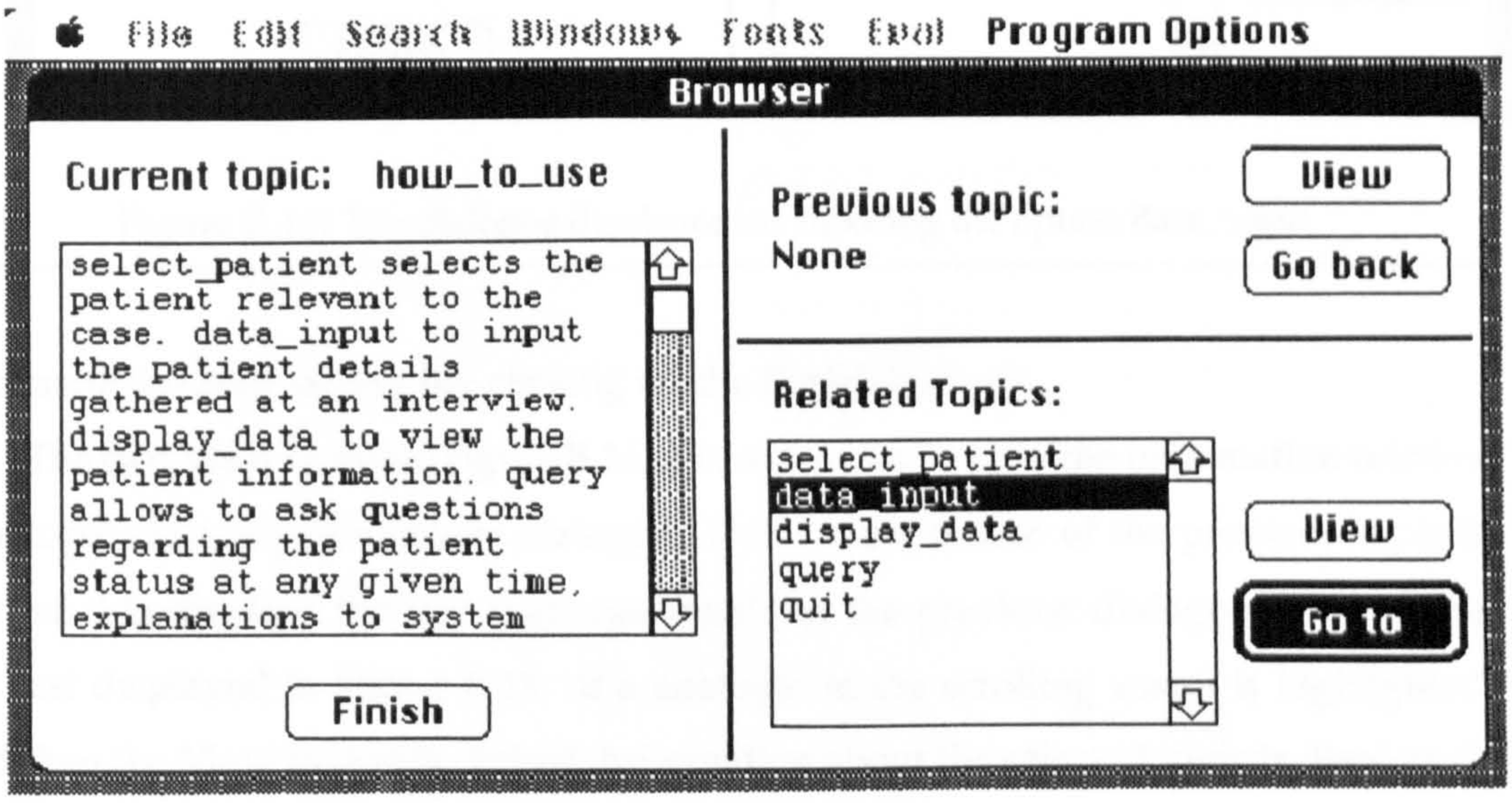


Figure B.15: The dialogue displayed on choosing the option **How to use the menu**

Detailed information of any of the topics in the scrolling menu can be obtained by highlighting the required option and clicking on the **Go to** button. If the option **data_input** is selected, the change in dialogue is shown in Figure B.16. The scrolling text of the new display contains the information about the selected topic, `data input`. If the option selected has any subtopics for which help is provided, the scrolling menu will contain those subtopics. The subtopics of the main menu option **Data Input** now appears on the scrolling menu (Figure B.16). The user may pursue information about any of the subtopics (by highlighting the required subtopic and clicking on the **Go to** button), go back to the previous dialogue (by clicking on the **Go back** button), or

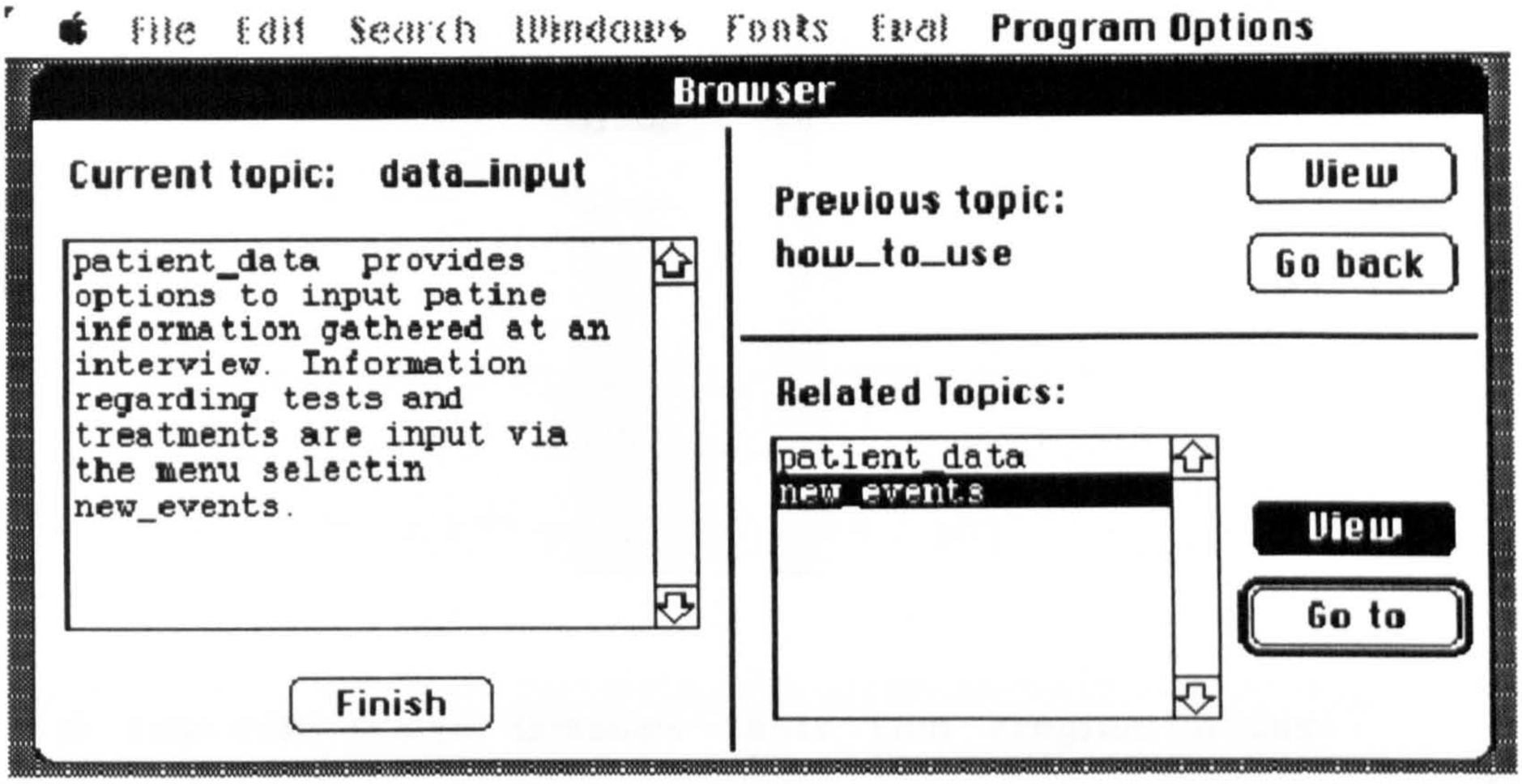


Figure B.16: The dialogue displayed on choosing the option **data_input**

terminate the help facility (by clicking on the **Finish** button).

The two **View** buttons (Figure B.16) allow the user to view the information relative to a topic via a supplementary dialogue. If the **View** button of the previous topic is clicked this displays the text that was visible in the previous dialogue, in this case the test displayed in Figure B.15. If a subtopic in the scrolling menu is highlighted and then the **View** button is clicked, information about the selected topic is displayed. For example, if the subtopic **new_events** is chosen then the supplementary dialogue displayed is shown in Figure B.17. Clicking on the **Ok** button in the display of Figure B.17 returns the display to the parent dialogue i.e. the dialogue relevant to the topic **data_input** given in Figure B.16. By clicking on the appropriate buttons the user can move forwards and backwards through the information structure.

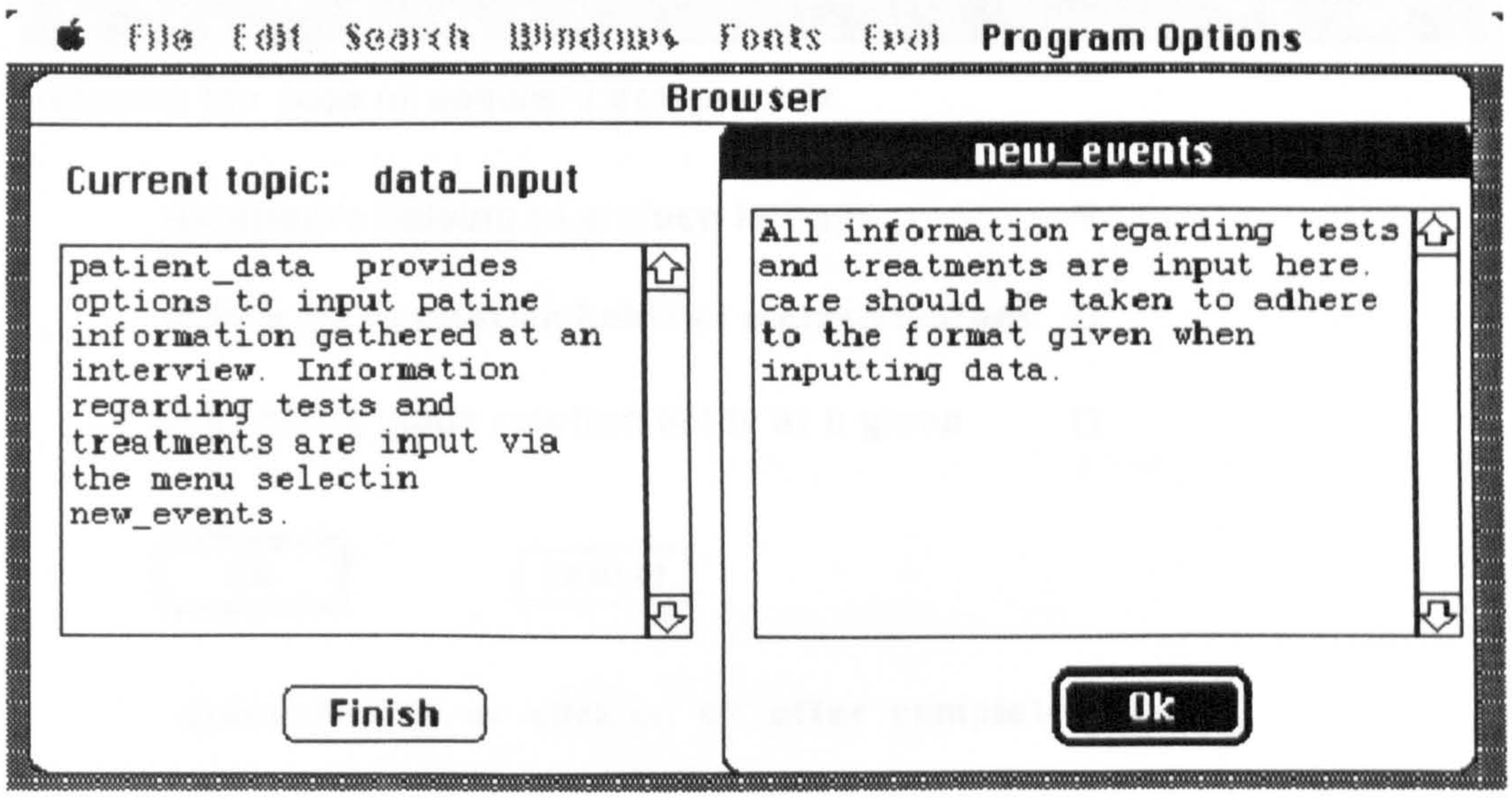


Figure B.17: The supplementary dialogue displayed on choosing the subtopic **new_events**

C Query Facility

Choosing the submenu option **inquiries** from the main menu option **Query** displays a dialogue (Figure C.1) which allows the user to query the knowledge base or ask explanations about the patient's status relevant to any given time.

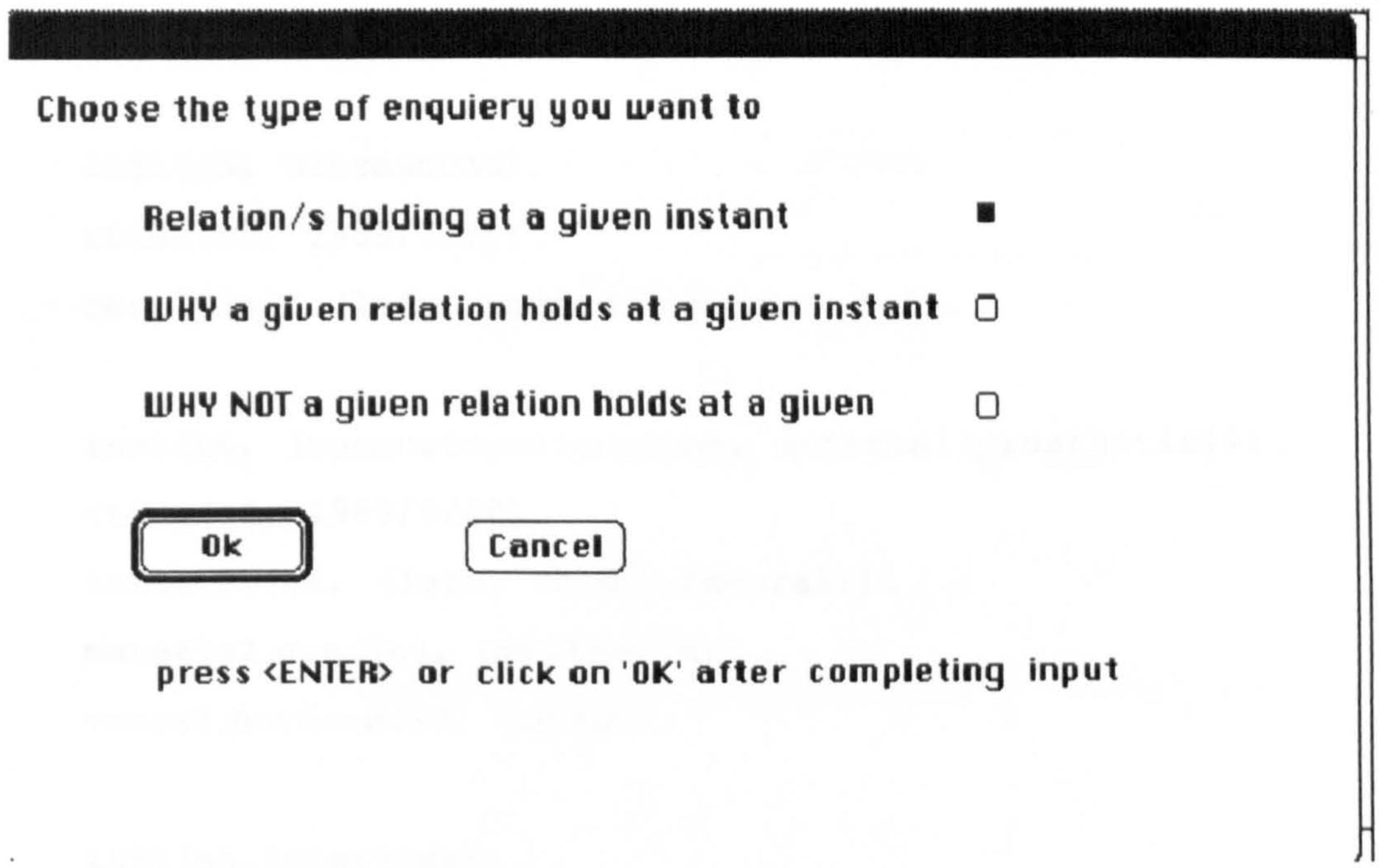


Figure C.1: The dialogue displayed on choosing the option **Query** from the main menu

It can be queried whether a particular property relevant to the patient holds true at a given time, or what the state of the patient is at a given time. The outputs of these two options are given relative to the case given in Case Study 1 of Chapter 7 Section 7.2. The output to the first type of query is illustrated in Appendix C.1. The second type is well illustrated in Chapter 7 Section 7.2. One simple illustration is given Appendix C.2. We have reproduced the event descriptions of Case Study 1 below for easy reference.

```
inst(e1, interview).
etime(e1, 1989/4/10).
report(e1, leg_pain, left, thigh).
report(e1, rest_pain, left).
result(e1, occlusive_arterial).

inst(e2, arteriogram).
etime(e2, 1989/4/25).
result(e2, (left, common_femoral), localised_dilatory).
result(e2, (left, upper_superficial_femoral,
                    localised_atherogenic_arterial).

inst(e3, ultrasound).
etime(e3, 1989/5/10).
result(e3, (left, common_femoral), 5.5).

inst(e4, (reconstruction:tube, material(prosthetic))).
etime(e4, 1989/5/30).
location(e4, (left, common_femoral)).
material_used(e4, prosthetic).
vessel_outcome(e4, patent).

inst(e5,interview).
etime(e5, 1989/6/5).
absent(e5, rest_pain, left).
absent(e5, leg_pain, left, thigh).
```

The Event Descriptions of Case Study 1

(Reproduced from Chapter 7 Section 7.2)

C.1 Finding Whether a Given Relation Holds True at a Specific Time

The dialogue displayed when the button relevant to the option Relation/s holding at a given instant in Figure C.1 is clicked is shown in Figure C.2.

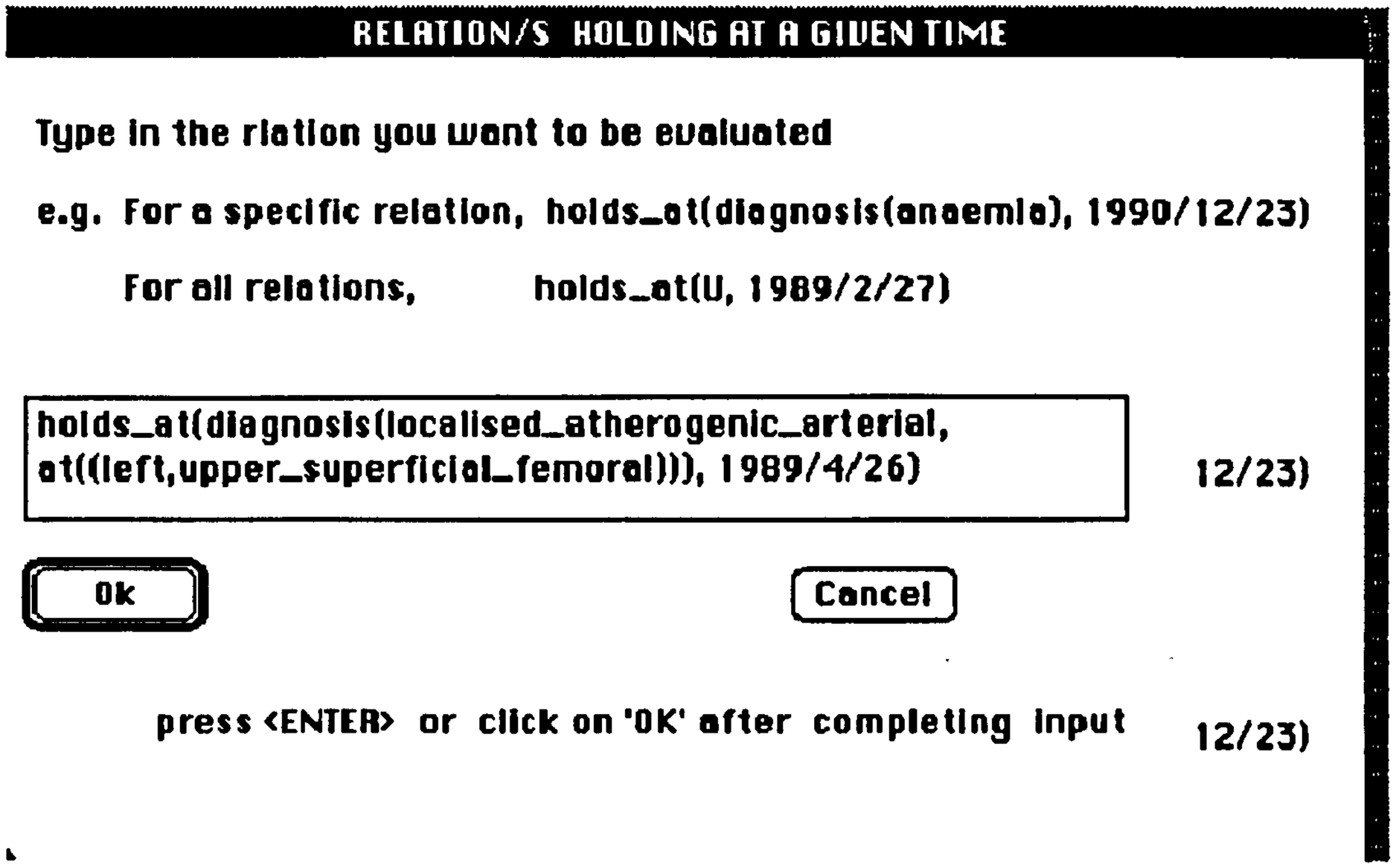


Figure C.2: The dialogue displayed on choosing the option **Relation/s holding at a given instant** in the window shown in Figure C.1

The relation need to be queried about is typed in as arguments for the `holds_at` predicate. For example if we pose the query,

```
holds_at(diagnosis(localised_atherogenic_arterial,
at((left, upper_superficial_femoral))), 1989/4/26)
```

then the answer given is shown in Figure C.3.

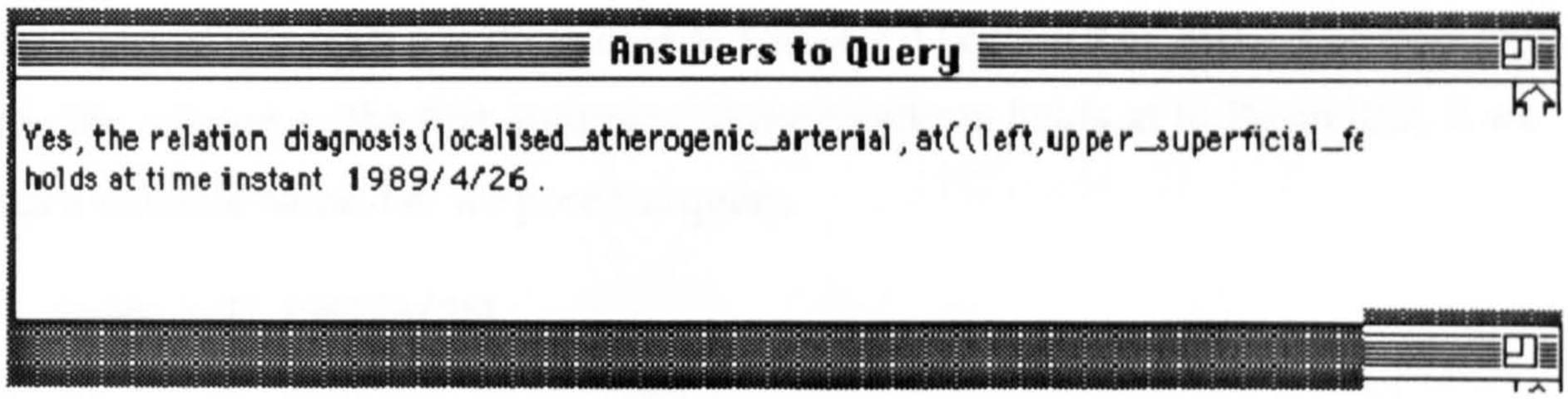


Figure C.3: The answer to the query, " holds_at(diagnosis(localised_atherogenic_arterial, at((left, upper_superficial_femoral))), 1989/4/26")

If the query,

holds_at(awaiting(treatment), 1989/4/26)

which is what will be expected, is posed the answer given is shown in Figure C.4.

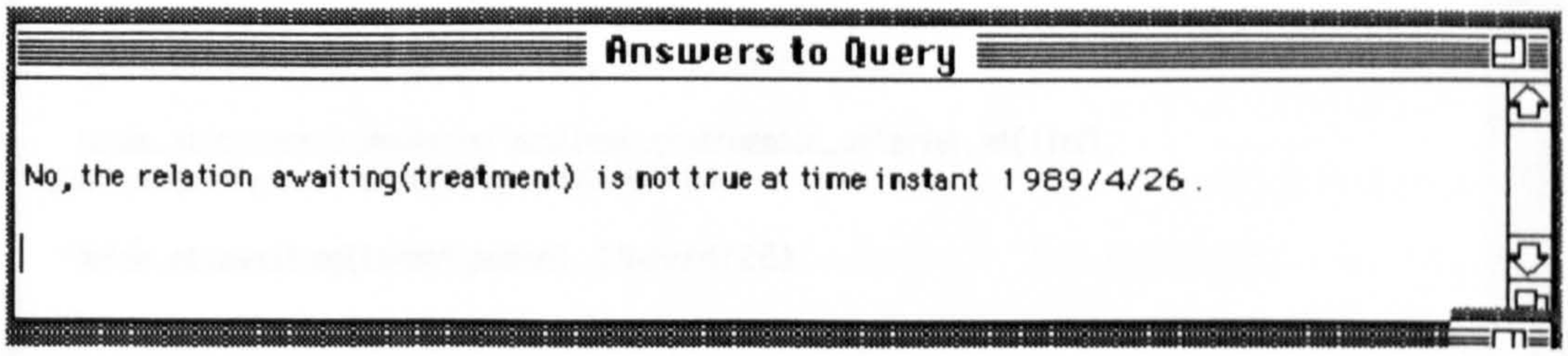


Figure C.4: The answer to the query " holds_at(awaiting(treatment), 1989/4/26)"

Any explanation for these answers or any other (as illustrated in Appendix D) can be displayed using the other two options (as appropriate) given in the dialogue in Figure C.1.

C.2 Finding What Relations Hold True at a Specific Time

The output of this option is illustrated in Chapter 7 Section 7.2. Instead of inputing a specific relation as the first argument to the predicate holds_at in Figure C.2, if we input a variable name, i.e. we pose the query,

holds_at(U, 1989/4/26)

then all relations holding true at that time will be displayed. The answer to the above query relevant to Case Study 1 is shown in Figure C.5.

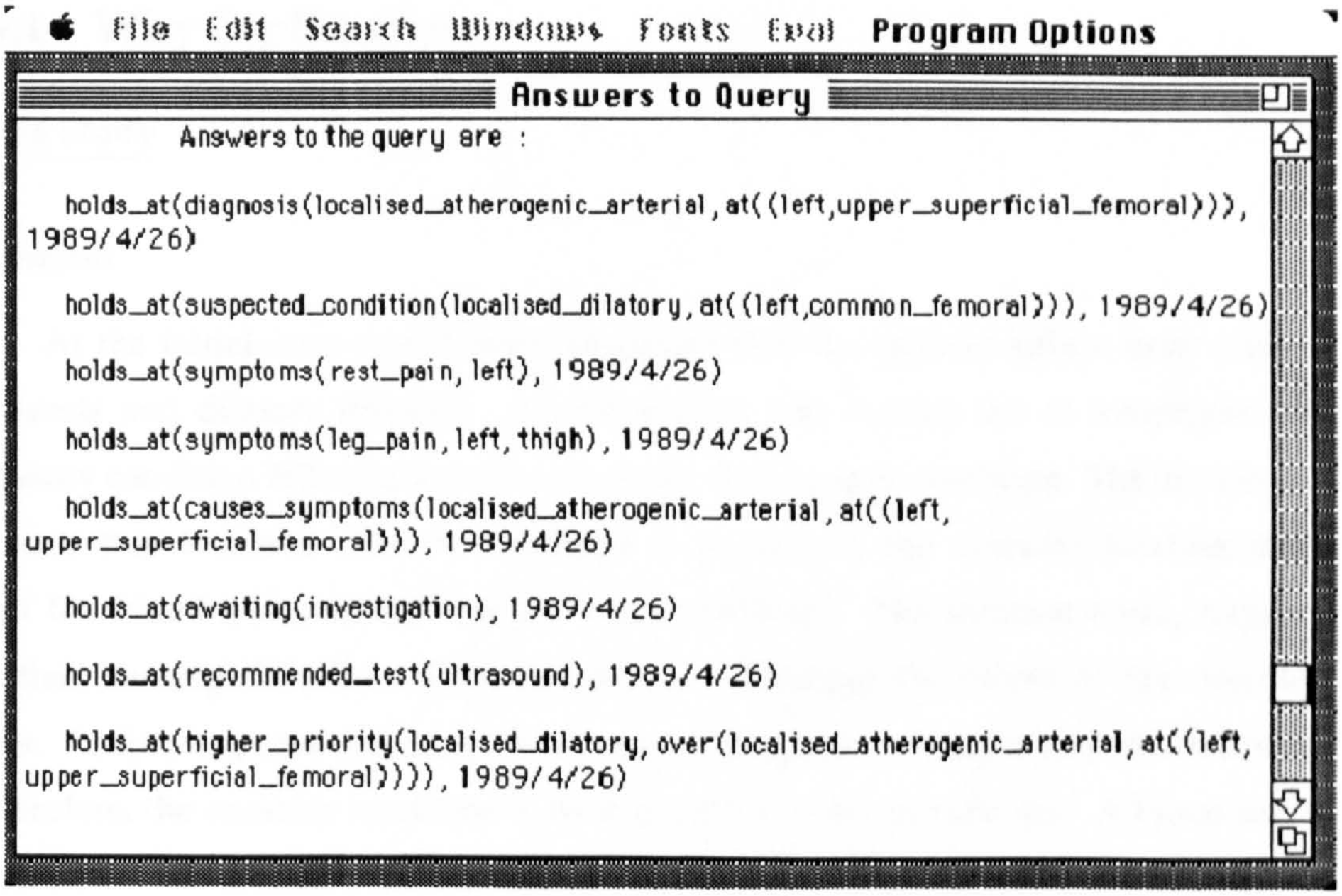


Figure C.5: The answer to the query "holds_at(U, 1989/4/26)"

D Illustration of Answer Justification

The user can ask explanations as to why a relation holds true at a given instant, or why a relation does not hold true at a given instant. These two options are illustrated with outputs in Appendix D.1 and Appendix D.2 respectively.

D.1 Why Explanation

Case Study

Scenario

At the initial interview it was diagnosed that the patient suffers from clinical anaemia and dilatory diseases. An ultrasound was carried out to investigate the dilatory condition followed by a blood test for the anaemic condition. The ultrasound reveals that localised dilatory condition is present at the vascular location aorta and the blood test confirms the anaemic condition. The location aorta, requires further investigations by an arteriogram to determine the extent of the diseased area. Performing an arteriogram is unsafe for the patient under anaemic conditions. Therefore, the anaemic condition is treated with a blood transfusion. A blood test is carried out to determine the success of the treatment. The results of the test reveals that the anaemic condition is eliminated. The arteriogram carried out next reveals that the segment consisting the vascular locations aorta and aortic bifurcation is diseased. The treatment *reconstruction:bifurcated* was performed to treat the dilated segment.

Illustration

The above described scenario is represented as event descriptions in Figure D.1. The state of the patient after the initial interview is presented by the relations given in Figure D.2.

```
inst(e1, interview).
etime(e1, 1988/3/1).
result(e1, dilatory).
result(e1, clinical_anaemia).
report(e1, abdominal_pain, 7).

inst(e3, ultrasound).
etime(e3, 1988/3/3).
result(e3, aorta, 6).

inst(e5, blood_test).
etime(e5, 1988/3/15).
result(e5, anaemia).

inst(e7, blood_transfusion).
etime(e7, 1988/3/27).

inst(e9, blood_test).
etime(e9,1988/4/9).
eliminates(e9, anaemia).

inst(e11, interview).
etime(e11, 1988/4/11).

inst(e13, arteriogram).
etime(e13, 1988/4/23).
result(e13, [aorta, aortic_bifurcation], localised_dilatory).

inst(e15, reconstruction:bifurcated).
etime(e15, 1988/5/20).
material_used(e15, prosthetic).
vessel_outcome(e15, patent).
```

Figure D.1: The case study

From the dialogue in Figure C.1, the second option (WHY a given relation holds at a given instant) is chosen. The dialogue given in Figure C.2 is displayed. If the relation,

```
holds_at(inappropriate(arteriogram), 1988/3/2)
```

is typed in, then the explanation given is shown in Figure D.3. A dialogue is displayed

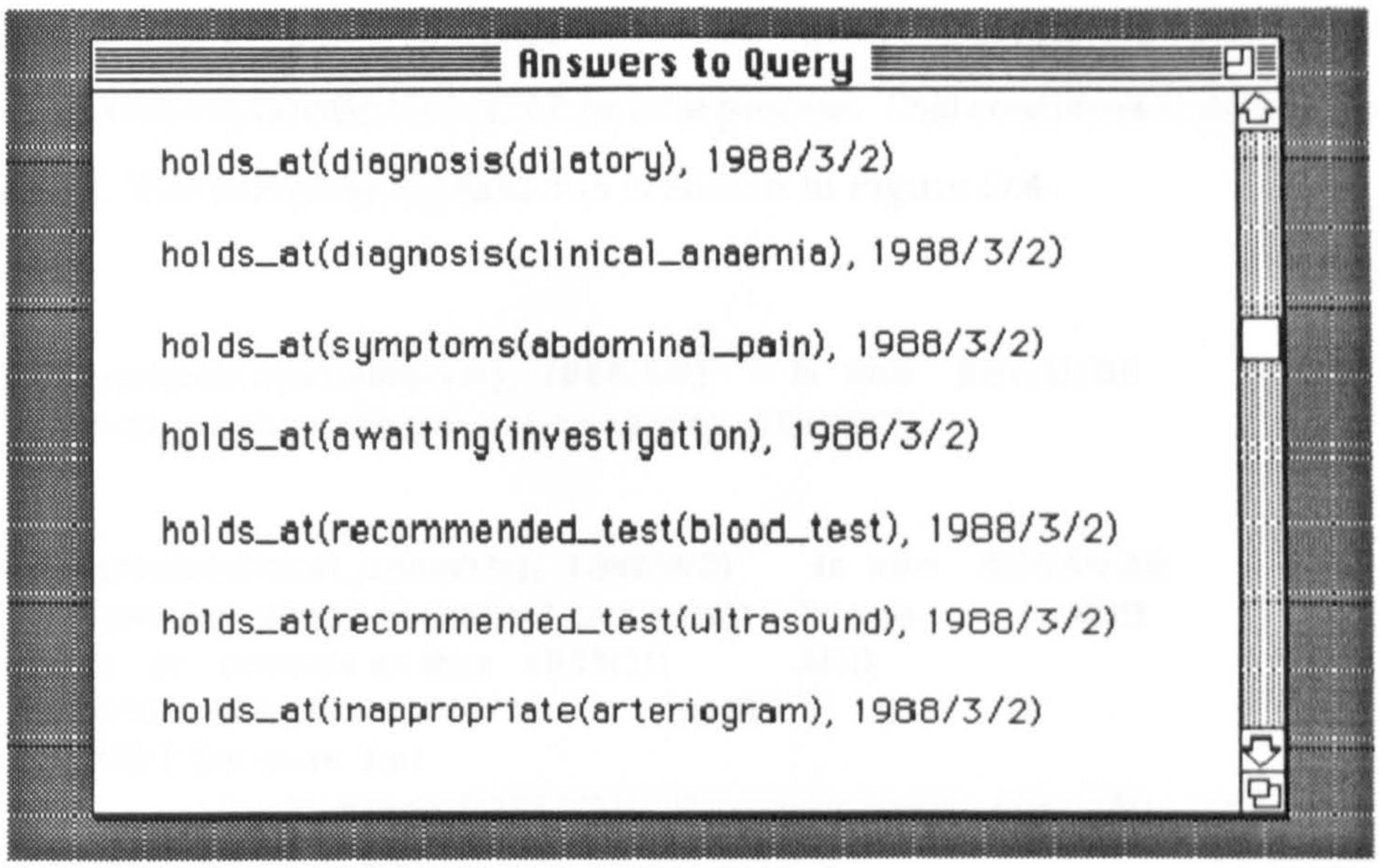


Figure D.2: Relations holding at 1988/3/2

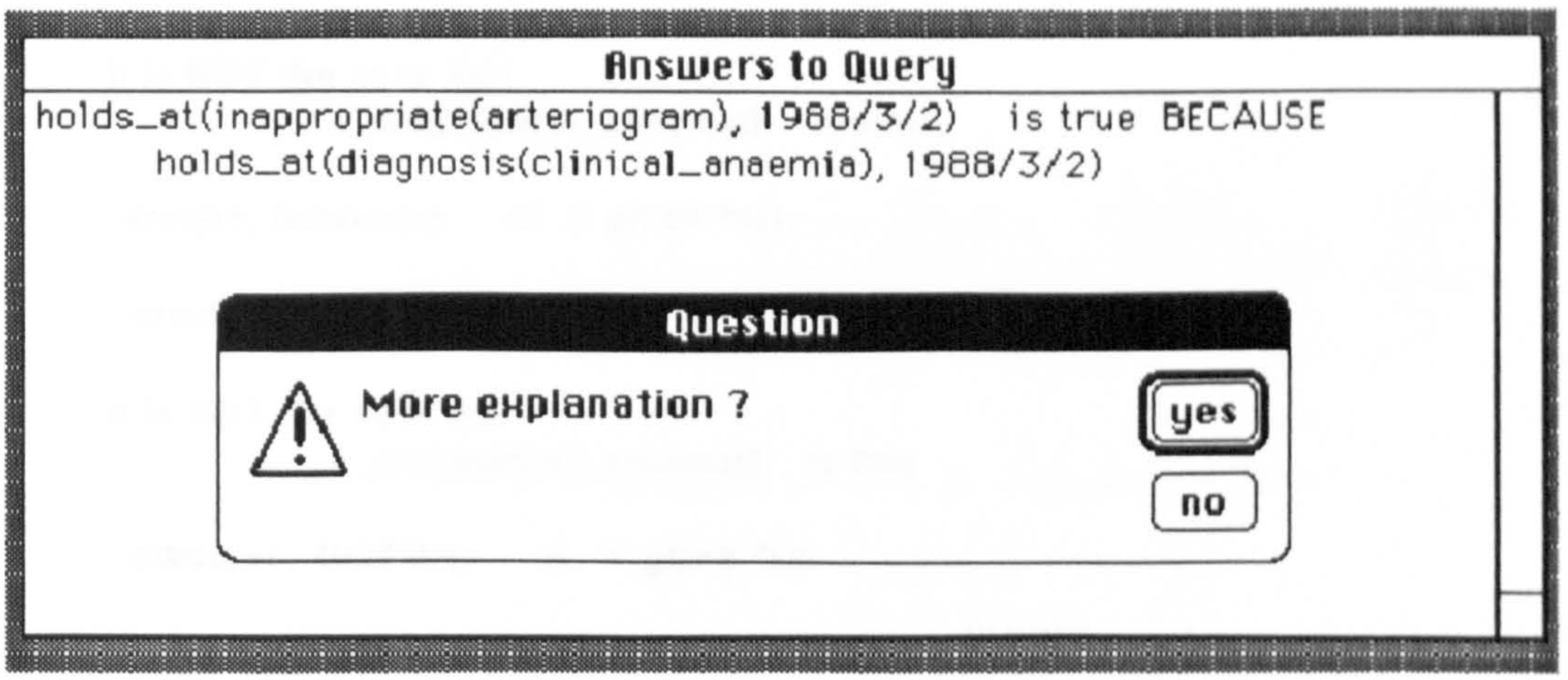


Figure D.3: Explanation why the relation, inappropriate(arteriogram) is true at 1988/3/2

inquiring whether further explanation is required (as shown in Figure D.3). If the answer **yes** is given, the body of the rule shown in Figure D.3 is explained (starting from the first clause). If the answer was **no** then the system quits from the explanation facility.

When the explanation to the first clause is completed, the second (if there is any) is explained and so on. After the explanation for a clause is given, the option is given either to get more explanation or quit from the process. This continues until all clauses are explained. The complete explanation is shown in Figure D.4.

```
holds_at(inappropriate(arteriogram), 1988/3/2)    is true  BECAUSE
        holds_at(diagnosis(clinical_anaemia), 1988/3/2)

holds_at(diagnosis(clinical_anaemia), 1988/3/2)    is true  BECAUSE
        initiates(e1, diagnosis(clinical_anaemia))  is true        AND
        event e1  occures at time  1988/3/1       AND
        1988/3/1  is before  1988/3/2       AND
        It is NOT the case that
                interrupted(1988/3/1, diagnosis(clinical_anaemia), 1988/3/2)  is true
        It is NOT the case that
                too_long_after(e1, diagnosis(clinical_anaemia), 1988/3/2)  is true

initiates(e1, diagnosis(clinical_anaemia))    is true  BECAUSE

        inst(e1, interview)  is true        AND
        the event interview  is a  diagnostic_task       AND
        result(e1, clinical_anaemia)  is true        AND
        It is NOT the case that
                a_drug(clinical_anaemia)  is true

        inst(e1, interview)    is  a given fact

        result(e1, clinical_anaemia)    is  a given fact

        It is NOT the case that
                a_drug(clinical_anaemia)  is true

        etime(e1, 1988/3/1)    is  a given fact

        before(1988/3/1, 1988/3/2)    is true
```

Figure D.4: Relations holding at 1988/3/2

The two conditions clinical_anaemia and dilatory do not interfere with each other. Hence, after the initial interview, tests are recommended for both conditions (Figure D.1) and either may be carried out first. The test ultrasound is carried out next.

D.2 Whynot Explanation

The relations that hold true after the test ultrasound, in the case described in Appendix D.1 is shown in Figure D.5.

```
Answers to the query are :

holds_at(diagnosis(clinical_anaemia), 1988/3/4)

holds_at(diagnosis(localised_dilatory, at(aorta)), 1988/3/4)

holds_at(symptoms(abdominal_pain), 1988/3/4)

holds_at(size(aneurysm, at(aorta), 6), 1988/3/4)

holds_at(causes_risk_of_event(localised_dilatory, at(aorta)), 1988/3/4)

holds_at(awaiting(investigation), 1988/3/4)

holds_at(recommended_treatment((reconstruction:tube,
        material(prosthetic)), for(localised_dilatory), at(aorta)), 1988/3/4

holds_at(recommended_test(blood_test), 1988/3/4)

holds_at(recommended_test(arteriogram), 1988/3/4)

holds_at(inappropriate(arteriogram), 1988/3/4)

holds_at(treatment_options_for(localised_dilatory, at(aorta),
        [(reconstruction:tube,material(prosthetic))]), 1988/3/4)
```

Figure D.5: Relations which hold true after the test ultrasound, at 1988/3/4

Now if we pose the query whynot the relation,

```
holds_at(recommended_treatment(blood_transfusion), 1988/3/4)
```

hold true, by selecting the third option of the dialogue shown in Figure C.1, the explanation given is shown in Figure D.6.

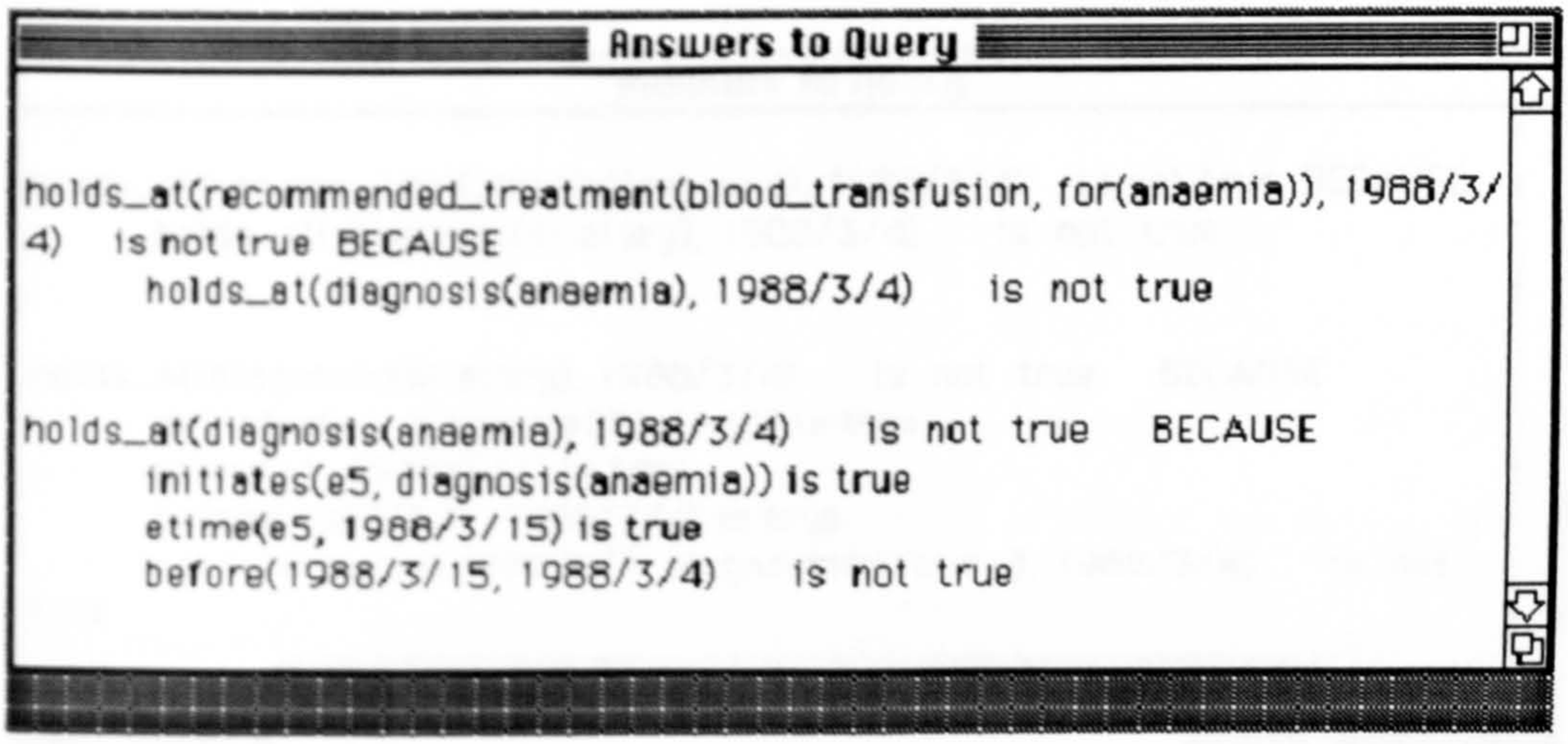


Figure D.6: The explanation as to why a blood transfusion is not recommended at 1988/3/4

If we ask why the relation,

holds_at(recommended_test(ultrasound), 1988/3/4)

does not hold true, the explanation given is shown in Figure D.7.

If the given explanation is not satisfactory, or it is not what the user expected, more explanation can be requested as in the case for "why" explanations. If more explanation was asked for at this stage (represented by Figure D.7), the next explanation that could satisfy the case is given. This is illustrated in Figure D.8. The full explanation for the above query is shown in Figure D.10.

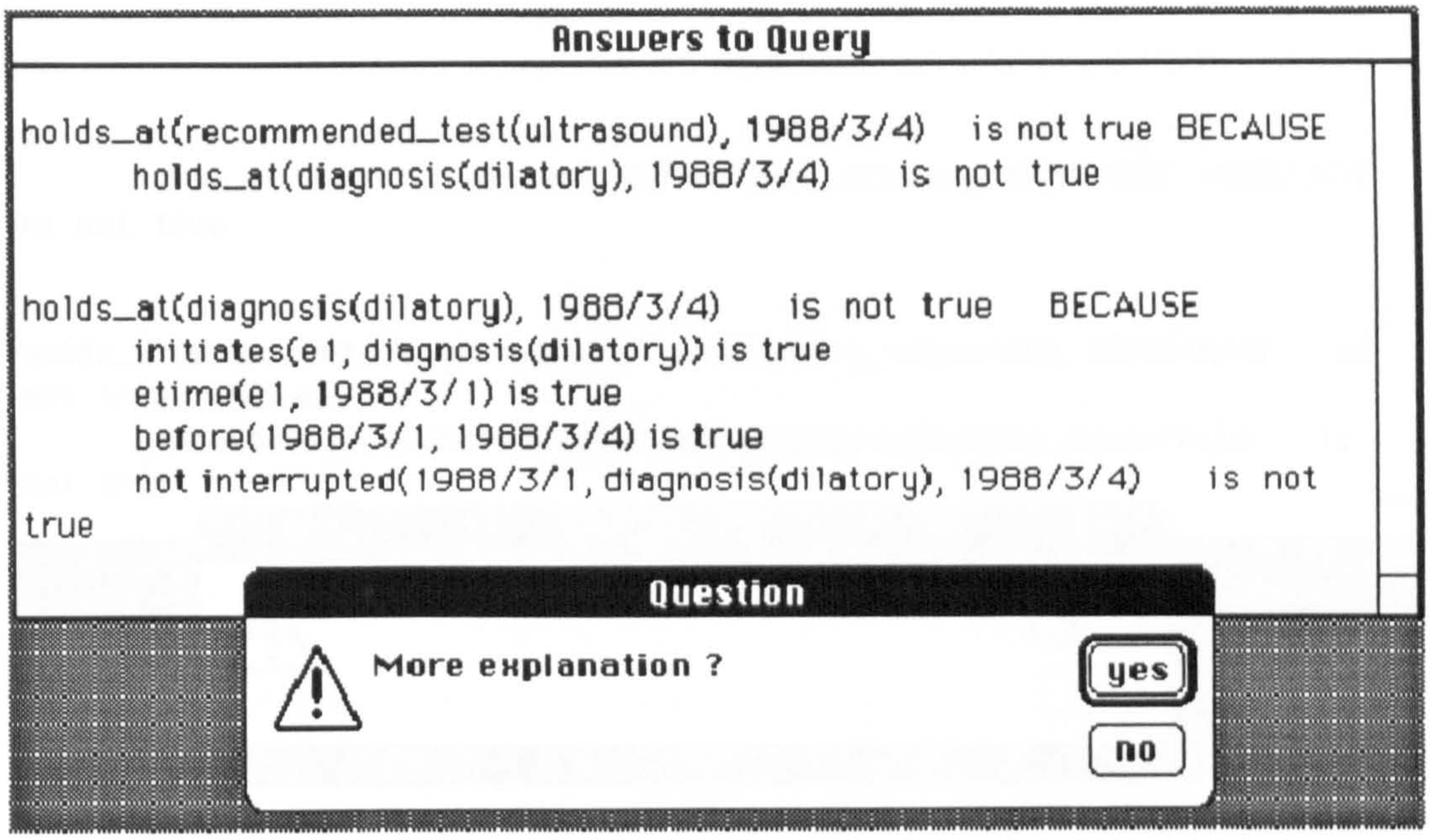


Figure D.7: The first explanation as to why an ultrasound is not recommended at 1988/3/4

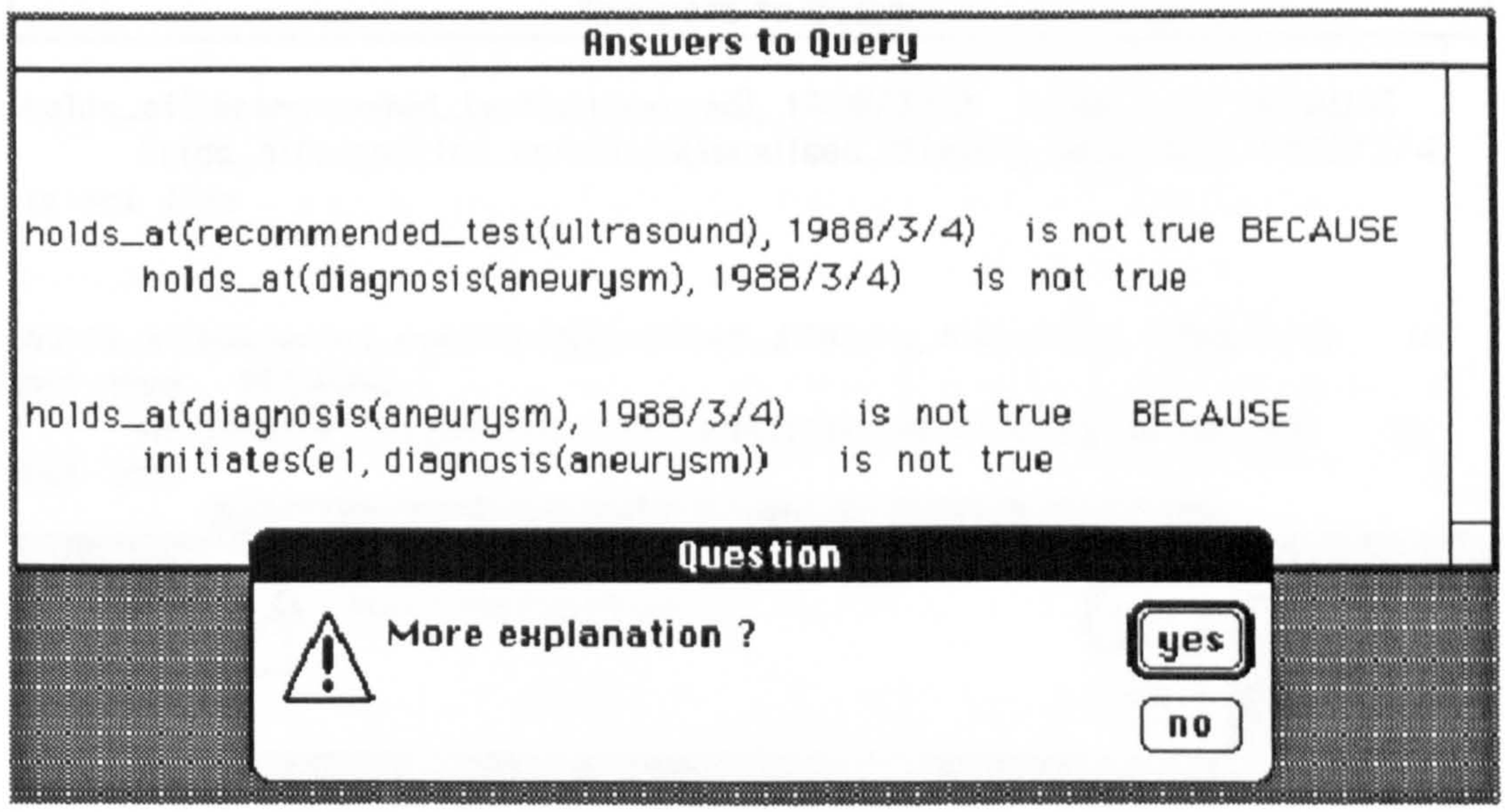


Figure D.8: The second explanation as to why an ultrasound is not recommended at 1988/3/4

Answers to Query

holds_at(recommended_test(ultrasound), 1988/3/4) is not true BECAUSE
holds_at(suspected_condition(localised_dilatory, at(aorta)), 1988/3/4) is not true

holds_at(suspected_condition(localised_dilatory, at(aorta)), 1988/3/4) is not true BECAUSE
initiates(e1, suspected_condition(localised_dilatory, at(aorta))) is not true

Question

More explanation ?

yes

no

Figure D.9: The third explanation as to why an ultrasound is not recommended at 1988/3/4

Answers to Query

holds_at(recommended_test(ultrasound), 1988/3/4) is not true BECAUSE
holds_at(suspected_condition(localised_dilatory, at(aorta)), 1988/3/4) is not true

holds_at(suspected_condition(localised_dilatory, at(aorta)), 1988/3/4) is not true BECAUSE
initiates(e1, suspected_condition(localised_dilatory, at(aorta))) is not true

Question

More explanation ?

yes

no

Figure D.10: The explanation as to why an ultrasound is not recommended at 1988/3/4

Bibliography

ABEYSINGHE, GEETHA & PAUL SOPER 1993. "Description of Processes with PML and Event Calculus". A report submitted to the Introduction of Process Technology (IOPT) Project – a DTI sponsored collaborative project between ICL, Praxis, ISS and Manchester University.

AIKINS, JANICE S., JOHN C. KUNZ, EDWARD H. SHORTLIFFE, & ROBERT J. FALLAT 1984. "PUFF: An Expert System for Interpretation of Pulmonary Function Data". *in* Clancey, W. J. & Shortliffe, E. H., (eds.), *Readings in Medical Artificial Intelligence*, chapter 19, pp. 444–455. Addison-Wesley Publishing Company.

ALLEN, JAMES F. 1983. "Maintaing Knowledge about Temporal Intervals", *Communications of the ACM*, 26 (11): 832–843.

ALLEN, JAMES F. 1991. "Time and Time Again: The Many Ways to Represent Time", *International Journal of Intelligent Systems, Special Issue, Temporal Reasoning, Part A*, 6 (4): 341–355.

ALONZO-BETANZOS, AMPARO, VICENTE MORET-BONILLO, & CARLOS HERNÁNDEZ-SANDE 1991. "Foetos:An Expert System for Fetal Assessment", *IEEE Transactions on Biomedical Engineering*, 38 (2): 199–211.

BARNETT, G. OCTO, NORMA S. JUSTICE, MICHAEL E. SOMAND, J. BARCLAY ADAMS, BRUCE D. WAXMAN, PETER D. BEAMAN, MONICA S. PARENT, FREDRIC R. VAN DEUSEN, & JACQUELINE K. GREENLIE 1979. "COSTAR - A Computer-Based Medical Information System for Ambulatory Care", *Proceedings of the IEEE*, 67 (9): 1226–1237.

BEDFORD, J M, B L ROSSER, & ROBERT A KOWALSKI 1990. "Representing Change in Air Traffic Flow Management Using the Event Calculus", *in Colloquium on Temporal reasoning*. IEEE. Colloquium organised by proffesional group c4 (Artificial

Intelligence) in association with the British Computer Society specialise group in Expert Systems on Temporal Reasoning.

BOLOUR, A, T L ANDERSON, L J DEKEYSER, & H K T WONG 1982. "The Role of Time in Information Processing: A Survey", *ACM SIGMOD record*, 12 (3): 27–50.

BORILLO, MARIO & BRUNO GAUME 1991. "Spatiotemporal Reasoning Based on an Extension of Event Calculus", *in* Kohonen, T. & Fogelman-Soulié, F., (eds.), *COGNITIVA 90, At The Crossroads of Artificial Intelligence, Cognitive Science and Neuroscience Conference held November 1991, Madrid*, pp. 337–344. Elsevier Science Publishers B.V.(North-Holland).

BRAUFFAERTS, A. & E. HENIN 1988. "Proof Trees for Negation as Failure: Yet Another Prolog Meta-Interpreter", *in* Kowalski, R. A. & Bowen, K. A., (eds.), *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, pp. 343–357, University of Washington, Seattle, August 15-19, 1988. The MIT Press.

BRUYNOOGHE, R.F. 1991. "PML Reference Manual". ICL/4R2F/00070 Issue 4.

BUCHANAN, BRYCE G. & EDWARD H. SHORTLIFFE, (eds.) 1984. *Rule-Based Expert Systems, The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Series in Artificial Intelligence. Addison-Wesley Publishing Company.

BURKE, J. P., D. C. CLASSEN, S. L. PESTOTNIK, R. S. EVANS, & L. E. STEVENS 1991. "The HELP System and its Application to Infection Control", *Journal of Hospital Infection, Supplement A, Hospital Infection - Towards the year 2000, Proceedings of the 2nd Hospital Infection Society, London, 2-6 September 1990*, 18: 424–431.

BUTLER, P. & S. ZIENTEK 1990. "PML Tutorial". STC Technology Ltd., Reference : STL/608/00071.

CAMPBELL, W. B., R. G. SOUTER, J. COLLIN, R. F. M. WOOD, I. G. KIDSON, & P. J. MORRIS 1987. "Auditing the Vascular Surgical Audit", *British Journal of Surgery*, 74 (2): 98–100.

CASTLEDEN, WILLIAM M., MICHAEL M. D. LAWRENCE-BROWN, HON MAN S. LAM, BRETT S. MCLOUGHLIN, DEAN S. THOMPSON, & JUAN LOPEZ 1988. "The Hollywood

Surgical-Audit Programme: A Computer-Based Discharge and Data-Collection System for Surgical Audit", *The Medical Journal of Australia*, 149: 70–74.

CHAN, DAVID 1988. "Constructive Negation Based On The Completed Databases", *in* Kowalski, R. A. & Bowen, K. A., (eds.), *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, pp. 111–125, University of Washington, Seattle. The MIT Press.

CLANCEY, WILLIAM J. & EDWARD H. SHORTLIFFE, (eds.) 1984. *Readings in Medical Artificial Intelligence, The First Decade.* The Addison-Wesley Series in Artificial Intelligence. Addison-Wesley Publishing Company.

CLARK, K L 1978. "Negation by Failure". *in* Gallaire, H. & Minker, J., (eds.), *Logic and Databases*, pp. 293–322. Plenum Press.

CLARK, K L & F G MCCABE 1982. "PROLOG: A Language for Implementing Expert Systems". *in* Hayes, J. E., Michie, D., & Pao, Y.-H., (eds.), *Machine Intelligence*, pp. 455–470. Ellis Horwood Ltd.

COLLEN, MORRIS F. 1991. "A Brief Historical Overview of Hospital Information System (HIS) Evolution in the United States", *International Journal of Biomedical Computing*, 29: 169–189.

CONNELLY, DONALD P., BRUCE H. SIELAFF, & EDWARD P. SCOTT 1990. "ESPRE-Expert System for Platelet Request Evaluation", *American Journal of Clinical Pathology, Supplement 1*, 94 (4): S19–S24.

CONRADI, REIDAR, CHRISTER FERNSTRÄM, ALFONSO FUGGETTA, & ROBERT SNOWDON 1992. "Towards a Reference Framework for Process Concepts", *in* Derniame, J. C., (ed.), *Lecture Notes in Computer Science 635, Software Process Technology*, pp. 3–17, Trondheim, Norway. 2nd European Workshop on Software Process Technology (EWSPT 92), Springer-Verlag.

CRANE, VICKI S 1988. "Economic Aspects of Clinical Decision Making: Applications of Clinical Decision Analysis", *American Journal of Hospital Pharmacy*, 45: 548–553.

CURTIS, BILL, MARK KELLNER, & JIM OVER 1992. "Process Modelling", *Communications of the ACM*, 35 (9): 75–90.

DASTA, JOSEPH F., MARIANNE L. GREER, & STUART M. SPEEDIE 1992. "Computers in Healthcare: Overview and Bibliography", *The Annals of Pharmacotherapy*, 26 (1): 109–117.

DEAN, THOMAS L & DREW V. MCDERMOTT 1987. "Temporal Data Base Management", *Artificial Intelligence*, 32: 1–55.

DENECKER, MARC, LODE MISSIAEN, & MAURICE BRUYNOOGHE 1992. "Temporal Reasoning with Abductive Event Calculus", *in* Neumann, B., (ed.), *10th European Conference on Artificial Intelligence, ECAI 92*, pp. 385–388, August 3-7, Vienna, Austria. John Wille & Sons.

DOJAT, MICHEL & CLAUDETTE SAYETTAT 1993. "Temporal Reasoning in a Medical Expertise", *in Proceedings of 5th Software Engineering and Knowledge Engineering Conference*, pp. 134–141, San Francisco.

ELLIS, DAVID, (ed.) 1987. *Medical Computing and Applications*, pp. 63–218. Ellis Horwood Limited.

EVANS, CHRIS 1989. "The Representation of Processes in the Event Calculus". 5 th draft, Logic Programming Group, Department of Computing, Imperial College of Science, Technology and Medicine, University of London.

EVANS, CHRIS 1990. "The Macro-Event Calculus: Representing Temporal Granularity". Revised in April 1990, Logic Programming Group, Department of Computing, Imperial College of Science, Technology and Medicine, University of London.

EVANS, CHRIS & MURRAY SHANAHAN 1989. "The Event Calculus". EQUATOR working document, 4th revised version, Department of Computing, Imperial College of Science, Technology and Medicine, University of London.

EVANS, R. SCOTT 1991. "The HELP System: A Review of Clinical Applications in Infectious Diseases and Antibiotic Use", *M. D. Computing*, 8 (5): 282–315.

FAGAN, LAWRENCE MARVIN 1980. *VM: Representing Time-Dependent Relations in a Medical Setting*. PhD thesis, Stanford University.

FAGAN, LAWRENCE M., JOHN C. KUNZ, EDWARD A. FEIGENBAUM, & JOHN C. OSBORN 1984. "Extensions to the Rule-Based Formalism for a Monitoring Task". *in* Buchanan, B. G. & Shortliffe, E. H., (eds.), *Rule-Based Expert Systems: The MYCIN Experiments and the Stanford Heuristic Programming Project*, chapter 22, pp. 397–423. Addison-Wesley Publishing Company.

FEILER, PETER H. & WATTS S. HUMPHREY 1993. "Software Process Development and Enactment:Concepts and Definitions", *in Continuous Software Process Improvement*, pp. 28–40, Berlin, Germany. Proceedings of the Second International Conference on the Software Process, IEEE Computer Society Press.

FOX, JOHN 1989. "Symbolic Decision Procedures for Knowledge Based Systems", Technical report, Imperial Cancer Research Fund, London. To appear in: H. Adeli(ed.), The Handbook of Knowledge Engineering, New York : McGraw-Hill 1989.

FOX, JOHN 1992. "Logic Engineering and Clinical Dilemmas", *in* Comyn, G., Fuchs, N. E., & Ratcliffe, M. J., (eds.), *Lecture Notes in Artificial Intelligence 636,Logic Programming in Action*, pp. 100–108, Zurich, Switzerland. Second International Logic Programming Summer School, LPSS'92, Spinger-Verlagg.

FOX, JOHN, ANDRZEJ GLOWINSKI, & MIKE O'NEIL 1989. "A Symbolic Theory of Decision-Making Applied to Several Medical Tasks", Technical report, Imperial Cancer Research Fund, London. To be published in: Proceedings of AIME 1989, Lecture Notes in Medical Informatics, Springer-Verlag.

FOX, JOHN, ANDRZEJ J GLOWINSKI, MICHAEL J O'NEIL, & DOMONIC A CLARK 1988. "Decision Making as a Logical Process". Imperial Cancer Research Fund Laboratories, 44 Lincoln's Inn Fields, London WC2A 3PX.

FREKSA, CHRISTIAN 1992. "Temporal Reasoning Based on Semi-Intervals", *Artificial Intelligence*, 54: 199–227.

GALLAIRE, HERVÉ, JACK MINKER, & JEAN-MARIE NICOLAS 1984. "Logic and Databases: A Deductive Approach", *ACM Computing Surveys*, 16 (2): 153–185.

GALLE, JOHAN 1992. "Applying Process Modelling", *in* Derniame, J. C., (ed.), *Lecture Notes in Computer Science 635, Software Process Technology*, pp. 230–236, Trondheim, Norway. 2nd European Workshop on Software Process Technology (EWSPT 92), Springer-Verlag.

GARDNER, REED M., OLAF K. GOLUBJATNIKOV, R. MYRON LAUB, JULIE T. JACOBSON, & R. SCOTT EVANS 1990. "Computer-Critiqued Blood Ordering Using the HELP System", *Computers and Biomedical Research*, 23 (6): 514–528.

GLOWINSKI, ANDRZEJ, MICHAEL O'NEIL, & JOHN FOX 1989a. "Design of a Generic Information System and its Application to Primary care", *in Lecture Notes in Medical Informatics, Proceedings of European Conference on Artificial Intelligence.* Springer-Verlag.

GLOWINSKI, ANDRZEJ, MICHAEL O'NEIL, & JOHN FOX 1989b. "Knowledge-Based Decision Support for General Practitioners : Designing Very Large and Versatile Systems". In Proceedings of the 5th International Expert Systems Conference, 1989 - Oxford: Learned Information.

GORDON, C., J. FOX, A. GLOWINSKI, & M. O'NEIL 1990. "The Design of the Oxford System of Medicine: an Overview", *in* Heidelberg, (ed.), *Medical Informatics Europe 90*, pp. 265–270. Springer Verlag.

GRANT, JOHN & JACK MINKER 1992. "The Impact of Logic Programming on Databases", *Communications of the ACM*, 35 (3): 67–81.

HAJNICZ, ELZBIETA 1990. "Role of the Present in Temporal Representation in Artificial Intelligence", *International Journal of Man-Machine Studies*, 32: 263–274.

HAMMOND, PETER 1993a. "OaSiS: Notes on a Prototype". Rigorously Engineered Decisions, DT1/SERC Project: ITD 4/1/9053, Safety Critical Systems Initiative, RED/ICRF/WP/810.2/1.

HAMMOND, PETER 1993b. "Oncology Prtocol Analysis and OaSiS Prototype Review". Rigorously Engineered Decisions, DT1/SERC Project: ITD 4/1/9053, Safety Critical Systems Initiative, RED/ICRF/WP/810.3/1.

HARMON, PAUL & DAVID KING 1985. *Expert Systems : Artificial Intelligence in Business.* John Wiley & Sons, Inc.

HLATKY, MARK A., ROBERT M. CALIFF, FRANK E. HARRELL, KERRY L. LEE, DANIEL B. MARK, LAWRENCE H. MUHLBAIER, & DAVID B. PRYOR 1990. "Clinical Judgment and Therapeutic Decision Making", *Journal of the American College of Cardiology*, 15 (1): 1–14. Guest Editor: Suzanne B. Knoebel.

HORACEK, HELMUT 1992. "Explanations for Constraint Systems", *in* Neumann, B., (ed.), *10th European Conference on Artificial Intelligence*, pp. 500–503, August 3-7, Vienna, Austria. John Wille & Sons.

JENSEN, C. S., J. CLIFFORD, S. K. GADIA, A. SEGEV, & R. T. SNODGRASS 1992. "A Glossary of Temporal Database Concepts", *SIGMOD Record*, 21 (3): 35–43.

JOHNS, NICKY & CLIVE SPENSER 1989. "LPA MacPROLOG Graphics Manual". Logic Programming Associates Ltd, Studio 4, Royal Victoria Patriotic Building, Trinity Road, London SW18 3SX,U.K.

JONES, NEILL 1993. "Primary Care Computing in Europe", *The Computer Bulletin, Computers in Medicine: The Healthy Way Forward*, 5 (3): 14–15.

KAHN, MICHAEL G. 1988. *Model-Based Interpretation of Time-Ordered Medical Data.* PhD thesis, University of California, Section on Medical Information Sciences, San Francisco CA.

KAHN, MICHAEL G., L. M. FAGAN, & S. TU 1991a. "Extensions to the Time-Oriented Database Model to Support Temporal Reasoning in Medical Expert Systems", *Methods of Information in Medicine*, 30 (1): 4–14.

KAHN, MICHAEL. G., JAY C. FERGUSON, H. SHORTLIFFE, & LAWRENCE M. FAGAN 1985. "Representation and Use of Temporal Information in ONCOCIN", *in Proceedings of the Ninth Annual Symposium on Computer Applications in Medical Care*, pp. 172–176, Baltimore.

KAHN, MICHAEL G., S. TU, & L. M. FAGAN 1991b. "TQuery: A Context-Sensitive Temporal Query Language", *Computers and Biomedical Research*, 24 (5): 401–419.

KOHANE, ISAAC SAMUEL 1986. "Temporal Reasoning in Medical Expert Systems", *in* Salmon, R., Blum, B., & Jorgensen, M., (eds.), *MEDINFO 86 Volume 5, Number 1-2, 5th World Congress on Medical Informatics, George Washington University, Washington,* pp. 170–174. Elsevier Science Publishers B.V. (North Holland).

KOHANE, ISAAC SAMUEL 1987. *Temporal Reasoning in Medical Expert Systems.* PhD thesis, Boston University Graduate School.

KOOMEN, JOHANNES A. G. M. 1991. "Reasoning About Recurrence", *International Journal of Intelligent Systems, Special Issue, Temporal Reasoning, Part B,* 6 (5): 461–496.

KOWALSKI, ROBERT A & MAREK SERGOT 1986. "A Logic-based Calculus of Events", *New Generation Computing, OHMSHA Ltd. and Springer-Verlag,* 4 (1): 67–95.

KOWLASKI, ROBERT A 1992. "Database Updates in the Event Calculus", *The Journal of Logic Programming,* 12 (1-2): 121–146.

KRAUSE, PAUL, JOHN FOX, & MIKE O'NEIL ANS ANDRZEJ GLOWINSKI 1993. "Can We Formally Specify a Medical Decision Support System?", *IEEE EXPERT,* 8 (3): 56–61.

LEE, R M, H COELHO, & J C COTTA 1985. "Temporal Inferencing on Administrative Databases", *Information Systems,* 10 (2): 197–206.

LEVY, M, P. FERRAND, & V CHIRAT 1989. "SESAM-DIABETE, an Expert System for Insulin-Requiring Diabetic Patient Education", *Computers and Biomedical Research,* 22: 442–453.

MAIOCCHI, ROBERTO & BARBARA PERNICI 1991. "Temporal Data Management Systems: A Comparative View", *IEEE Transactions on Knowledge and Data Engineering,* 3 (4): 504–524.

MARESH, JANET & DAVED WASTELL 1990. "Process Modelling and CSCW: An Application of IPSE Technology to Medical Office Work", *in* Diaper, D., Gilmore, D., Cockton, G., & Shackel, B., (eds.), *Human-Computer Interactions - INTERACT'90,* pp. 849–852, Cambridge, England. 3 rd International Conference on Human-Computer Interaction, Elsevier Science Publishers B. V. (North Holland).

MARTIN, JAMES 1976. *Principles of Database Management,* p. 4. Prentice-Hall Inc.

McKENZIE, EDWIN 1986. "Bibliography: Temporal Databases", *ACM SIGMOD Record*, 15 (4): 40–52.

McSHANE, DENNIS J. & JAMES F. FRIES 1988. "The Chronic Disease Data Bank-The ARAMIS Experience", *Proceedings of the IEEE*, 76 (6): 672–679.

McSHANE, DENNIS J., ALISON HARLOW, R. GUY KRAINES, & JAMES F. FRIES 1979. "TOD: A Software System for the ARAMIS Data Bank", *Computer*, 12 (11): 34–40.

MINKER, JACK, (ed.) 1988. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann Publishers, Inc.

NARDI, DANIELE & MARCO TUCCI 1989. "An Application of the Event Calculus for Representing the History of a Software Project", *in* Ghezzi, C. & McDermid, J. A., (eds.), *Lecture Notes in Computer Science 387*, pp. 176–190. 2nd European Software Engineering Conference 1989, University of Warwick, Coventry,UK, Springer-Verlag.

NAVATHE, S. B. & R. AHMED 1988. "TSQL: A Lanuage for History Databases", *in* Rolland, C., Bodard, F., & Leonard, M., (eds.), *Proceedings of the IFIP Working Conference on Temporal Aspects of Information Systems, Sophia - Antiopolis, France, 1987*, pp. 109–122. Elsevier Science Publishers B.V.

OSBURN, A. EUGENE, NORMAN M. NECHES, G. EDWARD SHISSLER, & DIANE KITTREDGE 1984. "Enhancement to COSTAR". *in* Blum, B. I., (ed.), *Information Systems for Patient Care, Computer and Medicine Series*, pp. 314–321. Springer-Verlag. 1981 IEEE, Reprinted with permission from The Fifth Annual Symposium on Computer Applications in Medical Care, Washington, D.C., November 1-4, 1982.

PAYNE, THOMAS H., ALLAN H. GOROLL, MARY MORGAN, & G. OCTO BARNETT 1990. "Conducting a Matched-Pairs Historical Cohort Study with a Computer-Based Ambulatory Medical Record System", *Computers and Biomedical Research*, 23 (5): 455–472.

PERLIS, DONALD, JENNIFER J. ELGOT-DRAPKIN, & MICHAEL MILLER 1991. "Stop the World - I Want to Think", *International Journal of Intelligent Systems, Special Issue, Temporal Reasoning, Part A*, 6 (4): 443–456.

PIMENTEL, STEPHEN G. & JOHN L. CUADRADO 1990. "The Event Calculus and Consistency Maintenance", *in* Matthews, M. M., (ed.), *Third International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE 90), Volume 1-2 1990*, pp. 1162–1166. ACM Special Interest Group Artificial Intelligence, ACM, New York.

PINTO, JAVIER & RAYMOND REITER 1993. "Temporal Reasoning in Logic Programming: A Case for the Situation Calculus", *in* Warren, D. S., (ed.), *Logic Programming, Proceedings of the Tenth International Conference on Logic Programming*, pp. 203–220, Budapest, Hungary, June 21-25, 1993. The MIT Press.

RAMONI, MARCO, MARIO STEFANELLI, LORENZO MAGNANI, & GIOVANNI BAROSI 1992. "An Epistemological Framework for Medical Knowledge-Based Systems", *IEEE Transactions on Systems, Man, and Cybernetics*, 22 (6): 1361–1375.

RECTOR, A. L., W. A. NOWLAN, & S. KAY 1991. "Foundations for an Electronic Medical Record". *in Methods of Information in Medicine*, pp. 179–186. F. K. Schattaeur Verlagsgesellschaft mbH.

RECTOR, A. L., W. A. NOWLAN, & S. KAY 1992. "Conceptual Knowledge: The Core of Medical Information Systems", *in* Lun, K. C., "Degoulet, P., Pierre, T. E., & Rienhoff, O., (eds.), *MEDINFO 92*, pp. 1420–1426. Proceedings of the Seventh World Congress on Medical Informatics, North-Holland Publishers.

RECTOR, A. L., W. A. NOWLAN, & S. KAY 1993. "Goals for Concept Representation in the GALEN Project". to be submitted to the Sixteenth Annual Symposium on Computer Applications in Medical Care, SCAMC 93.

REEVES, PHILIP 1993. "Local Proof Rules, Temporal Reasoning and Truth Maintenance". Medical Informatics R & D, Royal Brompton National Heart & Lung Hospitals, Sydney Street, London SW3 6NP, DILEMMA/RBH/TWP/WP08/PIR/1.0, a report prepared under AIM Project A2005.

REGGIA, JAMES A, BARRY T PERRICONE, DANA S NAU, & YUN PENG 1985a. "Answer Justificaiton in Diagnostic Expert Systems - Part 1:Abductive Inference and Its Justification", *IEEE Transactions on Biomedical Engineering, Computers in Medicine(special*

issue), BME-32 (4): 263–267.

REGGIA, JAMES A, BARRY T PERRICONE, DANA S NAU, & YUN PENG 1985b. "Answer Justificaiton in Diagnostic Expert Systems - Part II: Supporting Plausible justification", *IEEE Transactions on Biomedical Engineering*, BME-32 (4): 268–272.

ROBERTS, CLIVE 1989. "Describing and Acting Process Models with PML", *ACM SIGSOFT Software Engineering Notes, Representing and Enacting the Software Process, Proceedings of the 4th International Software Process Workshop*, 14 (4): 136–141.

SA, JIN & BRIAN WARBOYS 1992. "Integrating a Formal Specification Method with PML: A Case Study", *in* Derniame, J. C., (ed.), *Lecture Notes in Computer Science 635, Software Process Technology*, pp. 106–122, Trondheim, Norway. 2nd European Workshop on Software Process Technology (EWSPT 92), Springer-Verlag.

SADRI, FARIBA 1986. "Three Recent Approaches to Temporal Reasoning", Research Report 86/23, Imperial College of Science and Technology, University of London. revised in November 1986.

SERGOT, MAREK 1982. "A Query-the-user Facility for Logic Programming", Research Report 82/18, Imperial College of Science and Technology, University of London. To appear in Proceedings of the European Conference on Integrated Interactive Computer Systems P.Degano and E.Sandewall, eds.(North-Holland).

SHAHAR, YUVAL & MARK A. MUSEN 1993a. "RÉSUMÉ: A Temporal-Abstraction System for Patient Monitoring", *Computers and Biomedical Research*, 26: 255–273.

SHAHAR, YUVAL & MARK A. MUSEN 1993b. "A Temporal-Abstraction System for Patient Monitoring", *in* Frisse, M. E., (ed.), *Sixteenth Annual Symposium on Computer Applications in Medical Care*, pp. 121–127, Baltimore. McGraw-Hill Book Co.

SHAHAR, YUVAL, SAMSON W. TU, & MARK A. MUSEN 1992. "Temporal-Abstraction Mechanisms in Management of Clinical Protocols", *in Proceedings of the Fifteenth Annual Symposium on Computer Applications in Medicine, 1991*, pp. 629–633, Washington, D.C. McGraw-Hill.

SHANAHAN, MURRAY 1989. "Prediction is Deduction but Explanation is Abduction", *in International Joint Conference on Artificial Intelligence 1989*, p. 1055.

SHORTLIFFE, EDWARD H. 1987. "Computer Programs to Support Clinical Decision Making", *Journal of the Americn Medical Association*, 263: 1114–1120.

SHORTLIFFE, EDWARD H., BRUCE G. BUCHANAN, & EDWARD A. FEIGENBAUM 1984. "Knowledge Engineering for Medical Decision Making: A Review of Computer-Based Clinical Decision Aids". *in* Clancey, W. J. & Shortliffe, E. H., (eds.), *Readings in Medical Artificial Intelligence*, chapter 3, pp. 43–46. Addison-Wesley Publishing Company.

SHORTLIFFE, EDWARD H & LESLIE E. PERREAULT, (eds.) 1990. *Medical Informatics, Computer Applications in Health Care*. Addison-Wesley Publishing Company.

SIELAFF, BRUCE H, DONALD P CONNELLY, & EDWARD P SCOTT 1989. "ESPRE: A Knowledge-Based System to Support Platlet Transfusion Decisions", *IEEE Transactions on Biomedical Engineering*, 36 (5): 541–546.

SIELAFF, BRUCE H, EDWARD P SCOTT, & DONALD P CONNELLY 1991. "Design and Preliminary Evaluation of an Expert System for Platelet Request Evaluation", *The Journal of The American Association of Blood Bank*, 31 (7): 600–606.

SNODGRASS, RICHARD 1987. "The Temporal Query Language TQuel", *ACM Transactions on Database Systems*, 12 (2): 247–298.

SNODGRASS, RICHARD 1990. "Temporal Databases Status and Research Directions", *SIGMOD Record*, 19 (4): 83–89.

SNODGRASS, RICHARD & ILSOO AHN 1985. "A Taxonomy of Time in Databases", *in Proceedings of ACM SIGMOD International Conference on Data*, pp. 236–246. ACM.

SNODGRASS, RICHARD & ILSOO AHN 1986. "Temporal Databases", *IEEE Computer*, 19 (9): 35–42.

SNOWDON, ROBERT 1992. "An Example of Process Change", *in* Derniame, J. C., (ed.), *Lecture Notes in Computer Science 635, Software Process Technology*, pp. 178–195, Trondheim, Norway. 2nd European Workshop on Software Process Technology (EWSPT 92), Springer-Verlag.

SOPER, P J R, G K ABEYSINGHE, & C RANABOLDO 1990. "Temporal Aspects of Knowledge Based Systems for the Management of Hospital Patients", *in* Tjoa, A. M. & Wagner, R., (eds.), *Proceedings of the Interntional Conference on Database and Expert Systems Applications, Vienna, Austria*, pp. 354–359. Springer-Verlag.

SOPER, P J R, G K ABEYSINGHE, & C RANABOLDO 1991. "A Temporal Model for Clinical and Resource Management in Vascular Surgery", *in* Karagiannis, (ed.), *Proceedings of the International Conference on Database and Expert Systems Applications, Berlin*, pp. 549–552. Springer-Verlag.

SOPER, P J R, G K ABEYSINGHE, & C RANABOLDO 1992. "Knowledge Management in a Clinical Doamin". *in* Attia, F., Flory, A., Hashemi, S., Gouardères, G., & Marciano, J. P., (eds.), *EXPERSYS-92*, pp. 375–380. IITT International.

SRIPADA, S.M. 1988. "A Logical Framework for Temporal Deductive Databases", *in Proceedings of the VLDB 88, Los Angeles*, pp. 171–182. Morgan Kaufmann.

SRIPADA, S.M. 1990. *Temporal Reasoning in Deductive Databases*. PhD thesis, Imperial College, London.

STERLING, LENO & EHUD SHAPIRO 1986. *The Art of Prolog, Advanced Programming Techniques*. MIT Press Series in Logic Programming.

STOODELY, M. A. & J. M. SIKORSKI 1991. "OASYS: A Computerized Auditing System for Orthopaedic Surgery", *International Journal of Biomedical Computing*, 29: 119–131.

TU, SAMSON W., MICHAEL G. KAHN, MARK A. MUSEN, JAY C. FERGUSON, EDWARD H. SHORTLIFFE, & LAWERENCE M. FAGAN 1989. "Episodic Skeletal-Plan Refinement Based on Temporal Data", *Communications of the ACM*, 32 (12): 1439–1455.

ULLMAN, JEFFREY D. 1990. "Deductive Databases: Achievments and Future Directions", *SIGMOD Record*, 19 (4): 75–82.

VAN DEN AKKER, P J, L VAN BOCKEL, R BRAND, & R VAN SCHILFGAARDE 1991. "Computerised Vascular Data Management: A Flexible Modular Registry Suitable for the Evaluation of Long-Term Results in Patients Subjected to Multiple Interventions", *Journal of Vascular Surgery*, 5: 459–465.

WIEDERHOLD, GIO, JAMES F FRIES, & STEPHEN WEYL 1975. "Structured Organisation of Clinical Data Bases", *in AFIPS Conference Proceedings 1975, Volume 44*, pp. 479–485. AFIPS Press.

WIEDERHOLD, GIO, SUSHIL JAJODIA, & WITOLD LITWIN 1991. "Dealing with Granularity of Time in Temporal Databases", *Lecture Notes in Computer Science*, 498: 124–140.

WIEDING, J. U., T. KRETSCHMAR, & P. W. SCHÖNLE 1990. "Development of a Computer-Aided Reference System for Differential Diagnostics Support", *Methods of Information in Medicine*, 29 (2): 132–139.

WOLSTENHOLME, DAVID E. 1992a. "Improvements to Conditional Answers in Interactive Systems, Including the Use of Constraint-Solving". Current Address:BP Research International, Sunbury Research Centre, Chertsey Road, Sunbury-on-Thames, Middlesex TW16 7LN, U.K.

WOLSTENHOLME, DAVID E. 1992b. "Rule-Based Explanations in Logic Systems: Some Improvements". Department of Computing, Imperial College, London, SW7 2AZ; Currently at, BP Research International, Sunbury Research Centre, Chertsey Road, Sunbury-on-Thames, Middlesex TW16 7LN, U.K.