An Intelligent Computer Assisted Learning Environment For Acute Abdominal Pain

Thesis submitted in accordance with the
requirements of the University of Liverpool
for the degree of Doctor in Philosophy by

Caroline Elizabeth Johnson

September 1989

Abstract

The architecture for an intelligent computer assisted learning [ICAL] environment to teach tasks in a medical domain which changes over time and in response to student actions has been designed. The teaching strategy underlying the current prototype is discovery learning. This thesis describes the system architecture and focuses on the development of the teaching domain and pedagogical modules. The system is controlled by multiple processes: one is responsible for updating scenarios about which instruction exercises revolve, the other handles the user interface and services student requests. Scenarios are generated using a discrete-event simulation model, embedded within the system. The Knowledge Engineering Environment [KEE] has been used in the systems implementation.

Acquisition and representation of the domain knowledge is discussed, the prototype domain being acute abdominal pain. A framework for the representation of abdominal disease in terms of its underlying anatomy, pathology and physiology has been developed. This structure, which has been employed in the representation of a subset of abdominal diseases, provides a template with which further knowledge may rapidly be entered into the system. This would assist a course developer in extending the range of course material available.

Work on the pedagogical module has concentrated on exploring the learning opportunities afforded by the coupling of the user interface to the simulation module, rather than addressing interpretation of student's actions for the purpose of providing remedial feedback. Students' can experiment with different decision strategies by reseting a scenario at some predetermined state in the simulation model or can query the simulation model for insight into a problem's structure. Production of non-deterministic scenarios and individualised instruction is also facilitated.

Contents

List of Figures

List of Tables

Acknowledgements

Many thanks to Malcolm Taylor, my supervisor, for advising me to take part in post graduate studies, William Corbett and Paul Edwards the domain experts for encouraging me to produce papers for publication and helping prepare presentations, Colin Dunn for general support and Keith Halewood and Ian Finch for providing assistance when word processing this thesis. Last but not least I would like to thank Martin Hughes and my family for their understanding over the past four years.

This work has been supported by a Science and Engineering Research Council award, reference 85306079.

Chapter 1

Introduction

1.1 Overview Of Research

Computer assisted instruction aims to optimise instruction on well-defined course material to meet students' needs. Two types of approach have emerged: traditional computer assisted instruction [CAI] (also known as computer aided learning) and intelligent computer assisted instruction [ICAI]. Development of the first ICAI systems (also known as intelligent tutoring systems [ITS]) was motivated by the need for more individualised instruction than that offered by CAI.

ICAI systems have focussed on teaching subjects which are time independent. The principle reason for this is that emphasis has been placed on teaching strategies and student modelling, whilst the teaching domain has not been an area of major concern. Research into ICAI systems is still in its infancy, no detailed design methodology has emerged, beyond the recognition that the knowledge required for different tasks should be held in separate modules. Workers in the field[1, pages 2–4] striving for a standardized design method have proposed that in the development of an ICAI system, the primary objective should be to decide how the knowledge is to be modularised and what communication links between the modules are required. Each module can then be developed independently of the others. It would appear that ICAI designers have followed this direction and in attempting to demonstrate teaching capabilities have selected teaching domains which are well-understood and can easily be represented alleviating the problems of domain representation. Although further research is required in all areas[21] for people to recognise the true potential of the field the author thinks more complex teaching subjects must be considered. In this thesis, the architecture of a prototype ICAI system to teach multiple tasks in a complex medical domain is described. The design and development of the domain module is addressed, this module being developed in isolation of the others, in accordance with the suggested design principle.

Knowledge-based systems comprise a knowledge base of domain information, a database of case specific knowledge and an inferencing mechanism. The inferencing mechanism is used to derive further knowledge from the case specific and domain knowledge. Systems which carry out a single task often have associated only the domain knowledge which is specific to the task. The domain's structural, strategic and support knowledge is combined. General domain knowledge, such as knowledge of cause and different levels of knowledge abstraction, is implicit and thus inaccessible. For example, consider the rule:

IF The creature is an animal, *and*
The type of creature is herbivore, *and*
The vegetation in the creatures habitat is sparse,
THEN There is evidence that the creature may be suffering from malnutrition.

Implicit in the rule premises is the knowledge that an animal is a creature which can be either a omnivore, carnivore or herbivore. Also, the link between sparsity of vegetation causing malnutrition has as its rationale in the fact that herbivores eat only vegetation. If the domain model and strategic knowledge are separated, the domain knowledge becomes more explicit and accessible. Multiple tasks can then abstract from the domain model knowledge of relevance to them. The system investigated here has an explicit domain model which is used to generate a problem about which a tutorial focuses and to service the students' requests.

The problem solving strategies and factual knowledge existing ICAI systems attempt to impart to the student are generally atemporal. For example, in [9,69] children are taught mathematical subtraction and in [10] geographical facts. Few ICAI systems teach domain processes which are time dependent. The reason for this is that temporal reasoning is a relatively new research area and theories are only just beginning to emerge. Sophie[8] is a system which teaches electronic troubleshooting in a domain which changes over time. However, the domain model is updated only on student actions such as circuit faulting, and not on the explicit passage of time. Other programs designed to teach problem solving in domains that change over time, ignore time and focus instead on other factors of importance to the task. Shami and Knight[65] have developed a system to teach medical diagnosis. Their system assumes a patient's condition is invariant over the tutorial duration, an assumption which is clearly unrealistic if the patient has gastrointestinal bleeding. The problems presented to a student in the system discussed here change over time independently of student actions. The system architecture has embedded a simulator which uses the domain model to build a simulation model. The simulation model is updated dynamically over time and provides the material for presentation in a tutorial. At the end of a tutorial,

the simulation may be restarted at some previous point in time, allowing students to re-evaluate their decision strategies.

Student learning may be acquired by providing explanation or by the student monitoring the effects of her actions on a problem. The latter teaching mode, learning by discovery, is used in the system described. Although a facility for explanation has not been incorporated into the system, it could easily be added. The facility would be required to process the simulation model which describes the evolution of the domain problem, at any time during its development to determine the active components, their behaviour and relationships. Explanation at this level of detail is a feature of the representation and could not be done by traditional simulation models whose performance could only be found by the assessment of statistical information at the end of a simulation run.

Separating the domain model from the task knowledge which uses it constitutes a deep knowledge representation. This differs from surface level representations in which the domain model and task knowledge unite. A potential benefit of deep knowledge is that it supports better explanation since the knowledge structure is more explicit. Also, the representation provides a reference point for knowledge elicitation between a knowledge engineer and a domain expert. The domain model that has been developed contains deep knowledge. The extent to which it supports the claims is assessed. If further knowledge can easily be added to the system, the domain model may be extendable by a course developer with no programming experience. The number of different simulations generated by the system would then grow, increasing the quantity of tutorial material.

CAI programs are in general limited by their deterministic presentation of course material[43]. They have no real knowledge of how the domain behaves, this is prespecified by the course developer. Finally, the thesis will assess whether the separation of the domain knowledge from the mechanism which interprets it when generating a simulation run enables multiple, non-deterministic scenarios to be generated. If true, teaching sessions would be more individualised and the problems of mapping branching logic into the subject domain in CAI programs would be avoided. When coupling this possibility with that of easily extendable knowledge bases, the extension of course material would be rapid, overcoming the large development time inherent in CAI production.

1.2 The Domain

The subject domain for the prototype system is abdominal pain. This subject was selected for the following reasons:

- It is a large and complex dynamic domain which may be viewed from different perspectives, for example normal body processes (physiology), abnormal body processes (pathology) and anatomy. Some areas are not well understood.
- Multiple tasks may be carried out in the domain. These include diagnosis and management.
- Abdominal pain is a common clinical dilemma which is often mismanaged [18]. An increase in student knowledge would benefit a large patient community.
- Without computer assisted instruction in the clinical learning environment students can take only a passive role in the management of patients presenting to hospital with acute abdominal pain. The reason for this of course is that patients cannot be subjected to the risks of mismanagement by students. The introduction of clinical simulations into the learning environment provides a new learning experience. Students can actively carry out their management strategies whether beneficial or detrimental to patient health.
- When faced with a patient case of acute abdominal pain, medical students and junior medical staff can be assumed to have sufficient knowledge to carry out diagnosis and management using knowledge acquired from their pre-clinical courses. However, they experience problems in applying the relevant knowledge to a case. When experimenting with simulations students may reorganise their knowledge in order to understand patient behaviours.

From the teaching aspect, the depth at which patient management has been considered is at two levels; treatment of the underlying pathology and management of the physiology of patients who present to the Accident and Emergency Department. Since pathological processes may affect a patient's normal physiology the resultant pathophysiological processes may cause secondary diseases with possible fatal outcomes. For example, a patient with a bleeding duodenal ulcer may die of a heart attack, not as a direct outcome of an abnormal duodenum, but because the heart could not compensate for the blood loss. A student must control the secondary disease processes whilst diagnosing and treating the primary disease cause. In the previous example, the student should have administered fluid therapy to correct the fluid loss, then have treated the cause.

Most previous programs which incorporate a representation of clinical medicine have not attempted to explicitly describe both pathology and physiology. The reasons for this are:

- The relationships between pathology and physiology are not well understood at a deep level.

- Many physiological processes take place in the body, systems (for example the cardiovascular system) being used to classify them. Abnormalities in one process may propagate to others, possibly resulting in a combinatorial explosion.

- The representation of each process taking place in each body system is complex. Therefore researchers have modelled them in isolation.

The system described has a model of normal physiology and pathology associated with acute abdominal disease. Relationships between the two models are used to deduce a patient's pathophysiological response to disease.

1.3 Nature Of A Tutorial

A tutorial comprises a clinical scenario of a patient being admitted to an Accident and Emergency Department with acute abdominal pain. The objective of the tutorial is for the user of the system, a medical student or a house officer, to manage the patient and select a surgical treatment (possibly conservative) which agrees with expert medical opinion. Diseases are selected by the course tutor to match the users clinical competence. The tutorial duration is also modifiable.

Having been presented with a patient, the student's primary task is to make a diagnosis. To do this she may request patient histories, physical examinations, laboratory investigations, special investigations and blood pressure, temperature and pulse charts. However, whilst making a diagnosis the patient's condition may deteriorate to a state requiring management intervention. Patients with acute abdominal pain often present with shock, a physiological response to their underlying disease. Without intervention by intravenous fluid therapy, the shock may advance to a non-reversible state after which the patient dies. It is essential therefore that the student combines her management and diagnosis strategies. Of course, inappropriate management may also lead to the patient's demise.

When the student has arrived at a diagnosis and the patient's physiological condition is not too progressed she must select a surgical treatment. On the selection of an appropriate treatment a tutorial summary is produced, describing the actions carried out during the tutorial. The student may then restart the simulation from some time point within it to explore the effects of carrying out different decision strategies. The converse operation, advancement of the simulated time, is also provided.

1.4 Overview Of Thesis

A literature review of computer-based teaching systems for use in the medical domain is given in chapter 2 together with an overview of medical representations.

In chapter 3 discovery learning is considered with reference to cognitive science theories which support its benefit to learning. The architecture required by the prototype teaching system is discussed from a teaching and design perspective, its principle features being discovery learning, and comprehensibility and extensibility of the domain knowledge. An overview of the representation features provided by KEE and used in the implementation of the prototype system follows. With reference to the prototype domain, each of the systems components is described in subsequent chapters.

Domain knowledge acquisition and representation is discussed in chapters 4 and 5. Chapter 4 addresses the pathology, anatomy, pain character and sign and symptom complexes associated with acute abdominal pain. The physiology is covered in chapter 5. The simulation generator and the tutorial manager modules are discussed in chapter 6. The user interface and the student actions are described in chapter 7.

To test the expansibility of the domain knowledge further diseases where added to the system. Chapter 8 describes the incorporation of the new knowledge into the domain module. Finally, in chapter 9 the conclusions made on the basis of the results in chapter 8 are presented. This chapter also suggests extensions and further work which could be carried out to improve on the basic system.

Chapter 2

A Review Of Teaching Systems And Medical Representations

2.1 Introduction

The goal of computer assisted instruction is to optimise instruction on well defined course material to meet individual student's needs. Two generic types of program have been developed to address this goal; traditional computer assisted instruction (CAI) and intelligent computer assisted instruction (ICAI).

In this review the evolution of teaching systems will be discussed, the examples given being selected from the medical domain. No attempt will be made to examine individual systems at length, only their features of relevance to the discussion. As the discussion of the systems unfolds it will emerge that ICAI systems have an explicit representation of their domain knowledge. This characteristic is also common to those medical expert systems which have a deep representation of their domain knowledge. A selective survey of systems which have deep representations of pathology, physiology and anatomy and are relevant to the research described are also examined.

2.2 Traditional Computer-Assisted Instruction

The first instruction system was developed by Skinner[67] in the 1950's. The teaching approach adopted in his programs was drill and practice whereby subject material was presented to the student followed by questions. On the student correctly responding to a question textual reinforcement was issued by the program, incorrect responses were ignored. To ensure successful teaching, the material to be taught was split up into small sections and the questions required such small advances in knowledge that appropriate answers were almost guaranteed. Skinnerian type programs are also called *linear* programs, the name being derived from the path a student takes through the program, figure 2.1a.

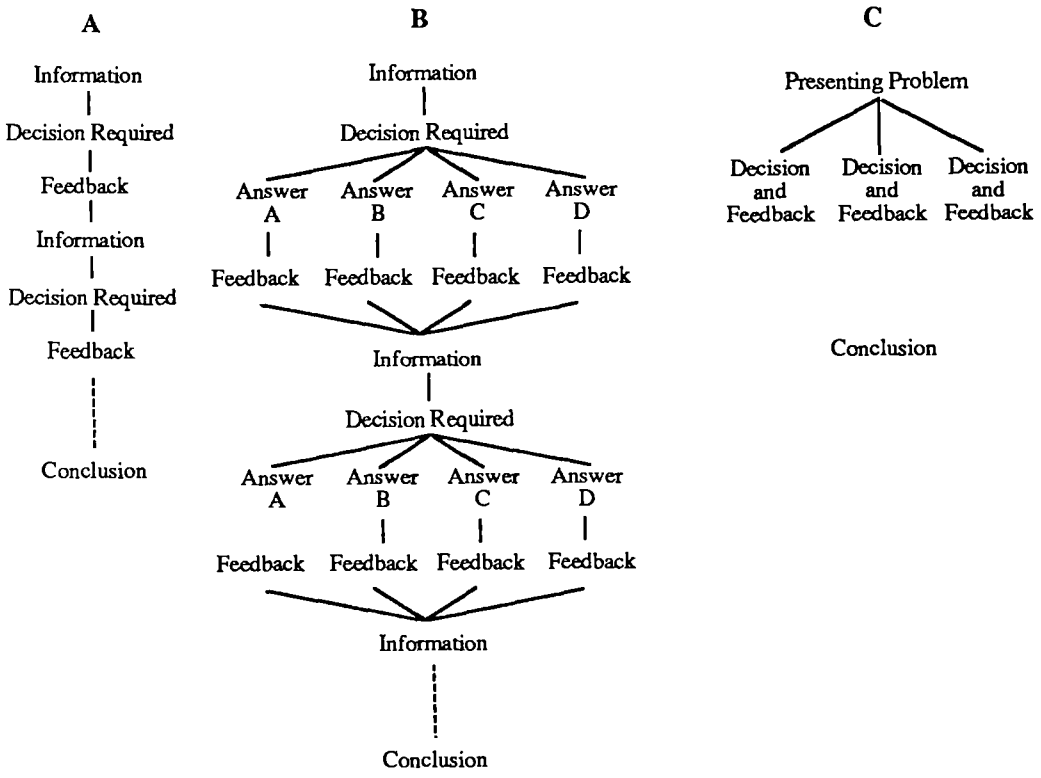


Figure 2.1: Structures Of (a) Linear, (b) Branching and (c) Open Types Of Traditional Teaching Programs

possible answers. On the selection of an answer feedback is generated. If the answer is correct the system will explain why it is correct, benefiting the student who has guessed the answer, before processing the next block. Incorrect answers result in explanation detailing why the answer is wrong and the current frame is reinvoked.

Criticisms of branching programs are that the student cannot select her own path through a program, this is predetermined by the course designer. For programs which aim to transfer factual material to the student selection of a topic can be performed at the program level, ie. each program deals with a specific topic[60]. However, programs which centre on patient management problems are severely limited by the branching type structure since they allow for only one route through the problem preventing the student pursuing the effects of incorrect management on a patient's condition. To allow the student more freedom of action in a patient management problem the *open* type of program was developed, figure 2.1c. These programs let the student chose what to do next within the confines of a multiple choice environment.

Open type programs model the patient as a static or dynamic entity. The static type assume the patient's condition is fixed over time, the student being required to diagnose the patient's illness. Schneiderman and Muller[63] developed a static type program which modelled pediatric disorders. Dynamic type programs assume the patient's condition changes over time in response to the student's actions, the student being responsible for diagnosing and treating the patient's condition. Examples of such programs include part of the Cardiff Computer Aided Learning Project[60] and the Liverpool Microtext Clinical Simulations[75,25,24]. Both types of program present to the student a patient together with the actions which may be requested, for example a full or brief history, examination, treatment and special investigations. A clock simulating real time is also presented in the dynamic type of program. On the selection of an action appropriate feedback is given. In the dynamic type of program, the simulated time is incremented by a predefined amount and the patient's condition is progressed (unless resuscitation is performed). It is important to note that time is not explicitly represented in the dynamic programs, instead time advance is implicit in the selection of actions.

The representation of teaching systems that have been discussed up to this point all have knowledge of only one problem which they can present to the student. For linear and branching programs which represent patient management problems, the change in the patient is fixed since only one route through the program exists. For open programs the initial presentation of the problem is fixed although in the dynamic type of program its development is dependent on the actions selected. In the late 1960's and early 1970's *generative* programs were developed which could generate teaching material of various difficulties for presentation, the problems not being prestored by

the course designer. Domains applicable to this type of program are required to be well defined. Mathematical representations of domains are best suited to this program design since the majority of the prestored teaching material can be omitted from the program and as many problems as the student requires can easily be generated by randomising the parameters used to model the problem.

In the medical domain few subjects can be represented mathematically because they are complex and have associated many variables which are context dependent or are not well understood. The scope for generative programs is therefore limited. Geddes and her colleagues[32] devised a generative program to generate a patient with one of four types of epilepsy. Given the type of epilepsy to be setup, the patient's age, weight, sex and other parameters are generated at random. These are displayed to the student and passed to equations representing the pharmacokinetic model. The student then selects a drug from a list. If the drug is inappropriate an error message is generated, and the student is requested to select another drug. If the drug is appropriate the dosage is prompted for and passed to the pharmacokinetic model. The model then generates the patient's response to the drug after prespecified time delays.

One of the earliest programs having generative capabilities was developed by Entwisle and Entwisle[26] for teaching disease diagnosis. This did not having a mathematical representation as do most later generative programs. For each of five diseases, the percentage of prevalence of 50 symptoms was represented. A disease for presentation to a student was selected at random. A random threshold value between 10 and 50 was then generated. All symptoms with a percentage prevalence greater than the threshold value were passed to the student who then had to diagnose the disease. Since the program had no explicit representation of the relationships existing between symptoms, for example causal relationships, its potential for explanation was limited.

One of the major problems of generative programs is generating feedback which is made more difficult as the complexity of the problem is increased to make it more realistic[16]. Programs having mathematical representations have only a shallow representation, permitting limited explanation. For this reason, *discovery* learning programs have been developed which allow the student to explore a domain model at their own pace, attributes of the domain being revealed as the student modifies the model. Unlike open type programs which have control over the initial presentation of a problem, the student may themselves select a problem by setting up the model accordingly and may later change the model to discover the results of "what if" questions. It is usually left to the student or groups of students to formulate explanations about the models behaviour, although in some centers human tutors provide assistance.

Simulation models which facilitate discovery learning normally have a mathematical representation. A popular subject area for such models in the medical domain

is physiology. The designers of the models strive to achieve a realistic representation of a subject but face problems of introducing errors through over simplification of the behaviour of the system or guessing behaviour which is not well understood, when identifying mathematical equations to represent the subject material.

The MAC family of physiological models are widely available and support discovery learning[22]. MacMAN, MacPUF, MacPEE and MacDOPE model the heart and peripheral circulation, the lungs and blood gas exchange, the kidneys and body fluids and pharmokinetics respectively. Using the models, students' can gain insight into normal and deranged physiology. They can study the results of changing one or more physiological parameters in the system being simulated over a period of time. The behaviour of the models are displayed in graphical and textual form. To overcome the difficulty of controlling the models, an instruction driver program, MACAID, has been developed to interface students' to the models. MACAID handles the setup of the models and provides feedback on the students actions[68].

Other clinical simulation models supporting discovery learning include a program to teach about the critically ill surgical patient[28] and the Didapharm-De system[73] which teaches about the effects of cardiovascular drugs. The first model assumes a patient fixed in time. The second model describes a patient whose condition may change dynamically in time. Both systems use graphics to display their output.

The limitations of traditional teaching programs arise from the representations used to develop them[43]. These include authoring languages which allow the course developer with little computing experience to easily develop tutorial software and high level languages. The limitations and their causes are summarised below:

- Feedback, when provided, is impoverished. The programs have no explicit knowledge of the student, her current understanding or previous actions or of the subject material. Where knowledge of the student does exist it is generally in the form of a number representing the student's current proficiency[61]. Lack of knowledge of a student restricts the potential for individualised instruction because it limits the programs' explanation capabilities. Subject knowledge is implicit in the programs' mathematical or frame-based representations and is therefore inaccessible for explanation purposes restricting the quality of the feedback.

- Although there are various strategies by which teaching material may be conveyed to the student, for example discovery learning and the various types of structured program, each program has only one. The teaching strategy is implicit in the program design. A single approach to teaching results in programs which are not optimised to meet all students' needs, since student learning varies in the population.

- Mixed-initiative dialogues are not supported. Programs which provide for discovery learning require the student to initiate the dialogue. All other programs themselves direct the dialogue using logic which has been predefined by the course designer denying the student the opportunity to change the focus of the tutorial.

- The subject and teaching knowledge are combined. This prevents the teaching knowledge being reused to instruct the student on a different topic.

2.3 Intelligent Computer Assisted Instruction

The pioneer of intelligent computer assisted instruction (ICAI), Carbonell[10] identified the major weakness of traditional computer assisted instruction programs as being that their internal representation of who was being taught, how to teach and what was being taught did not match that of a human. To overcome this limitation, he applied artificial intelligence techniques to teaching programs. In his description of a system called SCHOLAR, he singled out concepts which were to become the key principles underlying ICAI systems. These were that the system had explicit knowledge of the subject material being taught, the student, and how the subject material was to be presented to the student. Each type of knowledge was represented independently of the others. The different knowledge types interacted to derive the teaching sequence; this was not predetermined. Problems which were presented to the student could be solved by the system itself, in a way similar to an expert. The potential features of this representation were:

- The system could handle unanticipated responses. The explicit domain knowledge could be used to process them.

- Mixed-initiative dialogues could be accommodated. Knowledge of the subject being taught and the student's competence could both be used to determine the material to be presented.

- More detailed diagnostics of student errors could be generated using the programs knowledge of individual students. The student model would be generated dynamically to enable explanation in terms of the student's current understanding.

- The teaching strategies could be adapted to meet individual students needs since they are explicitly represented and not combined with the subject material.

- Knowledge of one type could be modified without recourse to change other knowledge types. The subject knowledge could therefore be altered without consideration of the procedures which interact with it during a tutorial.

The components of an ICAI system and their interaction are not universally agreed upon. Various architectures have been proposed by researchers in the field. These vary in the emphasis placed on domain knowledge and student modelling, an emphasis on domain knowledge indicating more student initiative is to be allowed by the program. Anderson's architecture[6] comprised four components; the domain expert which could solve problems in the domain, the bug catalogue comprising common misconceptions and errors in the domain, tutoring knowledge of how to teach the domain knowledge and the user interface which controls the interaction between tutor and student. Should a student's solution deviate from the ideal solution (derived by the domain expert component) the bug catalogue would be used to detect the error, and the student would then be guided back to the correct solution. Anderson assumed that only one correct solution to problems generated by his architecture existed, which was not always realistic.

Hartley and Sleeman[38] also propose four components; knowledge of the domain task, a model of the student behaviour, a list of teaching operations and means-ends guidance rules which relate teaching decisions to conditions in the student model. This proposal differs from Anderson's in that the student is explicitly modelled and the domain misconceptions are not, and the user interface is replaced by guidance rules which have knowledge of how to conduct an interaction with the student.

O'Shea and his colleagues[54] suggest a five ring model which is similar to Hartley and Sleeman's in that emphasis is placed on student modelling and teaching skills. The five components are a student history which represents the material which has been presented and the student responses, a student model which predicts the student's learning potential and current state of knowledge, the teaching strategy which decides the next type of teaching action to take, the teaching generator which yields a specific teaching actions and the teaching administrator which presents information to the learner and processes the learners response. Martin[48] has outlined a more general 'figure of eight' model which allows for constrainable learner control.

Yazdani[82] has classified ICAI architectures along a spectrum, one end of which has emphasis on domain knowledge, the other end on teaching skills. He states that:

> "the position on the spectrum is not simply a matter of convictions of the individual researchers, but is influenced by the nature of the expertise which is to be taught."

As the tasks become "more concrete and specific" student models and teaching knowledge is emphasised, for example the proposals of O'Shea et al., Hartley and Sleeman, and Martin. Teaching skills which are basically problem solving in a specific domain can best be achieved via monitors which watch over problem solving, as in Anderson's architecture.

Many researchers in the field of ICAI have investigated student modelling and teaching knowledge issues rather than those associated with the modelling of domain knowledge[21,30]. In consequence, the majority of ICAI systems that have been developed have knowledge of well defined domains, for example mathematics, rather than complex and ill defined domains such as medicine.

The classic example of a medical ICAI system is GUIDON[13], a system which was developed to explore whether the knowledge base of the expert system MYCIN[66] could be used to teach students how to diagnose bacterial infections of the blood. GUIDON has access to MYCIN's domain knowledge in addition to its own teaching expertise and knowledge of how to conduct a dialogue. It conveys general medical knowledge to a student, who takes the role of physician, through the discussion of a patient case. Comparison of the student's behaviour and MYCIN's forms a basis for the discussion. GUIDON has also been interfaced with the expert system PUFF[5], which has a similar representation to MYCIN, in order to teach about pulmonary function diagnosis.

Expert systems which incorporate medical knowledge and use artificial intelligence techniques may indirectly benefit user learning. Such systems will not be considered in the teaching context but will be discussed in the following section on the representation of medical knowledge.

2.4 Representations Of Medical Knowledge

Medical expert systems may be classified into two types, compiled and model-based. Compiled systems perform only one type of reasoning and combine their reasoning and medical knowledge into one structure, much of the medical knowledge being implicit. Examples of such systems include MYCIN and PUFF, which diagnose bacterial diseases of the blood and diseases of the respiratory system respectively. Both systems are represented by production rules. They have their reasoning and factual knowledge compiled into rules, for example consider MYCIN's rule:

IF (1) The infection is meningitis, *and*
(2) the subtype of meningitis is bacterial, *and*
(3) only circumstantial evidence is available, *and*
(4) the patient is greater than 17 years old, *and*
(5) the patient is an alcoholic

THEN There is suggestive evidence that diplococcus-phneumoniae is an organism causing the meningitis.

Implicit in the rule is knowledge of a diagnostic top-down refinement strategy (expressed in premises 1 and 2); the rule is inappropriate if laboratory evidence is

Causal relationships. The causal relationships between objects or processes are explicitly stated.

Models separate structure from function. The representation of parts of a system should not assume the purposes for which the parts are to be used. In practise models are generally compacted to make them more efficient, thus some selection of what entities are to be considered is made.

The potential benefits of model-based systems over other systems are that they provide:

- An organising schema about which large amounts of data can be represented.
- A context about which knowledge acquisition may be focussed.
- A basis for better explanation since all the knowledge is accessible.
- Reusable models. Since the reasoning models are abstract, they may be used in different systems.

The main argument against model based systems is that when compared to compiled systems they are potentially slower because they reason from first principles. However, as the processing power of computers increases, the time taken to reason from first principles will fall. Chandrasekaran remarks in [11], that a compiled system can become unmanageable and difficult to understand and extend as the size of the system increases. To alleviate these problems in MDX[11], a compiled system for assisting medical diagnosis, he organised domain and problem solving knowledge about *specialist* nodes arranged in a hierarchy. The specialist nodes were invoked in a top-down fashion, analogous to an establish/refine diagnostic strategy. Bratko[49], whose work will be considered in more detail later, found that humans have difficulty in defining complete rule sets. He went on to develop a program which generated rules containing compiled knowledge of electrocardiogram (ECG) descriptions from a deep model of the heart. The rules were then used to interpret ECG's.

Price and Lee's characteristics for model-based systems are ideals. Few of the medical systems described in the literature are truly model-based. Typically they have a single purpose and do not model basic medical principles and cause in sufficient depth to support additional reasoning. However, having studied these systems four types of medical knowledge contained within them may be identified. These are:

1. Strategic knowledge describing how a complex problem may be decomposed into more manageable parts.
2. Knowledge about entities organised into hierarchies.

3. Associational knowledge relating patient findings to disease or pathological states.

4. Anatomical and physiological knowledge.

The strategic knowledge contained within the systems generally addresses diagnosis or therapy, and either attempts to emulate human cognition (for example INTERNIST-1[57] reasons like a human diagnostician) or does not (for example MYCIN and CASNET[78] which both advise on diagnosis and therapy). Although strategic knowledge is important in advice giving systems, the prototype teaching system described in this thesis does not advise on problem solving so these issues will not be discussed further.

Hierarchies are commonly used in medical expert systems to structure knowledge of diseases and pathophysiological states. The organisation of such knowledge is usually about organ system (INTERNIST-1), etiology (NEOMYCIN) or a combination of both (CADUCEUS[57]). The distribution of entities about hierarchies generally depends on their use. For diagnosis, entities are structured in a way which makes explicit any significant diagnostic information. For example, in NEOMYCIN successively lower levels of the disease hierarchy represent more specific underlying causes of disease, each disease having only one placement in the hierarchy.. The hierarchy is used in conjunction with a top down refinement diagnostic support strategy. Pathophysiological states are usually omitted from the hierarchies unless they are clinically significant. Their absence can affect the performance of the system, as demonstrated by INTERNIST-1.

A hierarchy of diseases and important pathological states was used by INTERNIST-1, no entity residing at more than one node. The hierarchy was organised about organ system, such that liver disease, lung disease and kidney disease resided near the top of the hierarchy, each having subnodes which represented more specific disease categories until at the terminal level single disease entities were described. Patient findings were associated with each node using evokes and manifests links. Evokes links expressed the extent to which the possibility of the disease may arise given the finding. Manifests links specified the strength of expectation of a finding given the presence of a disease. The association of findings to diseases without consideration of the underlying pathological processes was found restrictive. A disease may have associated different pathological states. These states may be a consequence of the diseases progression over time or different underlying causes. By combining the findings associated with all possible presentations of a disease into a single disease description mistaken diagnoses may result. A disease description may account for all a patient's findings, but on pathophysiological analysis it may be shown to be incorrect. The group went on the develop CADUCEUS[57]. This system had a more structured knowledge base comprising a combined pathophysiological and disease hierarchy. Its representation overcame the limitations of INTERNIST-1.

The systems described so far link disease findings directly to disease descriptions and do not consider the diseases underlying physiological and pathological processes, excepting INTERNIST-1 which has limited pathological knowledge. Several research projects have looked at the ways causal models, which represent the domain in terms of the physiologic and pathophysiological processes underlying disease might assist decision making, their motivation being that the more extensive the domain representation the more potential there is for reasoning about it a a generalised fashion. Hence causal modelling fits in with the ideals of model-based systems. A problem of applying causal models to medicine should be noted. Many of the underlying causal mechanisms are either unknown or controversial.

All systems which represent the causal relationships between components may be considered to be causal. In this discussion however, mathematical models will be ignored since their causal relationships are usually implicit. In the early causal models disease manifestations were structured on the basis of the pathological states giving rise to such observations. These states in turn were organised into a causal graph. Links in the graph represented *caused-by* relations. Nodes with no predecessor represented disease findings, for example jaundice and pallor. Nodes with no successor represented specific entities, for example common duct stone, tumour. In between the extreme nodes are chains of nodes representing pathological derangements, for example cholestatis, conjugated hyperbilirubinemia. Such a model underlies CASNET.

In CASNET[78], medical knowledge is represented at three abstract levels: disease categories, pathological states and patient observations. Patient observations are associated with pathological states, which are causally linked to form a directed acyclic graph. The transition through states in this graph represents disease progression. For each state transition path there exists a disease classification and treatment plan. Given a specific state transition path, pathological states leading from the path constitute expected disease progression. Having administered a treatment plan, the expected disease progression is used to assess its performance and direct further patient assessment and treatment.

A feature of CASNET is that it represents temporal knowledge of disease by way of pathological state transitions. Unlike INTERNIST-1, it is able to use temporal knowledge when making diagnostic hypotheses, making it potentially more accurate. Its disease hypotheses can be evaluated over time by comparing the patients actual condition with that predicted by the model. Finally, CASNET supports reasoning about diagnosis and treatment, whereas INTERNIST-1 supports only diagnosis. By comparison with INTERNIST's domain representation the capability of CASNET to support multiple reasoning types results from its more decomposed representation.

ABEL[56] models its causal knowledge at three levels of detail. The highest and least detailed level is the clinical level. The lowest level, the pathophysiologic level, is the most detailed. In between is the intermediate level. Within a level, nodes are linked to form a causal graph. Focal links map important nodes on one level to nodes at a higher level. Different levels of detail allow for the description of high level concepts from multiple perspectives. For example high level concepts such as diseases may be described in terms of other diseases on the same level or in terms of lower level concepts such as physiological processes. The domain model was used by ABEL to give advice about diagnosis. A *patient specific model* (PSM) was constructed by instantiating some of the nodes in the domain knowledge to represent the current hypothesis about a patients disease. Laboratory investigation results and other low level knowledge was aggregated to higher levels to from disease hypothesis. To validate the hypotheses, low level concepts causally linked to the hypothesis were sought. Although ABEL was never implemented it was also intended to advise on therapy. Its deep representation, like CASNET's potentially supports multiple types of problem solving.

Qualitative models, which have knowledge of medical principles, are appearing in the literature. However, they are often standalone, not having been incorporated into a decision support system. Qualitative modelling is most appropriately applied to systems which cannot be modelled quantitatively because insufficient quantitative knowledge exists or when a quantitative model can be described but a lack of initial or boundary conditions prevents a solution being found in all cases. Physiological systems are so characterised and therefore provide the modelling domain in many qualitative research projects.

A qualitative model comprises a structural description of a mechanism. It is represented by state variables that are described qualitatively rather than quantitatively, and a set of constraints which always hold true. For example in a qualitative model of the cardiovascular system, the state variable pulse may take the values *very high, high, slightly-elevated or* normal. To simulate the working of the mechanism, certain state values are given qualitative values. These values are processed by the constraints to infer the behaviour of the other state variables. The changes are propagated through the model, thereby generating a snapshot of the behaviour of the system being modelled. This process is called envisionment.

Bratko and his collegues[49] developed KARDIO, a qualitative model of the electrical processes in the heart. He faulted different combinations of the hearts subsystems, each fault corresponding to a particular heart disorder (arrhythmia) and ran a qualitative simulation on each. The result was a complete knowledge base of ECG descriptions, automatically generated without the assistance of a human expert. Bratko showed that such a knowledge base would have been difficult to construct using only

human expert and medical knowledge. This suggests that model-based systems may address problems which are not suitable for compiled systems relying on traditional knowledge acquisition techniques. The knowledge base of ECG descriptions was later used by an expert system[49] to interpret ECG descriptions. Bratko's work illustrates that deep models may be used to generate compiled or surface knowledge of a domain. A problem solver can then use the surface knowledge to avoid the problem of reasoning from first principles, discussed earlier. Another feature of qualitative models is that execution of them reflects their causal structure. A trace of the execution may be used to explain the compiled knowledge. Execution traces of the heart model were used to explain the ECG descriptions generated from it, demonstrating another use of the deep model.

Qualitative models have one major limitation. When conflicting forces interact, it is difficult for the envisionment process to decide how the model behaves. This problem is usually resolved by considering each of the conflicting forces separately. For example, if one force causes a variable to increase and another causes it to decrease the system would consider the effects of increasing and decreasing the variable independently. This technique may however result in a combinatorial explosion if many conflicts are found. Another strategy is to activate a quantitative model whose behaviour in exactly defined to resolve conflicting forces. This technique has been used by Widman[79], in his model of the cardiovascular system, a domain reliant on multiple homeostatic feedback loops which are a source of conflicts.

The final type of knowledge seen in the systems reviewed was anatomical and physiological knowledge organised about hierarchies whose nodes or concepts contain knowledge about a single entity. The organisation of these hierarchies was abstract, in that it does not imply a single use. PATREC and RADEX[11], two systems used by MDX, have concept hierarchies. Knowledge contained within the concepts includes:

- Concept classifications, for example PATREC has concepts to classify physical phenomena like pain, laboratory data and drugs.
- Different attributes, for example the attributes of abdominal pain include site and frequency.
- Constraints on the values of the attributes, for example the sites of abdominal pain are constrained to the abdomen.
- Default assumptions about attributes of a concept, for example laboratory tests are always assumed normal.
- Inferential knowledge to enable the derivation of knowledge about a concept based on other concepts or attributes of the concept.

RADEX has knowledge of morphological procedures, organs, organ parts and deformities. For each morphological procedure information is stored describing the organs about which information can be gleaned by the procedure and special inferences which can be made given information about an organ's appearance. For each organ, its component parts and information about it which is obtainable from morphological procedures is stored. Organ parts are described by attributes which are useful when describing the anatomical structure of organs, for example surface area.

2.5 Conclusion

In medicine, traditional computer assisted instruction systems outnumber intelligent ones. A major limitation of the traditional systems is that they have no explicit domain knowledge, limiting their teaching potential. Intelligent programs feature an explicit domain representation which can potentially make them more powerful teaching aids. However, the medical domain is complex and generally not well understood making it an unfavourable prototype domain for developing explicit domain models. Few intelligent medical instruction systems have therefore been developed.

Although intelligent computer assisted instruction systems are few, medical decision support systems and standalone models are more prolific in number. However, many of these systems contain compiled knowledge which does not make explicit the medical principles required for effective teaching. Other systems have separate problem solving knowledge and medical knowledge representations. However the organisation of the medical knowledge is usually influenced by its use, limiting its application to other types of problem solving. Teaching systems require that the domain knowledge be accessible to multiple modules (synonymous to models), the interaction between modules being system dependent. For example, the domain model may be used by the user interface to answer a student query, or by the teaching strategy module to direct the focus of a tutorial. Until more explicit model-based medical representations are developed, the potential for development of medical ICAI systems will be limited.

The system described contains a model-based representation of the medical phenomena associated with acute abdominal pain. This model, described by the domain module, is accessed by other modules in the generation of tutorials. The system architecture is presented in the following chapter, chapter 3.

Chapter 3

System Architecture

3.1 Introduction

This chapter describes the architecture of an intelligent system for teaching about a problem which is dynamic in time and on student actions. The system has embedded a simulation generator to control the evolution of a simulation model which is presented to the student in the form of a scenario. By interacting with this model a student can explore her knowledge and learn.

Intelligent systems employ knowledge-based system techniques and act as if they possess human information processing capabilities and cognition. The extent to which these systems mimic human cognitive processes varies; some propose to model human intelligence whilst others do not. Those system which do not model human intelligence use AI for its engineering potential. The architecture described uses AI for the purposes of engineering elegance.

There now follows a discussion of the teaching strategy proposed for the system and its potential psychological implications for learning. The representation schemes provided in KEE and used in the systems implementation are described. This is followed by a discussion of the knowledge representations issues to be addressed by the system and the approach taken to resolve them. Finally, the configuration of a system which utilises this representation is presented together with the learning facilities it supports. The prototype system has addressed the teaching domain of acute abdominal pain although it has more general application.

3.2 Teaching Strategy

Teaching strategies are plans for presenting subjects to a student in order that the student comprehends the subject. They endeavor to impart to a student organising principles by which her domain knowledge can be structured to achieve efficient problem solving. All strategies use a domain model to establish the material to communicate

to the student. The extent to which a student can explore the domain model without intervention is fixed by the teaching strategy. 'Teacher-centered' strategies direct the student's interaction along paths which reveal aspects of the subject domain which are believed important in the student's creation of a cognitive model of the domain. Other strategies are more 'student-centered' and do not attempt to convey an explicit cognitive model of the domain, rather they assume that the student herself will realise a model.

Instruction in the clinical environment traditionally employed 'teacher-centered' teaching strategies. Given a patient, a student learnt by monitoring the teachers management strategy. The student was given no opportunity to investigate her decision strategies if they conflicted with expert opinion.

The introduction of computers into the clinical teaching environment provides for 'student-centered' teaching strategies in addition to traditional teaching techniques. Experimentation with subject matter acquired by conventional teaching methods may be supported, enabling the student to develop her own cognitive model of the domain.

3.2.1 Teaching Strategy Used Within The System

For subjects in which students have a basic knowledge but do not possess skills in applying it, for example clinical medicine, discovery learning is a valuable teaching strategy. Tools to assist the student in discovery have advantages over real experience. For the course controller, the risks inherent in allowing students to carry out actions in hazardous domains are avoided and tutorials are not confined to problems which are easily set up. Students benefit by being able to carry out their decisions whatever the implications; instruction is tailored to the individual and can take place at times demanded by them.

To maximise the potential for learning, the teaching strategy may initiate a tutorial dialogue to focus on important domain concepts to be learnt. The applicability of such dialogues is domain dependent. For subjects which are well understood, for example mathematics and French grammar, the amount of direction by the strategy may be great. However, for domains which are not well understood, for example areas of medicine, or when many solutions to a problem exist with no preferred solution, the use of tutorial dialogues is limited. In these circumstances student initiated dialogue is more beneficial than strategy initiated.

In the literature various nomenclatures have been used to describe the teaching strategies underlying instruction systems. Discovery learning or 'reactive' learning has been used by some authors to describe systems which provide feedback on the global task and/or on the student's immediate actions. Other authors use the term 'coaching' to describe systems which advise students on the global task. Strategies which assume

a 'teacher' initiated rather than 'student' initiated tutorial dialogue have been called tutoring systems. The author prefers to classify teaching systems which have been developed using AI techniques and have teaching strategies which provide remedial feedback or allow for mixed-initiative tutorials as ICAI or ITS, and systems which do not advise the student as ICAL.

The ultimate goal of the teaching system described is to provide an environment with a teaching strategy which highlights important domain concepts and *monitors* a student's behaviour, initiating remedial advice when appropriate. The student's actions will not be inhibited. Such a teaching strategy is 'student-centered'. Teaching systems which have 'student-centered' teaching strategies require emphasis to be placed on their domain knowledge rather than on their teaching strategies. This has been identified by Yazdani in the spectrum which he devised to illustrate how the role of knowledge varies according to the underlying teaching strategy (see page 23 in chapter 2).

Before determining when and what remedial advice should be given to a student an understanding of the domain and problems to be solved within it is required. The complexity and large-scale of knowledge associated with acute abdominal pain brought about an intensive study of the domain. Construction of the domain module followed, before that of the teaching strategy. This thesis addresses the development of the domain and discovery learning with no remedial advice components of the system thus the classification ICAL system. If strategic knowledge of how to plan a diagnosis and manage a patient were added to the system and used to assess the students performance, generating remedial feedback where appropriate, the system would become an ICAI system.

3.2.2 Cognitive Science Support For Discovery Learning

A theory which attempted to explain the educational significance of computer-based learning environments has been proposed by Papert[55]. He considered an environment to comprise primitive objects and actions which could be composed by a student within the constraints of the system to derive transitional objects and actions. In transforming the objects a student interprets the transitional objects as symbols representing real world entities and the actions as operations which they can relate to everyday experience. Having recognised the surface details of the system, over time its deeper properties emerge.

Lawler[46] has derived a theory for explaining the transition from a novice to an expert using the notions of *microworlds* and *miniworlds*. He defines a miniworld to be an

> "object focussed embodiment of some designed environment, as described by an expert in the domain"

and a microworld to be

> "a partial exploration of the complete generality of what might be possible in the miniworld".

A novice's experience can be described by interactions with microworlds of a miniworld. On each interaction the novice reconfigures within her mind some cognitive structure which contains aspects of the experience which are important to her. Having gained experience with one or more microworlds of the same miniworld the student will develop a concept[1] of the miniworld relinquishing the title of novice. When the student encounters microworlds of different miniworlds she may assimilate her microworld concepts to form conceptions[2] about the miniworlds' similarities and differences. As the student becomes more expert, abstract ideas about the character of miniworlds and their relations may develop.

Papert[55] has described a simulation as a special kind of microworld, one which tries to copy a certain part of reality. Lawler's theories provide a theoretical basis on which the discovery learning teaching strategy used by the described ICAL program may be deemed beneficial to student learning.

Assuming that each disease contained within the system is a miniworld and each of its different presentations is a microworld, simulation runs of different instances of the same disease would represent microworlds of the same miniworld. Students' engaging in a tutorial would initially interact with a single microworld or disease presentation. On resetting a simulation a different presentation of the same disease may be generated, exposing the student to two microworlds of one miniworld. By experimenting with different microworlds of the same miniworld the student may develop concepts, for example in gallbladder and duct disease on physical examination a patient's abdomen is initially tender, a mass then develops followed by rigidity. Repeated interactions with the system would enable several diseases or miniworlds to be explored. Conceptions of the domain would then emerge. For example having been presented with simulations of diseases in the gallbladder and ducts and appendix, a student should notice their physical examination findings are similar. They might then form the conception that the findings are a response to infection which is shared by both diseases. Further, such a response would be seen in all diseases with underlying infection.

Feltovich and his colleagues[27] have looked at the way in which the structure of an individual's knowledge affects her problem solving, in the medical domain. Following observations of problem solvers with different amounts of clinical experience they have proposed a theory of how a disease model develops over time, as a person moves

[1] Thing which the mind conceives after knowing many instances of some category which is devoid of all details except those specific or generic

[2] Formulated or widely accepted idea of what a thing should be

from novice to expert and its implications on diagnostic performance. Each disease was modelled in terms of the pathophysiology of the disease and the set of clinical manifestations that a patient with the disease should present. For an expert the number of diseases are extensive and they are represented in a hierarchy. The upper levels represent disease categories, sets of diseases that present similarly because of physiologic or clinical comparability. The middle and lower levels represent particular diseases and specialised forms of disease respectively. The novice's disease knowledge is dependent on the training materials she uses and the training experiences she encounters. Initially her disease knowledge is sparse, lacks extensive cross-referencing and is confined to the common versions of disease. With experience diseases are augmented and generalised into categorical clusters as disease similarities are found, and discriminated into finer distinct entities as differentiation points among and within the diseases are learned. The novices clinical expectations of the disease models are often imprecise but with experience these become tuned.

Feltovich's work on the evolving structure of knowledge with clinical experience has implications for computer-based training. To encourage the student to group diseases into categories she should experience many diseases. Real clinical experiences are limited by the distribution of patients in the training setting. Computer-based training which incorporates simulations can supplement a student's real-life experience of diseases. Further, the teaching systems may provide cues to emphasize the grouping of diseases. Disease classifications may be presented explicitly to the student or in the course of solving a problem involving a specific diseases relationships to other diseases may be highlighted. This will assist in the connection of new knowledge with situational cues which are likely to exist at some time when the new knowledge will by needed in a real diagnostic encounter. ICAI systems which diagnose cases in parallel to a student are particularly suited for this as they interact with the thought processes of the learner as she carries out diagnosis. The ICAL system described assumes the student discovers a classification for the diseases which it generates. The system's disease knowledge is represented by a disease taxonomy which is of potential benefit to student learning.

3.3 Knowledge Representation

A knowledge representation describes the structure used to store knowledge. These representations determine the manner in which knowledge can be processed, and thus influence knowledge usage. In system design, knowledge is often structured in a way which provides for efficient problem solving of one type of problem. Such performance is achieved by considering only the knowledge required to carry out the

task, making implicit the the assumptions on which the problem solving was based. In contrast to such *surface* representations, *deep* representations make explicit the assumptions on which reasoning is based and separate the structural knowledge of a problem from the procedural knowledge required to reason. Deep representations are deemed to be more human-like since they allow different types of reasoning over the same store of knowledge.

The system under investigation has a deep domain knowledge representation organised about a conceptual framework. Although the model is not the *ultimate* domain representation, in the sense that it completely specifies the domain it may assist learners in their comprehension of the domain. The model is used by the system to derive a scenario for presentation to a student. It has been implemented using an object-oriented programming paradigm.

3.3.1 Nature Of Scenarios

A scenario or microworld comprises a domain problem, ie a patient presenting with acute abdominal disease, and tasks or actions which a student may carry out on it. The problem has associated domain concepts and is dynamic in time, changing as a result of natural evolution or student action. Although the problem is changing continuously over time, there exist instances when its structure undergoes discrete changes. These points correspond to times when an event occurs which may change the domain concepts underlying the problem, for example pathological progression.

The actions which a student may carry out are of two types, those which modify the domain problem and those which do not. The former may effect changes at an instance in time, or over an interval, for example treatment. The later interpret the state of the problem at an instance in time, making no modifications to it; for example requesting laboratory investigations.

3.3.2 Other Issues To Be Addressed

The knowledge representation is required to meet the following criterion:

- The domain knowledge should be represented in a manner which reflects its natural structure and the domain principles should be explicitly represented. This would enable the domain model to be easily understood by domain experts, thereby making it more amenable to evaluation and acceptance in the learning environment. Additional knowledge could be added to the structure; its position in the representation would be apparent from the basic structure of the existing concepts. This is of importance in the medical domain where new aids to assist in patient management are continuously being released and would require repre-

sentation. This would allow further diseases to be incorporated into the model too.

- Scenarios presented to the student must be non-deterministic. They should focus on domain problems which a student is capable of solving, the problem and its initial presentation being generated at random on tutorial initialisation. Over time the problem will change because of student action or lack of it. To vary the evolution of different instances of a problem, each instance being presented in a different scenario, the values of state variables should be randomised within prespecified ranges.

- During a tutorial it should be possible to reset a scenario to some prior state. In this way students can explore different diagnostic and treatment strategies which may cause the disease to follow a new course thereby creating multiple microworlds of the same miniworld.

3.3.3 Approach Taken To Model Domain

To satisfy the knowledge representation requirements a classical conceptual domain model[71] incorporating deep knowledge has been explored. This model provides an explicit representation of the mechanisms underlying domain problems. A concept is a unit by which an object or event is perceived. They are described by components or *properties* which are either quantitative, called *dimensions* or qualitative, called *features*. Properties may be perceptual, functional or abstract entities.

The classical conceptual model is based on three assumptions:

1. The representation of a concept is a summary description of an entire class, rather than a set of descriptions of various subsets or exemplars of that class.

2. The properties of a concept are singly necessary and jointly sufficient to define that concept.

3. If concept X is a subset of concept Y, all defining properties of Y are nested in X.

From these assumptions it follows that to develop a conceptual representation of a domain, the domain has to undergo an abstraction process to select the concepts to be modelled and their properties. Concepts can describe an entire class of entity, or a specific instance of it, for example the concept pathological processes describes a class, the concept infection being an exemplar of the class. Every instance of a concept must exhibit all its properties which must not be disjunctive. This condition assumes that natural concepts are never disjunctive, for example the concept house could not have

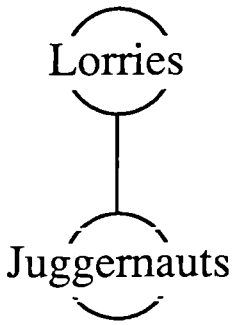


Figure 3.1: Concept Taxonomy To Express The Relationships Between Lorries And Juggernauts

the properties 'detached' and 'terrace' since for instances of house only one property would apply. Finally, every entity having the properties associated with a concept must be an instance of that concept.

Concepts are arranged in taxonomies, to express class relationships between concepts and to categorise them. Categorisation serves to determine whether a specific instance is a member of a concept or whether a particular concept is a subset of another. Using class relationships, inferences about concept properties can be made; for example consider the concepts lorries and juggernauts arranged taxonomically as shown in figure 3.1. Since juggernauts are lorries and lorries have engines the inference juggernauts have engines can be made.

The conceptualisation of a domain is dependent on the role it is to serve, the experience of the persons developing it and the comprehensibility of the domain. Chandrasekaran and Mittal[12,11] believe a domain representation in which domain concepts are implicit, is superior to a deep one for domains in which only one task type is to be carried out. Their remarks follow the development of MDX which performs medical diagnosis. Other researchers argue that for domains in which multiple tasks are to be carried out, representations in which domain concepts are explicit are better. Clancey[15] found that MYCIN's implicit knowledge representation was fine for diagnosing bacterial infections of the blood, but when he attempted to use the knowledge for teaching in the GUIDON project, the representation was inadequate. This prompted Clancey to develop NEOMYCIN, which contained an explicit version of MYCIN's strategic, support and structural knowledge.

The second factor to be considered when conceptualising a domain is the experience of its developers. Conceptual models are presupposed to be stable mental representations within individuals and across individuals. This implies that the properties of the world are shared by everybody, except for early development changes or physiological ones. Thus novices do not have the same conceptualisation of a domain

as an domain expert. This has been demonstrated in the medical domain by Feltovich [27] and discussed in the context of teaching. In this work medical experts were used in the derivation of the domain representation.

The final aspect to be taken into account when conceptualising a domain is domain comprehensibility. This has two issues, unclear cases and deriving domain properties. Unclear cases are cases which cannot be mapped to a specific concept because of disagreement by a consensus of people or uncertainty by an individual. To reconcile this problem, the classical view of conceptualisation does not stipulate that every person has the same perceptualisation of concepts, rather the properties they know of are the necessary ones. Hence unclear cases are the result of insufficient familiarisation of a domain. A second theory as to the existence of unclear cases has been proposed by Glass and Holyoak[33]. Their theory states that people have two definitions of concepts, a common and a technical one. The difference between the two conceptualisations leads to the problem of classifying unclear cases.

According to Smith and Medin[71], the problem of defining the properties of concepts is not that defining features do not exist, but that no-one has yet determined them. Possible reasons for their not being determined include the fact that perceptual properties have been sought, rather than abstract or functional ones.

If expert opinion is available and a domain is well-understood, the conceptual model derived is dependent on its use. For example, a model of medical diagnosis requires limited anatomical knowledge. If the same model was to be used to interpret Xray findings the anatomical knowledge would need to be more detailed. As a general heuristic, if a model is required to handle a different type of application in a similar domain, then this may be accounted for by a change in the role of an entity. A concept in one application might be thought of as a property in another, for example the gallbladder might be a concept in an Xray interpreting system and a property of a concept describing a disease in a medical diagnosis system. When such views occur it is an indication that the system has several levels.

The notion of multiple levels underlies the research area of knowledge depth, discussed in chapter 2. The more levels that are defined, the greater the number of concepts and possible uses to which the model may be employed.

The domain model underlying the architecture of the system under consideration could be perceived abstractly as a concept taxonomy, the root of which describes the domain in general (figure 3.2). Radiating from this node are concepts representing the major domain subdivisions. Each of these concepts are decomposed repeatedly into lower level concepts, until the primitives are reached. These may be decomposable but are selected to make the model more compact.

The domain model has multiple levels; entities are described by concepts and

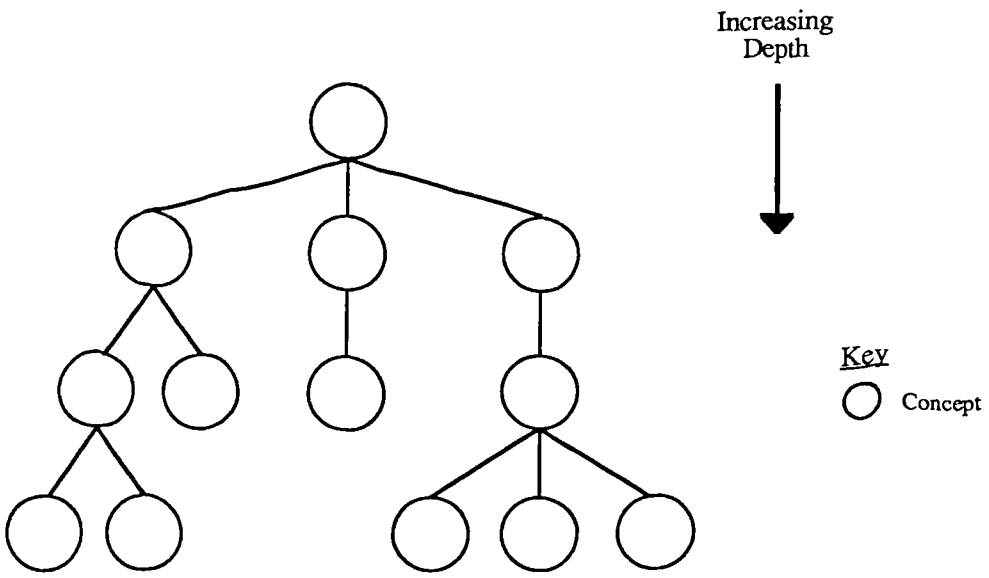


Figure 3.2: Conceptual Taxonomy

feature and dimension properties. Both numerical and symbolic reasoning is carried out using the model. This is evident by the model's dimensional and feature properties. To determine the values of the dimensional properties of a concept, functional properties associated with the concept are used. These generate numerical values which lie within some pre-defined range but aside from this are non-deterministic. That the dimensional properties of concepts may not be deterministic is not considered to conflict with the second basic assumption of classical conceptual models. It is assumed that all the dimensional properties of a concept are present, it is their value which may vary.

The concept taxonomy is used to construct a model of a domain problem. Problems are assumed to translate from one problem state to another, each problem state being described by a concept having features which describe other concepts. A problem is set-up by creating a hierarchy with root node describing the invariant properties of the problem (figure 3.3). Child nodes of the root are then created dynamically to represent the status of the problem at different intervals in time. Each child node is a subnode of the problem state it describes, and of the concepts which are features of the problem state. Using assumption three (page 38), the problem is then completely described over time by the set of all terminal nodes in the problem hierarchy. For example, consider a problem shifting between two states, A and B, over time. The problem states A and B would be represented as subconcepts of the domain problem node in the concept hierarchy. Features of each node, say x, y, z for A and l, m, n for B would also be represented by concepts in the concept hierarchy. The problem hierarchy's root concept would describe the problem's invariant properties and have two child nodes

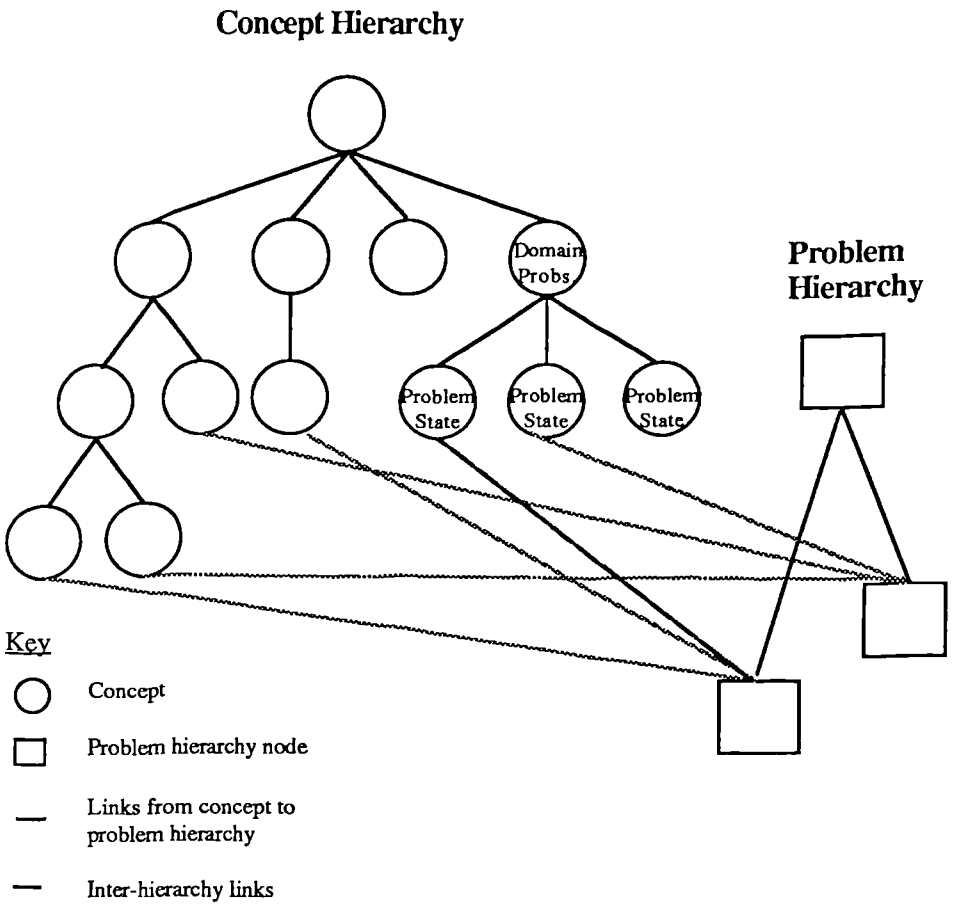


Figure 3.3: Problem Hierarchy

(terminals), one mapping to the problem state A, the other to B. The node associated with A would also have patient nodes x, y, z. Its variant properties, got from nodes A, x, y, z and invariant properties from the root, would provide a complete description of the problem during the time A was active. Similarly, the node associated with B would have properties inherited from l, m, n, the problem state B and the root node.

Application of such a domain model in the prototype system supports

A natural structure. Conceptual representations provide the foundations for theories in cognitive psychology of how people perceive the world[72].

Explanation. Domain entities are described at different levels of detail, from general to problem specific. Explanation at these levels is facilitated by processing the concept properties and connections.

Expansion. Additional 'top-level' knowledge could be added to the system without needing to consider its underlying entities. The concept taxonomy could be expanded by further categorisation of existing concepts. For example, a higher level concept could be created by aggregating lower level concepts in a combination

which had not been encountered previously.

Non-deterministic scenarios. Specific scenarios may be selected at random from those held in the 'general' domain representation. Provision for dimensional properties to be generated at random enables fine changes to the made to the top-level view of a problem. Thus different instances of a problem can be described.

Scenario reset. The problem hierarchy provides a scenario history. Scenarios may be reset at some prior point in their evolution by processing the problem hierarchy at the node describing the time interval containing the instance of time from which the scenario is to be reset. Having reset a scenario additional nodes may be added to the hierarchy on the transition to new problem states.

3.3.4 Knowledge Representation Facilities Provided By KEE

Knowledge-based systems [KBS] are able to reason about problems which have an irregular structure better than conventional programs, by their utilisation of symbolic processing. The potential application of this technique cannot be realised however, unless the knowledge which underlies it is made accessible for the purposes of inspection and modification. Tools, which provide an interface to the internal structures of a KBS are integrated into a KBS development environment. For this reason the development environment KEE was used to develop the prototype system. From within KEE, LISP[80] can be accessed. The representational structures provided in KEE that are of relevance in this work are described.

Units And Object-Oriented Programming

KEE provides a frame-based representation[7] for organising knowledge around units which have associated attributes. Units are of two types: class and member. *Class units* provide prototype descriptions of objects or entities, *member units* describe individual instances of objects or entities.

Slots associated with units describe their attributes. Two types of slots exist, member and own slots. *Member slots* occur only in class units and describe relationships involving members of the class. *Own slots* can occur in either unit type and express information which relates only to the units in which they occur.

The relationships existing between units are expressed by arranging them in hierarchies. Class units which are descendants and antecedents of other class units in a hierarchy are called *subclasses* and *superclasses* respectively. Unit hierarchies have associated an inheritance mechanism. This enables member slots to be passed down the hierarchy to subclasses as member slots and to member units as own slots. Additional information may be expressed in slots as facets which are inherited with member slots.

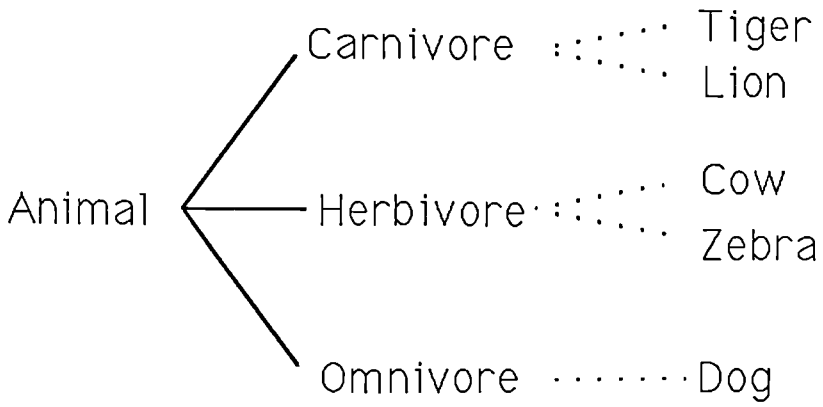


Figure 3.4: An Example Of A Unit Hierarchy

Facet types include *inheritance roles* which are rules expressing how the slot values of a units parents' are to be combined in order to derive the local slot value, *valueclass* which declares the slots datatype, *min.cardinality* and *max.cardinality* which restrict the number of values a slot can have and *default* which expresses the value of the slot if no other is given.

To illustrate features of units, consider figure 3.4. Animal, carnivore, herbivore and omnivore are all class units. Tiger and lion are instances of carnivore and are thus member units. Cow, zebra and dog are also member units. If the unit animal had as a member slot food type, this would be inherited by each of its subclasses carnivore, herbivore and omnivore. If this slot took the value meat for carnivore, it would be inherited by lion and tiger in own slots. An example of an attribute described by an own slot in a class unit is percentage of all animals which are carnivores. This would be local to the carnivore unit and would not be passed down the hierarchy.

Besides declarative knowledge, procedural knowledge can also be stored in slots as active values or methods. Methods are slots in which LISP functions are stored. By attaching different methods to the same slot in different subclasses of a class 'generic' operations can be defined, whose implementation details may vary. Methods are invoked by message passing, in which the name of the slot together with its arguments are transmitted. In this way units are able to interact with each other. Active values are functions which are attached to slots and may be activated whenever the active value is added to or removed from the slot, or the slot is accessed.

KEE supports object oriented programming. Conceptual objects within a system are represented by units, declarative knowledge describing their structure and procedural knowledge their behaviour. The inheritance mechanism enables instances of objects to be formed which share facts defined by their parents. Objects are active rather than passive structures, communicating with other objects by message passing.

Worlds

Worlds allow different views of a model to be maintained, where each world has associated a different set of assumptions. The model is represented in the *background* using units and objects. Facts or assumptions which are true in the background are also true in every world. The model may be viewed from the background or from different worlds, each view being dependent on the assumptions associated with it. Assumptions which are peculiar to worlds, these may be own values or unstructured facts, are held in assumption sets connected to the worlds.

Worlds may be grouped into hierarchies, where a child world inherits its parents assumptions by default. These assumptions can then be added to or removed from a world.

Assumption-based Truth Maintenance System

The assumption-based truth maintenance system ATMS provided by KEE, is an extension of De Kleer's ATMS[29] to include a contextual mechanism (worlds), nonmonotonic assumption retraction and integration into the frame system. De Kleer's work was based on Doyle's TMS[23]. The TMS was developed to handle inconsistencies which may arise when assertions are added to a model. In a logical knowledge base any derived facts that have been inferred from the inconsistency must be invalidated and removed to restore consistency. The TMS keeps a single consistent view of the world recording reasons or justification for each currently held belief. When new beliefs are introduced which render previous beliefs invalid, the assumptions on which the inconsistent beliefs were made have to be found. To do this Doyle uses dependency-directed backtracking. All beliefs supporting the contradiction are traced to uncover the assumptions leading to it. Assumptions are then removed until the contradiction is removed.

De Kleer uses worlds to hold different assumption sets. When a contradiction is found (an inconsistency, value class or cardinality violation) the world in which the contradiction occurs is invalidated (made *nogood*) and reasoning continues from another world which has associated a different set of consistent beliefs. In the ATMS the assumptions on which facts are based are remembered, having been set-up by deduction rules or attached to slots. Worlds then see deduced facts if their assumption set contains assumptions which support the deduction. Multiple deductions may be made since the ATMS uses a monotonic logic. In summary, the ATMS records justifications for beliefs, propagates justifications on the basis of new derivations and ensures appropriate derived facts are visible at any time.

Same-World Action Rules

Same world action rules are similar to production rules and enable the state of the background or a world to be modified. They can add or delete facts and run LISP code. All the conditions in the rule must be true in a world or in the background for the rule to be applicable. Logical connectives, AND, NOT and OR may be used to combine conditions. The rule conclusion is applied to the same context in which the conditions hold. The LISP forms in rules may contain variables which represent slots or units. The variables are assigned values when testing rules and become bound if a rule fires.

To invoke rules, forward and backward chainers are provided. Rules testing may be confined to specific worlds or to the background. The conflict resolution strategy used is least premise complexity. Rules may be grouped together in classes to achieve controlled problem solving, chaining being carried out on a class.

KEEPictures

KEEPictures is an object-oriented graphics toolkit. It contains a variety of picture primitives, each of which is represented as a class unit in the KEEPictures knowledge base. The uniqueness of each primitive is described by its units slots. Each slot represents a primitive attribute. Pictures may be customised by changing the values of attribute slots. Primitive pictures include:

- Boxstrings. These are rectangles containing text.
- Axes images. These can be used to construct graphs.
- Bitmaps.
- Straight lines with or without arrowheads.
- Polylines. These are groups of crooked lines, where each crooked line is a connected sequence of line segments with dots (or bitmaps) drawn at the endpoints. The line segments can be of variable thickness.

To design a picture, firstly a KEEpicture window, called a *viewport* has to be created. Pictures can then be placed onto this by instantiating their picture classes.

ActiveImages3

The ActiveImages3 package provides a variety of graphic displays for viewing and modifying slot values in KEE units. These graphic images may be integrated into an application.

Active images may be grouped together on the screen by attaching them to *image panels.* Each active image is attached to a slot and can be used either to display the slot values or to display and change the slot value. As the value of the slot changes, the active image is modified to reflect the change. Each active image is built up from KEEpicture units and can therefore by customised to meet the users needs.

3.3.5 KEE Data Structures Used To Implement The Domain Model

The domain and problem hierarchies have been implemented using KEE unit hierarchies. In the domain model each concept represented by a unit, with member slots describing its associated properties. KEE's inheritance mechanism ensures properties are passed down the hierarchy satisfying the third assumption of the conceptual model, page 38. Slots have a valueclass facet to provide typing. Dimensional properties may have the valueclass *number*; slots representing feature properties which are described by concepts represented at a different level in the hierarchy have valueclass *SUBCLASS.OF* or *MEMBER.OF* and those describing functional properties have the valueclass *method.*

The problem hierarchy is created dynamically in time. On initialising a scenario a class unit is created to represent the root unit of the hierarchy. This has member slots representing invariant properties of the problem and properties which are described by other domain concepts and are common to all problem states. The slots describing the variant properties have attached active values. The initial problem state is generated, call it *state1*, by creating a member unit of the class unit representing the root node of the hierarchy. This will inherit as own slots all member slots held in the root. The concept describing the initial problem state, a class unit in the domain hierarchy, is then linked to *state1* using a member link ie. *state1* describes an instance of the problem state. All properties associated with the problem state are passed to *state1* by inheritance. Any properties which are inherited from both the problem state and the root of the problem hierarchy are assumed to represent concepts in the domain model. Such properties are described by the slots which have attached active values. The active values are triggered when values are put into the slots they watch over, resulting in the creation of further member links from concepts in the domain hierarchy to *state1.* When inheritance is complete, *state1* will describe all the domain properties associated with the problem state.

Having set-up the problem, the next factor to be considered is its change in time. To overcome this problem, KEE worlds have been used (figure 3.5). Each problem state has associated a world, *state1* being associated with the 'current' world *world1.* The time at which the world is created and its associated problem state (a member unit in the problem hierarchy) are saved as world properties. All changes in the problem

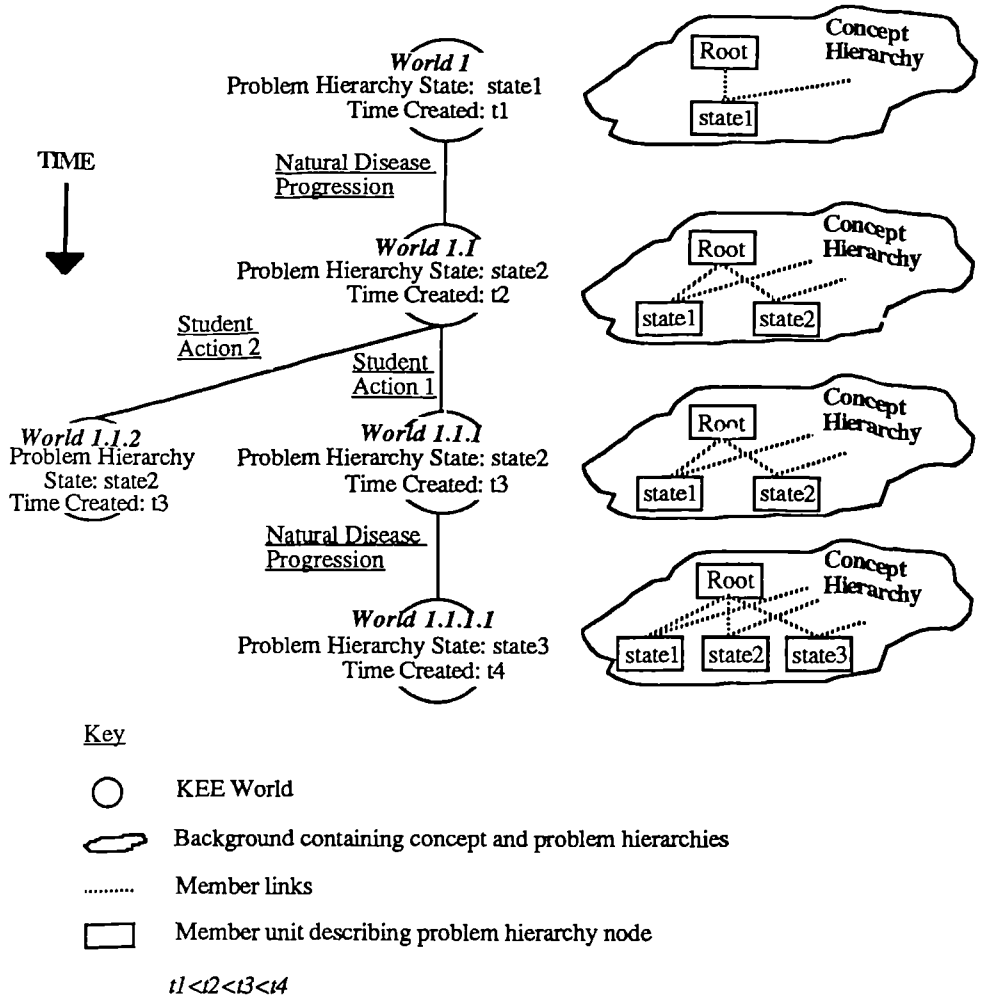


Figure 3.5: Representation Of Domain Problem Using KEE

which do not constitute a critical event[3] are stored as world assumptions. Since the states of the problem are described by a member unit with own slots, world views of these slots are permissible. When a critical event occurs, a child world of the current world is created and depending on the event type, a child node of the root node of the problem hierarchy may also be created. The child world inherits all the assumptions of its parent world and has properties describing its time of creation and the name of the node in the problem hierarchy with which it is associated. Expansion of the problem and world hierarchies continues until the simulation ends.

Using worlds' to represent different views of the problem enables partitioning of the scenario history. The benefit of this is that the scenario can easily be reset to some point back in time. To do this, the world from which the scenario is to be reset

[3]Critical events are events which result in a transition from one problem state to another, for example pathological progression, resetting of the simulation, and user requests such as the request for a special investigation.

has to be found. A child world of this world is then created. This world provides the required context from which to reinitiate the scenario since all its parents assumptions are inherited. The parent world's property list is accessed to find the problem state to associate with the child world. Figure 3.5 shows a reset of a simulation from time *t3*. The new world *World 1.1.2* represents the current context and is associated with the problem state *state 2*.

Actions a student may invoke on the domain problem are represented by same-world action rules and methods, depending on the action type. Actions which modify the domain are described by methods. They may invoke changes at an instance of time, for example surgery, or over an interval of time, for example fluid therapy. Actions which do not modify the domain but inspect it, for example laboratory investigations results are represented by rules.

3.4 System Components And Their Configuration

The prototype system comprises five modules each of which perform distinct functions; the domain knowledge, simulation generator, simulation model and student history, tutorial manager and user interface modules (figure 3.6). The simulation model and student history are combined within the same module because although both knowledge types are explicit it is not possible, in the current implementation to save the two in separate knowledge bases. A knowledge-based discrete event simulator is used to derive an interactive simulation model which can be used for experimentation purposes by the student.

This scheme differs from Sophies' I and II[8] in that the system described has an embedded simulation whereas they access a general-purpose electronic circuit simulator. In all the Sophie systems the functionality of the simulator is to provide the core of an inference engine. Sophie sets up experiments on the simulator to infer appropriate responses to student queries, for example, measurement requests, or to evaluate student hypotheses. Thus the simulator is invoked only to critique problem solving plans and to infer the responses of queries, never initiating changes in the problem to be solved.

Woolf and colleagues[81] have incorporated a simulation module in their Recovery Boiler tutor architecture. The purpose of this is to update the state variables every 1-2 seconds of real time.

An overview of the function and knowledge associated with each module follows, together with the process structure used to synchronize module processing.

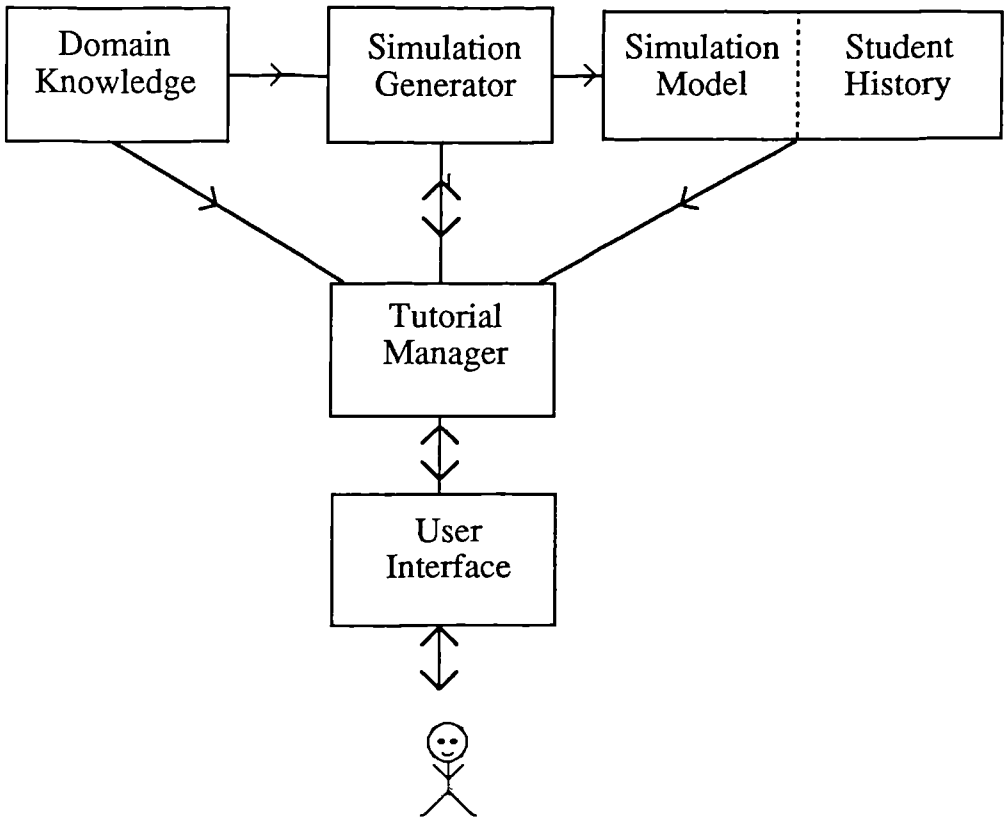


Figure 3.6: Architecture Of A Prototype ICAL System

3.4.1 Domain Knowledge

This module provides the general domain knowledge which is used by the simulation generator to construct an instance of a problem. Domain knowledge needed to service student requests is also held.

3.4.2 Simulation Generator

The simulation generator has a general structure which is described in this section. The application of this is discussed in chapter 6. Generation of a scenario in real time is controlled by this module. A problem instance is selected from the domain knowledge module to match a student's competence. The student's problem solving ability is stored in the tutorial manager. Having selected a problem deeper knowledge of the problems underlying concepts is used to set-up the simulation model.

On initialisation of a simulation run, a hospital date and time are generated at random and used to initialise *Simulated.real.time.start*. After fixed intervals of real time have elapsed the simulation model is updated.

The world view of the simulation is a model comprising objects which have associated behaviors (represented in methods). To update the model, certain objects are activated by passing messages to their methods. Further objects may become

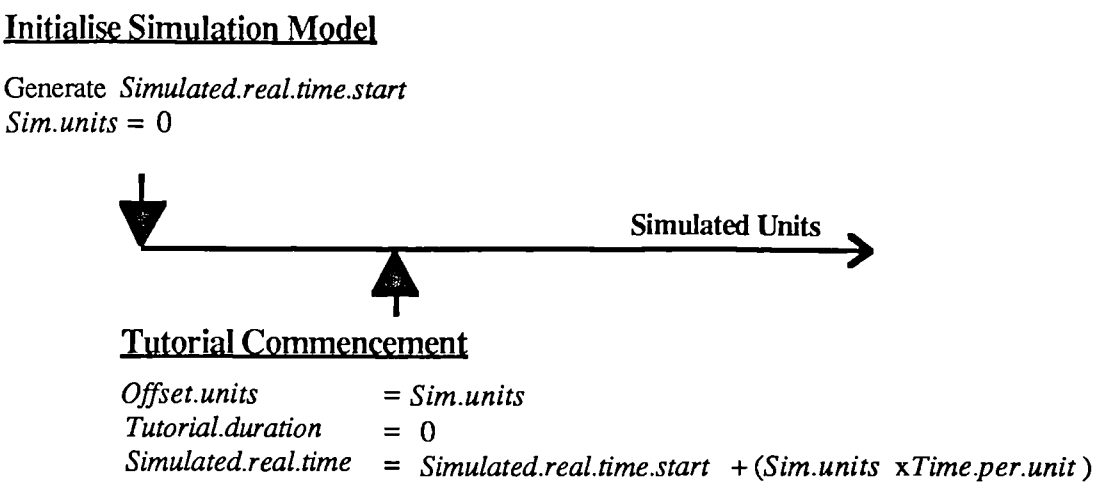


Figure 3.7: Relationship Between Simulated, Real And Tutorial Time

activated, as methods propagate messages through the model. Two lists are used to store events which operate on the simulation model, the next-event-list and the conditional-event-list. Objects with associated methods and other procedures which are invoked on every update are stored in the next-event-list whilst those which are invoked only when specified conditions are true are held in the conditional-event-list together with their associated conditions. On each update *Sim.units* is incremented by one, all elements of the next-event-list are executed together with those whose conditions are true in the conditional-event-list. After both lists have been processed an interrupt is set to cause a time delay before the next update. The delay may be overridden by the tutorial manager on certain student actions such as the advancement of simulated time or the request of operations which result in updating of the simulation model. This process continues until termination time is reached.

Certain problems are presented to the student at an advanced state and thus require the simulation to be updated *Offset.units* before introducing it to the student. To determine the number of simulated units which have elapsed during the tutorial, *Tutorial.duration*, the following relationship is used:

$$Tutorial.duration = Sim.units - Offset.units$$

Figure 3.7 illustrates the schema used to model time. *Simulated.real.time* represents the current hospital time being simulated. *Time.per.unit* describes the real time represented by each unit of simulated time and is assumed to be one in the prototype system.

3.4.3 Simulation Model

The simulation model comprises a history of the problem and its development over time. This model is constructed by the simulation generator. In generating a response to student queries, the tutorial manager may use the model to infer information. When the problem reaches a critical state, warnings may be sent by it to the tutorial manager.

3.4.4 Student History

This module contains a record of all the student actions carried out on the simulation model. At the end of a scenario the student history provides the focus for explanation of the domain problem. Also, it defines the points in time from which a scenario may be reset.

3.4.5 Tutorial Manager

The tutorial manager controls the time scales used by the simulation generator, selects appropriate problems, handles the servicing of student actions, generates a tutorial summary and passes warnings from the simulation model to the user interface. The time scales controlled by this module are *Time.per.unit* and *Tutorial.duration* which represent the elapsed real time between each simulation model update and the number of unit updates to be processed during the tutorial assuming processing is not prematurely terminated, respectively. Domain problems are categorised by the tutorial manager to match a students problem solving ability. Having obtained a measure of the student's competence from the course tutor, the module sends a problem to the simulation generator for setup. On student requests for action, the domain knowledge module is accessed to determine the general task knowledge associated with the action. Depending on this, the simulation model is inspected or the simulation generator is activated. At the end of a simulation, a depth first search of the student history is performed and explanation of the domain problem made. Explanation is achieved by describing the active concepts and their relationships at different times during a simulation.

3.4.6 User Interface

This module has knowledge of the screen display and mouse, which it uses to present messages and images to the user, and to recognise and comprehend user actions. User actions are of two types; A and B. Type A actions query the simulation model for knowledge which is passed to the student after a possible time delay. Type B actions modify the simulation model at a discrete point in time or over an interval of time.

3.4.7 Component Processes

The system has been designed for development in a multi-processing environment. Separate processes control the simulation generation and management of user requests. To ensure that the simulation generator and tutorial manager do not simultaneously access the simulation model a monitor is used. The monitor lock, which can be held by only one process, must be requested by any process intending to access the simulation model. If the model is not being accessed, the monitor lock is given to a process on its request. Whilst the lock is held by a process, further processes requesting it are blocked until the lock is surrendered. In this way the possibility of rendering the domain model inconsistent as a result of some unpredicted change following the combined effects of the simulation generator and tutorial manager is avoided.

The simulation generator process is repeatedly scheduled for activation after some predefined time delay. Assuming the tutorial manager process does not have the monitor lock, the simulation process updates the simulation after the time delay. If the monitor lock is not free, the process waits until the tutorial manager process finishes its execution and the lock becomes free. In the event of a user request when the simulation process is in possession of the lock, the tutorial manager process waits for the simulation process to be suspended before servicing the request.

When the maximum tutorial duration time has been reached, the simulation generator process initiates a tutorial summary and offers reset of the simulation. If simulation reset is not requested the process is removed from the process list, otherwise simulation generation proceeds as before.

3.5 Discussion

The architecture underlying the system being developed has general application. The domain model is represented as a conceptual model with concepts representing various views or levels of abstraction of the domain. A domain problem is derived dynamically over time by composing concepts. This problem or microworld is presented to a student together with actions which may be performed on it, in a scenario. By interacting with the problem its evolution over time may be modified. The problem may be reset to explore the outcome of applying alternative operations to it thereby exposing the student to multiple microworlds of a miniworld. Applying Lawler's theory of learning, experience with different miniworlds results in the student's comprehension of the domain.

The conceptualisation of the prototype domain, acute abdominal pain, is described in chapters 4 and 5. Miniworlds are assumed to be diseases and microworlds disease instances. Chapters 4 and 5 describe disease pathology and anatomy, and phys-

iology respectively. The acquisition of the domain knowledge is described to reveal how the problems which have been discussed and are inherent in domain conceptualisation have been overcome. Knowledge representation and implementation in KEE is also detailed. The simulation generator and model, student history and tutorial manager are focussed on in chapter 6. Actions which may be carried out on the model and the user interface are described in chapter 7. By increasing the number of miniworlds known to the model, its teaching potential is expanded. Augmentation of the domain concepts to extend the number of miniworlds is discussed in chapter 8.

Chapter 4

Pathology and Anatomy Associated With Acute Abdominal Disease

4.1 Introduction

A model facilitating the representation of all acute abdominal diseases has been developed. Knowledge of diseases has been represented at a deep level, in terms of the underlying pathology, anatomy and aetiology. Disease progression through time is described by a state transition network. Each disease state has associated pathological processes which determine some of the clinical symptoms and physical signs accompanying the disease, a physical cause and an anatomical site. Other disease attributes are generated by considering the sign and symptom complexes resulting from disruptions to normal physiological processes brought about by disease etiology. These complexes have been incorporated into an anatomical model and are setup when their cause presents in a disease. For example, the symptom complex biliary colic becomes activated when the biliary ducts are obstructed a gallstone.

As a disease moves from state to state, its physical cause may move to a different anatomical site possibly resulting in changing sign and symptom complexes. Also the disease pathology may change.

Following a description of the knowledge acquisition process, the resulting model of diseases and their anatomy and pathology is given. Diseases of the gallbladder and ducts have been focussed upon since this is representative of the domain as a whole. For each disease, its disease states provide a complete account of its possible presentations. From these prototype descriptions, a disease instance may be generated. The method for selecting an instance of disease presentation and the derivation of its clinical symptoms and physical signs is discussed. Implementation of the model in KEE

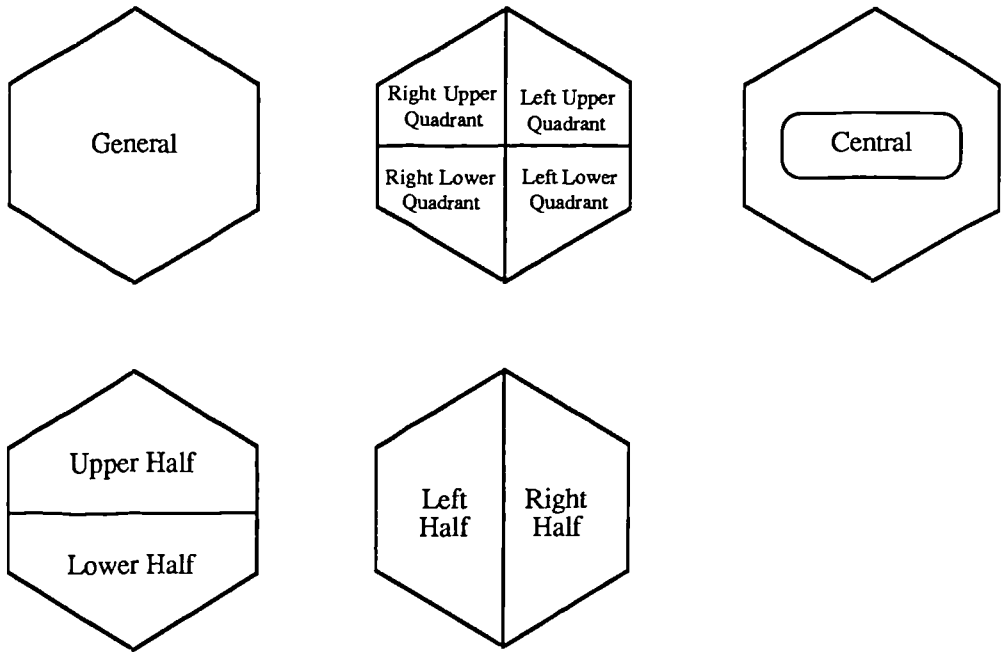


Figure 4.1: System Used To Classify The Site Of Acute Abdominal Pain

is detailed.

4.2 Knowledge Acquisition

The knowledge acquisition comprised three stages. During the first stage diseases occurring throughout the abdomen were considered in non-detailed terms. In the second stage diseases of the biliary tract were focussed on. A model describing these diseases was developed during stage three.

4.2.1 Stage One

During this stage discussions between the knowledge engineer and a domain expert took place. Diseases most frequently presenting to the Accident and Emergency Department as acute abdominal pain were identified and examined in a non-technical manner. Twenty diseases were recognised, seven of which had non-abdominal causes. The diseases were mapped onto a pain site classification system after DeDombal[19], whereby each disease resides at one of ten distinct sites (figure 4.1). To further sub-classify the diseases their pain character, ache or colic, was used (figure 4.2).

At the end of this stage the knowledge engineer produced a report describing each disease in terms of a progression network, nodes of which represented typical stages in a disease's development. A prototype description of each node was produced using a frame notation. Disease features of relevance to its diagnosis were represented by slots. Such features include history, clinical and physical findings and contributory factors like age and sex. Figure 4.3 illustrates the progression network describing the

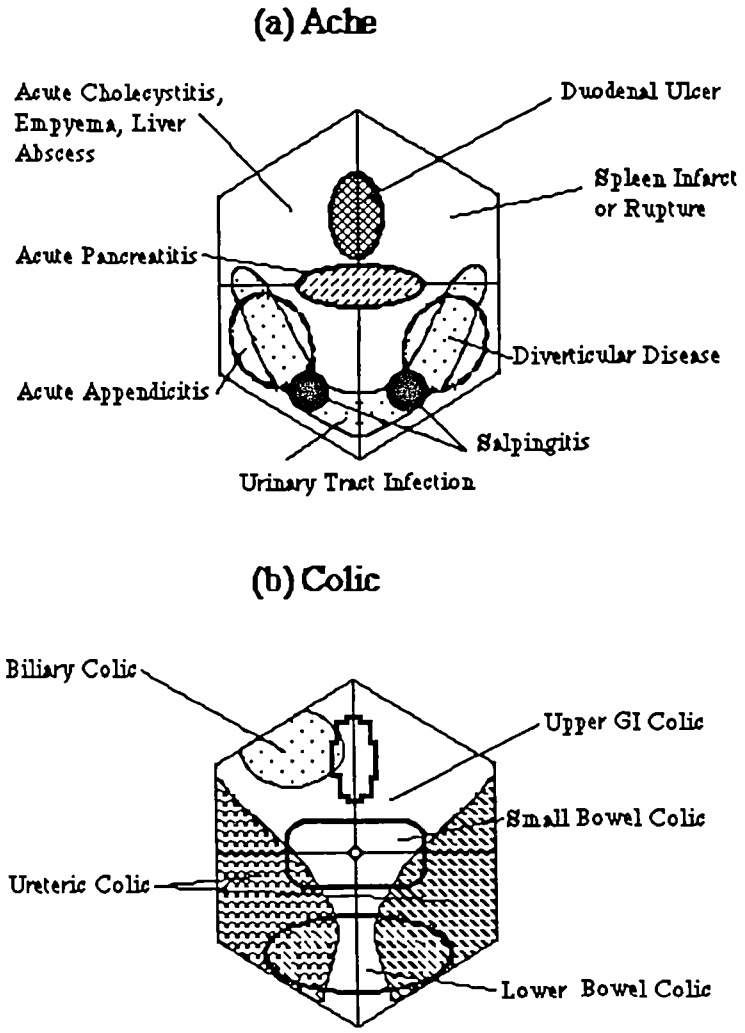


Figure 4.2: Disease Sites Of Diseases With Pain Of Type (a) Ache And (b) Colic

Disease State:	Acute Appendicitis
Symptoms and Clinical Signs:	Anorexia
	Thirst
	Constipation or Diarrhoea
	Respiration (shallow or faster)
	Tenderness in the right iliac fossa
	Rectal Examination (tender)
Haemodynamic Criteria:	Blood Pressure (normal)
	Temperature (37-38^{o}C)
	Pulse (> 90 beats per minute)
Contributory Factors:	Age (< 20 years)
	History no previous illness

Figure 4.4: Frame Describing The Disease State Acute Appendicitis

tigations, differential diagnosis, treatment and anatomical abnormalities were found. An anatomical abnormality is defined to be a physical object causing a deformity within a part of the anatomy, for example a gallstone causing an obstruction of the common bile duct.

To rationalise certain clinical attributes the abdomen's anatomy and physiological structure was used. These attributes were:

Pain Site At disease onset pain generally locates at the site where its associated anatomy lay in the embryo. The alimentary canal began as a hollow tube, the top, middle and bottom sections corresponding to the foregut, midgut and hindgut. The anatomy affected by a disease maps to one of these sections and from this the location of pain may be determined. Anatomy contained within the foregut, midgut and hindgut results in upper, middle or central and lower abdominal pain respectively. As a disease develops the pain locates at the affected anatomy's current site. This explains why the pain of acute appendicitis moves from the central abdomen to the right iliac fossa as the disease develops.

Radiation The spinal nerves determine where pain is felt. Areas of the body sharing the same spinal nerves may all reflect pain when not all are diseased. Hence the radiation of pain to the tip of the scapula, epigastrium and centre of the back in

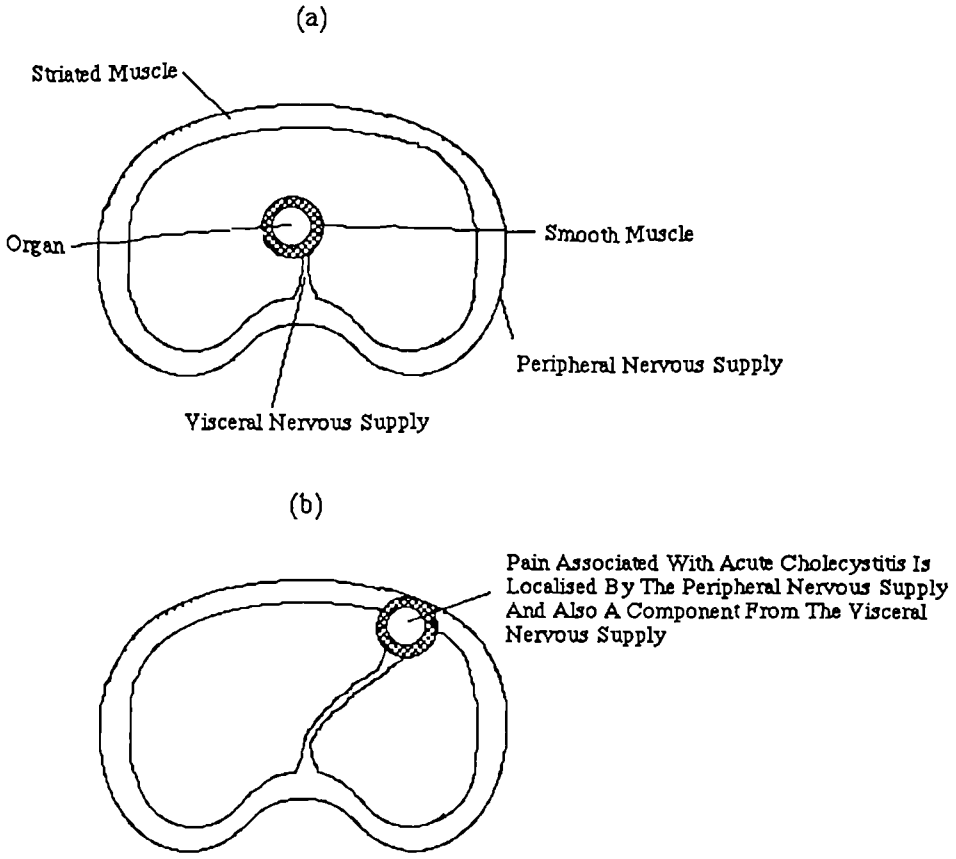


Figure 4.5: Nervous systems and muscle types of the acute abdomen when (a) organs are normal and (b) organs are inflammed

acute cholecystitis and abdominal pain with non-abdominal causes.

Localised/Diffuse Pain The nervous systems of the abdomen are shown in figure 4.5. Organs which are supplied by visceral nerves have associated diffuse pain when they become diseased. Those supplied by the peripheral nerves have associated localised pain. Over time the amount of inflammation in an organ may increase, causing the organ to impinge of the periphery of the abdomen. A disease presenting in this manner would be characterised by diffuse pain which became localised.

Muscle Tone and Inflammation Organs are surrounded by smooth muscle which losses its function when inflamed (figure 4.5). With inflammation there is a local reflex action which increases muscle tone to produce guarding of the area. When the inflammation becomes generalised the whole abdomen becomes rigid.

4.2.3 Stage Three

The disease state descriptions derived in stage two lead to the recognition of patterns of attributes common to different states. A domain expert revealed that disease attributes are related to pathology, symptom and sign complexes and pain character. The pathological processes associated with abdominal diseases were then reviewed using specialist pathological textbooks [36,51,77] supplemented by discussions with two experts. Clinical symptoms and physical signs manifested by the pathological processes were determined. The complexes biliary colic and jaundice were similarly assessed. It was observed that some anatomical abnormalities cause the symptom and sign complexes. This lead to the development of an anatomical model which had knowledge of the symptom complexes invoked by certain disease aetiology.

The disease state descriptions found in stage two were then reconsidered, the disease attributes pertaining to pathological processes and symptom complexes being abstracted out of the disease state specification and replaced by their appropriate cause. A complete model of disease progression associated with gallstones and cancer of the biliary tract was constructed. This incorporated general knowledge of pathological processes, anatomy and symptom and sign complexes. In this way the domain was represented at a deep level.

4.3 Model Of Disease Pathology And Anatomy

The model has explicitly represented general knowledge of patient profiles, diseases, anatomy, symptom complexes and types of pain. By abstracting knowledge of different types out of the disease specification, knowledge is not duplicated and modification of it is assisted. In appendix A a summary of the declarative domain knowledge assoicated with the model is given.

4.3.1 Patient Profile

Patient profile knowledge provides a description of a patient's previous medical history, name, age, sex and weight. These factors determine the likelihood of disease presentation [18].

Statistics on the incidence of different presentations of abdominal disease for males and females of various age groups is maintained by regional health authorities. The World Congress of the Organisation Of Gastroenterology has found that in temperate regions the incidence of abdominal disease does not vary considerably [17] so it was felt that the Merseyside Regional Health Authority's statistics would be representative of Western Europe. However, the disease classification standard used by the authority was not shared by the domain experts taking part in the project and although the data

Age Range	*Males*		*Females*	
	μ	σ	μ	σ
16-19	65.6	10.3	57.1	9.5
20-24	71.4	10.7	59.2	9.5
25-29	73	10.6	59.9	10.2
30-34	74.9	11.3	61.2	10.2
35-39	75.7	11.9	62.2	11.4
40-44	76.4	11.8	63.9	12.1
45-49	77.4	11.9	64.30001	11.5
50-54	75.2	10.9	64.6	12.6
55-59	73.80001	11.8	64.7	11.2
60-64	74.2	11.9	64.5	11.5

Table 4.1: Normal Distributions Of Weight By Age And Sex

was reputed to be widely available in practice it was difficult to obtain. Also other workers had analysed 6000 patient cases before arriving at disease likelihood measures [18]. It was decided that the effort required to get the data in a suitable form was not feasible given the general research aims. When modelling disease, sex and age are randomly generated (in the range 16 to 64). This age range is used to avoid the special presentation of diseases in the very young and elderly patient.

Patient weights are generated randomly from normal distributions of age and weight [62]. A separate distribution is available for either sex (figure 4.1). The polar method [45] is used to produce a random number, X_1, from the normal distribution $N(\mu, \sigma^2)$ as follows:

1. Generate U_1 and U_2 which come from the uniform probability distribution $U(0, 1)$.
2. Let $V_i = 2\ U_i$ for $i = 1, 2$ and let $W = V_1^2 + V_2^2$.
3. If $W > 1$, go back to stage 1. Otherwise let $Y = [(-2 \ln W)]^{0.5}$.
4. Generate $X_1 = V_1 Y$, a $N(0, 1)$ random variable.
5. Transform X_1 so that it comes from $N(\mu, \sigma^2)$ by setting $X_1 = \mu + \sigma X_1$.

4.3.2 Disease Knowledge

Disease progression over time is represented by a succession of disease states, each of which is active for a period of time satisfying a given range or until corrective treatment is given. All possible progression paths a disease may follow are described by a state transition network, figure 4.6. The arcs in the network represent disease progression, the arrows shown the direction in which states are related. States may become more or less severe. Each disease state has associated pathological processes

which are present over the entire state duration. On state transition the pathology may change mapping disease progression to pathological progression.

A disease may have multiple physical causes, for example tumour or gallstones. On setting up a disease a single cause is selected and is assumed to be present throughout the progression. For each disease state in a progression the physical cause resides in an anatomical part, resulting in a possible deformity, for example a partial obstruction. On state transition the physical cause may occupy a different anatomical part and the deformity may change. For example, gallstones in the gallbladder (with no associated abnormalities) may move to the common bile duct causing a partial obstruction, on the transition between the disease states asymptomatic gallstones and obstruction of the biliary tract.

The gross anatomical structure affected by a disease is used to classify its disease states. All diseases associated with the biliary tract are mapped to the pointer GB.AND.DUCT.DISEASES. In this way the disease taxonomy is constructed (figure 4.7).

The surgical treatment required to correct the disease at each state in a disease progression is given in the disease state specification. Where multiple treatments exist, they are all described. On selection of one of these treatments, the correct diagnosis is assumed and the simulation terminates. Disease states also have attributes describing their 'current' pain sites and the sites at disease onset.

4.3.3 Pathological Processes

The pathological processes which have been represented are shown in figure 4.8. They are arranged in a hierarchy to express *type-of* relationships, ie. localised and generalised peritonitis are progressive forms of infection, obstruction of the biliary tract and obstruction of the gastrointestinal tract are types of obstruction.

Each pathological process has clinical symptoms and physical signs. These are constant over the disease states for which they are active. For disease states with multiple pathologies the union of the disease attributes associated with each pathology is assumed present. An example of a combined pathology is infection and inflammation, inflammation being a response to infection. Any disease with infection has inflammation also. Inflammation may exist as a single pathology, for example in the disease state acute pancreatitis.

Each pathological type may be associated with the pain character ache or colic. For diseases with multiple pathologies no combinations of disease states occur which have pain characters of both ache and colic. The pain character is assumed constant over a disease state but may change on state transition if the pathological processes underlying the new state are different to those of the previous state.

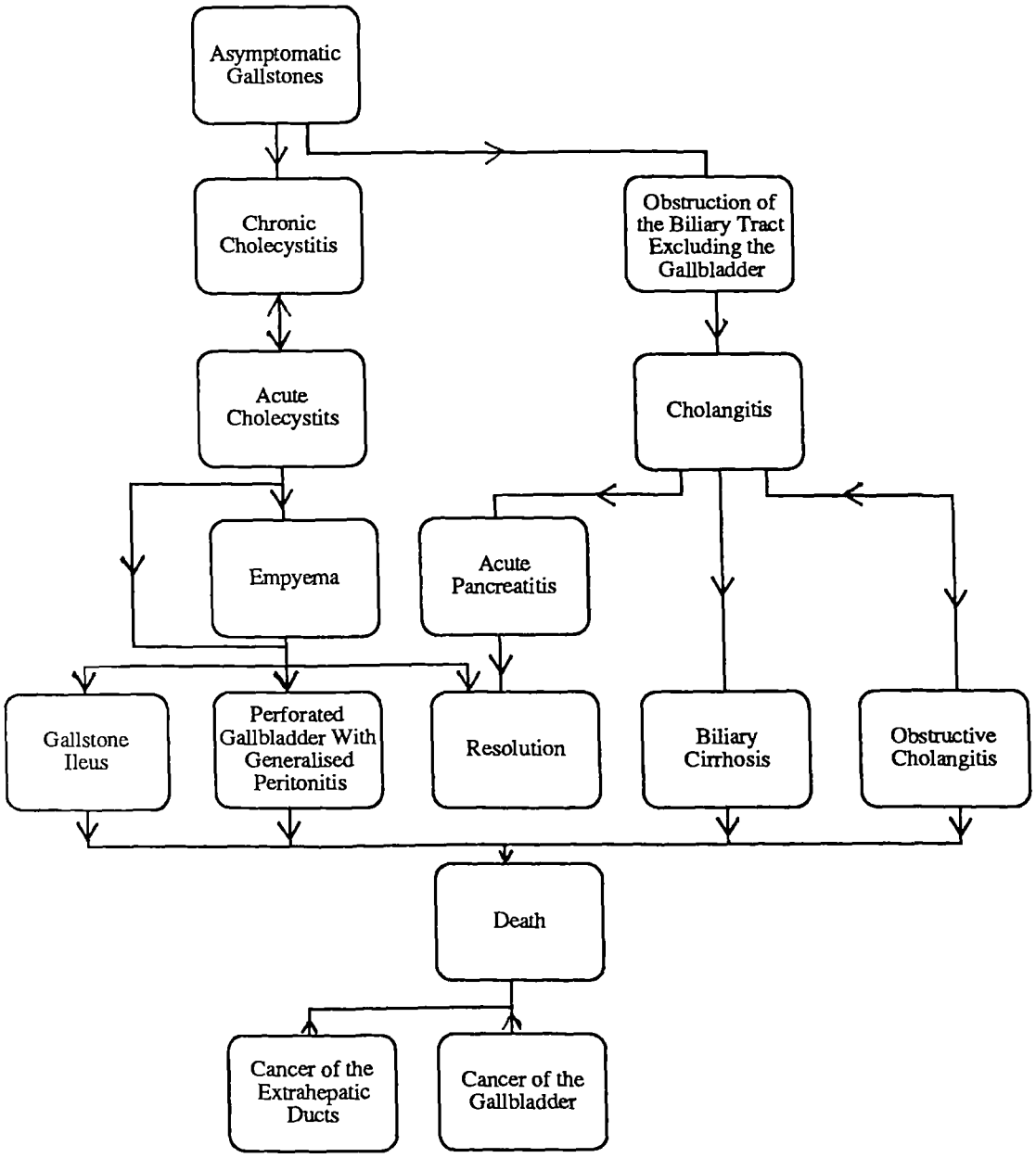


Figure 4.6: State Transition Network For Biliary Tract Disease

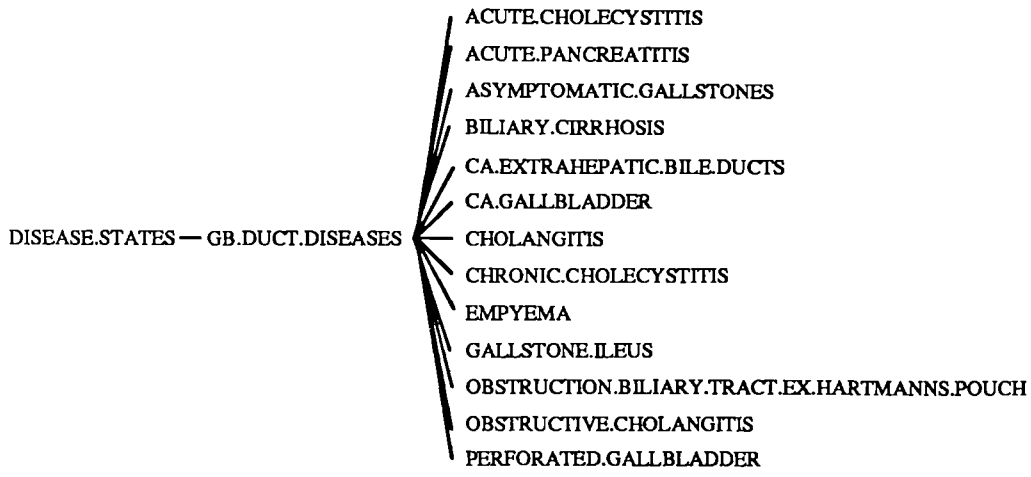


Figure 4.7: Disease Taxonomy

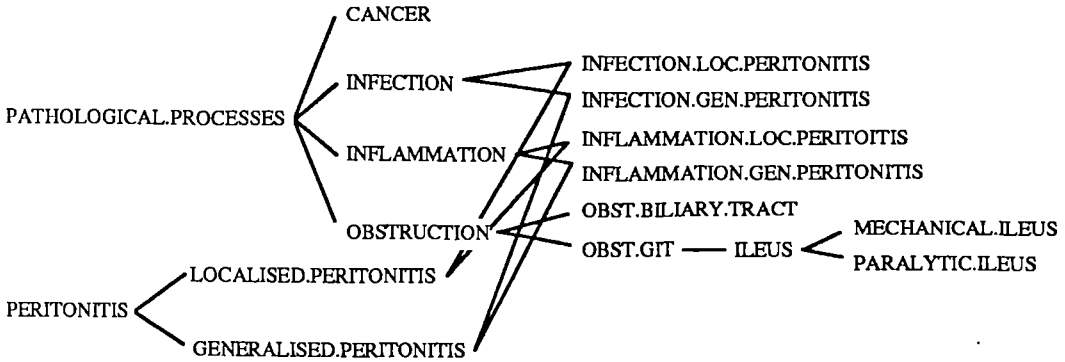


Figure 4.8: Pathological Process Hierarchy

4.3.4 Anatomy

The anatomy of the gastointestinal tract (figure 4.9) is represented by the hierarchy shown in figure 4.10. Each terminal node in this hierarchy represents an anatomical component which may be an anatomical part or a structure containing parts. Each anatomical part has knowledge of the parts it is connected to and the structure in which it is contained. Structures have knowledge of their subparts. For example, the gallbladder is a structure which has subparts Hartmann's pouch and the gallbladder body.

Using the knowledge of connections existing between subparts the connections of structures can be determined. For example, the cystic duct is connected to the common bile duct and Hartmann's pouch, Hartmann's pouch is connected to the cystic duct and the gallbladder body and the gallbladder body is connected to Hartmann's pouch (figure 4.11)[34]. Since the gallbladder contains the set of parts (Hartmann's pouch, gallbladder body) and these are connected to the set of parts (Hartmann's pouch, cystic duct, gallbladder body) the gallbladder must be connected to the cystic duct. The cystic duct is a subpart of the structure extrahepatic ducts, so the gallbladder connects to the extrahepatic ducts via the cystic duct. In this way the deep knowledge of anatomy is used to infer higher level knowledge.

The anatomical hierarchy is used to represent a patient's current anatomical status given a disease state. The cause of the disease state is inserted into an anatomical part and is propagated to any structure containing the part. For example, given that 'gallstones are in Hartmann's pouch' the relationship 'gallstones are present in the gallbladder' is inferred using the knowledge that Hartmann's pouch is a subpart of the gallbladder.

An anatomical abnormality caused by a disease is described in terms of the physical entity causing the abnormality, the deformity to the anatomy resulting from the cause and its severity (if appropriate). Deformities and their severity include partial and complete obstruction; physical causes include adhesions, internal hernia, torsion, gallstones and tumour. Deformities in certain anatomical structures cause disruptions in physiological processes and invoke symptom complexes.

The complexes obstructive jaundice and biliary colic and both related to abnormalities associated with diseases of the biliary tract and have been considered in detail. Both are invoked by an obstruction of the extrahepatic ducts. The causes of this that have been modelled are tumour and gallstones.

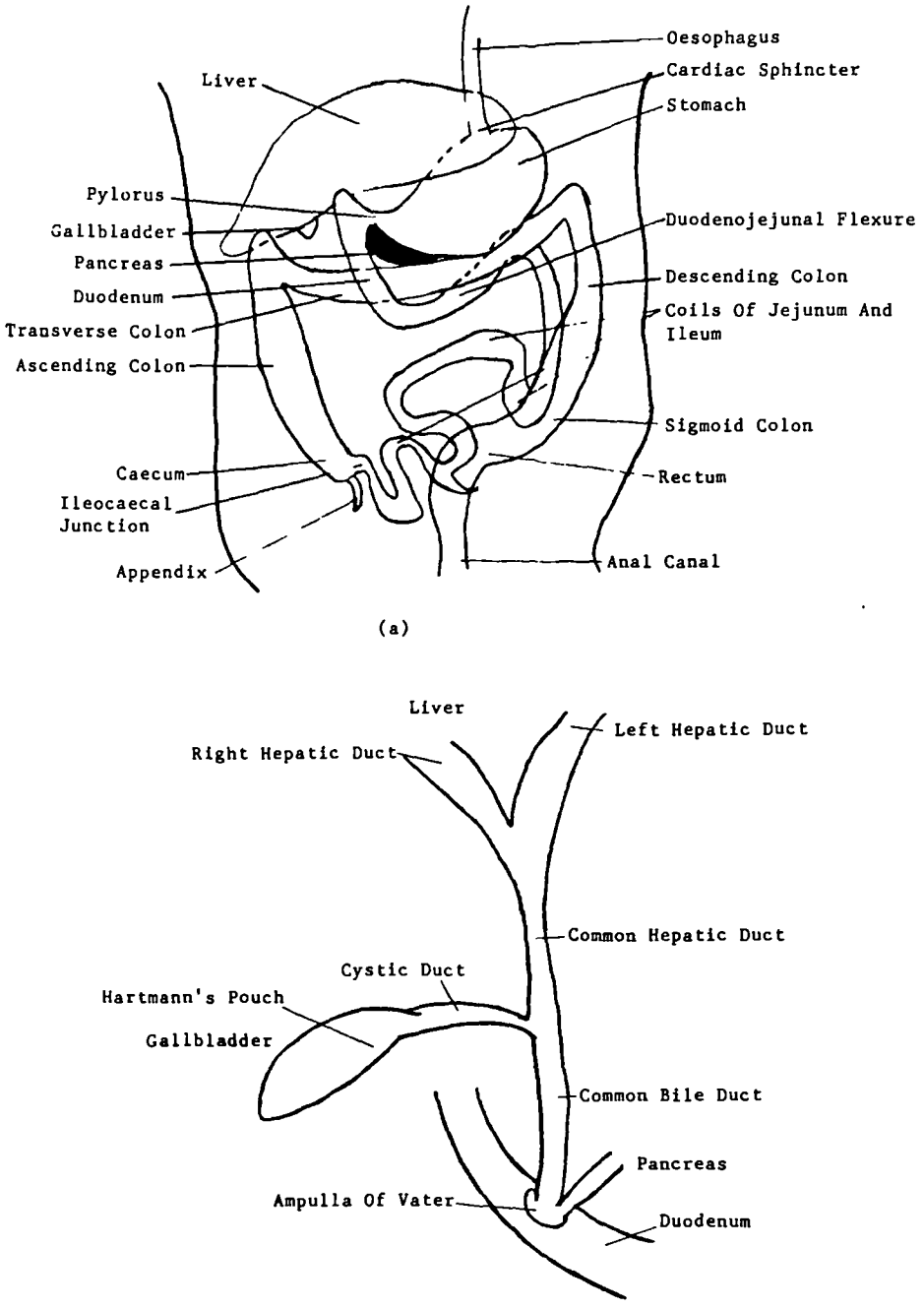


Figure 4.9: Anatomy Of The Gastrointestinal Tract

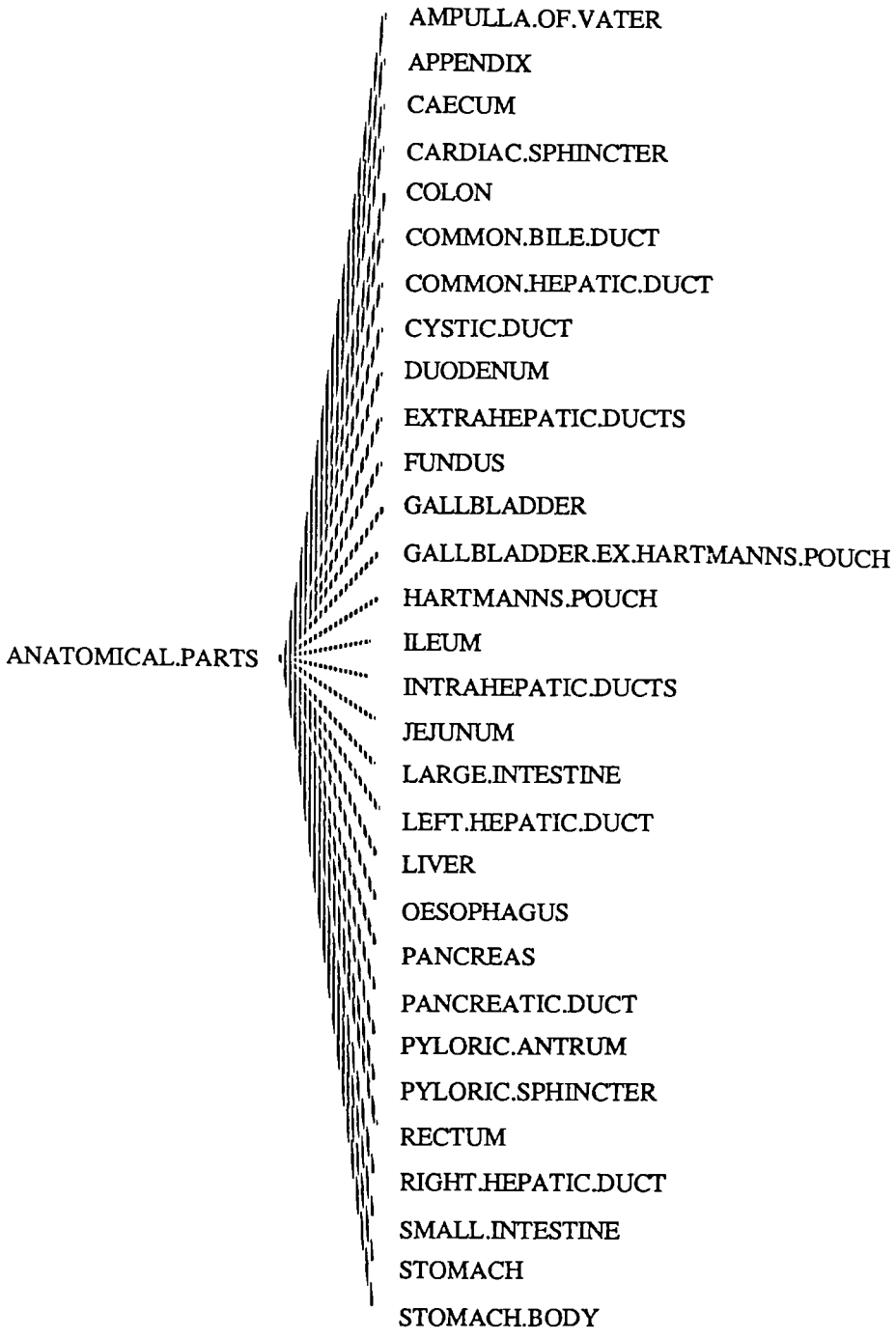


Figure 4.10: Anatomical Hierarchy

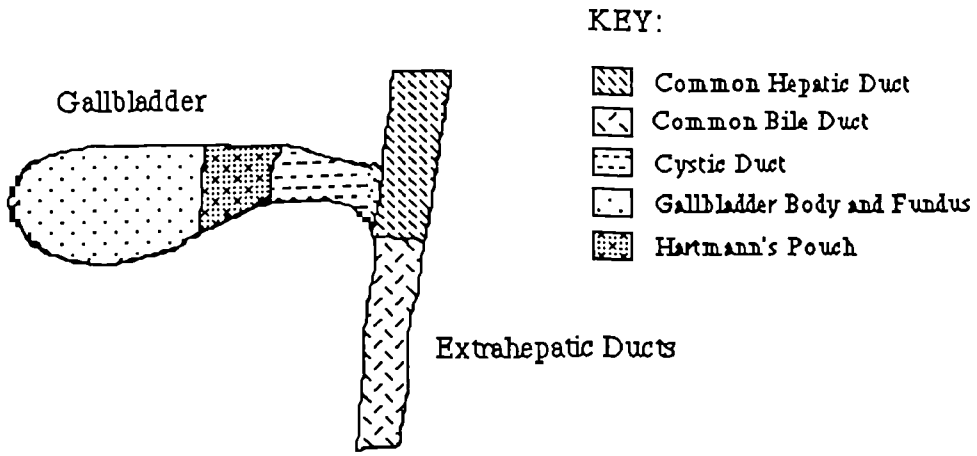


Figure 4.11: Anatomical Relationships Between The Gallbladder And Extrahepatic Ducts

4.3.5 Symptom and Sign Complexes

Biliary Colic

This is a symptom complex characterised by episodes of colicky pain punctuated by periods of complete relief. If the current disease state has pain type ache (activated by its pathological processes) both the pains are superimposed, periods of colic being combined with a persistent ache.

Obstructive Jaundice

A blockage of the extrahepatic bile ducts impedes the flow of bile into the duodenum. As a result, bile salts, bilirubin and alkaline phosphatase rise in the intravascular fluid[77]. When the serum bilirubin reaches a critical value, clinical signs appear. Excess of bile salts leads to pruritus, and excess of bilirubin makes the skin and conjunctivae appear yellow. Bilirubin is excreted in the urine giving it a dark colour. The faeces become pale coloured, and no urobilinogen is present in the urine. When the obstruction is removed there is a time delay before the clinical signs recede.

Representation Of Serum Bilirubin

The normal concentration of serum bilirubin in the intravascular fluid is 17 mmol/L [52]. When the obstruction occurs in the extrahepatic ducts there is a rise in serum bilirubin, dependent on the degree of obstruction.

It is assumed that after one day the serum bilirubin will rise to 40 mmol/L for a complete obstruction and 35 mmol/L for a partial obstruction. Thereafter, it will rise constantly by 8 mmol/L per day for a partial obstruction, 30 mmol/L per day for a complete obstruction. In a partial obstruction, the serum bilirubin is reduced because

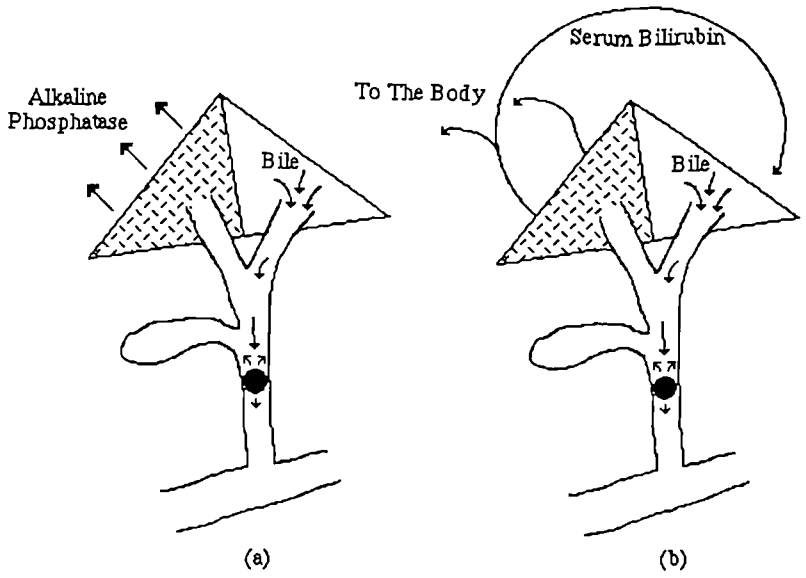


Figure 4.12: Removal of (a) alkaline phosphatase and (b) serum bilirubin from the body. In (a) increases of alkaline phosphatase are proportional to the degree of obstruction[77]. In (b) increases in the serum bilirubin are reduced by its transfer from the blood across the liver cell membranes to the bile.

it is passed via the intravascular fluid to a functioning part of the liver where it moves through the liver cell membranes and enters the bile (figure 4.12 a). After the critical value of 350 mmol/L has been reached, the serum bilirubin will rise no further.

The function to represent increasing serum bilirubin, SB, at time t hours, follows:

$$\text{Serum Bilirubin}(t) = \begin{cases} \frac{X-17}{24} + \text{SB}(t-1) & \text{if SB}(t-1) < X \text{ where } X \text{ is 40} \\ & \text{for a complete obstruction, 35} \\ & \text{for a partial obstruction and} \\ & \text{SB has been increasing for} \\ & \text{less than 25 hours} \\ \text{SB}(t) + c & \text{where } c \text{ is 8 for a partial obstruction,} \\ & \text{30 for a complete obstruction.} \end{cases}$$

It is assumed that all clinical symptoms and physical signs are evident when the serum bilirubin exceeds 40 mmol/L. When the obstruction causing the complex is removed, the serum bilirubin will fall at the rate of 1 mmol/L per hour if $\text{SB}(t) > 300$, thereafter by 30 mmol/L per day.

Representation Of Urine Urobilinogen

The urine urobilinogen takes its normal value of 4 mg in 24 hours when the serum bilirubin value is normal. If the serum bilirubin is increased and an obstruction is present the urine urobilinogen falls to 0 mg/24hrs. This value is retained until the obstruction is removed. It then takes its normal value.

Representation Of Alkaline Phosphatase

The normal alkaline phosphatase level in the intravascular fluid is assumed to by 110 IU[1] in Liverpool hospital laboratories. An obstruction of the bile ducts results in increased alkaline phosphatase levels, the increase being dependent on the time the obstruction has been present and the degree of the obstruction. If the liver becomes obstructed, alkaline phosphatase is passed to the blood where it remains since the liver cell membranes are unable to reabsorb it and there are no alternative routes by which it can enter the bile (figure 4.12b). The increase in alkaline phosphatase is directly associated with the proportion of the liver which is malfunctioning. In the presence of a partial obstruction, half the liver is assumed to be functioning normally, for a total obstruction the whole liver is assumed to be malfunctioning.

Normally the body removes 30 IU of alkaline phosphatase daily in the bile and 10 IU by other systems which are not well understood. In the presence of an obstruction, the alkaline phosphatase level rises daily by $n-$ (alkaline phosphatase removed by other systems) IU, where $n = 15$ for a partial obstruction, 30 for a complete obstruction. It is assumed that the alkaline phosphatase removed by other systems is a linear function of the alkaline phosphate level. When the alkaline phosphatase level reaches its maximum of 1200IU it remains constant until the obstruction has been removed.

Derivation of the equation modelling the removal of alkaline phosphatase from the body by systems other than the bile in the presence of a bile duct obstruction:

If at time t, AP is the intravascular alkaline phosphatase level and y is the AP removed by other systems,

$$y(\mathrm{AP}(t)) = a\,\mathrm{AP}(t) + b \tag{4.1}$$

where a, b are constants. When the alkaline phosphatase level is normal, AP_n, the alkaline phosphase removed by other systems is 10, therefore

$$b = 10 - a\,\mathrm{AP}_n.$$

[1]IU are international units.

When the alkaline phosphatase level is 1200, the alkaline phosphatase removed by other systems is p where $p = 25$ for a partial obstruction, 40 for a complete obstruction, and so

$$b = p - 1200\,a.$$

Substituting for b,

$$a = \frac{10 - p}{\mathrm{AP}_n - 1200}.$$

Therefore,

$$b = \frac{p\,\mathrm{AP}_n - 1200}{\mathrm{AP}_n - 1200}.$$

Substituting for a and b in equation 4.1,

$$y(\mathrm{AP}(t)) = \frac{1}{\mathrm{AP}_n - 1200}[(10 - p)\,\mathrm{AP}(t) + p\,\mathrm{AP}_n - 12000] \tag{4.2}$$

where $p = 25$ for a complete obstruction and $p = 40$ for a complete obstruction. Therefore,

$$\text{AP removed by other systems} = \begin{cases} 10 & \text{if no obstruction is present} \\ \text{equation 4.2} & \text{otherwise.} \end{cases}$$

When the obstruction is removed the excess alkaline phosphatase is removed from the intravascular fluid. This process occurs slowly[77] and is modelled by the exponential function derived below.

Derivation of the function modelling the removal of excess alkaline phosphatase following the removal of an obstruction

If y is the amount of alkaline phosphatase removed from the intravascular fluid when the alkaline phosphatase level is x at time t,

$$y(t) = a\,\exp(-kx(t)) \tag{4.3}$$

where a,k are constants. When the intravascular alkaline phosphatase level is 1200, the amount of alkaline phosphatase removed is p where $p = 30$ for a complete obstruction and $p = 15$ for a partial obstruction, and so

$$p = a\,\exp(-1200\,k)$$

whence

$$\ln(\frac{p}{a}) = -1200\,k \tag{4.4}$$

which can be rewritten as

$$\ln p - \ln a = -1200\,k \tag{4.5}$$

Also, when the intravascular phosphatase level reaches its normal value AP_n, the alkaline phosphatase removed is 10. Therefore,

$$10 = a \exp(-k\, AP_n)$$

whence,

$$\ln(\frac{10}{a}) = -k\, AP_n \tag{4.6}$$

Substituting for k in equation 4.4,

$$\ln(\frac{10}{a}) = \frac{\ln(\frac{p}{a})}{1200} AP_n$$

$$\ln a = \frac{1200 \ln 10 - AP_n \ln p}{1200 - AP_n}$$

$$a = \exp\{\frac{1200 \ln 10 - AP_n \ln p}{1200 - AP_n}\}$$

Rewriting equation 4.6 as

$$\ln 10 - \ln a = -k\, AP_n$$

and substituting equation 4.5 from it,

$$\ln 10 - \ln p = -k(AP_n - 1200)$$

Therefore,

$$k = \frac{\ln 10 - \ln p}{1200 - AP_n}.$$

And so, if $x(t)$ is the alkaline phosphatase level in the intravascular fluid the amount of excess alkaline phosphatase removed from the fluid at time t is $a \exp(-k\, x(t))$ where

$$a = \exp\{\frac{1200 \ln 10 - AP_n \ln p}{(1200 - AP_n)}\}$$

$$k = \frac{\ln 10 - \ln p}{1200 - AP_n}$$

and AP_n is the normal intravascular alkaline phosphatase level, assumed to be 110 IU, and $p = 30$ for a complete obstruction and $p = 15$ for a partial obstruction.

The model of intravascular alkaline phosphatase level changes that accompany a biliary tract obstruction is shown in figure 4.13.

4.4 Implementation Of Model In KEE

The general medical knowledge of diseases and their underlying anatomy, symptom and sign complexes, pain character, disease state knowledge and pathology is held in different unit hierarchies. These are used to instantiate an instance of a disease for a simulated patient. The patient case is represented by a unit hierarchy which is generated dynamically over time. Units within this represent the patient status over

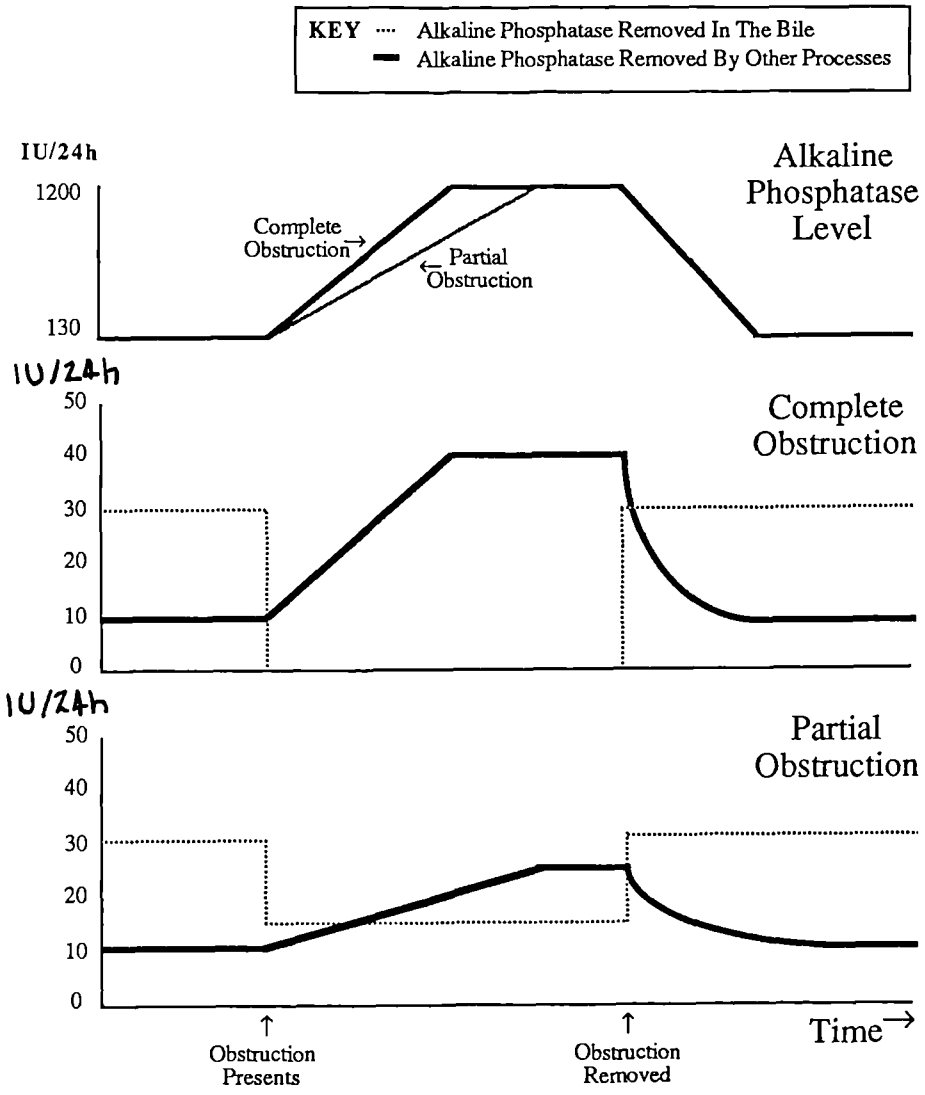


Figure 4.13: Consumption Of Alkaline Phosphatase In The Bile And By Other Systems In The Presence Of Complete And Partial Obstruction Of The Biliary Tract And The Resultant Alkaline Phosphatase Levels Of The Intravascular Fluid

intervals of time and are linked to disease states and other domain knowledge hierarchies. Disease attributes are inherited across the links. Knowledge of the intravascular fluid constituents serum bilirubin, alkaline phosphatase and urine urobilinogen are held in the intravascular fluid unit which is part of the physiological model and is described in greater detail in chapter 5. Graphs of the unit hierarchies and their associated units are given in appendices A and B.

4.4.1 Pathological Processes

The pathological processes illustrated in figure 4.8 are represented as subclasses of the unit UNDERLYING.PATH.PROCESSES. Each subclass has a member slot PAIN.CHARACTER which describes the pain character associated with the pathological type.

4.4.2 Pain Character

Units describing ache and colic are represented as subclasses of the unit hierarchy with root node PAIN.CHARACTER.

4.4.3 Disease States

The disease state taxonomy shown in figure 4.7 is represented as a unit hierarchy. The immediate subclasses of the root node, DISEASE.STATES, are used to partition disease states according to the anatomical structures which they affect. These subclasses have slots PRIMITIVE.STATES, MULTIPLE.ACUTE.STATES and CHRONIC.STATES describing the starting states in a disease progression, acute disease states which may exist concurrently and chronic disease states, respectively. The disease states are described by terminal nodes in the hierarchy, figure 4.14. They have slots:

UNDERLYING.PATH.PROCESSES Describes the states underlying pathological processes.

STATE.DURATION Specifies the range of times (in simulated hours) for which the state may be active.

SUCCESSOR.DIS.STATES A list of all possible next states in the disease progression.

CAUSES.WITH.SITES Defines the causes of the state and the possible anatomical parts afflicted. This knowledge takes the format of a list with sublist elements describing each cause and its anatomical sites. The cause is represented as a

triple with elements describing its physical form, the severity of the anatomical deformity resulting from its presence and the anatomical deformity.

TREATMENTS Provides a list of surgical treatments strategies which would correct the disease state. Correction of a disease state implies the disease is managed in an accepted surgical manner. The disease need not resolve, for example treatments will not cure gallbladder cancer.

PSITE.PRESENT Describes the site of pain when the disease state is active.

PSITE.ONSET Describes the site of pain at disease onset. This corresponds to a start state in a disease progression.

Other signs and symptoms which are not described by lower level domain concepts are defined by slots in the disease state unit for example, NAUSEA.

4.4.4 Anatomy

Each anatomical part is represented as a member of the class ANATOMICAL.-PARTS. The units describing anatomical structures, for example GALLBLADDER, have slots SUBPARTS which contain the structures components. Each of the components has a SUPERPART slot which describes the structure in which it is contained. All units also have a slot CONNECTED.TO which describes other anatomical components the part is joined to. Figure 4.15 illustrates the unit representing the common bile duct.

Anatomical part units' have a slot ABNORMALITY.PRESENT which has attached the active value CHANGE.ANATOMICAL.PATHOLOGY.MONITOR and take the default value normal. When a value is added to the slot, the structure in which the part resides is also given the value. On removing an abnormality from a part, its superpart structure accesses its other subparts to determine whether the abnormality is contained elsewhere within it. If not, the abnormality is removed from the superstructure. In this way anatomical changes are propagated to higher level components in the anatomical model.

The slot ASSOC.COMPLEXES in the anatomical part units describes any sign and symptom complexes which become activate if abnormalities exist in the anatomical part represented by the unit. Its value is a list with elements which are triples. Each triple describes the deformity, its severity and the complex invoked by it. Complexes are mapped to high level structures where possible to avoid knowledge duplication. For example, biliary colic is associated with the EXTRA.HEPATIC.DUCTS rather than each of its subparts. Invocation of the complex then occurs when an abnormality is propagated from an anatomical part to its superstructure.

Unit: **OBSTRUCTIVE.CHOLANGITIS** in knowledge base **AB-PAIN**
Superclasses: **GB.DUCT.DISEASES**

MemberSlot: ***CAUSES.WITH.SITES** from **OBSTRUCTIVE.CHOLANGITIS**
Inheritance: **OVERRIDE.VALUES**
ValueClass: **LIST**
Values: ((GALLSTONES COMPLETE OBSTRUCTION) (COMMON.BILE.DUCT COMMON.-
HEPATIC.DUCT CYSTIC.DUCT))

MemberSlot: ***SUCCESSOR.DIS.STATES** from **OBSTRUCTIVE.CHOLANGITIS**
Inheritance: **OVERRIDE.VALUES**
ValueClass: (UNION (SUBCLASS.OF **DISEASE.STATES**) (ONE.OF DEATH RESOLUTION))
Values: (**CHOLANGITIS**)

MemberSlot: ***UNDERLYING.PATH.PROCESSES** from **OBSTRUCTIVE.CHOLANGITIS**
Inheritance: **OVERRIDE.VALUES**
ValueClass: (SUBCLASS.OF **PATHOLOGICAL.PROCESSES**)
Values: **INFECTION INFLAMMATION**

MemberSlot: ***STATE.DURATION** from **OBSTRUCTIVE.CHOLANGITIS**
Inheritance: **OVERRIDE.VALUES**
ValueClass: **LIST**
Values: (48 48)

MemberSlot: ***PSITE.PRESENT** from **OBSTRUCTIVE.CHOLANGITIS**
Inheritance: **OVERRIDE.VALUES**
ValueClass: (ONE.OF CENTRAL GENERAL LH LLQ LUQ LOWH UPH RUQ RLQ RH)
Values: RUQ

MemberSlot: ***PSITE.ONSET** from **OBSTRUCTIVE.CHOLANGITIS**
Inheritance: **OVERRIDE.VALUES**
ValueClass: (ONE.OF CENTRAL GENERAL LH LLQ LUQ LOWH UPH RUQ RLQ RH)
Values: RUQ

MemberSlot: ***NAUSEA** from **OBSTRUCTIVE.CHOLANGITIS**
Inheritance: **OVERRIDE.VALUES**
Values: PRESENT

MemberSlot: ***TREATMENT** from **OBSTRUCTIVE.CHOLANGITIS**
Inheritance: **OVERRIDE.VALUES**
ValueClass: **LIST**
Values: (GALLSTONES (CHOLECYSTECTOMY EXPLORATION.OF.THE.COMMON.BILE.DUCT))

Figure 4.14: Frame Representing The Disease State Obstructive Cholangitis

Unit: **COMMON.BILE.DUCT** in knowledge base **AB-PAIN**
Superclasses: **ANATOMICAL.PARTS**

MemberSlot: ***ABNORMALITY.PRESENT** from **ANATOMICAL.PARTS**
Comment: Represents the cause of the pathology associated with this anatomical component.
AVUNITS: **CHANGE.ANATOMICAL.PATHOLOGY.MONITOR SETUP.COMPLEXES**
Inheritance: **OVERRIDE.VALUES**
Values: UNKNOWN

MemberSlot: ***ASSOC.COMPLEXES** from **COMMON.BILE.DUCT**
Comment: Tuples take the form (SEVERITY of abnormality, ABNORMALITY, COMPLEX).
Inheritance: **OVERRIDE.VALUES**
ValueClass: **LIST**
Values: (PARTIAL OBSTRUCTION OBST.JAUNDICE)
(COMPLETE OBSTRUCTION OBST.JAUNDICE)

MemberSlot: ***CONNECTED.TO** from **COMMON.BILE.DUCT**
Comment: Used to infer the connection relationships between anatomical parts. Connections are considered at the deepest level of anatomical description. Descriptions of connections may be made at a higher level by considering anatomical superparts.
Inheritance: **OVERRIDE.VALUES**
ValueClass: (MEMBER.OF **ANATOMICAL.PARTS**)
Values: **COMMON.HEPATIC.DUCT CYSTIC.DUCT AMPULLA.OF.VATER**

MemberSlot: ***SUPERPARTS** from **COMMON.BILE.DUCT**
Comment: Units for which this anatomical part is a component.
Inheritance: **OVERRIDE.VALUES**
ValueClass: (MEMBER.OF **ANATOMICAL.PARTS**)
Values: **EXTRAHEPATIC.DUCTS**

Figure 4.15: Unit Describing The Anatomical Part Common Bile Duct

An active value SETUP.COMPLEXES watching over the ABNORMALITY.-PRESENT slot compares any abnormalities added to the slot with those in the ASSOC.COMPLEXES slot. On finding a match, a symptom complex is activated by messaging its INITIALISATION slot with the arguments severity of abnormality, abnormality and site of abnormality. The complex is then added to the ACTIVE.COMPLEXES slot of the unit SIMULATION.GENERATOR.EVENTS.

4.4.5 Sign And Symptom Complexes

Sign and symptom complexes are represented by units in the sign and symptom complex hierarchy. Each complex has pathophysiological processes which become activated when the complex is setup. When critical values are reached, clinical signs and symptoms manifested by the complex are passed by inheritance to the currently active unit in the patient description hierarchy by creating member links from the complex to it. On removing the abnormality causing a complex, the complex may remain active but not progressive. Over time the pathophysiological processes will become corrected and the complex deactivated.

To enable explanation of sign and symptom complexes, the pathophysiological processes related to the complex are described declaratively together with text describing the reason for their activation. Figure 4.16 illustrates the symptom complex unit obstructive jaundice. Slots UNDERLYING.PATHOPHYSIOLOGICAL.PROCESS and RELATED.SUBPROCESSES describe the processes associated with the complex when it is first activated. Values contained within these slots describe the qualitative behaviour of each of process. Quantitive descriptions of the processes are represented procedurally by method slots.

The slot PROGRESSION.STATE describes a complexes progression state. Possible values for this slot are ACTIVE.NO.PROGRESSION, ACTIVE.PROGRESSION or NIL describing the complex when it is active but regressive, active and progressing and inactive respectively. When a complexes initialisation method is invoked with the triple argument (severity of deformity, deformity, site of deformity), the argument is stored in the complexes ETIOLOGY slot. If the complex is inactive its PROGRESSION.STATE slot is given the value ACTIVE.PROG. If the complex is active but is in the process of regression because the abnormality causing the complex had been removed but the pathophysiological derrangments resulting from it have not been resolved, the value of the PROGRESSION.STATE slot is changed from ACTIVE.NO.PROGRESSION to ACTIVE.PROGRESSION.

Having setup a complex, its associated processes are activated on each simulated hour. The underlying pathophysiological process and related subprocesses slots of the complex are called to determine the method slots of the complex which require

Unit: OBSTRUCTIVE_JAUNDICE in knowledge base AB-PAIN
Superclasses: JAUNDICE

MemberSlot: *ETIOLOGY from OBSTRUCTIVE_JAUNDICE
Comment: Pathological cause of complex.
Inheritance: OVERRIDE_VALUES
ValueClass: LIST
Values: UNKNOWN

MemberSlot: *INITIALISATION from OBSTRUCTIVE_JAUNDICE
Comment: Initialises the complexes associated with the complex.
Inheritance: METHOD
ValueClass: METHOD

MemberSlot: *PROGRESSION_STATE from OBSTRUCTIVE_JAUNDICE
Comment: Status of complex.
Inheritance: OVERRIDE_VALUES
ValueClass: (ONE_OF ACTIVE_PROGRESSION ACTIVE_NO_PROGRESSION)
Values: UNKNOWN

MemberSlot: *UNDERLYING_PATH_PROCESS from OBSTRUCTIVE_JAUNDICE
Inheritance: OVERRIDE_VALUES
Values: INCREASED SERUM_BILIRUBIN

MemberSlot: *RELATED_SUBPROCESS from OBSTRUCTIVE_JAUNDICE
Inheritance: OVERRIDE_VALUES
Values: (DECREASED URINE_UROBILINOGEN INCREASED_ALKALINE_-
PHOSPHATASE)

MemberSlot: *INCREASED_SERUM_BILIRUBIN from OBSTRUCTIVE_JAUNDICE
Comment: Bilirubin is normally secreted in the bile. If the bile flow to the small bowel is impededthen the bilirubin is secreted in the serum.
Inheritance: METHOD
ValueClass: METHOD

MemberSlot: *DECREASED_URINE_UROBILINOGEN from OBSTRUCTIVE_JAUNDICE
Comment: Bilirubin is converted to stercobilinogen in the small bowel. A reduction in the amount bilirubin entering the bowel causes a reduction in the amount of stercobilinogen passed to the blood and excreted by the kidneys as urine urobilinogen
Inheritance: METHOD
ValueClass: METHOD

MemberSlot: *INCREASED_ALKALINE_PHOSPHATASE from OBSTRUCTIVE_JAUNDICE
Comment: Increased hepatic function will result in increased alkaline phosphatase levels.
Inheritance: METHOD
ValueClass: METHOD

MemberSlot: *REGRESSIVE_PROCESSES from OBSTRUCTIVE_JAUNDICE
AVUNITS: MONITOR_REGRESSIVE_PROCESSES
Inheritance: OVERRIDE_VALUES
Values: UNKNOWN

MemberSlot: *DECREASED_SERUM_BILIRUBIN from OBSTRUCTIVE_JAUNDICE
Comment: On removal of the bile duct obstruction the serum bilirubin levels will fall. The excess serum bilirubin will be removed via the urine.
Inheritance: METHOD
ValueClass: METHOD

MemberSlot: *INCREASED_URINE_UROBILINOGEN from OBSTRUCTIVE_JAUNDICE
Comment: As bile is able to pass throught the bile ducts at its normal rate, the amount of bilirubin which is converted to stercobilinogen in the small bowel increases. Increase amounts of stercobilinogen pass into the blood and are removed by the kidneys resulting in an increase in urine urobilonogen.
Inheritance: METHOD
ValueClass: METHOD

MemberSlot: *DECREASED_ALKALINE_PHOSPHATASE from OBSTRUCTIVE_JAUNDICE
Comment: On the removal of the bile duct obstruction the alkaline phosphatase levels fall. The likely cause of this is natural breakdown.
Inheritance: METHOD
ValueClass: METHOD

MemberSlot: *STOOL_COLOUR from OBSTUCTIVE_JAUNDICE
Inheritance: OVERRIDE_VALUES
Values: PALE

MemberSlot: *ITCHING from OBSTRUCTIVE_JAUNDICE
Inheritance: OVERRIDE_VALUES
Values: PRESENT

MemberSlot: *URINE_COLOUR from OBSTRUCTIVE_JAUNDICE
Inheritance: OVERRIDE_VALUES
Values: DARK

Figure 4.16: Unit Describing The Symptom Complex Obstructive Jaundice

activation. The physiological parameters are then modified by these methods. The ON.PUT.ADD active value attached to each of the the physiological parameter slots adds the changes to the slots existing value. If the parameter reaches the critical value, when clinical signs and symptoms are observed, the symptom complex slot of the current patient status unit is given the value of the complex. This results in the creation of a member link from the complex to the current patient descriptor unit. On falling below the critical value, the sign and symptom complex is removed from the symptom complexes slot.

On removing an abnormality from the ABNORMALITY.PRESENT slot of an anatomical part, the SETUP.COMPLEX active value searches the ASSOC.COMPLEXES slot of the unit to determine whether any complexes are affected by the change. Each affected complex has its ETIOLOGY slot modified to remove the abnormality. On removing all values from a complexes ETIOLOGY slot, the active value MONITOR.COMPLEX.ETIOLOGY sets the PROGRESSION.STATE of the complex to ACTIVE.NO.PROGRESSION. The processes associated with the complex are then requalified and put into the slot REGRESSIVE.PROCESSES. Processes which were increasing become decreasing, those which were decreasing become increasing. On each update of the complex its processes are then corrected by calling methods. As each process is normalised it is removed from the REGRESSIVE.PROCESSES slot. When all the processes have been normalised the complexes PROGRESSION.STATE slot is set to NIL and the complexes name is removed from the ACTIVE.COMPLEXES slot of the unit SIMULATION.GENERATOR.EVENTS.

4.4.6 Disease Setup

The patient and their disease is represented by a frame hierarchy which is generated dynamically over time. The root node, PATIENT.PROFILE, contains patient profile knowledge, such as age, sex and weight and is created on simulation initialisation. A subclass unit of PATIENT.PROFILE, PATIENT.STATUS, contains slots describing the concepts underlying all disease states, for example pain character, underlying pathological processes. The active value, SETUP.ACTIVATOR, watches over these slots. On adding a value to any of the slots, SETUP.ACTIVATOR, creates a member link from the unit whose name is the slot value to the unit in which the slot is contained.

Having selected a disease to simulate, the root node in the progression is found using the disease's PRIMITIVE.STATES slot. A member of the class PATIENT.-STATUS is created, SNAP1, and linked to the class in the disease state hierarchy corresponding to the initial disease state. If a biliary tract disease was to be setup, the initial state may be obstruction of the biliary tract excluding the gallbladder (fig-

ure 4.17). SNAP1 is made a subclass of the disease state and inherits all its member slots. One of the disease attributes it inherits is a pathological process. The active value, SETUP.ACTIVATOR inherited from the unit PATIENT.STATUS then triggers a link to the pathological process unit so those attributes are inherited. A pain character, ache or colic, may be inherited which triggers the active value to create links from the pain character hierarchy to SNAP1. For obstruction of the biliary tract excluding the gallbladder, links to the pathological process obstruction of the biliary tract which has pain character colic results in the creation of member links between COLIC and SNAP1. The disease cause and site is generated at random using the slot CAUSES.WITH.SITES and used to setup the anatomical model. This may result in the activation of sign and symptom complexes. The state duration is found by generating a number at random in the range specified by the STATE.DURATION slot.

As the simulation proceeds and the state duration is exceeded the SUCCESSOR.DIS.STATES slot is used to select the next disease state. A second member unit, SNAP2, of PATIENT.STATUS is created and linked to the successor disease state. Member links are created to units in the pathological process hierarchy, disease attributes being inherited across the links. The CAUSES.WITH.SITES slot is used to derive an anatomical site associated with the disease state. This site maps to the same physical cause as the predecessor state. A deformity resulting from the cause together with its severity is also found. If the site of abnormality changes the anatomical model is modified, possibly resulting in the activation and regression of sign and symptom complexes. The duration of the new state is generated. On reaching the end of this state, other states are created similarly .

4.5 Conclusion

A *deep* model containing knowledge of diseases and their underlying pathological and anatomical principles has been developed. The domain knowledge is represented in general terms, not in the context of specific disease instances. Different types of domain knowledge is represented in separate program modules, for example the pathological and anatomical knowledge is contained in different frame hierarchies. Representation of knowledge at a deep level enables surface level inferences about diseases to be made without explicitly attaching such knowledge to the disease descripters. For example, the model can infer that diseases with obstruction of the extrahepatic ducts have symptom complexes obstructive jaundice and biliary colic without knowledge of these complexes being incorporated into disease state specifications. When modelling new disease states they need only be considered at a surface level, since their underlying concepts are

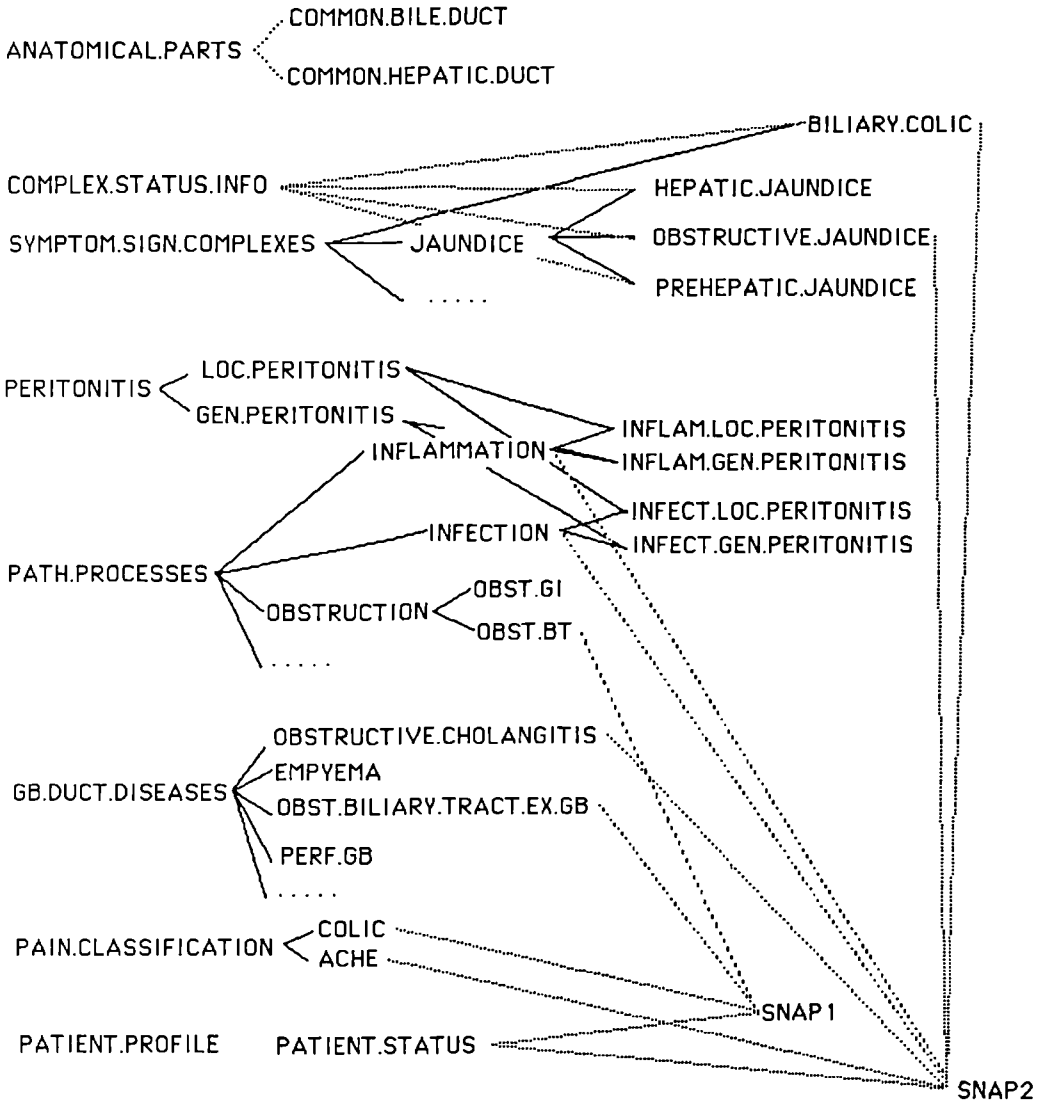


Figure 4.17: Representation of biliary tract disease with disease states obstruction of the biliary tract excluding the gallbladder (OBST.BILIARY.TRACT.EX.GB) progressing to obstructive cholangitis. SNAP1 describes the patient's status when the disease state OBST.BILIARY.TRACT.EX.GB is present. SNAP2 describes the patient after transition to obstructive cholangitis has been made.

already represented in the model. This reduces the likelihood of adding inconsistent and incomplete disease state specifications to the knowledge base.

Disease development is described by a state progression network, each state describing the disease presentation over an interval of time. State transition reflects changing pathology, etiology and affected anatomy. A specific disease instance is generated dynamically over time and is dependent on the treatments administered. The patient's condition is represented by two hierarchies, the patient status and anatomical hierarchies. The patient status hierarchy describes the patient's clinical symptoms and physical signs manifested by their disease state and its underlying pathology, pain character and symptom and sign complexes. The anatomical hierarchy defines the patient's current anatomical condition.

Chapter 5

Physiology and The Fluid Balance of The Body

5.1 Introduction

The pathology of acute abdominal disease is often associated with physiological derangements brought about by disturbances in the body fluid volume and composition. These derangements produce the syndrome of 'shock' which without intervention may progress to an irreversible state in which the patient will die. To avoid such complications physiological disturbances may be corrected by fluid therapy. It is of paramount importance that a patient's physiology is monitored whilst diagnosing the patient's disease and stabilised before corrective treatment of the underlying pathology is carried out. In the event of inappropriate fluid therapy being administered, secondary complications can occur which may ultimately lead to the patient's demise. Fluid therapy may therefore be crucial in sustaining life but can also end life. This issue makes it an important focus for teaching. The importance of physiology in the teaching of the acute abdomen has been neglected by teaching programs, which have tended to focus on diagnosis in isolation of management [20,65]. This omission restricts the realism of such programs. Other programs have addressed physiology in the context of the pathology of the cardiovascular system [47] or have provided a model for experimentation [73,4]. The later programs allow for the simulation of diseases remote to the heart but have no knowledge of which of their parameters require to be deranged or how.

The severity of shock can be assessed clinically by the signs and symptoms it manifests, or biochemically by investigating the physiological processes involved. Over time the patient's condition is thus described by pathological and physiological attributes, the pathological attributes being fixed over intervals of time; the physiological attributes being dynamic, dependent on processes associated with the pathology and

on the treatment being given.

This chapter describes the acquisition of knowledge associated with shock, its representation and incorporation into the disease simulator. Expansion of the domain knowledge to include shock constituted the addition of deep knowledge to the pathological process model, described in chapter 4. This deep knowledge was not directed for use by teaching components of the program but was required by the simulator to generate the physiology. The physiological model uses quantitative techniques to generate the fluid balance, blood pressure, heart rate and peripheral resistance. Changes in the processes associated with fluid balance, brought about by different pathologies, are explicitly represented. The physiology is updated at discrete points in time. A history of all patient data which is generated by the model, is measured in the real clinical situation and may be important in patient management is recorded, for example volumes of vomit and urine. All patient's physiological parameters are non-deterministic within the constraints of their diseases underlying pathology.

5.2 Knowledge Acquisition

The knowledge acquisition comprised two stages, each of which evolved a physiological model. The first stage was characterised by viewing the domain knowledge from a *surface* perspective whereby the behaviour of the cardiovascular system was discussed without reference to its underlying concepts, for example the physical principles underlying its normal function and their response to abnormal factors which may present in disease. This contrasted with the second stage in which the domain was viewed at a *deep* level, in terms of the domain principles fundamental to its behaviour.

5.2.1 Stage One

During this stage knowledge acquisition took the format of discussions between the knowledge engineer and one domain expert, the expert planning and sequencing the topics to be discussed. He selected the physiology required to be modelled, shock caused by sepsis, and in describing it introduced some domain terminology. The syndrome *chronic septic state* was identified, figure 5.1. This was made up of four consecutive phases; the hyperventilation, hyperdynamic, hypodynamic and irreversible hypodynamic phases. Each phase was characterised by clinical signs, haemodynamic parameters[1] and other descriptors which were described qualitatively. Diseases were linked to the chronic septic state by mapping pathologies associated with infection to each of its phases, figure 5.2.

[1]Parameters associated with the heart. These include heart rate, blood pressure, central venous pressure and peripheral resistance.

Phase1: Hyperventilation	Tachycardia - ↑ Pulse ↑ Respiration Rate ↑ Temperature Respiratory Alkalosis Flushing Sweating
Phase 2.1: Hyperdynamic	↑ Pulse Hypotension - ↓ Blood Pressure ↑ Cardiac Output Relative To Normal But Falling Overall ↓ Systemic Vascular Resistance ↓ CVP ↑ Temperature Metabolic Acidosis Hot, dry Skin Oliguria - ↓ Urine Output Rigors
Phase 2.2: Hypodynamic	↑ Pulse ↓ Blood Pressure ↓ Cardiac Output ↑ Systemic Vascular Resistance ↓ CVP ↑ Temperature Metabolic Acidosis Low Pulmonary Capillary Wedge Pressure Cold, Pale Skin Cyanosis ↓ Urine Output Mental Deterioration
Phase 3: Irreversible Hypodynamic	↓ Pulse ↓ Blood Pressure ↓ Cardiac Output ↑ Systemic Vascular Resistance Plethora - ↑ CVP ↓ Temperature Metabolic Acidosis High Pulmonary Capillary Wedge Pressure Cyanosis ↓ Urine Output Sweating

Figure 5.1: The Chronic Septic State

Chronic Septic State		*Pathological Processes*
Hyperventilation	$\longrightarrow$	Infection
Hyperdynamic Phase	$\longrightarrow$	Localised Peritonitis
Hypodynamic Phase	$\longrightarrow$	Generalised Peritonitis
Irreversible Hypodynamic	$\longrightarrow$	Death

Figure 5.2: Mapping Of The 'Chronic Septic State' To Pathological Processes

Units Of Effective Fluid Lost	Position of Patient	Heart Rate	Systolic Blood Pressure
0	Supine	Young person: 60 to 100 Old person: 80 to 100	Young person: 90 to 120 Old person: 110 to 120
	Erect	+25	No Change
1	Supine	No Change	No Change
	Erect	+25	No Change
2	Supine	+25	No Change
	Erect	+50	−10 to −20
3	Supine	+50	−10 to −20
	Erect	Maximum	−20 to −40
4	Supine	Maximum	65
5	Supine	Slow	< 40

Table 5.1: Table representing the blood pressure and heart rate changes associated with losses of effective fluid volume, in units of 500 ml. Effective fluid is the fluid available for circulation about the cardiovascular system.

Having been introduced to the concept chronic septic state the parameters defining it were then described in quantitative terms. To do this expert heuristic knowledge was used. A model to represent a *typical* patient's physiological response to the chronic septic state was developed using tables to structure the data. The tables were based on the assumption that physiological changes are brought about by changes in the body's fluid volume. These changes are manifested by pathophysiological processes, for example vomiting and diarrhoea, and the administration of intravenous fluid therapy.

Table columns described physiological parameters and rows their numerical values, each row corresponding to an absolute volume of body fluid (table 5.1). On setting up a patient of random age and sex, their 'normal' physiological status was described by a row of the table, referenced by a pointer. To update their physiological status, the net change in fluid balance would be determined and used to offset the tables' row pointer in a directions dependent on whether a net gain or loss in fluid balance had occurred. The row referenced by the pointer would then describe the change made to each physiological parameter in the patient's updated status.

The table representing the physiology contained no explicit knowledge of physiological principles, for example the haemodynamic principles associated with normal blood pressure and pulse and those brought about by pathological processes. These were implicit in the model and had not been considered in the discussions between the knowledge engineer and the domain expert. The physiology generated using the tables was deterministic and mapped only to diseases with infection.

Assessment of the chronic septic state model coincided with the introduction

of a second domain expert to the project. It soon became apparent that the first expert had simplified his view of the domain to avoid complicating issues which would have made the development of the model more difficult. When the second expert saw the typical patient's blood pressure and heart rate data, he clarified the relationship between blood pressure and heart rate and the order in which each fluctuated when fluid is lost from the body, ie. the heart rate initially rises as the blood pressure falls until a maximum value is reached, then both the blood pressure and heart rate fall, the rate of blood pressure decline increasing. However, he brought up the fact that increasing heart rate is a compensatory mechanism for fluid loss, and that older people are not as well able to compensate for fluid loss as younger people due to hardening and furring of the arteries. Therefore, a *typical* patient's response was a generalisation and not realistic. Also, the rate at which blood pressure falls is dependent on the type of fluid lost. For example, the blood pressure would fall more rapidly if one litre of blood was lost rather than one litre of vomit. The validity of the model was then queried and efforts were made to understand why the discrepancies had arisen.

Since the initial model was based on expert heuristic knowledge with little supporting factual knowledge to justify its appropriateness, evaluation of the model was difficult. Rather than seeking knowledge to support the model, it was decided to gain a deeper understanding of the principles underlying abnormal physiology. With this knowledge a revised physiological model would be constructed. The principles behind this model would be explicitly represented to overcome the problems of evaluation associated with the first model, the principles would be used to generate the physiological response as seen in patients of both sexes and ages 18 to 90, and also to potentially provide explanation for teaching purposes.

5.2.2 Stage Two

Knowledge obtained during this stage in the knowledge acquisition was gleaned from discussions with two domain experts, supplemented by textbooks. These discussions were structured by the knowledge engineer and involved more detailed questioning of the expert about the domain. An analysis of the haemodynamics associated with normal and abnormal blood pressure and heart rate was made. This constituted investigation into the anatomy of the body fluids, together with the normal and abnormal physiological processes which have an effect on it. Two types of shock that present in acute abdominal disease and have different causes were identified, septic and hypovolaemic shock. Although the blood pressure and heart rate trends in each shock type are similar, the rate at which they take place varies because they are associated with different pathophysiological processes. The blood pressure falls caused by falling blood volume may be more rapid in hypovolaemic shock when blood is lost, for example by

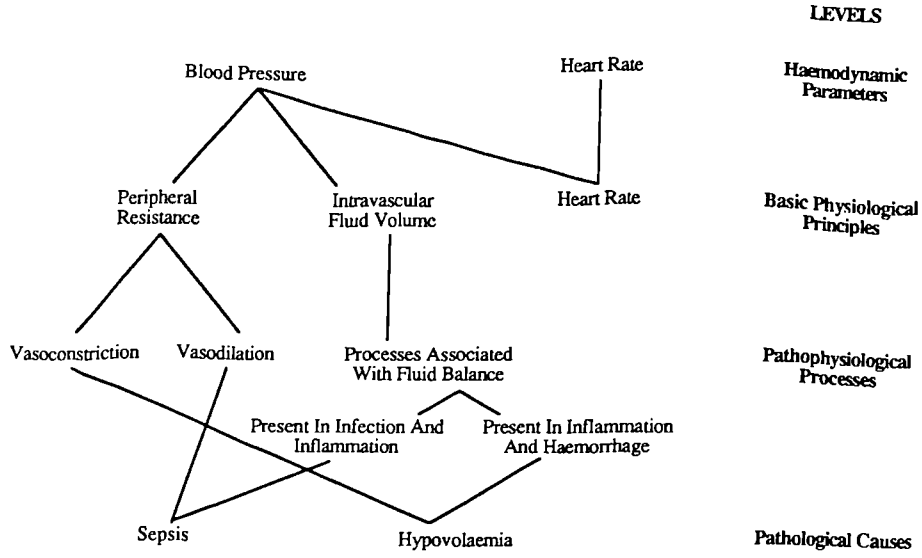


Figure 5.3: Multi-Level Representation Of The Physiology Associated With Blood Pressure and Heart Rate In Septic And Hypovolaemic Shock

haemorrhage, than in septic shock when the fluid loses do not have an immediate effect on blood volume. Heart rate compensates for falling blood pressure and its behaviour is thus dependent on the rate at which the blood pressure changes.

To enable the modelling of both shock types, a multi-level physiological model was derived, the cause of shock being represented at a low level, the effects of which are passed to a higher level view of the domain (figure 5.3). The model may be used in the simulation of all acute abdominal diseases unlike the previous one which modelled only diseases presenting with septic shock.

The quantification of the parameters needed to represent the domain concepts was performed by domain experts supplemented by facts acquired from textbooks. Unlike the first model, the domain concepts are explicitly represented and have associated mathematical functions. A patient's normal physiological parameters may be varied in accordance with their age, sex and weight, and their abnormal physiological processes may be modified within the confines of their disease, to generate a non-deterministic simulation.

To evaluate the model it was implemented in a high level language using discrete-event fixed time increment simulation techniques [45]. The reasons for this were:

1. By considering the physiology in isolation to the KEE model of diseases and their pathologies, more control could be had over the experimental conditions. The diseases which had been incorporated into the KEE program at the time of acquiring a physiological model utilised only a part of the pathological model, for example no disease had associated the pathological process obstruction of

the gastro-intestinal tract, thus the pathological model could not be completely tested. Also, the mapping of certain physiological parameters into the disease knowledge had not been made.

2. The hardware running KEE was not easily accessible to the domain experts who were to evaluate the model. A portable evaluation program was therefore developed, written in BBC Basic. Since numerous experiments could be carried out on the model, it was decided that the distribution of a program to the experts provided for better evaluation than mailing selected summary listings.

The model was tuned, using feedback from the experts, to yield results which are consistent with the clinical situation (appendix C). Once the feasibility of the basic concepts underlying the model had been established, the model was ready for implementation in KEE where the features of an object-oriented programming environment could be exploited.

5.3 Physiological Model

The physiological model has knowledge of the anatomy of the body fluids, the pathophysiological processes which are associated with fluid loss and gain and the haemodynamics associated with blood pressure and heart rate. At discrete points in time the change in fluid volume brought about by the pathophysiological processes is found and used to generate the blood pressure and heart rate consistent with septic or hypovolaemic shock.

5.3.1 The anatomy of body fluids

A proportion of the body's weight, which is dependent on age and sex, is comprised of fluid. Using statistics given in [52], the total fluid volume is approximated by the following relationships:

$$\text{Total Fluid Volume(litres)} = \begin{cases} \frac{\text{weight}}{4700}[-3 \times \text{age} + 2404] & \text{for females} \\ \frac{\text{weight}}{4700}[-8 \times \text{age} + 2964] & \text{otherwise.} \end{cases}$$

This is contained in two compartments, the extracellular fluid (ECF) and the intracellular fluid. The extracellular fluid which makes up approximately 2/3 of the bodies fluid is divided up into two further compartments, the intravascular and extravascular fluids (figure 5.4). The extravascular fluid comprises 75% of the extracellular fluid, the remaining 25% residing in the intravascular space (ie. blood).

The pathophysiological processes which are associated with shock initially effect changes in the extracellular fluid. If fluid is removed from the extracellular compartment, the body compensates for the loss by transferring fluid from the intracellular

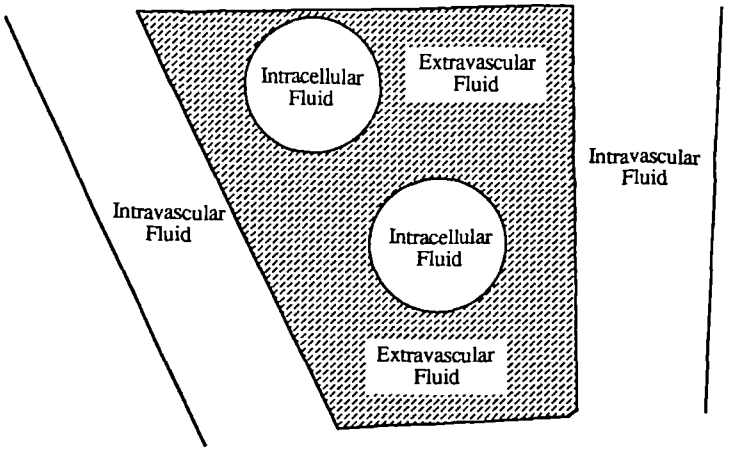


Figure 5.4: Anatomy Of The Body Fluids

space to the extracellular space. This compensatory mechanism is slow and for the purposes of the model is ignored. In addition to intracellular shifts, fluid losses from either the intravascular or extravascular spaces are compensated for by equilibration between the compartments. These shifts are modelled by evaluating the change in extracellular fluid volume, and adding 75% of this change to the extravascular fluid volume, 25% to the intravascular fluid volume.

Intravenous fluids enter the body via the intravascular fluid space and are then distributed about the whole of the ECF space. The redistribution is dependent on the fluid type. For colloids, which tend to stay in the intravascular space, it is assumed that 90% remains in the intravascular space and 10% goes to the extravascular space. Crystalloids redistribute about the intravascular and extravascular spaces in the ratio of their fluid volumes.

5.3.2 Pathophysiological Processes Associated With The Bodies Fluid Volume

A block diagram illustrating the concepts underlying fluid balance, as perceived by the knowledge engineer, is shown in figure 5.5. The intermediary systems such as the alimentary canal and the kidney are linked to the ECF by distribution systems. The terminals, for example urine, blood and gastrointestinal secretions represent the physical products made up from the body fluid and yielded by the pathophysiological processes taking place within the intermediary systems. The coordinating system, management, is illustrated which is external to the body but has an effect on it.

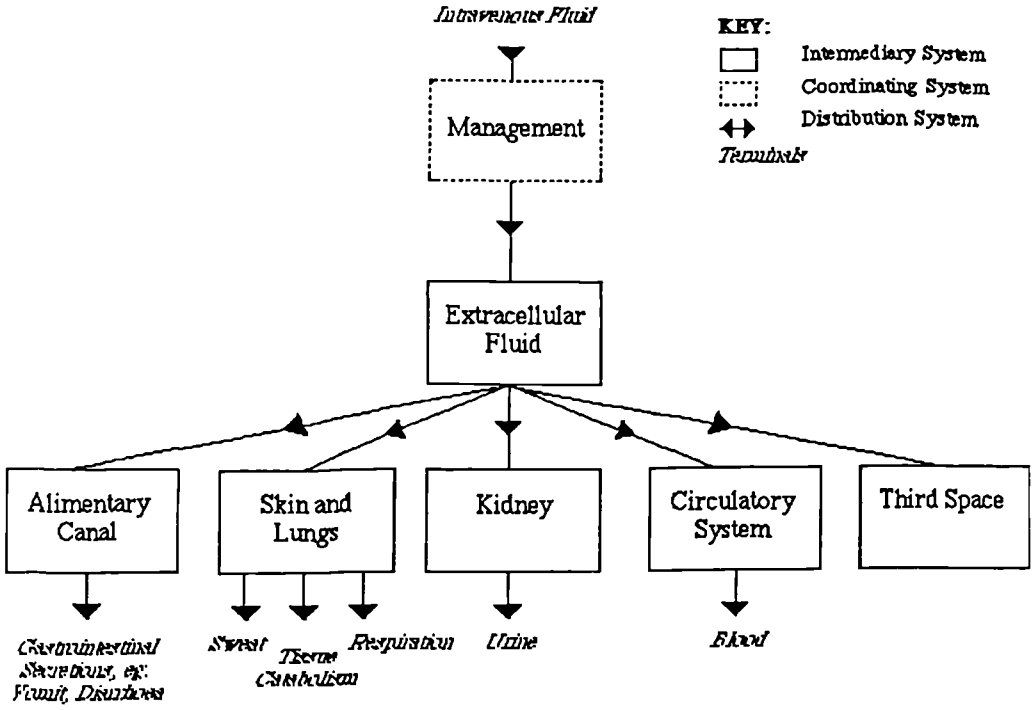


Figure 5.5: Block Diagram Illustrating The Extracellular Fluid Balance

Fluid Loss From The Kidney

Urine output has been represented as a linear function of the systolic blood pressure. This satisfies the conditions that:

1. The normal urine output is $6/7 \times$ (normal body weight in kgs) ml/hr and this occurs at time t_0, when the blood pressure is normal. This has been inferred from the fact that a 70kg male produces 60ml of urine/hr [35].

2. When the blood pressure is 80 mmHG the urine output is zero.

When the blood pressure falls below 80, no urine is produced.

Derivation of the urine output function:

If at time t, y is the volume of urine produced and BP(t) the blood pressure,

$$y(\mathrm{BP}(t)) = a\,\mathrm{BP}(t) + b \text{ where a, b are constants.} \tag{5.1}$$

When the blood pressure equals 80, the volume of urine produced is 0, therefore

$$b = -80 \times a.$$

When the blood pressure is normal (BP(0)), the volume of urine produced is $6/7 \times$ normal body weight in kgs, and so

$$b = \frac{6}{7}\,(\text{normal body weight in kgs}) - a \times \text{BP}(0).$$

Substituting for b,

$$a = \frac{6 \times (\text{normal body weight in kgs})}{7 \times (\text{BP}(0) - 80)}.$$

Therefore,

$$b = -80 \times \frac{6 \times (\text{normal body weight in kgs})}{7 \times (\text{BP}(0) - 80)}.$$

Substituting for a and b in equation 5.1,

$$y(\text{BP}(t)) = \frac{6 \times (\text{normal body weight in kgs})}{7 \times (\text{BP}(0) - 80)}[\text{BP}(t) - 80] \tag{5.2}$$

In addition to the above relationships between blood pressure and urine output further relationships associated with the medical condition anuria were identified. These stated that if the systolic blood pressure falls below 80 for half an hour or more and the fluid balance is restored causing an increase in blood pressure above 80, the urine output will remain zero until a time delay of (number of hours the blood pressure is below 80)^3 hours has passed. When urine output resumes, it will be twice its normal volume for a period of 24 $\times$ (no of hours the blood pressure is below 80). Therefore

$$\text{Urine Output} = \begin{cases} \text{Equation 5.2} & \text{if BP}(t) > 80 \text{ and anuria} \\ & \text{is not present;} \\ 0 & \text{if BP}(t) < 80 \text{ or the '0 urine out-} \\ & \text{put' stage of anuria is present;} \\ 2 \times \text{urine output}(t = 0) & \text{otherwise.} \end{cases}$$

Fluid Loss From The Skin And Lungs

The physiological processes whose combined effect is fluid loss from the skin and lungs and have been modelled are active sweating, tissue catabolism, normal fluid loss through the skin and respiration. It is assumed that the nett fluid loss from tissue catabolism, normal loss through the skin and respiration is constant and amounts to one litre per day [51].

The fluid loss associated with active sweating has been defined by experts to be:

Vol. of fluid lost = 0.05 $\times$ (Rise in temperature above normal in °C) litres/hour.

The rise in temperature above normal (in °C) has been linked to the severity of infection presenting with a disease. For diseases with infection, localised and generalised peritonitis the rise in temperature is assumed to be 1°C, 1.5°C and 1.5°C respectively.

For infection and generalised peritonitis the rise is assumed constant and fluctuates between $\pm 0.3°C$ of the specified temperature rise. Localised peritonitis is associated with a swinging temperature which varies between $\pm 2°C$ of the rise specified. Textbooks state the existence of a relationship between temperature and sweating but do not quantify it, instead they give ranges of absolute volumes of fluid lost, such as between 0 and 4 litres per hour [64].

Fluid Loss From The Alimentary Canal

The volumes of vomit and diarrhoea output from the alimentary canal have been represented by exponential functions. Since the diseases which had been modelled are not usually associated with diarrhoea, only upper gastrointestinal losses have been considered in detail.

Derivation of the volume of vomit equation which assumes no fluid therapy:

At time t_0 the volume of gastrointestinal loss produced is MAXVOM and at time $t_1, \ldots, t_n$ the volume is 90% of that produced on the last vomit. *MAXVOM is* assigned the value VOMWEIGHT $\times \frac{5900\text{xECF}}{24}$, where VOMWEIGHT is an integer used to weight MAXVOM according to the vomiting severity. The factor $5900 \times$ ECF represents the daily volume of vomit. Justification for this is given in [51] where it is stated that all the intestinal secretions from the mouth to the duodenum may be vomited within 24 hours of disease onset.

If V is the volume of vomit produced at time t,

$$V(t) = ae^{-kt} \text{ where } a,k \text{ are constants.} \tag{5.3}$$

At time $t = 0$, the volume of vomit produced is MAXVOM, and so

$$\text{MAXVOM} = ae^{-k \cdot 0}, \text{ whence } a = \text{MAXVOM}.$$

At time $t = 1$, the volume of vomit produced is $0.9 \cdot$ MAXVOM, and so

$$0.9 \cdot \text{MAXVOM} = ae^{-k}$$

Since $a =$ MAXVOM,

$$0.9 = e^{-k}, \text{ whence } k = -\log_e(0.9).$$

Substituting for a and k in equation 5.3 gives

$$V(t) = \text{MAXVOM } e^{\log_e(0.9)t} \tag{5.4}$$

It is assumed that the frequency of vomiting is associated with a diseases pain type. For colics, the pain is episodic and vomiting occurs every fourth hour and for

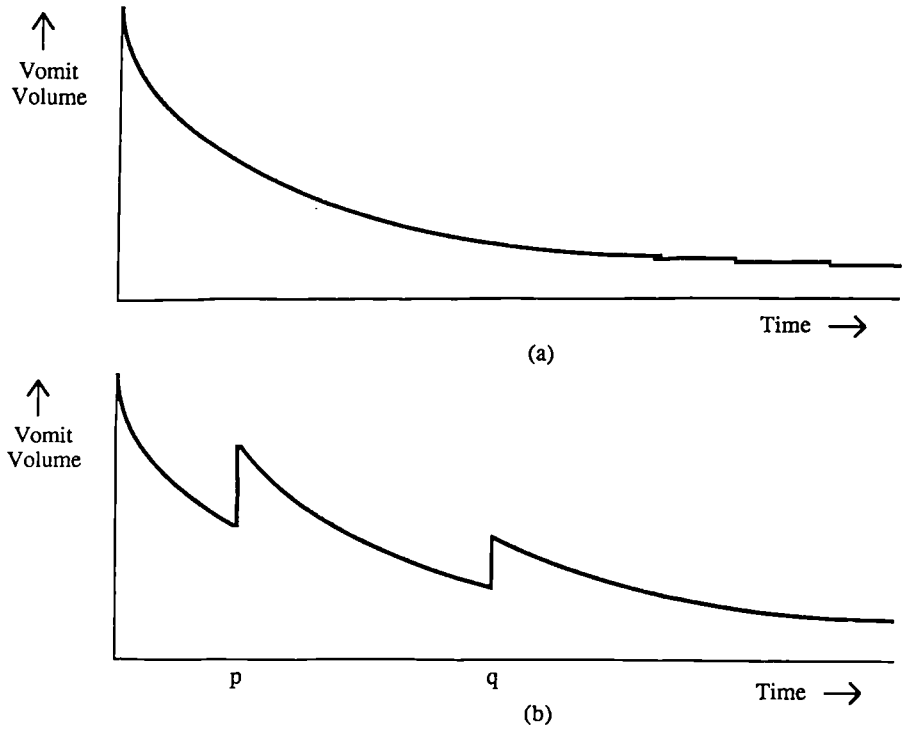


Figure 5.6: Vomit volume curves illustrating the (a) non-administration and (b) administration of intravenous fluids where fluids are given at times p and q

aches the pain is constant and vomiting occurs every hour.

Extension of vomit function to handle the input of fluid via infusion to the extracellular space:

When fluids are administered and the extracellular fluid volume is increased, the vomit function (equation 5.4) has to be modified to allow for the generation of increasing vomit volumes. To facilitate this a correction factor has been incorporated into equation 5.4 to transform the variable t to some point back in time and thus generate a greater volume of vomit (figure 5.6).

If h is the volume of infused fluid which may be vomited at time t in addition to that returned by equation 5.4, and CF is a correction factor

$$V(t) + h = \text{MAXVOM}\, e^{\log_e(0.9)(t+\text{CF})}$$

Therefore,

$$\text{CF} = \log_e[\frac{V(t)+h}{\text{MAXVOM}}] \times \frac{1}{\log_e 0.9} - t. \tag{5.5}$$

Thereafter $t = t + \text{CF}$ is substituted into equation 5.4. On adding more fluid to the extracellular space the correction factor, CF, is recomputed by substituting $t = t + \text{old CF}$ into equation 5.5, and adding the result to the original correction factor value, CF.

Fluid Changes Associated With The Third Space

The losses of fluid to the third space are assumed to be dependent on the severity of inflammation presenting in a disease. For diseases with inflammation, localised and generalised peritonitis the loss of fluid to the third space is assumed to be in the ranges [0.5, 1], [1.5, 2] and [2, 2.5] litres/24 hrs respectively, where the duration of the inflammation is disease and treatment dependent. It has been assumed that on disease resolution all fluid lost to the third space is made effective by returning it to the body's fluid compartments.

Losses Of Fluid From The Circulatory System

The fluid loss from the circulatory system which has been considered is haemorrhage. It is a recognised loss but not particulary relevant to the diseases considered here. It is supported by the model.

Fluid Gain From Transfusion

Intravenous fluids are categorised into 2 types: colloids and crystalloids. The type of fluid to be administered, its volume and rate of administration are directly input to the model.

5.3.3 Haemodynamics Associated With The Blood Pressure And Heart Rate

There is a well understood relationship between the systolic blood pressure (BP), the peripheral resistance (PR) and the cardiac output (CO) [35]:

$$\mathrm{BP} \propto \mathrm{PR} \times \mathrm{CO}.$$

There is also a relationship between the cardiac output, heart rate (HR) and stroke volume (SV):

$$\mathrm{CO} = \mathrm{HR} \times \mathrm{SV}$$

resulting in,

$$\mathrm{BP} \propto \mathrm{PR} \times \mathrm{HR} \times \mathrm{SV}.$$

Since the stroke volume is difficult to measure by the bedside, a simplified equation is used, as follows:

$$\mathrm{BP}(t) = \mathrm{PR}(t) \times [\mathrm{K} + 0.5\,\{\mathrm{HR}(t-1) - \mathrm{HR}(0)\}] \times \mathrm{IV}(t) \tag{5.6}$$

where K is a constant, HR(0) is the normal heart rate at $t = 0$, HR$(t-1)$ is the heart rate at time $t-1$, and PR(t) and IV(t) are the peripheral resistance and intravascular fluid volume at time t.

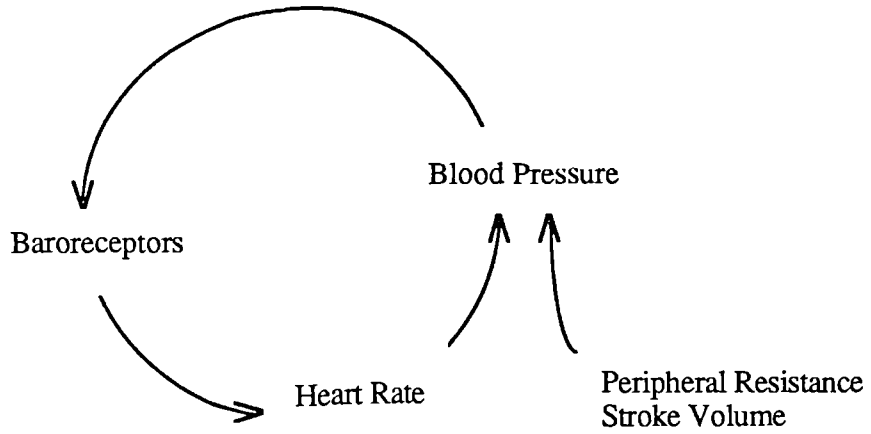


Figure 5.7: Feedback Loop Between Heart Rate And Blood Pressure

Age	*Blood Pressure*
< 50	90 to 120
≥ 50	110 to 120

Table 5.2: Normal Range Blood Pressure Categorised According To Age

There exists a closed loop relationship between blood pressure and heart rate (figure 5.7). Blood pressure is dependent on heart rate, but on sensing changes in the blood pressure the heart rate may be modified by the baroreceptors. The model assumes the heart rate to be dependent on the blood pressure at an instance in time, and the blood pressure to be dependent on the difference between the normal heart rate and that at the instance of time prior to that when the blood pressure is measured. If the heart rate is greater than its normal value it will exert a positive influence on the blood pressure, less than normal, a negative influence. K represents the effects of normal heart rate on the blood pressure. Also, a correspondence between the intravascular fluid and stroke volumes is assumed.

On simulation initialisation (time $t = 0$), the peripheral resistance is set to 0.75 together with a normal range blood pressure (table 5.2). It is assumed that the heart rate is stable on and prior to the start of the simulation, so HR(-1) = HR(0). Substitution of the normal peripheral resistance, blood pressure and intravascular fluid volume into equation 5.6 is used to define K.

The diastolic blood pressure at time t is generated using the relationships:

$$\text{Diastolic Blood Pressure} = \begin{cases} \text{SBP}(t) - 50 & \text{if SBP}(t) > 60 \\ \text{SBP}(t) - 40 & \text{if SBP}(t) > 40 \\ 0 & \text{otherwise} \end{cases}$$

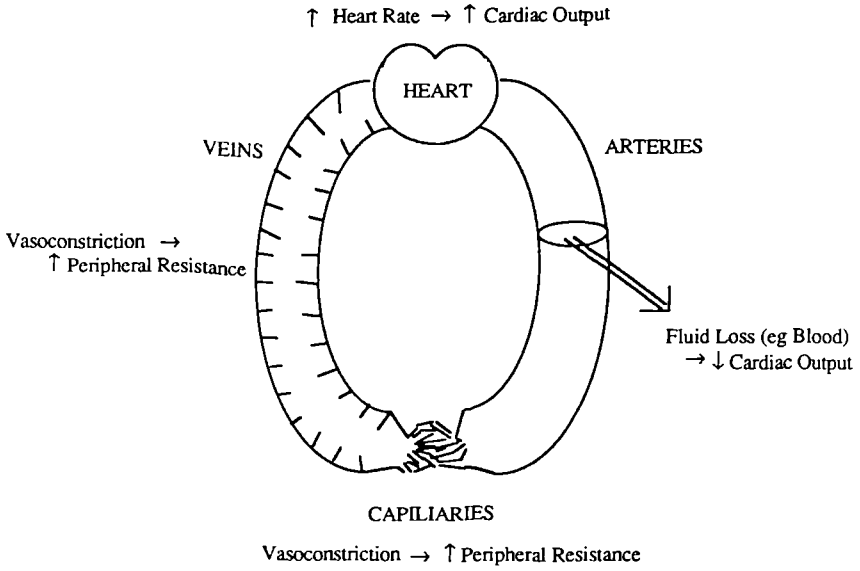


Figure 5.8: Principles Underlying Hypovolaemic Shock

where $\mathrm{SBP}(t)$ is the systolic blood pressure at time t.

Representing the peripheral resistance and heart rate as parameters of the blood pressure equation enables the compensatory mechanisms they both provide to counteract falling blood pressure to be modelled explicitly. This is of importance when modelling hypovolaemic and septic shock since the behaviour of the heart rate and peripheral resistance is different in each.

In hypovolaemic shock fluid loss reduces the cardiac output (figure 5.8). To compensate for this, vasoconstriction takes place. This increases the peripheral resistance, maintaining the blood pressure at a value greater than that were the compensation not to have taken place. When maximal vasoconstriction has occurred the heart rate increases, reducing the drop in cardiac output. When the heart rate reaches a maximum value, no further compensations take place. The heart rate falls, which together with falling stroke volume results in a blood pressure drop which cannot be stemmed without fluid therapy.

Vasoconstriction is modelled by increasing the peripheral resistance to a maximum value of 1 to offset the fall in intravascular volume. The heart rate is represented by reciprocal and linear functions of the blood pressure The compensatory effects for falling blood pressure effected by the heart rate are implicit in the equations. Heart rate at time t is represented by the functions:

1. For $\mathrm{HR}(t=0) \leq \mathrm{HR}(t) \leq 140$,

$$\mathrm{HR}(t) = \frac{\text{Scalefactor}}{\mathrm{BP}(t)} \tag{5.7}$$

where scalefactor= 10000, and

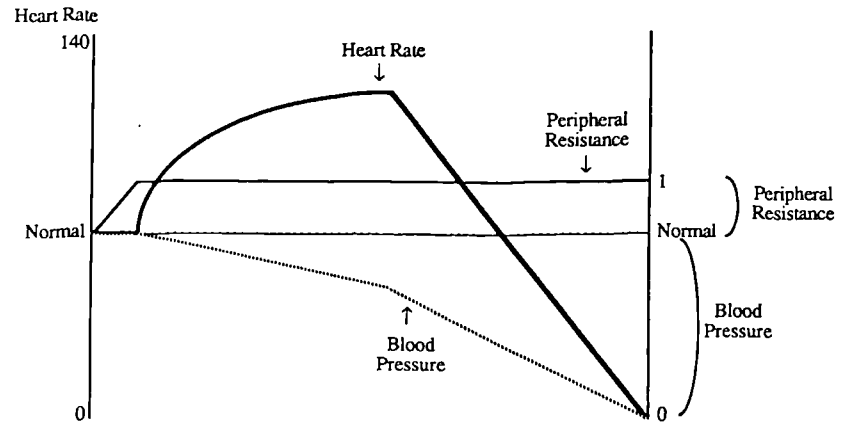


Figure 5.9: Relationships Between Peripheral Resistance, Heart Rate And Blood Pressure In Hypovolaemic Shock

2. For $0 \leq \mathrm{HR}(t) \leq p$ where p is the last heart rate value generated which satisfies equation 5.7 and x is the corresponding blood pressure,

$$\mathrm{HR}(t) = \frac{p}{x} \times \mathrm{BP}(t). \tag{5.8}$$

The relationships between heart rate, blood pressure and peripheral resistance are shown in figure 5.9.

If the blood pressure falls below 40 for more than four hours it is assumed that resuscitation is impossible (figure 5.10). However, if the blood pressure falls below 40 for four hours or less and the cardiac output is increased by fluid therapy, the heart rate will rise linearly to 120 before falling to its normal value. To model this, additional heart rate functions are required:

Increasing heart rate above 40 If $0 < \mathrm{HR}(t) \leq 120$, $q < 40$ is the blood pressure when the cardiac output is first increased, y is the heart rate corresponding to q and x is the blood pressure corresponding to the maximum heart rate generated using equation 5.7,

$$\mathrm{HR}(t) = \frac{120 - q}{x - y} \times \mathrm{BP}(t) + \frac{qx - 120y}{x - y}.$$

Falling heart rate from 120 to normal If $\mathrm{HR}(0) \leq \mathrm{HR}(t) \leq 120$, x is the blood pressure when the heart rate is 120, $\mathrm{HR}(0)$ and $\mathrm{BP}(0)$ are the normal heart rate and blood pressure values at time $t = 0$,

$$\mathrm{HR}(t) = \frac{120 - \mathrm{HR}(0)}{x - \mathrm{BP}(0)} \times \mathrm{BP}(t) + \frac{x \times \mathrm{HR}(0) - 120 \times \mathrm{BP}(0)}{x - \mathrm{BP}(0)}.$$

Should the blood pressure fall below 40 and resuscitation takes place, the heart rate is modelled by the functions above, not by equations 5.7 and 5.8.

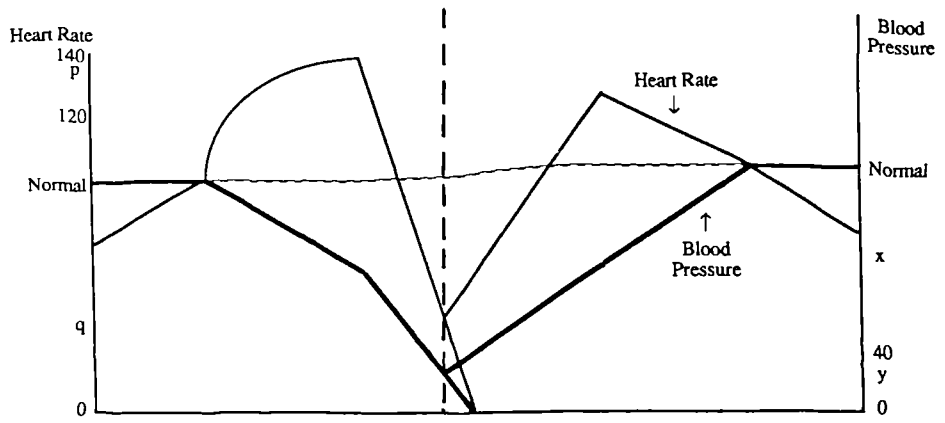


Figure 5.10: Relationships Between Heart Rate And Blood Pressure

To prevent the blood pressure rising above its normal value, the peripheral resistance and heart rate will fall below their normal values providing decompensatory mechanisms. The peripheral resistance will drop to its minimum value of 0.5 before the heart rate falls. Falling heart rate has been represented as a linear function of the intravascular fluid volume satisfying the conditions that when the heart rate is 60 the intravascular fluid volume is 1.5 times its normal value and when it is normal the intravascular fluid volume is normal. The function is defined below:

$$\mathrm{HR}(t) = \frac{2\,[60 - \mathrm{HR}(0)]}{\mathrm{IV}(0)} \times \mathrm{IV}(t) + 3\,\mathrm{HR}(0) - 120$$

where HR(0) and IV(0) represent the normal heart rate and intravascular fluid volume values at time $t = 0$ and IV(t) the intravascular fluid volume at time t.

In septic shock vasodilation occurs, reducing the peripheral resistance (figure 5.11). The amount of vasodilation has been mapped to the extent of infection (figure 5.12). During localised infection it is assumed that the peripheral resistance falls to 0.7, for generalised infection it falls to its minimum value of 0.5. The peripheral resistance is assumed lower in generalised infection because vasodilation occurs not only in the abdomen but throughout the circulation. Falling peripheral resistance leads to falling blood pressure if the cardiac output is not increased. To compensate for this, the heart rate is increased, using the same principles as described for hypovolaemic shock. The difference between the heart rate in septic and hypovolaemic shock is its precursor; falling peripheral resistance and intravascular volume for septic shock, falling intravascular volume only for hypovolaemic shock.

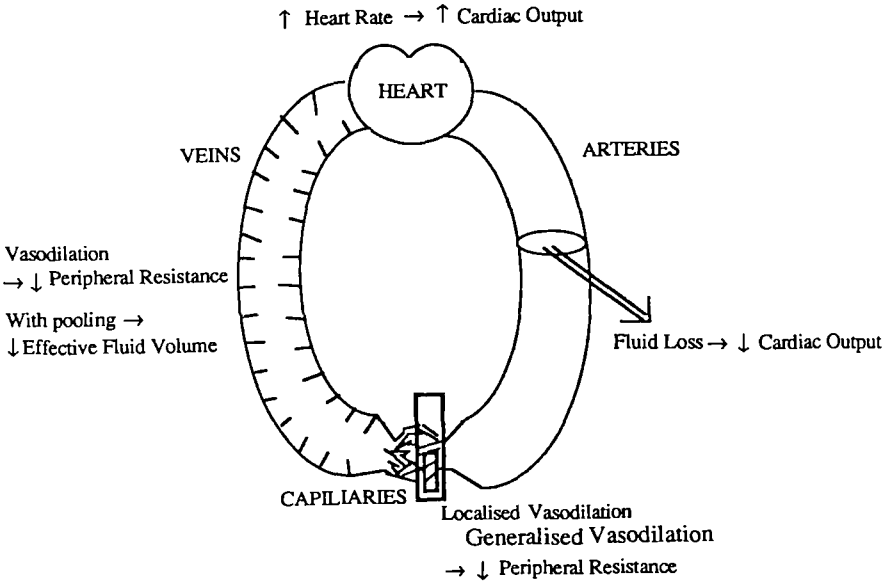


Figure 5.11: Principles Underlying Septic Shock

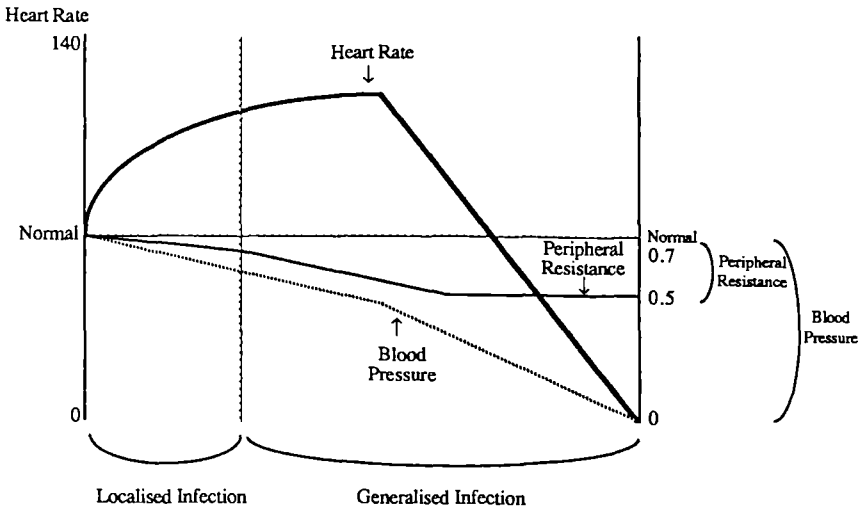


Figure 5.12: Relationships Between Blood Pressure, Peripheral Resistance and Heart Rate In Septic Shock

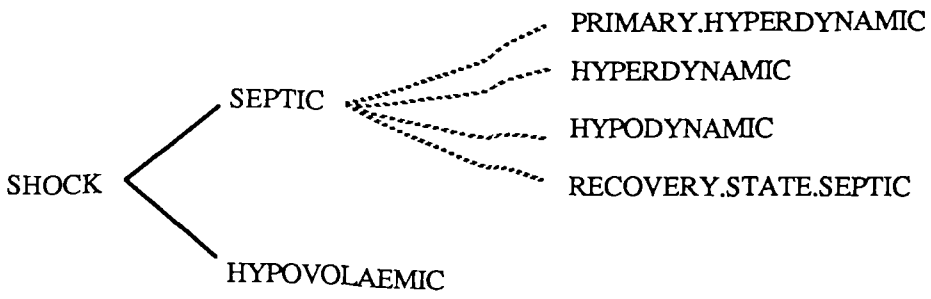


Figure 5.13: Shock Hierarchy

5.4 Combining The Physiological Model With Other Domain Knowledge

Incorporation of the physiological model into the existing disease representation described in chapter 4 necessitated the addition of pathological knowledge not associated with history and physical examination findings to the pathological process hierarchy. Knowledge of the temperature rises accompanying infection has been added to the INFECTION unit and its subclasses. Third space losses have been added to the INFLAMMATION unit and its subclasses. For disease states with infection and inflammation, third space and temperature attributes are determined by the inheritance resulting from subclass links to each pathological process unit.

The shock entity is modelled as a hierarchy with subclasses representing the different types of shock (figure 5.13). The stages seen in the progression of septic shock have been represented as member units of the class SEPTIC. Each pathology is mapped to either hypovolaemic shock or a stage of septic shock. Septic shock is linked to pathologies with infection such that the stages primary hyperdynamic, hyperdynamic and hypodynamic are linked to units INFECTION, INFECTION.LOCALISED.-PERITONITIS and INFECTION.GENERALISED.PERITONITIS respectively. The RECOVERY.STATE.SEPTIC unit describes the regression of septic shock. All other pathologies have hypovolaemic shock. When a disease state is setup, shock types are inherited from the pathological units associated with the disease. If a disease state inherits hypovolaemic and septic shock, because it has pathological processes associated with infection and others which are not, its shock type is assumed to be septic. The most progressed septic shock stage is selected if more than one stage is present. The shock type and for sepsis its progression stage, is assumed constant throughout a disease state.

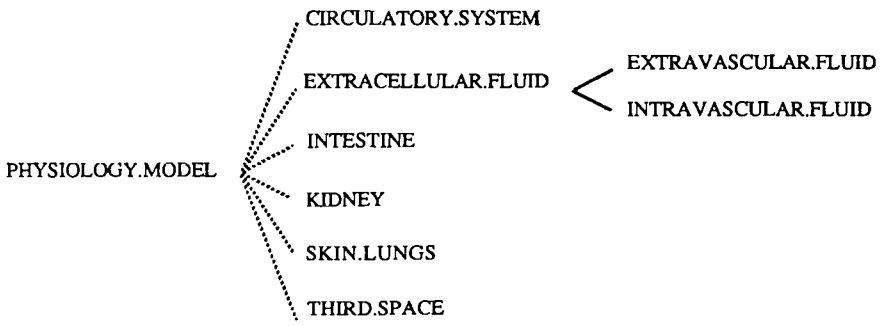


Figure 5.14: Hierarchical Representation Of Physiological Model

5.5 Representation Of Physiological Model Using KEE

The intermediary systems of figure 5.5 are represented by KEE units, arranged in a hierarchy as shown in figure 5.14. The physiological processes (which produce the products described by the terminals in figure 5.5) associated with each intermediary system are described by the elements of a list contained within the SUBSYSTEMS slot of each unit. For example, the SUBSYSTEMS slot of the unit SKIN.LUNGS has the value (TEMPERATURE RESPIRATION.AND.TISSUE.CATABOLISM). All processes are described further by methods residing in the units.

Each physiological process has been represented at two levels, a general and detailed level. At the general level all the processes components are declared. Each of these components is then described quantitatively by a method. For example, the physiological process temperature has as a component the fluid loss resulting from increased temperature. This is then described quantitively by the method VOL.OF.-FLUID.LOSS.TEMP (figure 5.15).

To model the distribution system, messages are passed between the units representing the intermediary systems. On each update of a patient's physiology, the member units of the physiological model are activated in the sequence EXTRACELLULAR.FLUID, CIRCULATORY.SYSTEM, KIDNEY, SKIN.LUNGS, ALIMENTARY.CANAL and THIRD.SPACE. To activate a unit, the list held in its SUBSYSTEMS slot is abstracted and a message is sent to each of the list elements. The processes messaged represent processes which are associated with fluid balance or haemodynamics.

Methods representing processes which effect changes in the bodies extracellular fluid volume put the volume change they produce in the ECF.LOSS slot of the unit in which they reside. If a gain in fluid volume is yielded by a process, its value is negated before putting it in the ECF.LOSS slot. An active value, ON.PUT.ADD, watching over all ECF.LOSS slots adds new ECF.LOSS values to the existing slot value.

Unit: **SKIN.LUNGS** in knowledge base **AB-PAIN**
Member of: **PHYSIOLOGICAL.MODEL**

MemberSlot: ***SUBSYSTEMS** from **PHYSIOLOGICAL.MODEL**
Inheritance: **OVERRIDE.VALUES**
ValueClass: **LIST**
Values: (TEMPERATURE RESPIRATION.AND.TISSUE.CATABOLISM)

MemberSlot: ***TEMPERATURE** from **SKIN.LUNGS**
Comment: General level description of physiological model.
Inheritance: **OVERRIDE.VALUES**
ValueClass: **LIST**
Values: (VOL.OF.FLUID.TEMP)

MemberSlot: ***VOL.OF.FLUID.TEMP** from **SKIN.LUNGS**
Comment: Contains LISP function which puts the volume of fluid lost due to sweating in the ECF.LOSS slot.
Inheritance: **METHOD**
ValueClass: **METHOD**

MemberSlot: ***RESPIRATION.AND.TISSUE.CATABOLISM** from **SKIN.LUNGS**
Comment: Contains LISP function which puts the volume of fluid lost due to respiration and tissue catabolism in the ECF.LOSS slot.
Inheritance: **METHOD**
ValueClass: **METHOD**

MemberSlot: ***ECF.LOSS** from **PHYSIOLOGICAL.MODEL**
Comment: Value is world dependent and represents the extracellular fluid lost between two successive updates of the physological model as a result of active processes within this intermediary system.
AVUNITS: **ON.PUT.ADD ON.GET.ZERO**
ValueClass: **NUMBER**
Values: UNKNOWN

Figure 5.15: KEE Representation Of The Intermediary System SKIN.LUNGS

To recompute the extracellular fluid volume, the ADJUST.ECF.FLUID.VOL method, in the unit EXTRACELLULAR.FLUID, gets the values from the ECF.LOSS slots and finds their sum (figure 5.16). On getting a value from an ECF.LOSS slot, an active value, ON.GET.ZERO, puts the value 0 in the slot. To the sum the fluid gain from intravascular transfusion is added. The total extracellular fluid volume change and 75% and 25% of it are put in the CURRENT.VOL slots of the EXTRACELLULAR.FLUID, EXTRAVASCULAR.FLUID and INTRAVASCULAR.FLUID units respectively. The active value, ON.PUT.ADD, attached to each CURRENT.VOL slot adds the change to the old fluid volume. In this manner, changes in the extracellular fluid volume are propogated to the EXTRAVASCULAR.FLUID and INTRAVASCULAR.FLUID units.

The unit CIRCULATORY.SYSTEM comprises methods to generate the peripheral resistance, the heart rate and blood pressure, in that order, on each simulation update. The heart rate is modelled by five LISP functions, the functions being called PULSEFUNCn where $n = 1, 2, \ldots 5$. To generate the value of the heart rate, the method PULSE.FUNC evaluates the current heart rate function whose name is held in the slot PULSE.FUNC.IN.USE. The value returned is then appended onto the list of previously generated heart rate values, stored in the slot CURRENT.PULSE.

Heart rate functions have parameters which define them, for example the parameters A and C are used to define the pulse functions which have the general form $y = Ax + C$, and constraints which must be satisfied for the function to be active. The parameters and constraint values associated with each heart rate function are held in active value units such that those associated with PULSEFUNCn are held in the active value unit PULSE.FUNC.n.GUARDIAN where $n = 1, 2, \ldots 5$. When a heart rate function is put into the PULSE.FUNC.IN.USE slot, its active value unit is attached to the CURRENT.PULSE slot. On adding a value to the CURRENT.PULSE slot, the AVPUT method of the attached PULSE.FUNC.GUARDIAN checks that the new heart rate value conforms to the constraints of the active heart rate function. If violation of the constraints occurs, the name of a different heart rate function is put into the PULSE.FN.IN.USE slot. The active value, ATTACH.-GUARDIAN.TO.CURRENT.PULSE watching over the PULSE.FN.IN.USE slot, then attaches the PULSE.FUNC.GUARDIAN associated with the new heart rate function to the slot CURRENT.PULSE and puts the name of the old PULSE.FUNC.GUARDIAN in the AV.TO.DELETE slot of the unit PULSE.FUNC.GUARDIAN.MANAGER. This active value also watches over the CURRENT.PULSE slot. When it is activated and more than one PULSE.FUNC.GUARDIAN units are attached to the CURRENT.-PULSE slot, it detaches the outdated guardian, the name of which is held in its AV.-TO.DELETE slot. Detachment of PULSE.FUNC.GUARDIANs' is performed by the

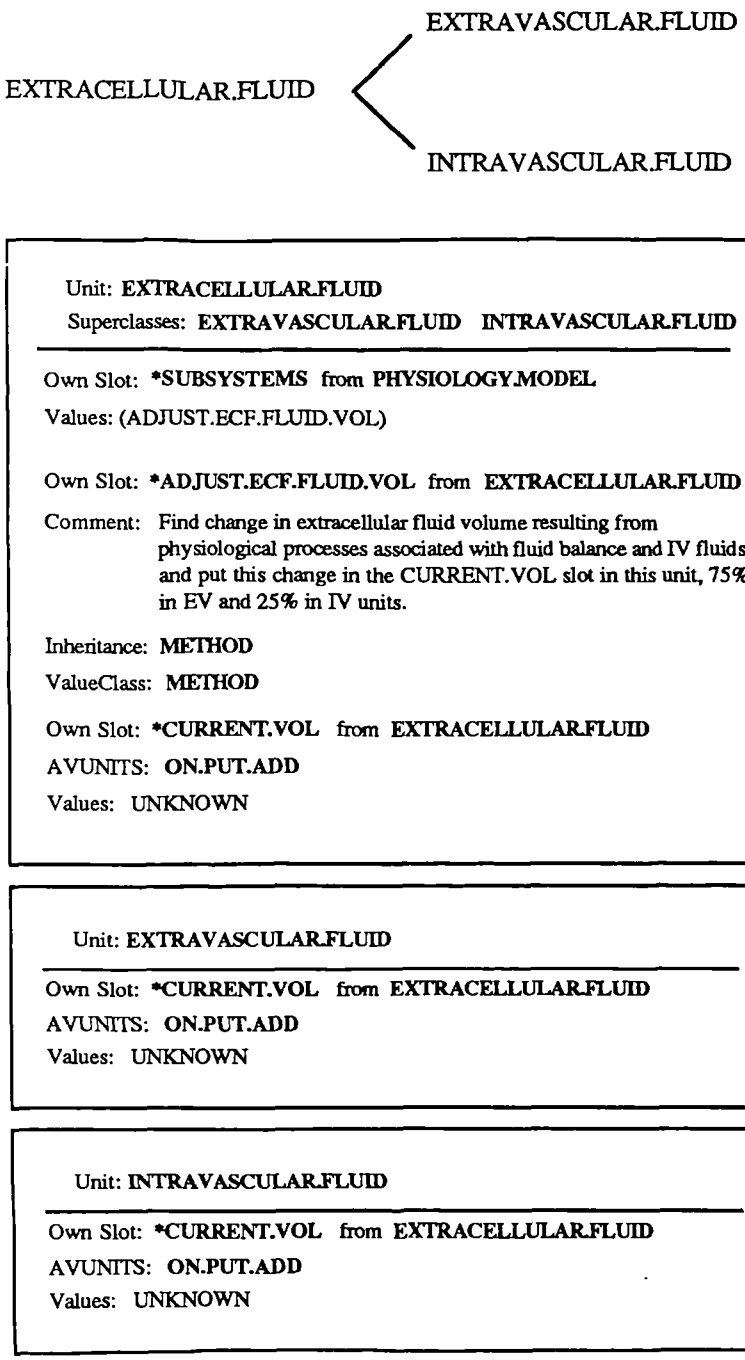


Figure 5.16: Representation Of The Body Fluid In KEE

PULSE.FUNC.GUARDIAN.MANAGER unit because an active value cannot remove itself from a slot.

On adding a PULSE.FUNC.GUARDIAN to the slot CURRENT.PULSE, the guardians AVADD method determines whether the heart rate function has already been defined. It is assumed that if the current heart rate function is PULSEFUNCn then PULSEFUNC$(n-1)$ where n is an integer and $1 < n < 6$ has been defined. To check whether other linear heart rate functions have been defined, the value of the slot A, held in the heart rate function's associated PULSE.FUNC.GUARDIAN is retrieved. If the value is null, the function has not been defined (no heart rate functions are of the form $y =$ constant). The functions parameters are then generated and stored in the slots A and C of the PULSE.FUNC.GUARDIAN.

Table 5.3 describes the constraints and parameters underlying the heart rate model. To illustrate its use consider the following scenario. Assume that the slot PULSE.FUNC.IN.USE has the value PULSEFUNC1, and the slot CURRENT.PULSE is watched over by the active values PULSE.FUNC.1.GUARDIAN and PULSE.FUNC.-GUARDIAN.MANAGER. To generate the next heart rate value, the method PULSE.-FUNC would be activated. This would evaluate the function PULSEFUNC1 whose name would be held in the PULSE.FN.IN.USE slot. PULSEFUNC1 would return a value, say 120. The value 120 would be added to the slot CURRENT.PULSE. Activation of the AVPUT slots in the active value units monitoring the slot would result in no changes being made to the knowledge base. If the value 150 had been returned however the PULSE.FUNC.1.GUARDIAN's AVPUT slot would have detected that PULSEFUNC1's pulse constraints had been violated. It would then put the value PULSEFUNC2 in the PULSE.FN.IN.USE slot and if this function had not been invoked before, add the value ((HR(0) 150)(BP(0) 83)) to the slot PULSE.BP.RANGE in the unit PULSE.FUNC.2.GUARDIAN. The blood pressure is assumed to be 83 when the heart rate is 150. Putting PULSEFUNC2 in the PULSE.FUNC.IN.USE slot would activate the ATTACH.GUARD.TO.CURRENT.PULSE unit's AVPUT method which would put the value PULSE.FUNC.1.GUARDIAN in the AV.TO.DELETE slot of the unit PULSE.FUNC.GUARDIAN.MANAGER and attach PULSE.FUNC.2.GUARD-IAN to the slot CURRENT.PULSE. Since the slot A in PULSE.FUNC.2.GUARD-IAN is empty, the value 150/83 is put in it, thereby defining PULSEFUNC2. The active value PULSE.FUNC.GUARDIAN.MANAGER would then be invoked. This would detach the unit PULSE.FUNC.1.GUARDIAN, whose name is stored in its AV.-TO.DELETE slot. On generating further pulse values transition may be made to PULSEFUNC1 if the blood pressure exceeds 83 and to PULSEFUNC3 if the blood pressure falls below 40 for less that 4 hours and effective fluid therapy is given.

All parameters describing the circulatory system are held in the unit CIRC-

Pulse Func	PARAMETERS			CONSTRAINTS		
	A	C	Others	HR	BP	IV Vol
1			10,000	[HR(t=0), p]	[BP(t+0), x]	
2	$\frac{p}{x}$			[p, o]	[x, 0]	
3	$\frac{120-q}{x-y}$	$\frac{(qx-120-y)}{(x-y)}$		[q,120]	[y,x]	
4	$\frac{120-HR(t=0)}{72-BP(t=0)}$	$\frac{72-HR(t=0)-120xBP(t=0)}{72-BP(t=0)}$		[120, HR(t=0)]	[x, BP(t=0)]	
5	$\frac{-(HR(t=0)-60)}{3xIV(t=0)}$	$\frac{4xHR(t=0)}{3}$		[HR(t=0), 60]		[IV(t=0), 4·IV(t=0)]

Table 5.3: Constraints And Parameters Associated With The Heart Rate Functions

ULATORY.-SYSTEM. To generate the peripheral resistance, the method PR.FUNC messages the method PR.FUNC in the shock unit whose name is stored in the slot SHOCK.TYPE. The value returned is stored in the slot CURRENT. PERIPHERAL.-RESISTANCE. Systolic blood pressure is generated using the method SYSTOLIC.BP. This uses the intravascular fluid volume, current peripheral resistance and initial and current heart rates held in slots of the same name in the units INTRAVASCULAR.-FLUID and CIRCULATORY.SYSTEM. The value returned is added to the list of previously generated systolic blood pressures in the slot CURRENT.SYSTOLIC.BP. The method DIASTOLIC.BP then generates the diastolic blood pressure and stores it in the slot CURRENT.DIASTOLIC.BP.

The peripheral resistance methods in the shock hierarchy units use the following algorithms to generate the peripheral resitance (PR):

For septic shock

$$PR = Current.PR + PR.CHANGE$$

where PR.CHANGE is:

$$\frac{\text{PR on entering shock state} - \text{MIN.PR}}{\text{Duration of Disease State}}$$

for the PRIMARY.HYPERDYNAMIC.SEPTIC and HYPERDYNAMIC.SEPTIC units, where MIN.PR is 0.7 and 0.5 for each of the respective units;

$$PR.CHANGE = 0$$

for the HYPODYNAMIC.SEPTIC unit since maximum vasodilation will have been reached on mapping to the HYPERDYNAMIC.SEPTIC unit at a previous stage in a disease progression; and

$$\frac{0.75 - \text{PR at time of treatment}}{\text{Recovery Time}}$$

for the RECOVERY.STATE.SEPTIC unit, where the period of recovery is associated with each treatment unit.

For hypovolaemic shock

Normal.bp _ Bp.on.simulation.initialisation;
Former.bp = Bp.on.previous.physiology.update;
Current.bp = Bp(current time) generated using current PR(current time-1)
Former.PR = PR(t-1);

if (Current.bp = Former.bp)

(* Either HR increase=IV vol. decrease, HR decrease=IV vol. increase or increase HR and IV vol. are constant. *)

then PR = Former.PR
elif (Current.bp < Former.bp) AND (*Former.PR* < 1)

(* Patient is lossing IV fluid and HR is normal. PR is increased to compensate for the fluid loss. *)

then Change.in.PR.required = $\frac{(\text{Former.bp}*\text{Former.PR})}{\text{Current.bp}}$;

if (Change.in.PR.required > 1)

(* Vasoconstriction cannot alone compensate for the fluid loss. The PR is set to its maximum value. HR will increase as a secondary compensationary mechanism. *)

then PR = 1
else PR = Change.in.PR.required

(* By increasing the PR the potential fall in the BP is totally diverted. *)

fi

elif (Current.bp > Normal.bp) AND (Former.PR > 0.5)

(* The effective blood volume will be greater than its normal value
To compensate for this vasodilation will occur, causing the PR
to fall to its minimum value of 0.5. After the PR has reached a
minimum the HR will fall. *)

then Change.in.PR.required = $\frac{\text{(Former.bp*Former.PR)}}{\text{Current.bp}}$;

if (Change.in.PR.required < 0.5)

(* Vasodilation cannot *alone compensate for the excess in IV*
fluid volume. The PR is set to its minimum value. HR will
decrease as a secondary compensationary mechanism. *)

then PR = 0.5
else PR = Change.in.PR.required

(* By decreasing the PR the potential increase in the BP is
totally diverted. *)

fi

else

(* BP is less than its normal value but may only be compensated
for by changing the HR. *)

PR = Former.PR

fi

The minimum peripheral resistance and hourly change in peripheral resistance for septic shock are represented by the slots MINIMUM.PR and PR.CHANGE. On setting up the shock stage, the peripheral resistance change is derived. The method PR.FUNC then adds the change to the current peripheral resistance to derive the new peripheral resistance.

To initialise the model, methods declared in each physiological unit's INIT.-SUBSYSTEMS slot, are processed in a way analogeous to that by which the SUB-SYSTEMS slot is processed. PULSE.FUNC.1.GUARDIAN is attached to the slot CURRENT.PULSE. The value PULSEFUNC1 is put in the slot PULSE.FN.IN.USE before the active value ATTACH.GUARDIAN.TO.CURRENT.PULSE is attached to

it. The shock type is determined and put into the SHOCK.TYPE slot of the CIRCULATORY.SYSTEM unit.

Values associated with the physiology are generated dynamically over time and held in KEEworlds. As the pathology progresses, its temperature, third space and vomiting attributes may change. These changes will be reflected by the fluid balance which will indirectly affect the pulse, blood pressure and urine output. For diseases with septic shock, the shock stages invoked may become more progressed as a disease develops. The transition to an advanced septic shock stage is accompanied by a decreasing peripheral resistance which may lead to falling blood pressure.

5.6 Extensions To Physiology

The conceptual schema about which the physiology model is arranged facilitates the incorporation of additional domain principles. To illustrate this fluid concentration and composition is addressed.

5.6.1 Assumptions Underlying Fluid Composition and Concentration

The extracellular extravascular (EV) and intravascular (IV) fluid spaces contain electrolytes and other components. It is assumed that in the absence of disease these components have a fixed concentration. The IV space contains the same components as the EV space in addition to proteins, cells (carrying haemoglobin) and urea. A semi-permeable wall, through which electrolytes and water may pass, separates the EV and IV spaces. When fluid balance changes occur the electrolyte concentration changes in both spaces in the ratio of the fluid volume changes. The effect of this is that the concentration of electrolytes will remain the same in both fluid compartments.

Different types of fluid affecting the bodies fluid balance have different compositions and concentrations. The regulation of sodium and potassium by the kidney, the electrolyte composition of vomit and diarrhoea and the electrolyte changes associated with intravenous fluid theraphy are discussed in more detail to enable description of the extensions to the physiological model required to model extracellular concentration and composition changes.

To compensate for losses and gains in sodium and potassium the body is able to shift electrolytes from its intracelluar reserves to the extracelluar space. The importance of these reserves on the balance of an electrolyte depends upon the difference in its EV and intracellular concentration. For electrolytes such as sodium, where the concentration of the intracellular reserves is much less than the normal EV concentration the influences of the reserves are minimal, only coming into effect when the EV sodium level falls below that of the intracellular fluid. Conversely, for electrolytes

such as potassium, which have a much greater intracellular concentration than EV concentration, losses of electrolytes are better compensated for by the reserves, restoring the EV electrolyte concentration at its normal level longer. In addition to the shift of electrolytes from the intracellular space to the extracelluar space, the kidney is able to regulate the amount of electrolytes output in the urine. It may concentrate or dilute electrolytes output in the urine depending upon their excess or deficit in the IV fluid.

Modelling Of Sodium Regulation By The Kidney

Given that the normal concentration of IV sodium is 142 mmol/litre [31], the following assumptions have been made regarding the concentration of sodium in the urine:

- When the IV sodium concentration is greater than its normal value, excess sodium is output to maintain the normal IV concentration. Sweating is a process which incurs large losses of fluid containing little sodium.
- Should the IV sodium concentration become ≤ 120, no sodium is output in the urine.
- For IV sodium concentrations of between 120 and normal, the sodium concentration of the urine is the same as the intravascular fluid concentration.

The following knowledge is implicit in the model:

- The kidney is functioning normally and retains its functionality no matter how high the extracellular sodium concentration rises.
- If transfusions are given which contain little or no sodium, the extracellular sodium concentration becomes very dilute. To compensate for this the kidney does not excrete sodium in the urine.
- The patient is too ill to eat so sodium gain by food consumption is ignored. Intracellular shifts of sodium are assumed insignificant and are also ignored. The only means of inputing sodium to the body is by infusion.

Secondary conditions arising from sodium imbalance are hyponatremia and hypernatremia. The former is assumed to occur if the IV sodium concentration is $<$ 130mmol/litre and has been so for two or more hours; the later if the IV sodium concentration is $\geq$ 150mmol/litre, also for two or more hours. An algorithm for generating the urine sodium concentration is given in figure 5.17.

Norm.IV.Conc = 142;

Generate ECF.vol(t);

Current.IV.Conc = $\frac{\text{IV.conc}_{(t-1)} \times \text{ECF.vol}_{(t-1)}}{\text{ECF.vol}_{(t)}}$;

Generate urine volume;

(* Calculation urine sodium concentration. *)

if Current.IV.Conc $\leq$ 120

then Urine.Sodium.Conc = 0

elseif 120 $\leq$ Current.IV.Conc $\leq$ Norm.IV.Conc

then Urine.Sodium.Conc = Current.ECF.Conc

else (* IV sodium concentration is above its normal value. *)

$$\text{Urine.Sodium.Conc} = \text{Current.ECF.Conc} + \frac{\text{Current.IV.Conc}[\text{ECF.vol}_{(t-1)} \times \text{Norm.IV.Conc}]}{\text{Urine Volume}}$$

fi

Figure 5.17: Algorithm To Model The Kidney's Sodium Regulation

Modelling Of Potassium Regulation By The Kidney

In the event of no disease, the intracellular concentration of potassium is much greater (150 mmol/L) than the extracellular concentration (5 mmol/L) [35]. When the extracellular potassium concentration is reduced, electrolyte is shifted from the cells to the extracellular space to restore the concentration to its normal value. It is assumed that 30 mmol of potassium is output in the urine and 10 mmol in the faeces daily, in the absense of disease. These losses occur in disease, and have been allowed for by adding 40 mmol of potassium daily to the extracellular fluid, the source of which is assumed to be the cells.

Unlike sodium, if the IV potassium concentration is low ($\leq$ 2.5mmol/L), potassium is not completely reabsorbed. Instead it is shifted from the cells to restore the concentration to 2.5 mmol/L. The ratio of intravascular to urine potassium is assumed to be 4:1 for intravascular concentrations of upto 5 mmol/L. When the intravascular potassium concentration rises above 5 mmol, excess potassium is output in the urine to restore the balance to 5 mmol, inaddition to potassium with concentration 5/4 $\times$ (Volume of fluid lost in litres) mmol. An algorithm for generating the urine potassium concentration is shown in figure 5.18.

Normal.IV.Conc = 5 mmol/L;

Ratio.IV.To.Urine.Potassium = $\frac{1}{5}$;

Current.ECF.Conc = $\frac{\text{IV.Potassium.Conc}_{(t-1)} \times \text{ECF.vol}_{(t-1)}}{\text{ECF.vol}_{(t)}}$ mmol/L;

Add fixed amount of potassium to Current.ECF.Conc to represent that consumed by the urine and faeces (40 mmol/L);

Calculate Urine.vol;

(* Generate concentration of potassium in the urine. *)

if Current.ECF.Conc $\leq$ 2.5

then Urine.Potassium.Conc = 2.5 $\times$ Ratio.IV.To.Urine.Potassium;

IV.Potassium.Conc = 2.5

elif 2.5 < Current.ECF.Conc $\leq$ Normal.IV.Conc

then Urine.Potassium.Conc = Current.ECF.Conc $\times$ Ratio.IV.To.Urine.Potassium;

IV.Potassium.Conc = $\frac{\text{Current.ECF.Conc} \times \text{ECF.vol}_{(t)} - \text{Urine.Potassium.Conc} \times \text{Urine.vol}}{\text{ECF.vol}_{(t)}}$

else (* IV potassium concentration is excessive. *)

Urine.Potassium.Conc = Ratio.IV.To.Urine.Potassium.Conc $\times$Normal.IV.Conc+ $\frac{\text{ECF.vol}_{(t)}(\text{Current.ECF.Conc} - \text{Normal.IV.Conc})}{\text{Urine.vol}}$;

IV.Potassium.Conc = Normal.IV.Conc

fi

Figure 5.18: Algorithm To Model The Kidney's Potassium Regulaton

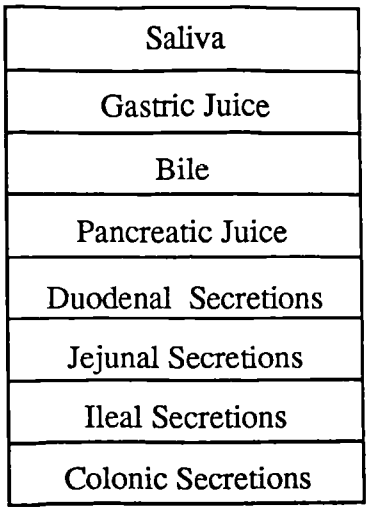


Figure 5.19: The Sequence Of Alimentary Secretions

Modelling Of Vomit And Diarhoea Concentration And Composition

Within the alimentary canal secretions are produced which have different electrolyte concentrations. These secretions may be arranged in the order of their relative position of production in the alimentary canal (figure 5.19). The sequence of secretions may then be inferred using the anatomical model, given that each secretions site of production maps to an anatomical part. For example bile maps to the gallbladder and gastic juice maps to the stomach. The stomach is located above the gallbladder, inferred by following the CONNECTED.TO relationships beginning with the oesphagus, so gastric juice comes before bile in the sequence of alimentary secretions.

The cause of each abdominal disease is associated with an anatomical part. This provides a reference point for determining the secretions involved in the intestine loss. It is assumed that all alimentary secretions above the reference point are found in vomit, below the reference point in diarrhoea. The concentration of electrolyte n in vomit or diarrhoea containing secretions a through b is found as follows:

$$\text{Concentration of electrolyte } n = \frac{\sum_{i=a}^{b} \text{mmol of electrolyte } n \text{ in } i}{\sum_{i=a}^{b} \text{Daily vol. of i}}$$

The number of mmol of electrolyte n lost =

$$\text{Volume of fluid lost} \times \text{Concentration of electrolyte } n$$

Modelling Of Intravenous Fluid Concentration And Composition

Different types of colloid and crystalloid solutions are available for transfusion. The electrolyte composition (mmol/L) of various types of colloids and crystalloids is

Solution	Na^+	K^+	HCO_3^-	Cl^-	$Mg\_$
Crystalloids:					
Sodium Chloride	154			154	
Sodium Chloride and Glucose	30			30	
Glucose					
Sodium Bicarbonate	150		150		
Sodium Lactate	131	5	29	111	2
Colloids:					
Dextran 40	154			154	
Gelatin	154			154	
Plasma	142	4.5	26	103	2.2

Table 5.4: Electrolyte Concentration Of Crystalloid And Colloid Solutions

shown in table 5.4[31] . When giving fluid, its electrolytes are distributed between the IV and EV spaces in the same ratio as the fluid volume (page 92). For example, giving 1 litre of the crystalloid sodium bicarbonate, which has a sodium concentration of 150mmol/L, 15 mmol would be distributed within the EV space, 135 mmol within the IV space.

5.6.2 Proposed Implementation Of Fluid Composition And Concentration

The extracellular fluid components would be described by slots with values representing their concentrations. Member slots defining components common to both the EV and IV fluids would be held in the unit EXTRACELLULAR.FLUID and passed down to the EXTRAVASCULAR.FLUID and INTRAVASCULAR.FLUID units by inheritance. Components exclusive to the intravascular fluid would be represented by own slots in the INTRAVASCULAR.FLUID unit. The *normal* component concentrations would be supplied by default values.

The composition of each fluid associated with changing fluid balance would be modelled by considering each of the fluids components separately. The components would be declared in the slot describing the process and quantified by methods. For example the slot describing urine generation in the unit KIDNEY would contain a list with elements including VOLUME, SODIUM.CONCENTRATION and POTASSIUM.CONCENTRATION. Methods to generate the SODIUM.CONCENTRATION and POTASSIUM.CONCENTRATION, would be based on the algorithms which have been described and would return the urine sodium and potassium concentration in mmol/L. These values would be stored in the slots NA.CONC and K.CONC of the unit KIDNEY. Methods to perform this operation would be declared in the slot COMPOSITION.AND.CONCENTRATION, listed in the SUBSYSTEMS slot of the unit EXTRACELLULAR.FLUID. To determine the change in IV and EV fluid com-

position and concentration each of the physiology model's units would be scanned to determine the composition and concentration of processes associated with it. Together with the extracellullar fluid volume changes, the new IV and EV concentration would be found.

A part of the fluid composition and concentration has been implemented to assess whether the additional knowledge it would require is compatible with the existing knowledge representation. The areas addressed were the electrolyte concentrations of vomiting and diarrhoea, and of intravenous fluids.

To model the alimentary secretions the unit ALIMENTARY.CANAL has been defined with slots representing the general attributes of a secretion. These are the volume of the secretion produced, its calcium, bicarbonate, potassium and sodium concentration in the absence of disease and the predecessor and successor secretions in relation to the conceptualisation of the alimentary canal. To link a secretion to an anatomical part, the anatomical part associated with the production of a secretion is made a member of the unit ALIMENTARY.CANAL. For example bile is associated with the gallbladder so the unit GALLBLADDER is a member of the unit ALIMENTARY.CANAL. Values pertaining to specific secretions are held in anatomical units. For example, the value of the sodium concentration of the bile is held in the NA.CONC slot of the GALLBLADDER unit.

The LISP function CONC.LOSSES.ALIM.CANAL, which has arguments *level, loss.type* and *volume*, returns a list of the electrolyes lost in *volume* litres of *loss.type* vomit or diarrhoea together with the total number of mmol of each electrolyte lost. *Level* provides a reference point for determining the secretions involved (figure 5.20).

The intravenous fluids have been represented by a unit hierarchy. The root unit, IV.FLUIDS, represents a prototype fluid with member slots describing each electrolyte. This has subclasses COLLOID and CRYSTALLOID. Member units of these subclasses describe fluids of types colloid and crystalloid. For example, the colloids DEXTRAN.40, GELATIN and PLASMA are member units of the subclass COLLOID. The electrolyte concentrations of specific fluids are held in the own slots which describe electrolytes and have been inherited from the IV.FLUIDS unit. When generating the volume of fluid input into the extracellular space the number of mmol's of each electrolyte in the fluid is found by multiplying the volume (in litres) by its concentration in mmol/L. This is then distributed about the space s where s represents the IV or EV space as follows:

$$\text{Conc. electrolyte in } s = \frac{\text{Fluid vol. input into } s}{\text{Fluid vol. input into the ECF space}} \times (\text{mmol electrolyte input})$$

The extensions which have been incorporated into the knowledge base are shown in appendices A and B.

```
(CONC.LOSSES.ALIM.CANAL
 (LAMBDA (LEVEL LOSS.TYPE VOL)

      (*Returns a list of the total number of mmol's of the electrolytes NA, K, CL and
       HCO3 lost in VOL litres of fluid of type LOSS.TYPE (either vomit or diarrhoea)
       from the reference point LEVEL, a part of the alimentary canal. It is assumedthat
       all secretions are mixed in proportion to the volumes of each secretion produced daily.)

(LET* ((LOSS.TYPE.LINKS (COND
                            ((EQUAL LOSS.TYPE 'VOMIT) 'SUCCESSOR.SECRETION)
                            (T 'PREDECESSOR.SECRETION)))
       (TOT.DAILY.VOL.ALL.SECRETIONS
            (TOT.DAILY.VOL.SECRETIONS.PRODUCED LEVEL LOSS.TYPE.LINKS)))

      (MAPCAR '(NA K CL HCO3)
        '(LAMBDA (X)
           (LIST X (IQUOTIENT (TIMES 100 VOL (CALC.ELEC.CONC X LEVEL
            LOSS.TYPE.LINKS)) TOT.DAILY.VOL.ALL.SECRETIONS)))))))

(TOT.DAILY.VOL.SECRETIONS.PRODUCED
 (LAMBDA (LEVEL DIRECTION)

      (*Returns the total daily volume of secretions produced (in litres) from the unit LEVEL
       in the alimentary canal to an end unit in the direction DIRECTION.)

(COND
  ((NULL (GET.VALUE LEVEL DIRECTION))
   (GET.VALUE LEVEL 'DAILY.VOL.SECRETION))
  (T (+ (GET.VALUE LEVEL 'DAILY.VOL.SECRETION)
        (TOT.DAILY.VOL.SECRETIONS.PRODUCED (GET.VALUE LEVEL
          DIRECTION) DIRECTION))))))

(CALC.ELEC.CONC
 (LAMBDA (ELEC LEVEL DIRECTION)

      (*Returns the sum of the electrolyte concentration multiplied by the daily volume
       of secretion produced for each secretion from LEVEL to the end of the alimentary
       canal in the direction DIRECTION.)

(COND
  ((NULL (GET.VALUE LEVEL DIRECTION))
    (TIMES (GET.VALUE LEVEL (PACK* 'NORMAL. ELEC '.CONC))))
  (T (+ (TIMES (GET.VALUE LEVEL (PACK* 'NORMAL. ELEC '.CONC))
        (GET.VALUE LEVEL 'DAILY.VOL.SECRETION))
     (CALC.ELEC.CONC ELEC (GET.VALUE LEVEL DIRECTION) DIRECTION))))))
```

Figure 5.20: LISP Function Returning The Electrolyte Concentration Of Intestinal Secretions

5.7 Conclusion

A model has been constructed to represent the physiological changes associated with fluid imbalance in the context of the pathological processes demonstrated in acute abdominal diseases. Previous programs have addressed pathophysiological changes within the confines of single systems, for example the urinary system. This work has considered pathological changes in general and their effects systemically.

The physiological model is a quantitative model which utilises symbolic and mathematical reasoning. The pathological processes discussed in chapter 4 have been mapped to septic and hypovolaemic shock. Septic shock is associated with the pathological process infection and becomes more pronounced as the infection progresses. Pathologies not incorporating infection are associated with hypovolaemic shock. The pathological attributes which affect fluid balance, for example temperature and third space losses, together with knowledge of intravenous fluid administration are passed to the model in symbolic form. The attributes are quantified within the model using mathematical functions are are used to generate physiological parameters which would be recorded at the bedside, for example blood pressure and heart rate.

The model is updated hourly, since this is the frequency with which the physiological parameters it generates are measured clinically. Also the treatment modalities available for utilisation in septic and hypovolaemic shock result in changes in the patients status within an hour of administration.

A patient's normal physiological parameters are varied in accordance with their age, sex and weight which are described by the patient profile detailed in chapter5. Having initialised a normal patient's physiological status, it may be varied within the confines of a disease. This is achieved by randomising the functions representing the physiological processes and the pathological progression a disease by follow, ie. a disease transition. In this way the physiology produced is non-deterministic.

An object-oriented representation has been used to structure the model. Physiological processes associated with different systems of the body, for example the urinary and cardiovascular systems, are described by separate objects and listed each objects SUBSYSTEMS slot. Each process is represented at two abstract levels, a declarative and procedural level. The declarative level describes the concepts of the process which have been modelled, for example fluid volume. Each of the concepts is represented by a mathematical function at the procedural level. The declarative and procedural levels are described by slots with valueclass LIST and METHOD respectively.

By structuring the model about an object oriented representation in this manner, it is easily expandable and comprehensible. The nomenclature describing each physiological object enables knowledge to be easily accessed given that the person searching

the knowledge base is familiar with the medical domain. The declaration of knowledge which is represented procedurally at another conceptual level makes the model easier to understand. It is expected that this will assist in the models evaluation and make it more acceptable to the user. Expansion of the model has been demonstrated by the partial implementation of fluid composition and concentration.

Chapter 6

The Tutorial Manager, Simulation Generator And Model, And Student History

6.1 Introduction

This chapter describes the configuration of the tutorial manager, simulation generator, and simulation model and student history modules of the prototype system. The simulation generator is coupled to the tutorial manager and domain knowledge components to provide a co-operative system [53] which builds a simulation model. The simulation model is a discrete model, the state variables[1] changing at a finite number of points in time. The inferencing procedure for activating events[2] is a 'combined fixed-increment time advance' event-scheduling, activity scanning algorithm [76].

Activation of events in an event-scheduling algorithm is driven by time only, no other world conditions are used to generate event routine[3] activation. In the next-event time advance approach the time of occurrence of future events is found. The simulation is advanced to the time of occurrence of the first of these events and the system state is updated to take into account the event. The time of occurrence of future events is reviewed and the simulation clock is updated to the next most imminent event. The process of advancing the simulation clock from one event time to another is continued until some predefined stopping condition is reached. In the fixed-increment time advance approach [41] the simulation clock is incremented in steps of Δt time units, for some appropriate choice of Δt. After each update of the clock, a check is made to determine whether any events have occured during the previous time interval

[1] Variables used to describe the system at a point in time.

[2] Instanteous occurrences which may change the state of the system.

[3] Program code associated with an event which describes the changes to be made to the state variables when the event occurs.

of length Δt. If one or more events are scheduled to occur during the interval they are scheduled to occur at the end of the interval, their order of priority given by some predefined rule set. The system state is then updated accordingly.

The activity scanning strategy [41] assumes a system comprises components which engage in activities subject to specified conditions. An event may be activated on a time condition being met, as in event scheduling, and also because other conditions within the system have been satisfied. Each component has associated with it an *activity routine* which models active phases of the component, a *clock* which indicates the time at which the component may next by considered for activation and a *condition routine* to determine whether conditions other than the passage of time have to be met before the component may be activated. A control procedure scans the activities in priority order for time eligibility and other activation conditions, executing the activity routine of the first component whose activation conditions are met. On activation, the scan starts again in priority order, the process continuing until no conditions fire. Time is then advanced asynchronously, the scanning procedure being repeated until a termination condition is reached.

To overcome the inefficiencies associated with rescanning on all the activities following the activation of one, a more structured activity list was proposed by Tocher [76]. This minimised the number of activities rescanned on each phase of the control procedure and used the next-event time advance event-scheduling mechanism to ensure that on each scan at least activity would be activated. Tocher's 3-phase approach utilised the following protocol:

- Advance time using the next-event time mechanism.
- Schedule time dependent events for prior execution.
- Scan conditional events.

Events associated with the simulation model generated by the system described are dependent on time and/or other conditions. The inferencing mechanism for activating events is a variant of Tocher's algorithm, figure 6.1. It uses the fixed-increment time advance approach to update the simulation clock, the time increment being one simulated hour. Application of the next-event time advance mechanism would be inappropriate since periods of inactivity of the model would be skipped over by jumping the clock. During periods of inactivity the student may request actions on the simulation model, so it is important that these periods are modelled. Since the physiology model is updated every simulated hour on each phase of the control procedure at least one event occurs.

The similarities between artificial intelligence and simulation techniques has been described in the literature. O'Keefe [53] has noted that both expert systems

```
Perform initialisation; (*Set ending-time, put initial events into event list*)
Set sim.hours=1;
While (sim.hours < ending − time) do
        (*proceed next-event list*)
        While (last entry in next-event-list hasn't been processed) do
                retrieve next-element next-event-list;
                execute next-element next-event-list
        EndWhile;
        (*proceed conditional-event list*)
        While (last entry in conditional-event-list hasn't been processed) do
                Set i to next-element process-conditional-list;
                if (condition-routine-i evaluates to T)
                then execute-activity-routine i
                fi
        EndWhile;
        Set interrupt to occur after a fixed time period;
        sim.hours=sim.hours + 1
EndWhile
```

Figure 6.1: Event Scheduling/Activity Scanning Algorithm For Simulation Generator

and simulation comprise modular representations of a system which is driven by some inferencing mechanism. Adelsberger and his colleague [3] have made analogies between frames and entities of event scheduling and activity scanning. Both representations are charactised by attributes. Also, the conditional events in simulation, for example the condition routines in activity scanning, can be compared with the production rule. Research has begun to explore whether the use of artificial intelligence techniques in simulation overcomes the limitations of conventional simulation methods.

Stelzner et al.[74] have stated that conventional approaches to simulation are limited because:

- The obscurity of programming languages makes it difficult for users to validate the accuracy of a simulation model's output.
- Modified models must be re-compiled before the effects of modifications can be seen. This results in a lengthy turn-around time.
- Experience of a language is imperative for the development and modification of a model, other than to change certain parameters.
- Models are inextensible.

Knowledge-based approaches are seen by Stelzner and his colleague to overcome these restrictions. Models are made more understandable by the use of declarative knowledge, visual icons and symbols and are interactive. Parameters, stored as slot values in frames, are accessible for inspection by the program or a user. In conventional simulations values are stored in arrays and are accessible to the program only. Finally, the models are extensible. They may be extended and used for the same task, or new inferencing routines may be added to enable the simulation of different tasks. Object-oriented simulation toolkits include SIMKIT [74] which works with KEE and DEVS [83].

The simulation described has been developed using an object-oriented representation and has been implemented in KEE. The benefits of this representation that have been identified by Stelzner et al. have been exploited. Inaddition, KEE's worlds representation has been used to partition the simulation model such that it may be reset from points where critical events (see chapter 3, page 48) occured.

6.2 Coupling Of The Knowledge Types

The interface between the different knowledge types is shown in figure 6.2. Details of the tutorial to be setup are passed to the tutorial manager via the user interface together with a request to commence tutorial generation. The tutorial manager gives

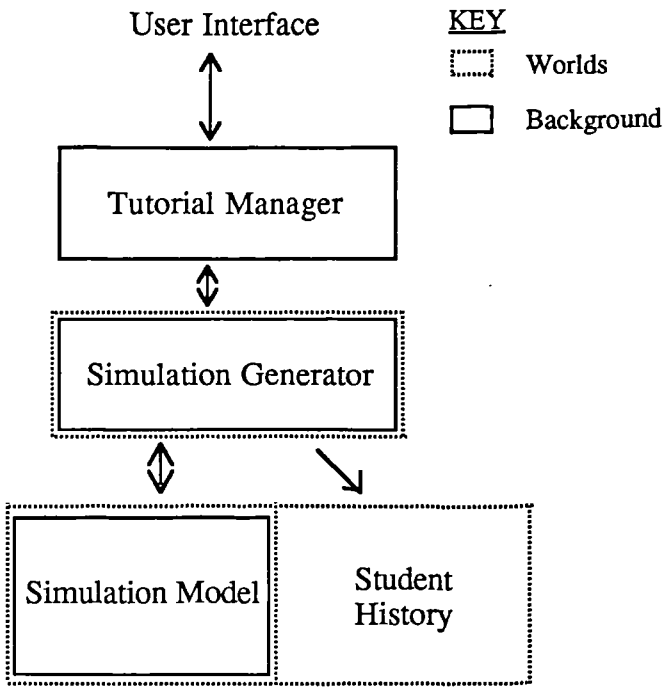


Figure 6.2: Interconnections Between The Tutorial Manager, Simulation Generator, Student History And Simulation Modules

the tutorial specification to the simulation generator. The simulation generator defines a simulation model conforming to the specification. The model's disease state, pathological and pain character knowledge is represented in the background or problem hierarchy, the anatomical, physiological and sign and symptom complex knowledge is held in a world hierarchy. For all simulated times, a patient's condition is described by a problem state in the background, and the physiology and anatomy models viewed in the context of a world. On disease state transition a node in the problem hierarchy is created together with a world in the world hierarchy.

A process, SIM.PROCESS, is created to update the simulation model by one simulated hour after a fixed amount of real time has elapsed. When the simulation process is inactive, the tutorial manager process (KEE's ttyprocess) may respond to student actions, described in detail in chapter 7. These actions constitute critical events and result in the spawning of a world.

World properties define the simulated hours over which the world was active, its corresponding problem state in the problem hierarchy and the reason for its creation. These properties define the student history and are processed by the tutorial manager in the production of a tutorial summary. They are also used to reset the simulation.

6.3 Tutorial Manager

The tutorial manager is represented by an object with methods describing its functionality and other slots describing attributes of the tutorial to be setup. The name of each slot followed by its inheritance role and valueclass and a description of its use is given below:

PROBLEM.COMPLEXITY

OVERRIDE

ONE.OF MULT.ACUTE ACUTE ACUTE.CHRONIC MULT.ACUTE.CHRONIC

Maximum complexity of the problem to be simulated. The problem may comprise an acute or multiple acute disease states, with or without a chronic disease state.

DISEASE.CLASS

OVERRIDE

ONE.OF GB.DUCT.DISEASES

Class of diseases from which an instance is to be simulated.

TUT.DUR.SIM.TIME

OVERRIDE

NUMBER

Number of hours to be simulated before a summary is produced, assuming early termination is not made.

TUT.DUR.REAL.TIME

OVERRIDE

NUMBER

Maximum real time(in minutes) the tutorial is active before a summary is produced.

REAL.TIME.OF.SIM.HR

OVERRIDE

NUMBER

Real time delay (in seconds) between successive simulation updates. Equivalent to Tutorial duration real time/Tutorial duration simulated time.

GENERATE.SIMULATION.RUN

METHOD

METHOD

Calls SIMULATION.SETUP.MANAGER of the unit PS.ACTIVATOR to setup a simulation conforming to the tutorial specification defined by slots in this unit.

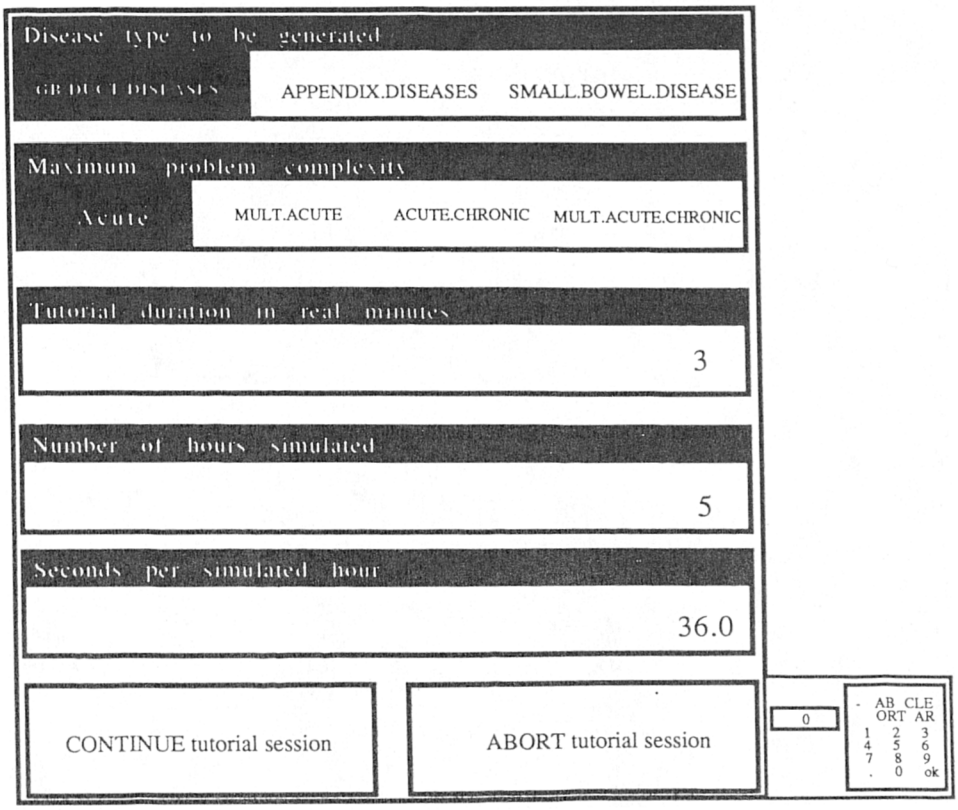


Figure 6.3: Viewport Interfacing To The Tutorial Manager

HALT.PATIENT.SIMULATON

METHOD

METHOD

Deschedules the simulation generator by setting the global variable **term** to 'true'. Messages the user interface to report that simulation generation has been suspended.

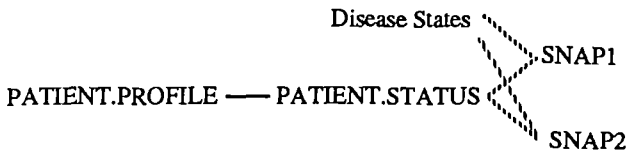
OPEN.DEFAULT.SETTINGS

METHOD

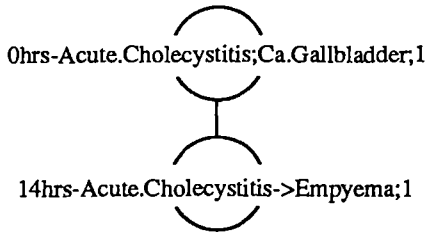
METHOD

Opens the tutorial profile viewport.

A viewport, figure 6.3, is used to interface the TUTORIAL.MANAGER object with the course tutor. Active images attached to the viewport allow the manager's non-method slots to be modified. For example, mousing on the image title, 'tutorial duration in real minutes' would activate a keypad to enable the tutor to modify the TUT.DUR.-REAL.TIME slot. On changing the value of one of the slots REAL.TIME.OF.SIM.HR, TUT.DUR.REAL.TIME or TUT.DUR.SIM.TIME, an active value updates the other slot values to conform to the change.



(a)



Anatomical model: View1
Snap: Snap1
Associated simulated hours: 0-13
Physiological Parameters For Times 0-13

Anatomical model: View2
Snap: Snap2
Associated simulated hours: 14, 15···
Physiological Parameters For Times 0-15···

(b)

Figure 6.4: Representation of a patient's changing status where (a) shows the knowledge held in the background and (b) in worlds

6.4 Simulation Generator/Simulation Model/Student History

6.4.1 General Schema

The simulation generator comprises two objects; PS.ACTIVATOR and SIMULATION.GENERATOR.EVENTS. PS.ACTIVATOR contains procedural knowledge to initialise a simulation model, effect a state transition and reset a simulation. SIMULATON.GENERATOR.EVENTS contains declarative knowledge required in the generation of the simulation model, for example knowledge of events, the current disease states being generated, active complexes and the hospital clock. Hospital time is the time being simulated (*Simulated.real.time* in chapter 3).

On initialisation of the simulation model the simulation clock, represented by the global variable **sim.time** is set to zero. The properties of the simulation model which are invariant over intervals of simulated time are held in the background. These are described by the patient profile, patient status and problem state frames which are arranged in a hierarchy (figure 6.4a). The patient profile is constant throughout a simulation and is the root node of the hierarchy. The problem states provide "snapshots' of the patient's changing condition and so are named *snaps*. A new snap is created on the transition of a disease from one state to another. Specification of the attributes common to each snap is provided by the object PATIENT.STATUS.

In chapter 4 the simulation of one disease progression over time was described. The simulation generator makes provision for the simulation of multiple concurrent acute and chronic disease states. Disease states are manipulated using a *triple* disease

state data structure. This has the format:

```
( DISEASE-STATE
  CAUSE
  SITE )
```

where CAUSE is a triple with elements:

```
( PHYSICAL-ENTITY-CAUSING-ABNORMALITY
  SEVERITY-OF-DEFORMITY
  DEFORMITY-CAUSED-BY-ABNORMALITY ).
```

For each chronic disease it is assumed that unless the disease is correctly treated it is described by one state which does not progress but remains fixed throughout a simulation. Chronic diseases are held in the STATIC.DIS.STATES slot of the unit SIMULATION.GENERATOR.EVENTS, each disease being described by a disease state triple. Acute diseases are described by triples with format:

```
( SIMULATED-HOUR-OF-STATE-PROGRESSION
  CURRENT-DISEASE-STATE
  SUCCESSOR-DISEASE-STATE )
```

where CURRENT-DISEASE-STATE and SUCCESSOR-DISEASE-STATE are described by disease state triples. Acute disease specifications are held in the slot DYNAMIC.DIS.STATES of the unit SIMULATON.GENERATOR.EVENTS.

Each 'snap' represents single or multiple disease states and their associated pathologies over an interval of time. Integers appended onto the end of the names of snap frames represent the relative times of creation of snaps. For example, 'snap1' was created before 'snap2'. The global variable, **hr.dis.state.update**, represents the simulated hour at which the next disease state transition takes place. On setting up a disease state **hr.dis.state.update** is found by scanning the active acute disease states held in the DYNAMIC.DIS.STATES slot of the unit SIMULATION.GENERATOR.-EVENTS and selecting the earliest simulated hour of state progression, from the disease specifications.

Knowledge of the physiology, sign and symptom complexes, and anatomy associated with the disease states (represented in the background) is described by a world hierarchy, each world having associated one snap. The root world corresponds to snap1. On disease state transition a child world of the world associated with the snap representing the old disease state is created, to hold world knowledge associated with the new disease state. This is illustrated in figure 6.4 where snap1 and snap2 correspond to the worlds 0hrs-ACUTE.CHOLECYSTITIS;CA.GALLBLADDER and 14hrs-ACUTE.CHOLECYSTITIS→EMPYEMA respectively. World transition paths, inferred from the processing of the world hierarchy in a top-down manner the name of the root world being held in the slot ROOT.WORLD.NAME of the unit SIMULATION.GENERATOR.EVENT, represents anatomical and physiological progression

of the simulated disease over time. Child worlds are also spawned by the simulation generator when student requests are issued.

The nomenclature for worlds is as follows:

Root world :: 0hrs- Presenting-condition
Other worlds :: Simulated-hour-world-created hrs- Reason-world-created

where

Disease-state-progression	:: [Progression-state ;] Progression-state
Presenting-condition	:: [Presenting-condition ;] Disease-state
Reason-world-created	:: Disease-state-progression / Student-request
Disease-state-progression	:: [Progression-state ;] Progression-state
Progression-state	:: Disease-state → Disease-state
Disease-state	:: Subclass of a class of diseases
Simulated-hour-world-created	:: An integer

and student requests are detailed in chapter 7.

Worlds have properties as follows:

Time.world.start Describes the simulated hour at which the world is created.

Time.world.end Describes the simulated hour at which the world ends.

Path.state The 'snap' describing the disease states and pathology associated with the world. For example the worlds 0hrs-ACUTE.CHOLECYSTITIS;CA.GALL-BLADDER and 14hrs-ACUTE.CHOLECYSTITIS→EMPYEMA in figure 6.4 have path.state properties 'snap1' and 'snap2' respectively.

Cause The reason the world was created. Takes value NATURAL.PROGRESSION or MANAGEMENT.

Spec.cause The specific reason for world creation. For worlds with cause Natural.-progression, the specific cause is a particular progression, for example Acute.-Cholecystitis→Empyema. For worlds with cause MANAGEMENT the specific cause is the type of management.

Seq.no The level of the world in the world hierarchy and the order it was created in when taking into account other worlds on the same level and having the same parent. Figure 6.5 illustrates the seq.no properties of worlds in a typical world hierarchy. The root world has sequence number W1. The seq.no of child worlds is generated by concatenating the parent world's seq.no, the '.' character and the sum of (number of sister worlds + 1). By comparing the sequence numbers of

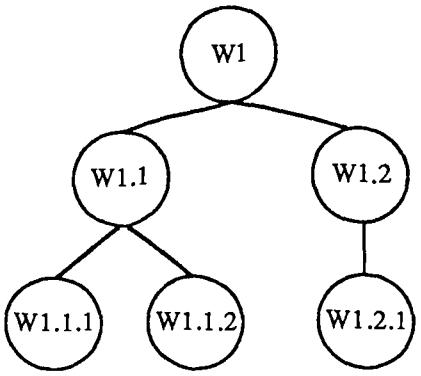


Figure 6.5: Sequence Numbers In A Typical World Hierarchy

worlds on the same level in the world hierarchy and sharing the same parent, their order of creation may be found. For example, W1.1 was created before W1.2, and W1.1.1 was created before W1.1.2. Also, worlds on the same level radiating from the same parent have the same start time. Thus worlds with seq.no's W1.1 and W1.2 have the same Time.world.start property, as do worlds with seq.no's W1.1.1 and W1.1.2. World transition paths are W1-W1.1-W1.1.1, W1-W1.1-W1.1.2 and W1-W1.2-W1.2.1.

Each patient scenario has associated a world transition path. Processing the seq.no properties of the world hierarchy to determine the transition paths and their order of creation enables a summary of a tutorial to be derived. This is presented in textual and graphic format. Abstracting worlds from the world hierarchy with cause properties which are not Natural.progression and processing the worlds seq.no properties enables the student's history to be found.

Using worlds in the representation of the simulation model enables the status of the anatomy and physiology models to be saved at points in time when events of significance occur, for example state transition and student requests. The models represent a patient's current anatomy and physiology status. By making changes to the models, in the context of worlds' multiple views of the models may be maintained. Parameters derived by the physiology model, for example heart rate and blood pressure values, are added to the world from which the physiology model was viewed at the time of their generation. World inheritance ensures that these parameter values are passed down the hierarchy so a history of each parameter is contained in each world. These histories are used to produce charts, for example blood pressure charts.

Updating of the simulation model is controlled by the process SIM.PROCESS. This process is activated after a time delay of at least the number held in the slot REAL.TIME.OF.SIM.HR of the unit TUTORIAL.MANAGER seconds. Global vari-

ables used by this process are **term** and **stop**. **Stop** is set to true if the tutorial is to be terminated. If a simulation run has ended, because the tutorial duration has been exceeded or the disease has been correctly managed, *term* is set to true. A tutorial summary is then generated and the student is given the option to reset the simulation. If a reset is required *term* is reset to NIL, otherwise *stop* is set to true.

The simulation model is updated by processing the conditional and unconditional simulation events, held in the slots UNCONDITIONAL.EVENT.LIST and CONDITIONAL.EVENT.LIST of the unit SIMULATION.GENERATOR.EVENTS. All events held in the unconditional event list are performed on each update. These events are:

- Update simulation clock by incrementing *sim.time*.
- Update hospital time and date, stored in the slots CLOCK in the unit SIMULATION.GENERATOR.EVENTS and DATE in the unit PATIENT.PROFILE respectively. To do this add one to the value of hospital time, and take the modulus 24. An active value watching of the CLOCK slot will modify the date if the hospital time takes the value one. Hospital time has associated an image panel with active images attached to the CLOCK and DATE slots.
- Update physiology model by calling each method described in the SUBSYSTEMS slot of each physiology unit.

The conditional events have the format:

((condition) action ··· action)

where the condition and actions are Lisp forms. When processing the conditional event list, each condition is evaluated and if true, each action is evaluated. Events in the conditional event list are:

- If any sign and symptom complexes are active, ie. the ACTIVE.COMPLEXES slot of the unit SIMULATION.GENERATOR.EVENTS is not empty, process them.
- If the blood pressure, heart rate and temperature requires recording for charting purposes ie. the sum of the slot value FREQ.DATA.REC and first element in the list of the slot TIMES.DATA.RECORDED of the unit CHART.MANAGER is equal to *sim.time*, add *sim.time* to the list of times in TIMES.DATA.-RECORDED. The charts generated by the system are detailed in chapter 7.
- If disease state transition occurs, update the simulation model. Disease state transition takes place when *sim.time* is equal to *hr.dis.state.update*. The

```
While (NOT *stop*) do
  if (NOT *term*)
  then  block process for value of slot REAL.TIME.OF.SIM.HR in the
        unit TUTORIAL.MANAGER seconds;
        (WITH.MONITOR *sim.lock* evaluate UNCONDITIONAL.EVENT.LIST;
              evaluate CONDITIONAL.EVENT.LIST)
  fi;
  if (*term*)
  then if (zoom.ahead.process is in process list)
        then delete zoom.ahead.process;
                  create tutorial summary;
        fi;
        if (reset simulation=T)
        then *term* = NIL;
              (WITH.MONITOR *sim.lock* select world from
                    which to reset the simulation;
                    setup simulation reset)
        else *stop*=T
        fi;
Endwhile;
remove the simulation model from the system
```

Figure 6.6: The Control Procedure Underlying The Simulation Generator Process

method SNAP.UPDATE.MANAGER in the unit PS.ACTIVATOR will update the simulation model. This is described in the next section.

- If the tutorial duration time has been reached, ie. *sim.time* is equal to *tutorial.duration*, the time.world.end property of the current world, *world*, is set to *sim.time*. The method HALT.PATIENT.SIMULATION to stop the generation of the simulation model.

The algorithm used by SIM.PROCESS is shown in figure 6.6.

6.4.2 Algorithms Used In The Generation Of The Simulation Model

The initialisation of the patient's disease, management of disease state transition and reset of the simulation environment on the termination of a tutorial is carried out by methods in the object PS.ACTIVATOR. These algorithms underlying these methods

are described:

SIM.SETUP.MANAGER

Initialise KEE worlds;
Set *sim.time* to 0;
Set *offset* to 0;
(Setup initial disease state and root world* *world* *)*
Call method PATHOLOGICAL.SETUP;
Put *world* in slot ROOT.WORLD.NAME in unit SIMULATION.GENERATOR.EVENTS;
Set *hr.dis.state.update* to the next simulated hour disease state transition occurs;
Add properties SEQ.NO, START.TIME, CAUSE, SPEC.CAUSE and PATH.STATE to
the world *world*;
(Create PATIENT.PROFILE unit *)*
Call method PAT.PROFILE.SETUP;
Call method PHYSIOLOGY.SETUP;
Make PATIENT.PROFILE a superclass of the unit PATIENT.STATUS;
Setup hospital time clock;
(Assume blood pressure, temperature and heart rate data is recorded *)*
(at simulated hour 0 *)*
Put value 0 in list in slot TIMES.DATA.RECORDED in unit BP.PULSE.TEMP.DISPLAY;
Set *tutorial.duration* to the value in the slot TUTORIAL.DURATION in the unit
TUTORIAL.MANAGER;
Set *stop* to NIL;
Set *term* to NIL;
Create monitor *sim.lock*;
Create process SIM.PROCESS and add to process list

PATHOLOGICAL.SETUP

(Select problem for simulation *)*
Get value from slot DISEASE CLASS of unit TUTORIAL.MANAGER;
Get value from slot PROBLEM.COMPLEXITY of unit TUTORIAL.MANAGER;
if problem complexity involves chronic disease **then**
Randomly select a disease from the slot CHRONIC.STATES in the unit
represented by DISEASE.CLASS if chronic states defined
fi;
if multiple acute states are described by problem complexity **then**
  Randomly select a disease from the slot POSSIBLE.CONCURRENT.ACUTE.-
  STATES in the unit DISEASE.CLASS if available, otherwise select a disease
  from the slot ACUTE.STATES
else select a disease at random from the slot ACUTE.STATES of the
  unit represented by DISEASE.CLASS
fi;

(Create root world and set global variable* *world* *representing the current world to it *)*
Set *world* to *sim.time*-hrs-[chronic.state;]acute.state(s);
Create world *world*;
(Setup chronic state and store in the unit SIMULATION.GENERATOR.EVENTS *)*
Put chronic state definition in the STATIC.DIS.STATES slot of the unit
SIMULATION.GENERATOR.EVENTS in the world *world*;
(Setup snaps in the problem hierarchy and initialise the global variable* *path.state* *which *)*
(represents the current snap *)*
Set *path.state* to SNAP0;
Create unit SNAP0, a member unit of PATIENT.STATUS;
(Generate cause and site for acute states, store as disease state specifications in the slot *)*
(DYNAMIC.DIS.STATES in the unit SIMULATION.GENERATOR.EVENTS and *)*
(setup anatomy in the context of the root world *)*
for each acute state **do**
  Using CAUSES.WITH.SITES slot of the disease state select a cause and site at random;
  Put cause at anatomical site in the context of the world *world*;
  Generate the successor disease state with the same physical cause, possibly at a
  different site using the SUCCESSOR.DIS.STATE slot of the acute state;
  (Create the disease specification and add to the DYNAMIC.- *)*
  (DIS.STATES slot of the unit SIMULATION.GENERATOR.EVENTS *)*
  Set disease specification to
  (duration (current disease state defn.)(next disease state defn.));
  Add disease specification to the slot DYNAMIC.DIS.STATES of the unit
  SIMULATION.GENERATOR.EVENTS in world *world*
od;
(Generate shock type and put in the SHOCK.TYPE slot of the *)*
(unit CIRCULATORY.SYSTEM *)*
if (*path.state* does not have a SHOCK.TYPE value which is a
  member of the CLASS SEPTIC) **then**
  Put HYPOVOLAEMIC in the slot SHOCK.TYPE of *path.state*
else Put the most advance septic shock state in the SHOCK.TYPE slot of
  path.state
fi;
Put value of the slot SHOCK.TYPE in the unit *path.state* into the SHOCK.TYPE
slot of the unit CIRCULATORY.SYSTEM

PATIENT.PROFILE.SETUP

Generate sex at random and name;
Generate age at random in the range 16 to 64;
Generate weight;
Generate hour, day, month and year hospital time begins.
Store sex, name, age, weight, day.month.yr and hr disease start in the slots

SEX, NAME, AGE, WEIGHT, DAY.MTH.YR and HR.DIS.START of the unit PATIENT.PROFILE

PHYSIOLOGY.SETUP

Message the elements of the INIT.SUBSYSTEMS slots of the units EXTRACELLULAR.-FLUID, CIRCULATORY.SYSTEM, KIDNEY, SKIN.LUNGS, INTESTINE and THIRD.-SPACE

SNAP.UPDATE.MANAGER

(Update pathology, assign* *hr.dis.state.update* *the value of the most imminent *)*
(state transition and adjust the physiology to match the new disease states *)*
Call the method UPDATE.PATHOLOGY in the unit PS.ACTIVATOR;
Redefine *hr.dis.state.update*;
Call the method UPDATE.PHYSIOLOGY in the unit PS.ACTIVATOR

UPDATE.PATHOLOGY

(Get active disease states and find acute states which require updating *)*
Set chronic states to the value of the STATIC.DIS.STATES slot of the unit SIMULATION.GENERATOR.EVENTS in the world *world*;
Set acute states to the value of the DYNAMIC.DIS.STATES slot of the unit SIMULATION.GENERATOR.EVENTS in the world *world*;
Set acute states no progression to the acute states whose end times (from the disease state specification) are not equal to *hr.dis.state.update*;
Set progression states to the difference of the acute states and the states which have resolved or have not progressed;
Remove the times from the disease specifications of the progression states which then are of format ((current dis. state triple) (next dis.state triple));
(Derive textual description of the progressions being modelled *)*
if (NULL progression.states) **then**
    SPEC.PROG is acute state → resolution
else SPEC.PROG is [name of current dis.
    state → next dis. state;] name of current dis. state → next dis. state;
(Set old snap name to current snap and generate new snap name *)*
Set old snap name to *path.state*;
Set new snap name to SNAPn where n is one greater than n in *path.state*;
(Determine whether all disease states have resolved (no chronic states, acute states *)*
(with no progression or progression states) or whether states are still active *)*
if ((NULL progression states) AND (NULL acute states no progression) AND
    (NULL chronic states)) **then**
    (All diseases have resolved so remove state definitions from DYNAMIC.DIS.STATES *)*
    (slot, put END.TIME property in* *world* *and halt patient simulation *)*

Remove values of form (*sim.time* (Resolution NIL NIL) NIL)) from the slot DYNAMIC.-
DIS.STATES of the unit SIMULATION.GENERATOR.EVENTS;
Put *sim.time* in END.TIME world property of world *world*;
Call method HALT.PATIENT.SIMULATOR in the unit TUTORIAL.MANAGER with
the argument 'Patient's disease has resolved'
else

(Chronic and/or acute states active. Setup next state transition for disease states *)*
(which have progressed. Create a new snap and world. Update the anatomical model *)*
(and DYNAMIC.DIS.STATES slot of the unit PATIENT.SIMULATION.EVENTS *)*
Set *path.state* to new snap name;
Create unit *path.state*. a member of the unit PATIENT.STATUS;
Create child world of *world* with name *sim.time*hrs-spec.prog;
Assign world properties where CAUSE is Natural.Progression, SPEC.CAUSE is
spec.prog, PATH.STATE is *path.state*, SEQ.NO is dependent on *world* and TIME.-
WORLD.START is *sim.time* plus one;
Put property END.TIME with value *sim.time* in world *world*;
Set *world* to child world of *world*;
Setup acute disease states which do not progress and chronic disease states by creating
links from disease states to *path.state*;
(Anatomical knowledge associated with disease states which do not progress is *)*
(passed to* *world* *via world inheritance. Deal with disease states which have progressed *)*
for all states which have progressed **do**
Remove disease state specification from DYNAMIC.DIS.STATES slot of the unit
SIMULATION.GENERATOR.EVENTS;
od;
if (NOT (NULL progression states)) **then**
(Acute states have progressed. If a state progresses to death, the anatomy *)*
(associated with the state does not change *)*
Update anatomical model for disease states which do not progress
to death by extracting the sites from the current and successor
disease state triples.
if (the current and successor sites are the same) **then**
skip
else remove the anatomical abnormalities from the current state if no other states
are associated with it;
Add an abnormality to the new site
fi;
for all progression states **do**
(Generate the next state definition by switching the next state triple to the *)*
(current state triple, generating the duration of the state described by this triple *)*
(and deriving a new successor state triple. The successor state triple has the *)*
(same physical cause as the current state triple but a possible different site *)*
if (the new disease state to be setup is RESOLUTION) **then**

Put the value SHOCK.TYPE.ON.RESOLUTION in the slot SHOCK.TYPE
of the unit *path.state*;
Set successor state definition to NIL
else generate successor state definition using the slot SUCCESSOR.DIS.STATES
in the disease state described by the current disease state triple
fi;
Generate the time the current disease state ends using the DURATION
slot of the DISEASE.STATE frame;
Add the next acute disease state specification to the DYNAMIC.DIS-
STATES slot of the unit SIMULATION.GENERATOR.EVENTS
od
else *(* Acute states have resolved but either or both chronic states and acute states *)*
(which have not progressed remain.*)*
fi;
(Determine SHOCK.TYPE slot value of* *path.state* *)*
if (the SHOCK.TYPE slot of *path.state* does not contain a septic state)
then Set the SHOCK.TYPE slot of *path.state* to HYPOVOLAEMIC
else Set the SHOCK.TYPE slot of *path.state* to the most advanced septic shock state
fi

UPDATE.PHYSIOLOGY

if (the SHOCK.TYPE slot of *path.state* is not equal to that described by
the physiology model) then
Put value in the slot SHOCK.TYPE of the unit *path.state* in the SHOCK.TYPE
slot of the unit CIRCULATORY.SYSTEM;
Call the method INIT.PR.CHANGE in the unit described by the SHOCK.TYPE value
fi

RESET.SIMULATION

Delete active processes associated with the simulation generation;
Close all windows and viewports associated with the tutorial;
Delete the unit PATIENT.PROFILE;
Delete all 'snap' units;
Initialise KEE worlds

6.4.3 Resetting The Simulation

The patient scenario may be reset from points in the simulation model corresponding to the times worlds end. On requesting to reset the simulation, the type of world from which to recommence the simulation run is requested. World types are MANAGEMENT, NATURAL.PROGRESSION and SIM.RESET. The properties of all

the worlds in the world hierarchy are then processed and a list of worlds found which have a cause property equal to the world type from which to reset the simulation. The list is then presented to the student in the form of a menu, from which a menu item is selected.

For example, consider the world hierarchy shown in figure 6.7a. If the student requested a simulation reset from a world of type MANAGEMENT, the following list of worlds would be generated:
(1hrs-LABORATORY.INVESTIGATION.SERUM.BILIRUBIN 4hrs-HISTORY.PAIN 4hrs-EXAMINATION.ABDOMINAL 8hrs-TREATMENT.LONGMEYER'S 15hrs-SPECIAL.INVESTIGATION.ISOTOPIC.SCAN)

On selection of a world, a simulation reset is performed by creating a child world of the world; and resetting the simulation clock to the time the parent world ended, the problem state *path.state* to the snap associated with the parent world, the tutorial duration *tutorial.duration*, the current world, *world* and the time of disease state transition, *hr.dis.state.update*. The algorithm used by the reset procedure follows:

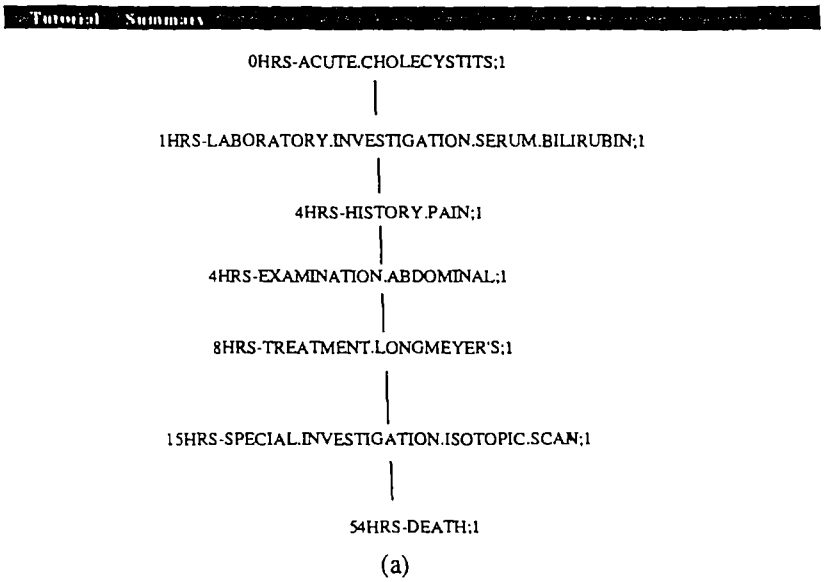
Set **sim.time** to the value of the property TIME.WORLD.END of the parent world;
Set **path.state** to the value of the property PATH.STATE of the parent
world;
Set **tutorial.duration** to the sum of **sim.time** and the value of the slot TUTORIAL.-DURATION.SIM.HRS in the unit TUTORIAL.MANAGER;
Setup **hr.dis.state.update** by scanning the DYNAMIC.DIS.STATES slot of the SIMULATION.GENERATOR.EVENTS slot;
Set child world name to be **sim.time**hrs-SIMULATION.RESET;extn.no;
Create child world;
Add properties CAUSE = SIM.RESET,SEQ.NO, TIME.WORLD.START = *sim.time*,
PATH.STATE = *path.state*,
SPEC.CAUSE = To explore different patient management strategies;
Set **world** to child world.

Figure 6.7b show the changed world hierarchy following reset from the world 4hrs-HISTORY.PAIN;1.

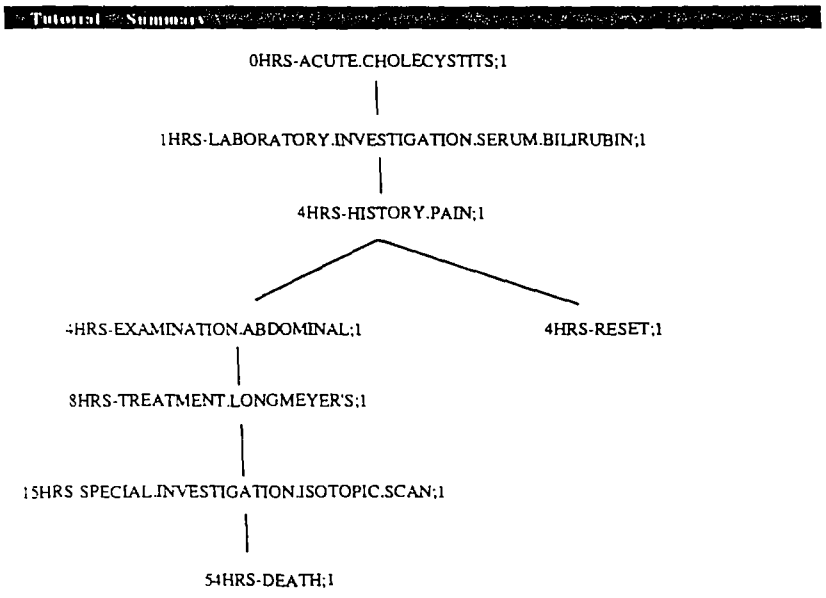
Having reset a simulation, the physiological parameter values will be passed from the parent to be new world by inheritance, as will the state of the physiological and anatomical models. Simulation can then proceed as before.

6.4.4 Tutorial Summary

The tutorial summary is created by processing the properties of worlds in the world hierarchy to generate a textual description of the tutorial history. Formatting of the output reflects the sequence with which critical events occured.



(a)



(b)

Figure 6.7: Changes on the world hierarchy imposed by simulation reset - before (a) and after (b) reset.

Firstly, the world transition paths are found and ordered according to their relative time of creation in a tutorial. To do this the world hierarchy is processed in a 'top-down', 'depth-first' manner, the SEQ.NO properties of worlds being used to sequence the hierarchy branches. For example, given two sibling worlds, with SEQ.NO properties 1.1 and 1.2, the branch of the hierarchy starting with the node whose SEQ.NO is 1.1 is processed first. The world hierarchy is represented as a list, branches of the hierarchy being described as sublists. The LISP functions for transforming the hierarchy into a list are given below:

```
(DEFINEQ (GET.LOWER.WORLDS (WORLD)
      (* The function ORDER.BY.SEQ.NO orders the list of sibling
      worlds supplied as argument according to their SEQ.NO
      properties. The KEE functions GET.WORLD.CHILDREN and
      GET.WORLD.NAME return the immediate child worlds and the
      world name of the world reference supplied as an argument.)

(LET ((CHILD.WORLDS (ORDER.BY.SEQ.NO (GET.WORLD.CHILDREN WORLD))))
(COND ((NULL CHILD.WORLDS) (LIST (GET.WORLD.NAME WORLD)))
      ((EQUAL 1 (LENGTH CHILD.WORLDS))(CONS (GET.WORLD.NAME WORLD)
                        (GET.LOWER.WORLDS (CAR CHILD.WORLDS))))
      (T (LIST (GET.WORLD.NAME WORLD)(GET.LOWER.WORLDS (CAR CHILD.WORLDS))
          (PROCESS.LOWER.WORLDS (CDR CHILD.WORLDS)))))))

(DEFINEQ (PROCESS.LOWER.WORLDS (WORLDS)
(COND ((EQUAL (LENGTH WORLDS) 1) (GET.LOWER.WORLDS (CAR WORLDS)))
      (T (GET.LOWER.WORLDS(CAR WORLDS))
        (PROCESS.LOWER.WORLDS (CDR WORLDS))))))
```

For example, the world hierarchy shown in figure 6.8 would be transformed into the list:

(A B (C D) (E (F G) (H I J)))

by calling the function GET.LOWER.WORLDS with argument A, the root world.

The tutorial summary is formatted such that the description of every critical event in a scenario is indented by a fixed amount, outdenting being used to highlight alternative events arising from tutorial reset. Figure 6.9 illustrates the outline of a typical tutorial summary, that arising from the world hierarchy shown in figure 6.8. The Lisp functions to format the output are given below:

```
(DEFINEQ (PRINT.LOWER.WORLDS (WORLDS POSN)
(COND ((NULL WORLDS) NIL)
      ((ATOM WORLDS)
       (PRINTOUT T T .TAB POSN (WORLD.SUMMARY.INFO WORLDS)))
      ((NOT (LISTP (CAR WORLDS)))
```

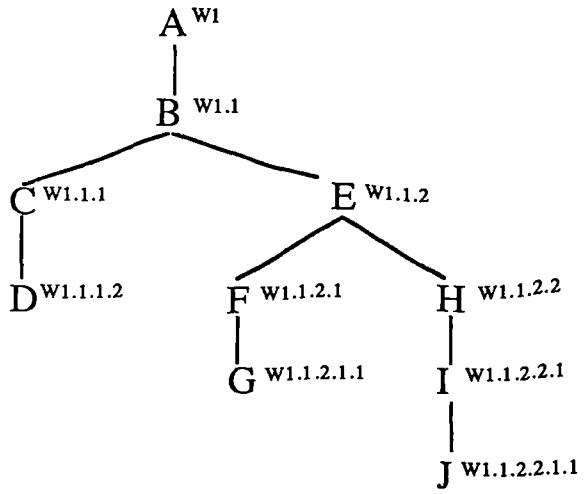


Figure 6.8: World Hierarchy Illustrating SEQ.NO World Properties

```
(PRINTOUT T T .TAB POSN (WORLD.SUMMARY.INFO (CAR WORLDS))
        (COND ((NULL (CDR WORLDS)) (TERPRI))
              (T (PRINT.WORLD.TIME (CAR WORLDS) POSN 'END.TIME)
              (PRINT.LOWER.WORLDS (CDR WORLDS (IPLUS POSN 3))))))
     ((EQUAL 1 (LENGTH (CAR WORLDS)))
      (PRINTOUT T T .TAB POSN (WORLD.SUMMARY.INFO (CAAR WORLDS)))
      (PRINT.LOWER.WORLDS (CDR WORLDS) POSN))
     (T (PRINT.LOWER.WORLDS (CAAR WORLDS) POSN)
        (PRINTOUT T (PRINT.WORLD.TIME (COND ((LISTP (CAAR WORLDS))
                                             (LAST (FLATTEN (CAAR WORLDS)))
                                             (T (CAAR WORLDS)))
POSN 'END.TIME))
        (PROCESS.SUBLIST.WORLDS (CDAR WORLDS) (IPLUS POSN 3))
        (PRINT.LOWER.WORLDS (CDR WORLDS) POSN)))))

(DEFINEQ (PROCESS.SUBLIST.WORLDS (WORLDS POSN)
(COND ((EQUAL 1 (LENGTH WORLDS))(PRINT.LOWER.WORLDS WORLDS POSN))
      (T (PRINT.LOWER.WORLDS (CAR WORLDS) POSN)
        (PROCESS.SUBLIST.WORLDS (CDR WORLDS) POSN)))))
```

where the function WORLD.SUMMARY.INFO returns the CAUSE and SPEC.CAUSE properties, separated by a '-' character, of its world argument; (PRINT.WORLD.TIME *world posn point*) prints the start or end time (represented by its point argument) in hospital and simulated time of its world argument on a new line, at horizontal position posn + 3; and PRINTOUT is a KEE printing function [2]. To invoke the output the function PRINT.LOWER.WORLDS is called with the argument (GET.LOWER.-WORLDS *root world of the hierarchy*).

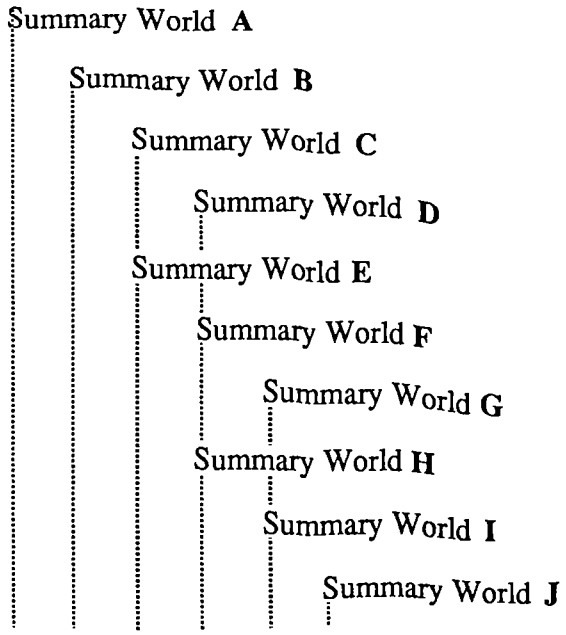


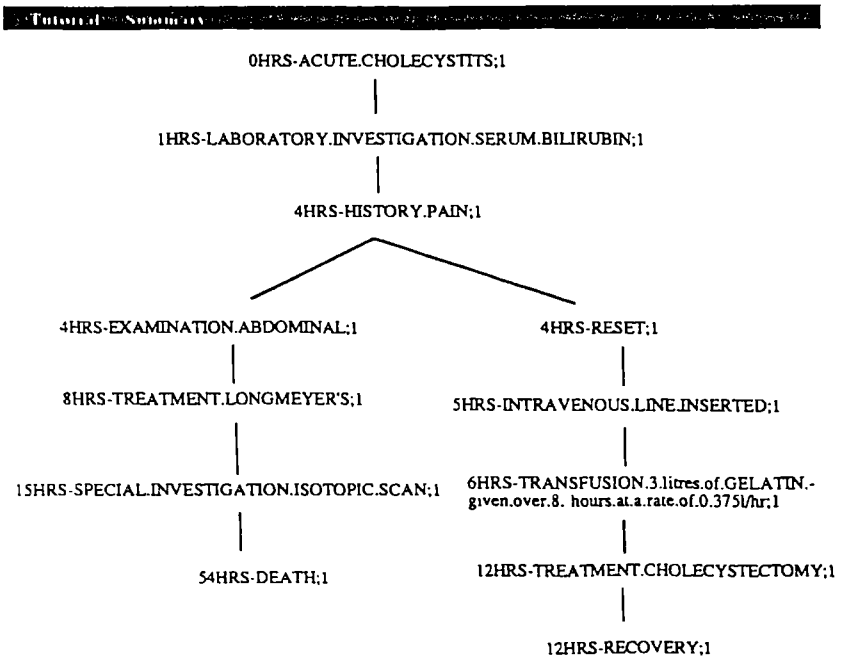
Figure 6.9: Outline Of Tutorial Summary

For each critical event, the time the event occured (in simulated and hospital time), the event type and a description of it is reported. This information is found using the world properties TIME.WORLD.START, CAUSE and SPEC.CAUSE. Simulated times are converted to hospital time by adding **sim.time** to the hour the simulated began in hospital time (held in the slot HR.DIS.START of the unit PATIENT.PROFILE), taking the modulus 24 to establish the hour, and adding the integer division 24 to the date the simulation began in hospital time (held in the slot DAY.MTH.YR of the unit PATIENT.PROFILE) to determine the date.

To illustrate the tutorial summary resulting from a tutorial which has undergone simulation reset consider figure 6.10. Figure 6.10a shows the world hierarchy, as shown by the world browser (a KEE facility used to view all worlds), resulting from a tutorial in which a reset has been performed. The results of applying two different management strategies are shown, one resulting in patient recovery and the other death. Assuming that the hospital time on simulation initialisation is 14hrs on the 1-2-88, the TIME.WORLD.START, SEQ.NO, CAUSE and SPEC.CAUSE world properties being illustrated in the figure, the resulting tutorial summary is shown in figure 6.10b. The world browser provides a graphic description of the summary.

6.5 Conclusion

The simulation generator and tutorial manager are controlled by different processes and are defined by separate modules. Each module is represented by a set



(a)

14hrs / 1/02/88: Simulated hr 0

Natural Progression - Acute Cholecystitis

15-16hrs / 1/02/88: Simulated hr 1

Management - Laboratory Investigation Serum Bilirubin

18-19hrs / 1/02/88: Simulated hr 4

Management - History Pain

Management- Examination Abdominal

22-23hrs / 1/02/88: Simulated hr 8

Management - Treatment Longmeyer's

5hrs / 2/02/88: Simulated hr 15

Management - Special Investigation Isotopic Scan

20-21hrs / 3/02/88: Simulated hr 54

Natural Progression - Death

Simulation Reset - To explore different patient management strategies

19-20 hrs / 1/02/88: Simulated hr 5

Management - Intravenous line inserted

20-21 hrs / 1/02/88: Simulated hr 6

Management - Transfusion 3 litres of GELATIN given over 8 hours at a rate of 0.375l/hr

2-3 hrs / 2/02/88: Simulated hr 12

Treatment - Cholecystectomy

Natural Progression - Recovery

(b)

Figure 6.10: Example of a tutorial summary. In (a) the world hierarchy is shown, the corresponding summary is demonstrated in (b)

of objects. A semaphore or monitor ensures that both processes do not update the simulation model simultaneously.

The simulation model is non-deterministic and generated dynamically, a combined 'fixed-increment time advance' event scheduling, activity scanning algorithm being used by the simulation generator to control the update. The model is described by a unit hierarchy in the background, and world hierarchy. Aspects of the problem being simulated which are invariant over intervals of simulated time, for example disease states, patient profile, are represented in the background as nodes in a hierarchy. For each point of simulated time, one node in the hierarchy provides the current focus of background knowledge. Attributes of the problem which change on each update and are dependent on the background knowledge, for example the physiology, are held in worlds. Each world corresponds to one node in the background problem hierarchy, transition from one problem state to another being accompanied by the creation of a child world in the world hierarchy. Child worlds are also created when the student requests actions. World properties are used to define the reason for the creation of each world.

By processing the world hierarchy, the student's patient management pathway can be found, and a new management strategy effected. Alteration of management strategy invariably results in progression through different disease states. An additional facility, not implemented could allow the student to determine the disease state and progression to be generated. This facility would require to reset the simulation from the time a disease state transition took place, prompting the student for the next disease state defined by the diseases SUCCESSOR.DIS.STATE slot.

The benefits of using a knowledge-based approach for the simulation are that declarative knowledge held in slots in the background or worlds is accessible to the program or user. This assisted in separating the different program modules, since one module could access anothers knowledge held in slots, rather than declare global variables or rely on the passing of knowledge via the arguments of methods. The knowledge was accessible to the student, via active images and the world browser. These facilities, together with the ability to assess the knowledge base via queries from the Lisp listener and various display options, are provided by the development environment and were also of benefit to the knowledge engineer when building the system. Knowledge is understandable since it is organised about the concepts with which it is associated. For example, the tutorial specification knowledge is held in the tutorial manager object.

Chapter 7

User Interface And Student Actions

7.1 Introduction

The user interface module is responsible for the presentation of the tutorial to the student and the recognition of student requests. A tutorial comprises five stages (figure 7.1). The selection of the clinical problem for generation and the tutorial summary and reset option stages have been described in chapter 6 in the context of the tutorial manager. This chapter focuses on the remaining stages in a tutorial, when the student requests actions and receives feedback about them.

The selection of student actions is not constrained to a plan predetermined by the system but is decided by the student. Categories of available actions are displayed in a panel, a menu being associated with each specifying instances of the action type. Certain menu items have associated submenus to further the selection of choices. The arrangement of the menu items serves to reinforce the student's factual knowledge.

The actions are of two types; those which enable the student to obtain further information about the patient's problem and others which may change the patient's condition (figure 7.2). Each action has associated a control procedure. Before accessing the simulation model the control procedure obtains the monitor, *sim.lock*, to ensure the simulation model is not updated unexpectedly whilst a student request is being processed. For the first type of action (type A), the procedure then queries the simulation model for knowledge which is directly presented to the student or is used to infer other knowledge which is passed to the student. The control procedure of the second type of action (type B), may change the evolution of the simulation model by interfacing with the simulation generator to modify the domain concepts associated with the patient's problem. These concepts may be changed at an instance or over a period of simulated time. Each control procedure also creates a new world in the simulation model. Since

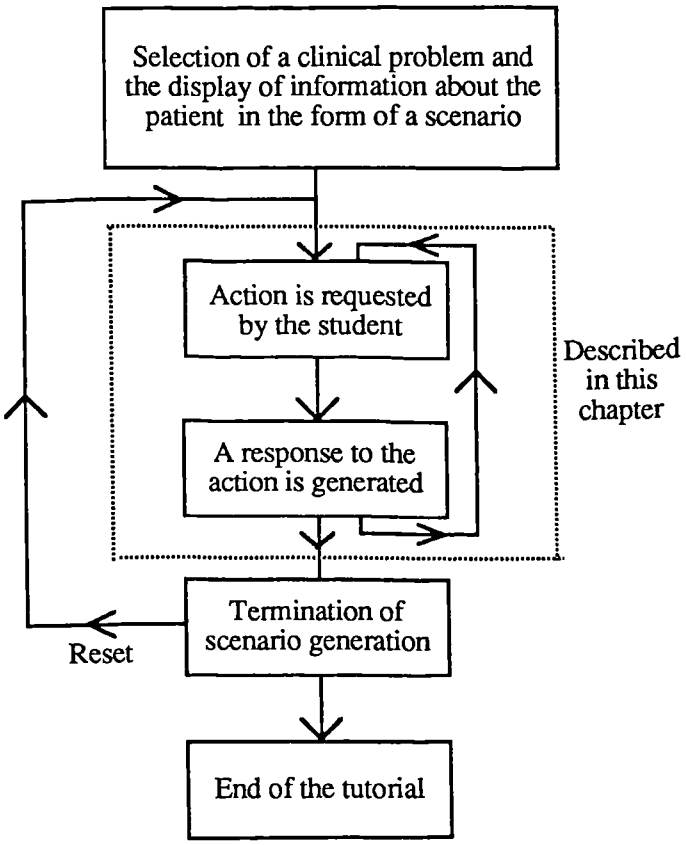


Figure 7.1: Tutorial Stages

the simulation may be reset from points in a scenario where new worlds were created, the creation of new worlds by the control procedures allows for the tutorial to be reset from points when student actions were carried out.

The user interfaces features are described. These include help information, the display of the patient's status and hospital time and a facility to manage the student's requests for actions and deliver feedback associated with the actions. The student actions supported by the system are constrained by the representation of the simulation model. The characteristics of the model which facilitate the action types provided by the system are discussed, together with the representation formalisms used in the implementation of their control procedures. Specific actions are then described. These actions reside in the systems tutorial manager module.

7.2 User Interface

The user interface consists of three screens; start up, tutorial and summary. The startup and summary screens have been described in chapter 6. The screen layout during a tutorial is shown in figure 7.3. The student actions available are defined by a KEE picture viewport. Mousing on any icon in the viewport activates a menu from which the student can chose an item. In the top left hand side of the screen a help

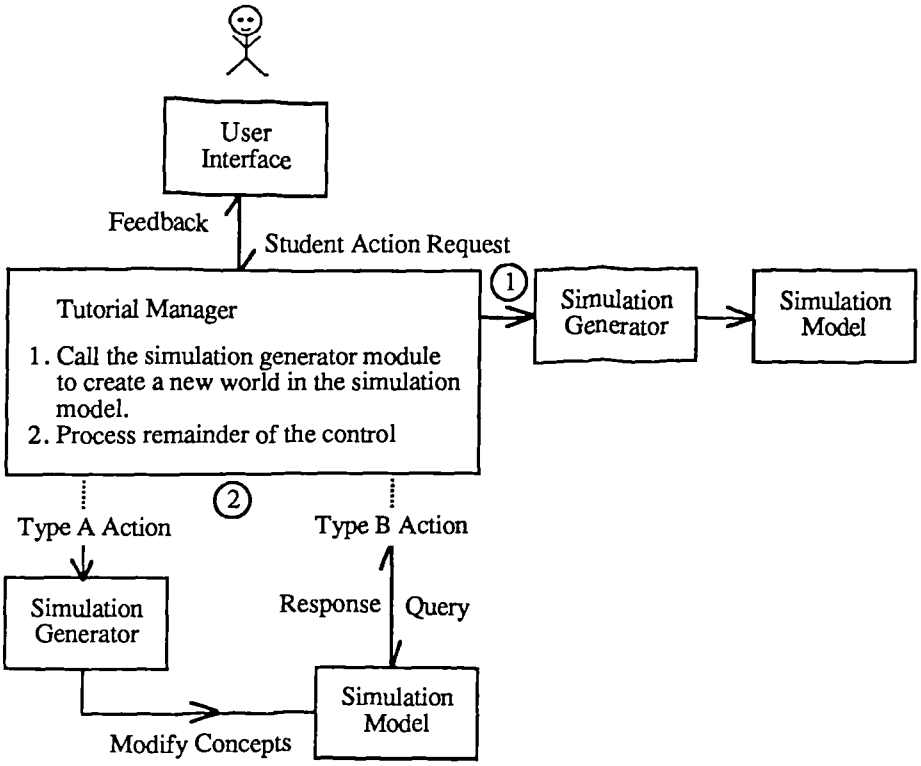


Figure 7.2: Processing Of Student Actions

window resides, in which textual descriptions of the items in the student action menus is displayed, together with prompts for data to be input. The clock windowpane describes the simulated hospital date and time. The windowpane comprises images attached to the REAL.TIME and DATE slots of the unit SIM.GEN.EVENTS. At the bottom left hand of the screen is the patient condition window. Information about the patient's status is displayed in this window. The top and middle right hand areas of the screen are reserved for the display of diagrams, represented by KEE Pictures. Finally, active images to abort and restart the tutorial are positioned at the bottom right hand side of the screen.

7.3 Student Actions

Student actions are classified according to their function; type A actions provide the student with more information about the simulation model and type B modify the model.

The knowledge contained within the simulation model may satisfy the student request or may be used in conjunction with other knowledge to infer information to service the request. If information of the patient's physiological or current clinical status is required the current world and its associated snap (unit in the patient status

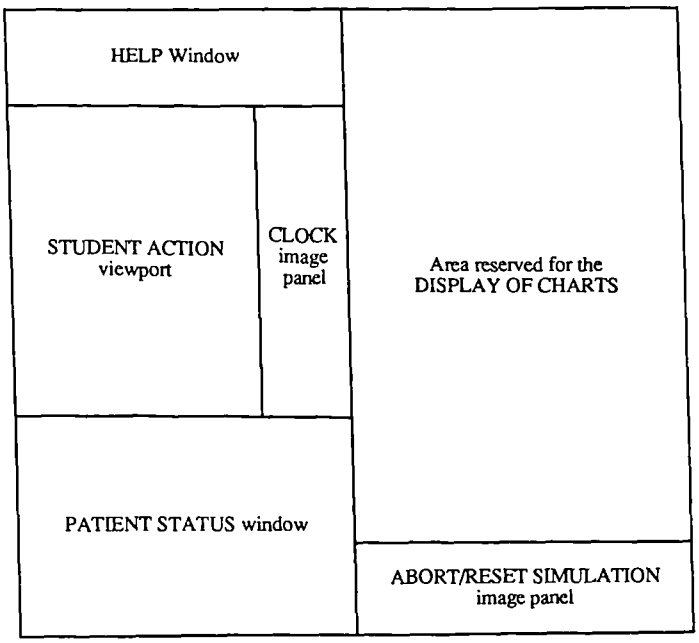


Figure 7.3: Screen Management During The Stages Of A Tutorial When The Student Can Request Actions

hierarchy) is used, otherwise a search of the world and patient status hierarchies is performed to obtain knowledge of the patient's clinical history.

Knowledge of clinical symptoms and signs is held in the patient status unit hierarchy, in the background. Each terminal node or snap describes the patient's condition over an interval of simulated time. The patient attributes represented by slots in the snaps are either:

- normally absent, for example vomiting, or
- normally present but take abnormal values, for example yellow conjunctivae.

It is assumed that a patient's clinical attributes which are not represented by a snap take normal values.

When querying the simulation model for the status of a patient attribute, say $attribute_x$, which is normally absent and is not represented by the current snap, '$attribute_x$ not present' is returned, otherwise the value of the slot representing the attribute is returned. Patient attributes which are normally present may be described by slots or may not. Normal values of attributes represented by slots are held in the background, in worlds or are generated by methods contained in the slots. Attributes whose normal values are held in the background may become abnormal by assigning the attribute an abnormal value in the context of a world. The KEE inheritance mechanism FROM.ANY.CONTEXT ensures that a slots background value is not present in

Patient Attributes	Representation Details
Clinical	
Norm Present-Abnormal Value	Described by a slot whose value is held in a snap in the background
Norm Absent-Present	As above
Norm Present-Normal Value	Not represented
Norm Absent-Absent	As above
Physiological	
Norm Present-Abnormal Value	Described by a slot in the physiological model containing a method or a value which is world specific
Norm Absent-Present	Described by a slot in the physiological model whose value is world specific
Norm Present-Normal Value	Parameters associated with the blood pressure, heart rate and fluid volume are described by slots in the physiological model whose values are world specific. Values of parameters associated with fluid composition are held in slots in the background or are described by methods
Norm Absent-Absent	Not represented

Table 7.1: Summary Of How Patient Attributes Are Represented In The Simulation Model

a world if a world specific value has been defined [2]. Attributes which are represented by slots whether they take normal or abnormal values are physiological parameters, for example serum bilirubin and white cell count. All clinical attributes which are normally present are only represented when they take abnormal values and are described by snaps in the background. Table 7.1 shows when patient attributes are described in the simulation model.

Many of the student action procedures which query the simulation model for information of the patient's clinical status use the LISP function ATTRIBUTES.-PRINTER (figure 7.4). This function takes as argument a list with sublist elements comprising the attributes to be queried preceeded by their default value. For each attribute, the function searches the current snap in the patient status hierarchy, for a slot representing the attribute. If such a slot exists, the attribute name and slot value is returned, otherwise the attribute name and default value is returned. Specifying the default value of a patient attribute as normal assumes that the student knows how to qualify the value normal. More descriptive feedback is achieved by defining a more

```
(ATTRIBUTES.PRINTER
(LAMBDA (L)
(* For each sublist of L, the CAR of the list represents the default
   value and the CDR the attributes whose values are sought. The
   current snap described by *PATH.STATE* is scanned for a slot
   representing the attribute. If a slot exists its value is printed,
   otherwise the default value is printed. )

(MAPC L '(LAMBDA (X)
        (LET* ((DEFAULT.VALUE (CAR X)))
            (MAPC (CDR X)
            '(LAMBDA (ATTRIBUTE)
              (LET* ((VALUE (GET.VALUES *PATH.STATE* ATTRIBUTE))
                  (PRINTOUT T ATTRIBUTE :)
                  (COND ((NULL VALUE) (PRINTOUT T DEFAULT.VALUE))
                        (T (PRINTOUT T VALUE)))))))))
```

Figure 7.4: LISP Function Used To Query The Simulation Model

illustrative default value. For example, the default value white is more descriptive than a default value normal when used in the context of the attribute conjunctivae.

Some student actions, for example special investigations, require the values of multiple patient attributes in order to deduce appropriate feedback. Such actions are represented by SAME.WORLD.ACTION rules. The premise of each rule defines the values of attributes in the simulation model which are required to satisfy the response described by the rule conclusion.

The values of the physiological parameters blood pressure, heart rate and temperature are passed by world inheritance to the current world. They are thus easily accessible to the control procedures which are associated with student actions for plotting charts.

Type A actions may be processed as soon as they are requested or after a time delay, as occurs in reality. Actions which are delayed are added to the slot CONDITIONAL.EVENT.LIST in the unit SIMULATION.GENERATOR.EVENTS, the time of activation forming the condition on which the action is triggers.

Type B actions may change the evolution of the simulation model by modifying the models underlying concepts. For treatment actions, the active disease states may be changed possibly manifesting changes in the physiology. Intravenous fluid ad-

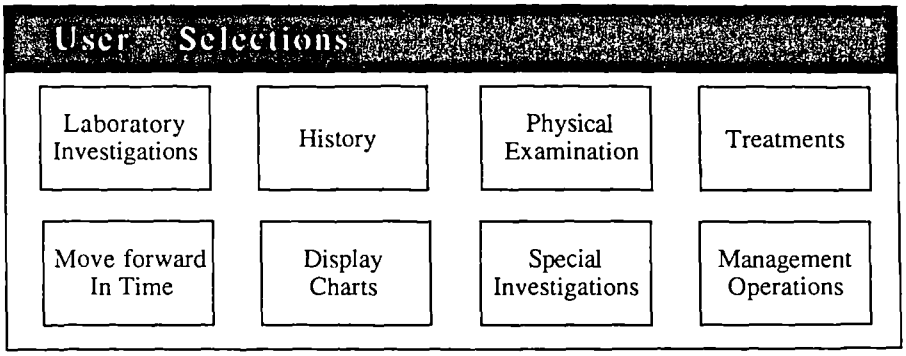


Figure 7.5: Viewport Describing Student Actions

ministration changes the processes associated with fluid balance thereby changing the physiology. Other type B actions do not directly change the patient's condition but interface with the simulation generator to modify the presentation of the tutorial, for example the action to advance the simulation clock. These modifications may however invoke disease state transition causing disease progression.

Categories of student action are held in a viewport, each category being represented by a box string KEE picture (figure 7.5). Mousing on a box string results in the display of a menu describing the actions available of the chosen type. Actions belonging to the same category often share the same control procedure. On the selection of a menu item a control procedure is invoked with arguments defined by the item. These arguments are of different types as follows:

(UNIT SLOT) The control procedure will query the patient attribute described by the slot SLOT in the unit UNIT.

(RULE CLASS) The control procedure will forward chain on the rule class described by the argument RULE CLASS.

(SUBTYPE) The control procedure will call the method SUBTYPE residing in the unit which represents the chosen box string or will use SUBTYPE in the course of its own execution.

The categories of type A and B student action which have been implemented and appear in the viewport shown in figure 7.5 are now described in detail:

7.3.1 Type A

Laboratory Investigations

The laboratory investigation options available are shown in figure 7.6. On the selection of an investigation the following control procedure is executed with arguments

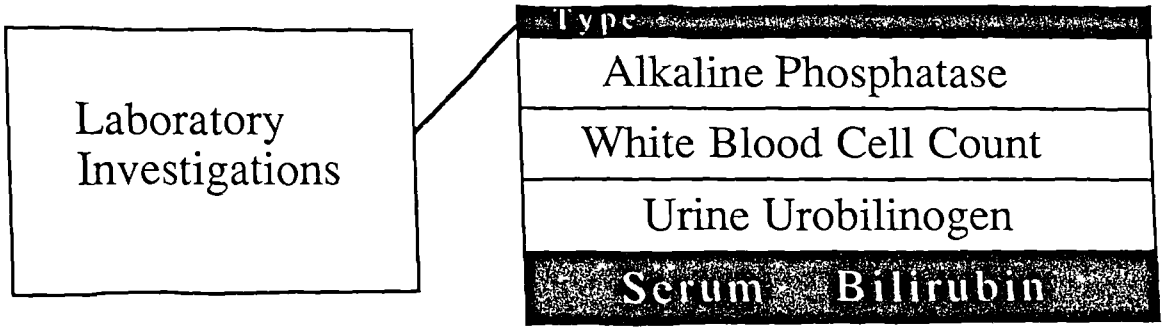


Figure 7.6: Menu Of Available Laboratory Investigations

(UNITname SLOTname) where (UNITname SLOTname) is:

Investigation	*UNITname*	*SLOTname*
Alkaline Phosphatase	INTRAVASCULAR.FLUID	CURRENT.ALKALINE.PHOSPHATASE
Serum Bilirubin	INTRAVASCULAR.FLUID	CURRENT.ALKALINE.PHOSPHATASE
White Cell Count	INTRAVASCULAR.FLUID	WHITE.BLOOD.CELLS
Urine Urobilinogen	KIDNEY	URINE.UROBILINOGEN

Control Procedure:

```
(WITH.MONITOR *SIM.LOCK*
  (SETQ UNITS (SLOT.FACET.VALUES (SLOTname UNITname) 'UNITS))
  (SETQ LAB.INV (REMOVE 'CURRENT (MAKE.LIST.PERIODS.SPACE SLOTname))
  (PRINTOUT T "Laboratory Investigation:", LAB.INVEST,
    "Current", LAB.INVEST,"value is",
  (COND ((EQUAL (SLOT.FACET.VALUES (SLOTname UNITname) 'VALUECLASS)
    'METHOD) (UNITMSG UNITname SLOTname))
    (T (GET.VALUE UNITname SLOTname 'OWN *WORLD*))), UNITS)
  (* Create a child world of the current world *world* with CAUSE and SPEC.CAUSE
  properties equal to MANAGEMENT and the selected investigation respectively.)
  (WORLD.CREATE *WORLD* 'MANAGEMENT
    (APPEND '(Laboratory Investigation:) LAB.INVEST )))
```

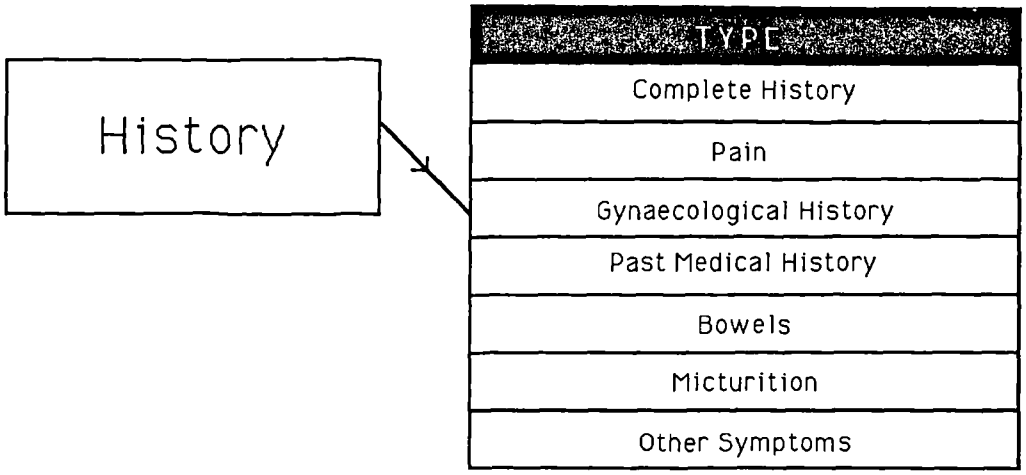


Figure 7.7: Menu Of History Options Available

History

The history options provided are shown in figure 7.7. On the selection of an option the following control procedure is executed with an argument describing the type of history requested,

Control Procedure:

```
(WITH.MONITOR *SIM.LOCK*
   (PRINTOUT T History:, HISTtype)
   (UNITMSG THISUNIT HISTtype)
   (* Create a child world of the current world *world* with
   CAUSE and SPEC.CAUSE properties equal to MANAGEMENT
   and the selected history respectively.)
   (WORLD.CREATE *WORLD* 'MANAGEMENT
   (APPEND '(History:) HISTtype)))
```

Each of the history option methods query the current snap using the function ATTRIBUTES.PRINTER (figure 7.4) except for the method describing the pain option which also searches the world and patient status hierarchies to infer the sites of pain. The arguments supplied by each method to the ATTRIBUTES.PRINTER function are:

Method	*Arguments*
BOWELS	((Normal STOOL.COLOUR) (Not Present CONSTIPATION DIARRHOEA BLOOD. IN.STOOLS MUCUS.IN.STOOLS))
GYNAECOLOGICAL.HISTORY	((Not Present CHANGE.IN.NORMAL.PERIOD.PATTERN PREGNANCY ABNORMAL.BLEEDING))
MICTURITION	((Normal URINE.COLOUR) (Not Present FREQUENCY DYSURIA PAIN.ON.PASSING.URINE))
OTHER.SYMPTOMS	((Not Present VOMITING NAUSEA ANOREXIA ITCHING LOSS.OF.WEIGHT) (Normal CONJUNCTIVAE.COLOUR))
PAST.MEDICAL.HISTORY	((Not Present PREVIOUS.SIMILAR.PAIN PREVIOUS.- MAJOR.SURGERY DRUGS.FOR.ABDOMINAL.PAIN PREVIOUS.INDIGESTION
PAIN	((Not Present RADIATION))

The PAIN method also prints information on the sites of pains of type ache and colic, and their relieving and aggravating factors. To do this the value of the slot, PAIN.CHARACTER, in the current snap is found and the pain site at onset and at present in each active disease state. The pain sites are defined in the disease state units in the disease taxonomy. If only one type of pain exists, the pain site at onset and at present is printed and the function ATTRIBUTES.PRINTER is called with the argument, ((Not Present AGGRAVATING.FACTORS RELIEVING.FACTORS PAIN.CHARACTER)). If multiple pain types are present, the pain sites associated with each are listed, together with their aggravating and relieving factors. These factors are represented by slots in the unit describing the pain type.

Physical Examination

The types of physical examination considered are shown in figure 7.8. As in the history action, each option has associated a method which calls the function ATTRIBUTES.PRINTER with the patient attributes and their default values of relevance to the option. These arguments are shown below:

Method	*Argument*
RECTAL	((Not Present BLOOD.IN.STOOLS MUCUS.IN.STOOLS SWELLINGS RECTAL.TENDERNESS ABNORMAL.ANATOMICAL.STRUCTURES EMPTY.RECTUM) (Normal STOOL.COLOUR))
VAGINAL	((Not Present DISCHARGE VAGINAL.TENDERNESS))
ABDOMINAL	((Not Present SCARS DISTENSION GUARDING MASS MUSCLE.- PROTECTION REBOUND.TENDERNESS RESTRICTED.ABDOMINAL.- MOVEMENT TENDERNESS) (Normal BOWEL.SOUNDS) (Negative MURPHY'S.SIGN))

Special Investigations

Options of this type are shown in figure 7.9. The results of special investigations are delivered after a time delay. Each type of investigation has associated SAME.-WORLD action rules which describe the results demonstrated by the investigation

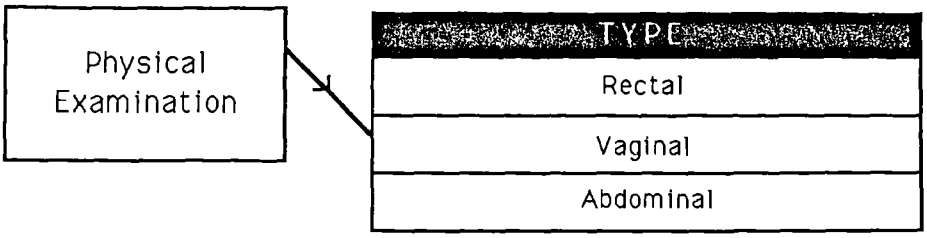


Figure 7.8: Menu Of Available Physical Examinations

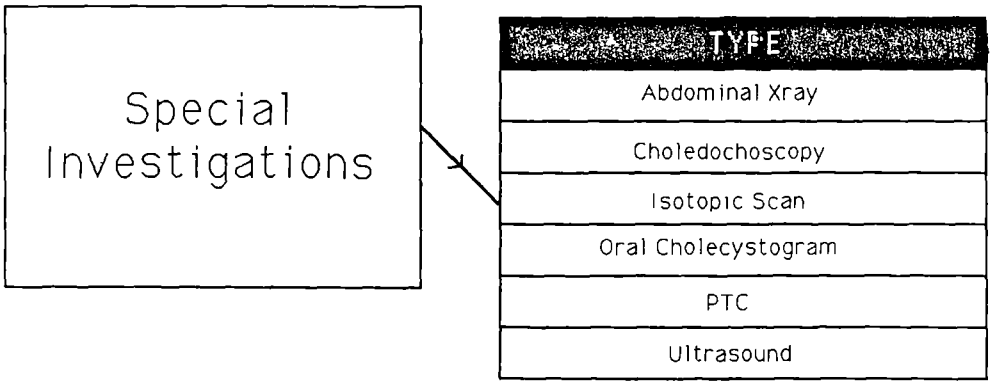
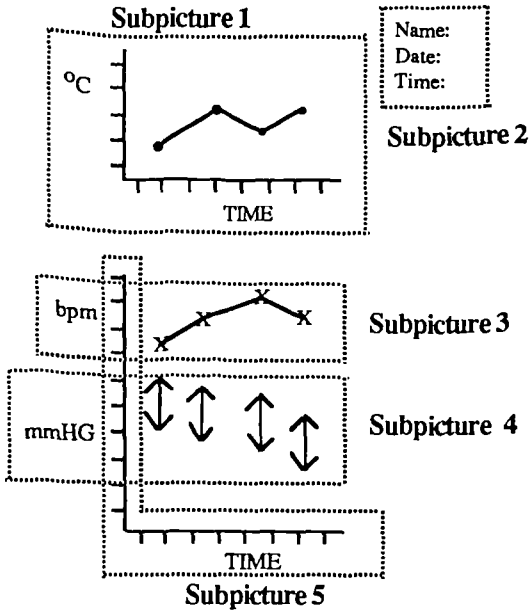


Figure 7.9: Menu Of Available Special Investigations

when certain patient conditions present. The rules for each investigation are distributed about rule classes. A slot, WAIT.TIME, in the rule classes describes the time delay between the request for an investigation and the delivery of the result. The special investigation rules are given in appendix D.

The control procedure associated with each special investigation firstly gets the monitor *sim.lock* and then checks to see that the investigation is not already awaiting processing. If not it adds the special investigation request to the CONDITIONAL.-EVENT.LIST slot of the unit SIMULATION.GENERATOR.EVENTS, the condition being a delay of WAIT.TIME simulated hours. It also creates a child world of the current world with CAUSE and SPEC.CAUSE properties 'Management' and 'Special Investigation: *selected investigation type*' respectively.

When the request is activated by the simulation generator module, the rule class is forward chained on in the context of the current world. Each rule which fires adds the patient features demonstrated by the investigation to the slot called RES.TO.BE.-PRINTED, in the SPECIAL.INVESTIGATIONS.BS unit. When no more rules will fire, the value held in the slot is presented to the user. If no rules fired the slot is empty and the message no abnormalities found is printed. The list element associated with the request in the SIMULATION.GENERATOR.EVENTS unit is then removed.



Key: Subpicture 1 comprises 3 axes, a line, 2 boxstrings and a polyline
Subpicture 2 comprises a boxstring
Subpicture 3 comprises a boxsting and a polyline
Subpicture 4 comprises 3 polylines and a boxstring
Subpicture 5 comprises 3 axes, a line and a boxstring.

Figure 7.10: Configuration Of Pictures Representing The Blood Pressure, Heart Rate And Temperature Chart

Display Charts

Blood pressure, heart rate and temperature charts have been implemented. Although each of these charts could be displayed individually using the principles to be described, the system plots all the parameters on the same chart.

To display patient parameters graphically, KEE pictures has been used. The layout of pictures in the viewport TEMP.PULSE.BP.CHARTS.VP in shown in figure 7.10. Both horizontal axes comprise two axes, a line and a bitmap picture to overcome the problem of numbering an axis scale with integers which are not necessarily increasing, for example hours 22 to 10.

When a student requests a blood pressure, heart rate and temperature chart it is generated dynamically. The control procedure servicing the request gets the monitor *SIM.LOCK* and creates a child world of the current world. This world has CAUSE and SPEC.CAUSE properties 'management' and 'Display charts: *Temperature, heart rate and blood pressure*' respectively. The control procedure then calls methods held in the DISPLAY.CHARTS.BS unit to generate the subpictures describing the chart. These methods consider the patient parameters generated over the last 24 hours of the

simulation or throughout the simulation if fewer than 24 hours have been generated.

The simulated hours at which a patient's blood pressure, heart rate and temperature are recorded are described by a list in the slot TIMES.DATA.RECORDED in the unit BP.PULSE.TEMP.DISPLAY. The methods which generate the polylines representing the systolic, diastolic, heart rate and temperature, abstract from the physiological model the physiological parameters generated at the hours data was recorded.

The component pictures of the horizontal axes which are displayed depend upon the axes scale. The axes labels are centered about the displayed pictures. Figure 7.11 illustrates an example chart. Two active images appear below the chart. Mousing on the images allows the charts to be advanced or moved back in time by a maximum of 12 hours.

To chart only the blood pressure, heart rate or temperature subparts of the BP.-PULSE.TEMP.DISPLAY viewport, the subpictures associated with the other charts should be closed and the viewport zoomed onto the chart of interest.

7.3.2 Type B

Move Forward In Time

The control procedure of the move forward in time option gets the monitor *SIM.LOCK*. Have prompted for the number of hours to advance the simulation (*ZOOM.AHEAD.TIME*), it then creates a child world of the current world with CAUSE and SPEC.CAUSE properties 'SIMULATION.ADVANCE' and by '*ZOOM.-AHEAD.TIME* hrs'. The process ZOOM.AHEAD is then added to the process list.

The function executed by the process ZOOM.AHEAD is shown in figure 7.12. The simulation generation process (*SIM.PROC*) is woken repeatedly until the required number of updates has been performed. The WAKE.PROCESS command makes the simulation process ready for activation but does not suspend the zoom ahead process. To force the activation of *SIM.PROC* the process *ZOOM.AHEAD* is suspended for at least 50 milliseconds.

The use of a separate process to advance the simulation avoids the incorporation of checks for simulation termination in the control procedure associated with the time advance. Should termination of the simulation generation occur whilst the zoom ahead process is active, the simulation generator process deletes it.

Management Operations

The management operations available for selection are shown in figure 7.13. They can be subdivided into actions associated with intravascular fluid therapy and others which alter the frequency physiological parameters are recorded. On the selection

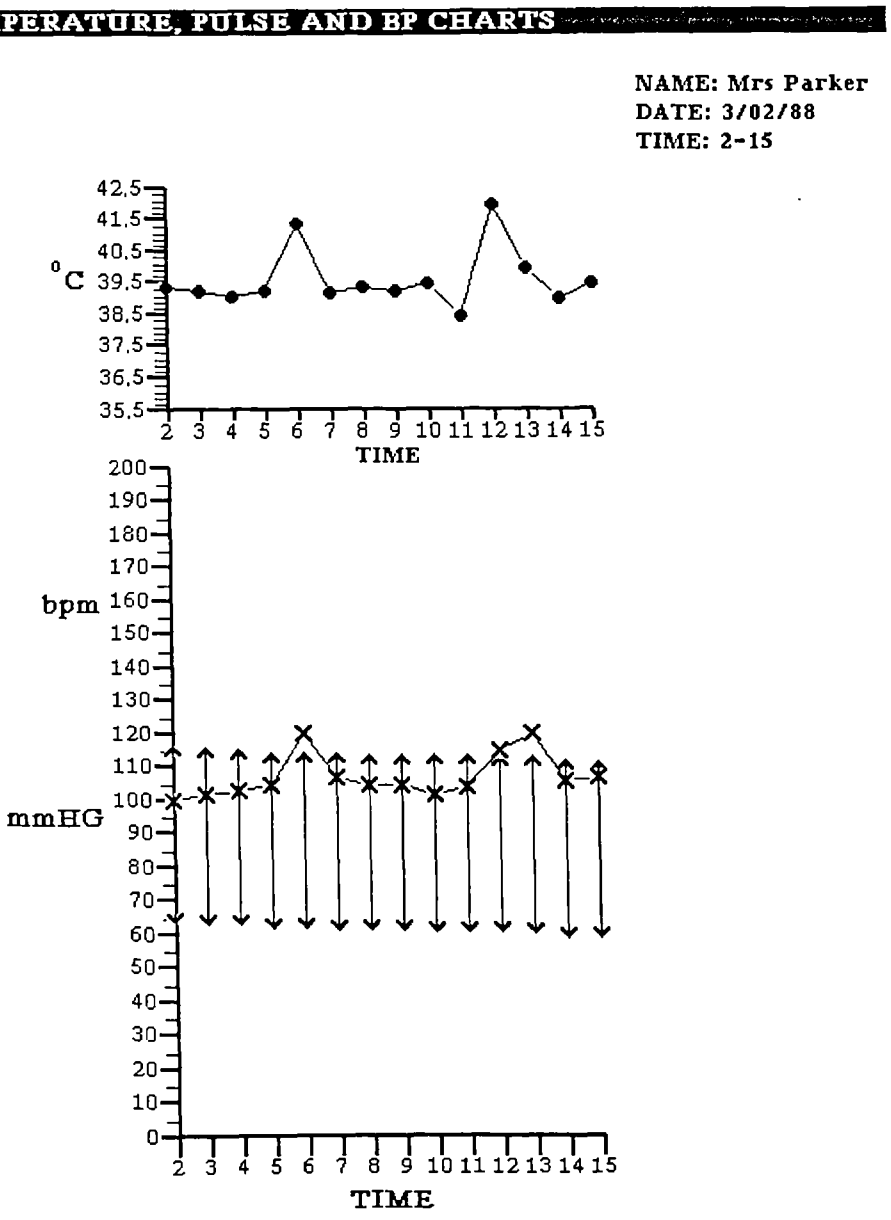


Figure 7.11: An Example Of The Blood Pressure, Heart Rate And Temperature Chart As Seen In Empyema Of The Gallbladder

```
( ZOOM.AHEAD.PROCESS
   (LET ((COUNTER 1))
     (UNTIL (GREATERP COUNTER *ZOOM.AHEAD.TIME*)
       do (WAKE.PROCESS *SIM.PROC*)
         (BLOCK 50)
         (SETQ COUNTER (ADD1 COUNTER)))))
```

Figure 7.12: Function Executed By The Process Which Advances The Simulation In Time

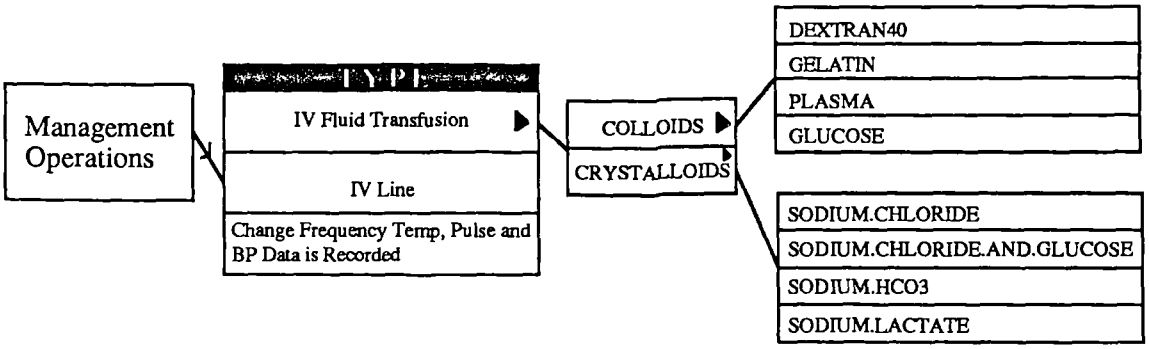


Figure 7.13: Selections Associated With The Management Operations Menu

of any action its control procedure gets the monitor *SIM.LOCK* and creates a child world of the current world with CAUSE property 'MANAGEMENT'.

The intravascular fluid options all share the same control procedure. This takes the argument (UNITname SLOTname ARGUMENTS) and messages the slot called SLOTname in the unit UNITname, a member of the unit MANAGEMENT.OPERATIONS with the arguments ARGUMENTS.

The action INSERT.IV.LINE option calls the SETUP method in the unit INSERT.IV.LINE. This method puts the value PRESENT in the IV.LINE slot of the PATIENT.PROFILE unit (in the context of the current world). It also defines the new world's SPEC.CAUSE property to be intravenous line inserted.

Each fluid transfusion option calls the SETUP method in the unit TRANSFUSION with an argument describing the fluid type. Before prompting for the volume of fluid to be administered and duration of administration, the value of the IV.LINE slot of the unit PATIENT.PROFILE is got. Should the value not be equal to PRESENT the option is terminated, since an intravenous transfusion cannot be performed without an IV line. Having obtained the volume and duration of the transfusion, the rate

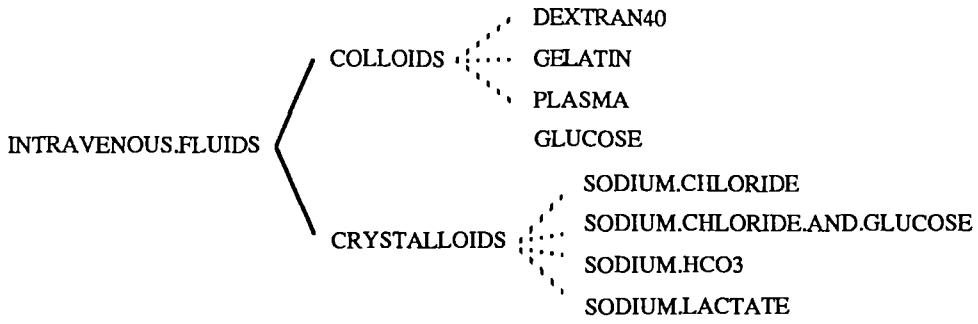


Figure 7.14: Frame Hierarchy Representing Intravascular Fluid Types

of administration per hour is found. Using the unit hierarchy of IV fluids member links (figure 7.14) the class of the fluid can be found. Depending upon the class the distribution of the hourly volume of fluid about the extracellular fluid space is found. The time at which the transfusion ends is put into the TIME.END.TRANS slot of the TRANSFUSION unit. Finally, the SPEC.CAUSE property of the newly created world is set to Transfusion: x litres of *fluid.type* given over y hours at a rate of z l/hr, where x represents the volume of fluid of type *fluid.type*, y represents the duration of the transfusion and z the rate of administration.

The default frequency with which physiological data is recorded for charting purposes is every four hours. Selection of the CHANGE.FREQ.DATA.RECORDED option allows the frequency to be modified. Having prompted for the required frequency, it is put into the FREQUENCY.DATA.REC slot of the unit BP.PULSE.TEMP.DISPLAY in the context of the new world. The world's SPEC.CAUSE property is set to 'change the frequency with which physiological data is recorded'.

Treatments

The treatment options available are shown in figure 7.15. Each of the options share the same control procedure. On the selection of an option the procedure is invoked with an argument listing the operations associated with the selection, for example (LAPAROTOMY ADHESION.DIVISION). The procedure obtains the monitor *SIM.LOCK* and creates a child world of the current world with CAUSE and SPEC.-CAUSE properties 'MANAGEMENT' and 'Treatment: *selected treatment*' respectively. For each of the active and static disease states, which are described by slots in the SIMULATION.GENERATOR.EVENTS unit, the operation(s) required to correct the state are found by getting the value from the TREATMENT slot of the unit describing the state. The disease states which are corrected by the chosen operations are found to-

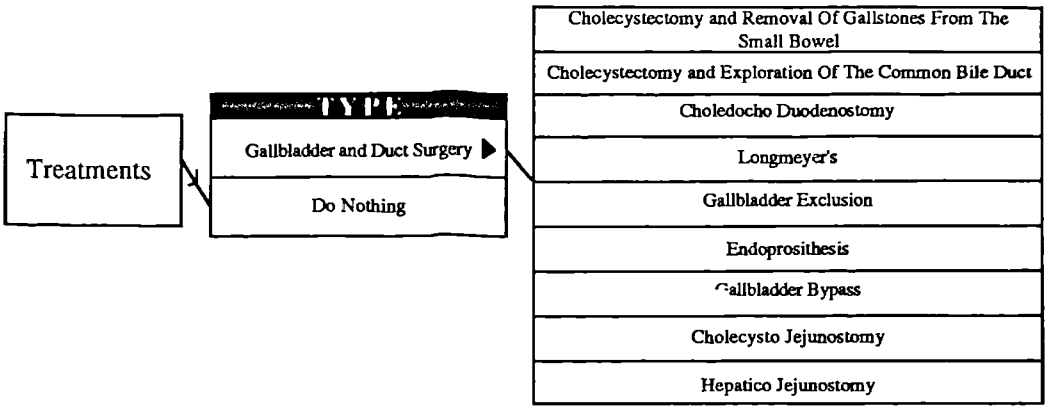


Figure 7.15: Selections Associated With The Treatments Menu

gether with the operations which partially correct the states. If partially treated disease states are present, the message:

> 'The following treatment(s) are appropriate in the treatment of the patient's disease but are needed in combination with other treatments. To cure these patient problems the treatment(s) should be reselected together with other treatment(s) or a totally different treatment strategy followed.'

is displayed followed by the operations associated with the partially corrected disease states. All inappropriate operations are displayed with the message:

> 'The following treatments were totally inappropriate - no abnormalities were found'.

When corrected disease states exist, a new snap in the patient status hierarchy is created. The snap name is used to define the *PATH.STATE* property of the new world. Member links are created from each of the existing disease states (the corrected states not being considered) and the anatomical abnormalities associated with the states found. When removing the corrected disease states from the slots describing the active and static disease states in the SIMULATION.GENERATOR.EVENTS unit, the anatomical abnormalities underlying the states are found. If the abnormalities are not present in the existing states, they are removed from the anatomical model. The shock type described in the new snap is compared with the old snap to determine whether the physiological model requires updating. If the shock types differ, the new shock type is put into the SHOCK.TYPE slot of the unit CIRCULATORY.SYSTEM. Should all the disease states be corrected, the END.TIME property of the new world is set to *SIM.TIME* and the HALT.PATIENT.SIMULATOR slot of the TUTORIAL.MANAGER unit messaged with the argument 'A surgical decision has been made which matches that of an expert surgeon'. The simulation will then terminate.

7.4 Conclusion

KEE's active image and picture facilities have been used to develop the systems user interface. KEE pictures has been applied to the dynamic creation of images, reducing the development time required to produce detailed graphic images. Its object-oriented design has been used to structure knowledge associated with the concepts represented by the picture units. For example, the student actions are displayed in a viewport, with picture units representing different categories of student action. Each action has associated a control procedure which is represented by a method in the unit. Use of KEE pictures encourages a structured representation of user interface and student action knowledge which is easily extended. This feature of the system is considered important because the student action knowledge will increase as the domain knowledge is extended, for example addition treatments options will be required.

The representation of the simulation model influences the student actions which may be incorporated into the system. Two basic types of action have been identified; those which query the model and those which may change its underlying concepts. The feedback from the action types may be immediate or may have associated a time delay. In generating a response to student queries, slots values may be queried directly or may be used by rules to infer knowledge which is not explicitly represented in the model. Actions which modify the simulation model may change the patient's evolving condition. This increases the non-deterministic nature of the system.

Chapter 8

Extending The Domain Knowledge To Cover Additional Diseases

8.1 Introduction

A teaching system for discovery learning provides the student with more opportunity to glean the subject domain's concepts if the number and variety of problems demonstrating the concepts is increased. Presenting the student with an assortment of domain problems also enables them to broaden their problem solving skills.

The combining of the subject and teaching knowledge in the representation of traditional teaching programs prevents the subject knowledge being extended. Different tutorials can only be achieved by the development of further programs. The development time of each tutorial hour is fixed and is generally regarded as being 300 man-hours. Knowledge-based systems can potentially be extended because their subject and teaching knowledge are explicit and held in separate modules. Although their initial construction time is usually longer than traditional teaching programs, the number of man-hours needed to generate further hours of tutorial time falls relative to the total system development time, figure 8.1.

For the system described a large number of different tutorials may be generated using only knowledge of gallbladder and duct diseases because the disease states in the gallbladder and duct disease state transition network are linked bidirectionally. If the links between states were directed however, the disease knowledge would support only 18 different tutorials. The time taken to model gallbladder and duct diseases was two years.

To increase the variety of clinical problems presented in tutorials, the domain knowledge has been extended to cover two further disease classes. These are appendix

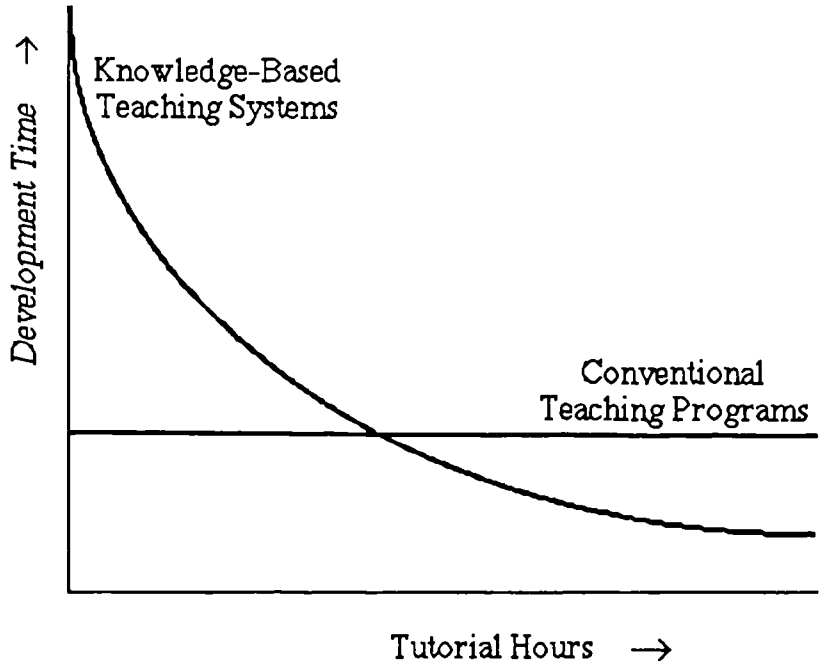


Figure 8.1: Time Taken To Develop Tutorials For Teaching Systems

and small bowel diseases. Special investigation knowledge has been extended to enable students to experiment with the investigations in the context of the new diseases. Also, the treatment knowledge has been supplemented to allow for the correction of the new diseases. In addition to allowing for the assessment of whether the expansion of the domain knowledge module facilitates the generation of further tutorials, incorporation of the new diseases into the system provides for the evaluation of the disease representation.

8.2 Schema For Acquiring And Representing Further Domain Knowledge

The model for representing abdominal disease (chapters 4 and 5) had only been used to describe gallbladder and duct diseases occurring in the right upper quadrant of the abdomen. The diseases to be added to the system were selected because they present with pain occurring at sites remote to the right upper quadrant, their underlying pathological processes had been incorporated into the domain knowledge representation but did not map to existing disease states, and one disease class, appendix diseases, frequently presents to the Accident and Emergency department whereas the other, small bowel diseases, do not.

When acquiring knowledge of further diseases, the disease model provides a focus for discussion. For each disease class, a disease state transition network was developed. States in the network were then described using the disease specification provided by the slot names and valueclass facets of the unit DISEASE.STATES in

the disease unit hierarchy. The causes of each disease state were found, together with the anatomy they involve. Pathological processes associated with each state were also found, together with treatments and state duration.

The rules sets representing the special investigations whose results signified the presence of the new diseases were revised. Rules describing the findings associated with the new diseases were added to the rule sets. The student action, 'treatment' was modified too to allow for the selection of treatments' of relevance to appendix and small bowel diseases.

8.3 Expansion Of Disease Knowledge

The representation of each disease class added to the disease model follows:

8.3.1 Small Bowel Diseases Characterised By Obstruction

Two causes of small bowel obstruction were considered, adhesions and internal hernia. Adhesions and internal hernias are associated with a *low* obstruction of the ileum and a *high* obstruction of the jejunum respectively. The disease state transition network for these small bowel diseases is shown in figure 8.2. Frames corresponding to each disease state are as follows:

(DISEASE STATE **NON-STRANGULATED.INTERNAL.ENTRAPMENT.OF.THE.-SMALL.BOWEL**
(CAUSES.WITH.SITES *((INTERNAL.HERNIA HIGH OBSTRUCTION) (JEJUNUM) ((ADHESIONS LOW OBSTRUCTION) (ILEUM))))*
(STATE.DURATION *(12 24)*)
(SUCCESSOR.DIS.STATES*(RESOLUTION STRANGULATED.INTERNAL.ENTRAPMENT.OF.THE.SMALL.BOWEL)*)
(UNDERLYING.PATH.PROCESSES *(MECHANICAL.ILEUS INFLAMMATION)*)
(CONSTIPATION *PRESENT*)
(PAIN.SITE.AT.PRESENT *(INTERNAL.HERNIA LUQ)(ADHESIONS ILL-LOCALISED)*)
(PAIN.SITE.AT.ONSET *CENTRAL*)
(TREATMENT *(HERNIA (HERNIA.REPAIR.OPERATION)) (ADHESIONS ((DO.NOTHING) (LAPAROTOMY ADHESION.DIVISION)))*)

(DISEASE STATE **STRANGULATED.INTERNAL.ENTRAPMENT.OF.THE.SMALL.-BOWEL**
(CAUSES.WITH.SITES *((INTERNAL.HERNIA HIGH OBSTRUCTION) (JEJUNUM)) ((ADHESIONS LOW OBSTRUCTION) (ILEUM))*)
(STATE.DURATION *(12 24)*)
(SUCCESSOR.DIS.STATES *(SMALL.BOWEL.ABSCESS PERF.SMALL.BOWEL)*)
(UNDERLYING.PATH.PROCESSES *(MECHANICAL.ILEUS INFLAMMATION INFECTION)*)

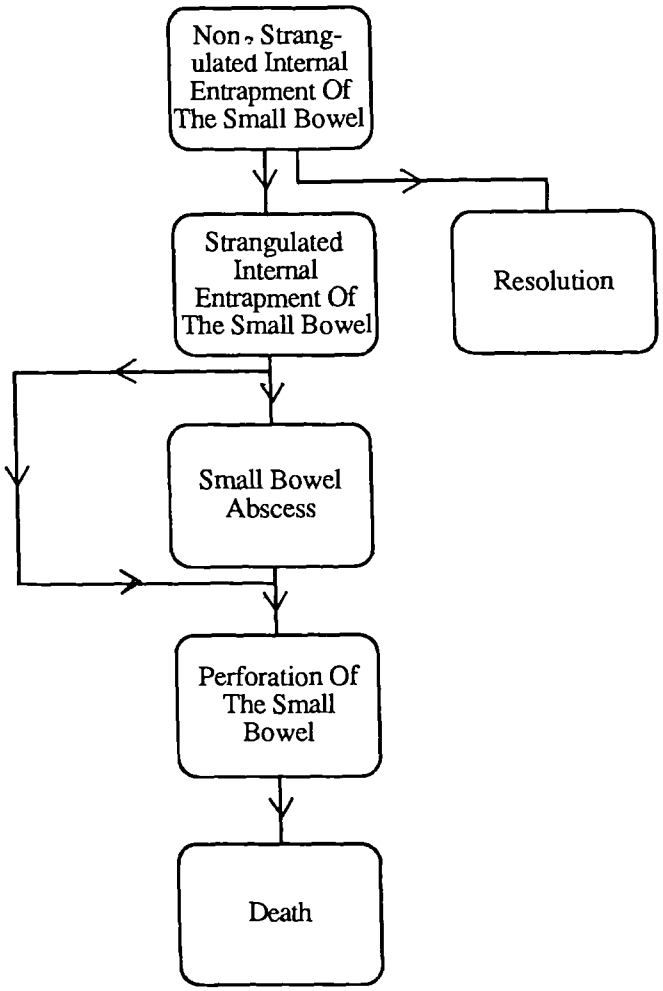


Figure 8.2: Disease State Transition Network For Small Bowel Diseases

(CONSTIPATION *PRESENT*)
(PAIN.SITE.AT.PRESENT *(INTERNAL.HERNIA LUQ) (ADHESIONS ILL-LOCALISED)*)
(PAIN.SITE.AT.ONSET *CENTRAL*)
(TREATMENT *(HERNIA ((HERNIA.REPAIR.OPERATION SMALL.BOWEL.RESECTION)))(ADHESIONS ((DO.NOTHING) (LAPAROTOMY ADHESION..DIVISION))))*

(DISEASE STATE **SMALL.BOWEL.ABSCESS**
(CAUSES.WITH.SITES *((INTERNAL.HERNIA HIGH OBSTRUCTION) (JEJUNUM)) ((ADHESIONS LOW OBSTRUCTION) (ILEUM))*)
(STATE.DURATION *(48 48)*)
(SUCCESSOR.DIS.STATES *(PERF.SMALL.BOWEL)*)
(UNDERLYING.PATH.PROCESSES *(MECHANICAL.ILEUS INFECT.LOC.PERITONITIS INFLAM.LOC.PERITONITIS)*)
(CONSTIPATION *PRESENT*)
(EMPTY.RECTUM *PRESENT*)
(PAIN.SITE.AT.PRESENT *(INTERNAL.HERNIA LUQ) (ADHESIONS ILL-LOCALISED)*)
(PAIN.SITE.AT.ONSET *CENTRAL*))
(TREATMENT *(HERNIA ((HERNIA.REPAIR.OPERATION SMALL.BOWEL.RESECTION))) (ADHESIONS ((LAPAROTOMY ADHESION.DIVISION DRAINAGE.OF.ABSCESS SMALL.BOWEL.RESECTION))))*))

(DISEASE STATE **PERF.SMALL.BOWEL**
(CAUSES.WITH.SITES *((INTERNAL.HERNIA HIGH OBSTRUCTION) (JEJUNUM)) ((ADHESIONS LOW OBSTRUCTION) (ILEUM))*)
(STATE.DURATION *(1 1)*)
(SUCCESSOR.DIS.STATES *(DEATH)*)
(UNDERLYING.PATH.PROCESSES *(PARALYTIC.ILEUS INFECT.GEN.PERITONITIS INFLAM.GEN.PERITONITIS)*)
(EMPTY.RECTUM *PRESENT*)
(PAIN.SITE.AT.PRESENT *GENERAL*)
(PAIN.SITE.AT.ONSET *CENTRAL*)
(TREATMENT *(HERNIA ((LAPAROTOMY HERNIA.REPAIR.OPERATION SMALL.BOWEL.RESECTION)))(ADHESIONS ((LAPAROTOMY ADHESION..DIVISION SMALL.BOWEL.RESECTION))))*))

The new disease knowledge has been incorporated into the disease taxonomy by creating the unit SMALL.BOWEL.DISEASES, a subclass of the unit DISEASE.STATES, and associating each of the disease states in figure 8.2 with its subclasses. The starting state of a progression, NON-STRANGULATED.INTERNAL.ENTRAPMENT.OF.THE.SMALL.BOWEL is held in the PRIMITIVE.STATES slot of the SMALL.BOWEL.DISEASES unit.

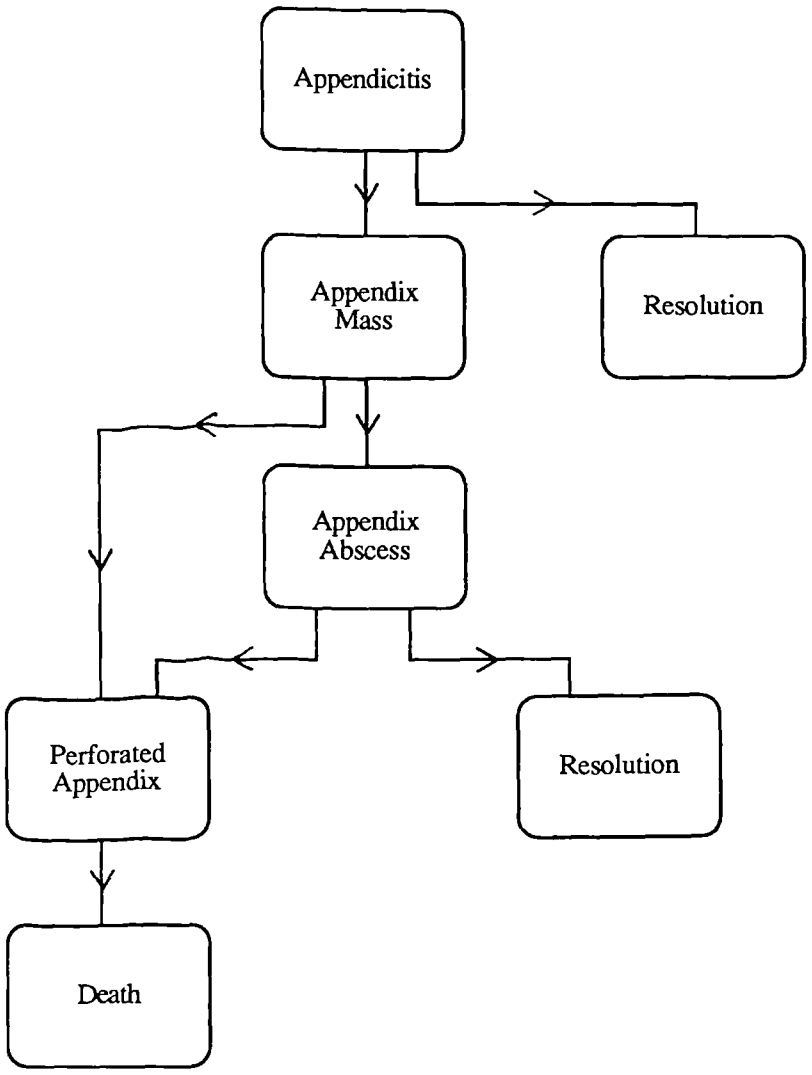


Figure 8.3: Disease State Transition Network For Appendix Disease

8.3.2 Appendix Diseases

Obstruction of the appendix by a faecolith has been modelled. The progression of the disease over time has been represented by a state transition diagram (figure 8.3), nodes of which correspond to the disease states, described by the frames below:

(DISEASE STATE **APPENDICITIS**
(CAUSES.WITH.SITES *((FAECOLITH NIL OBSTRUCTION)(APPENDIX))*)
(PAIN.CHARACTER *COLIC*)
(STATE.DURATION *(48 48)*)
(SUCCESSOR.DIS.STATES *(RESOLUTION APPENDIX.MASS)*)
(CONSTIPATION *PRESENT*)
(ANOREXIA *PRESENT*)
(VOMITING *PRESENT*)
(NAUSEA *PRESENT*)
(PAIN.SITE.AT.PRESENT *CENTRAL*)

(PAIN.SITE.AT.ONSET *CENTRAL*)
(SHOCK.TYPE *HYPOVOLAEMIC*)
(TREATMENT *(FAECOLITH ((APPENDICECTOMY)))*))

(DISEASE STATE **APPENDIX.MASS**
(CAUSES.WITH.SITES *((FAECOLITH NIL OBSTRUCTION)(APPENDIX))*)
(UNDERLYING.PATH.PROCESSES *(INFECTION INFLAMMATION)*)
(STATE.DURATION *(48 48)*)
(SUCCESSOR.DIS.STATES *(APPENDIX.ABSCESS PERF.APPENDIX)*)
(CONSTIPATION *PRESENT*)
(MASS *(PRESENT IN THE RIGHT LOWER QUADRANT)*)
(PAIN.SITE.AT.PRESENT *RLQ*)
(PAIN.SITE.AT.ONSET *CENTRAL*)
(RECTAL.TENDERNESS *(ON THE RIGHT SIDE)*)
(TREATMENT *(FAECOLITH ((DO.NOTHING)))*))

(DISEASE STATE **APPENDIX.ABSCESS**
(CAUSES.WITH.SITES *((FAECOLITH NIL OBSTRUCTION (APPENDIX))*)
(UNDERLYING.PATH.PROCESSES *(INFECT.LOC.PERITONITIS INFLAM.LOC.-PERITONITIS)*)
(STATE.DURATION *(48 48)*)
(SUCCESSOR.DIS.STATES *(RESOLUTION PERF.APPENDIX)*)
(ANOREXIA *PRESENT*)
(VOMITING *PRESENT*)
(NAUSEA *PRESENT*)
(PAIN.SITE.AT.PRESENT *CENTRAL*)
(PAIN.SITE.AT.ONSET *CENTRAL*)
(SHOCK.TYPE *HYPOVOLAEMIC*)
(TREATMENT *(FAECOLITH ((DRAINAGE.OF.APPENDIX.ABSCESS)))*))

(DISEASE.STATE **PERF.APPENDIX**
(CAUSES.WITH.SITES *((FAECOLITH NIL OBSTRUCTION (APPENDIX)))*)
(UNDERLYING.PATH.PROCESSES *(INFECT.GEN.PERITONITIS INFLAM.GEN.PER-ITONITIS)*)
(STATE.DURATION *(1 1)*)
(SUCCESSOR.DIS.STATES *(DEATH)*)
(PAIN.SITE.AT.PRESENT *GENERAL*)
(PAIN.SITE.AT.ONSET *CENTRAL*)
(TREATMENT *(FAECOLITH ((LAPAROTOMY APPENDICETOMY)))*))

To implement the new disease states a subclass, APPENDIX.DISEASES, of the class unit DISEASES.STATES has been created. The PRIMITIVE.STATES slot of the new unit has the value (APPENDICITIS), which represents the root note of

the disease state progression network. Subclasses of the unit APPENDIX.DISEASES represent the disease states in the network described by the frames above.

8.4 Expansion Of The Special Investigation Knowledge

The only special investigation which has been considered and is of relevance to appendix and small bowel diseases is abdominal Xray. For small bowel diseases, the Xray demonstrates a distended small bowel and fluid levels. Loops of bowel are also revealed, multiple loops for a low small bowel obstruction, fewer loops for a high small bowel obstruction. Xray findings, when present, for appendix diseases are a sentinel loop visualised in the right iliac fossa and a loss of the pro-peritoneal fat line.

To model these findings rules were added to the rule class ABDOMINAL.-XRAY.RULES, to check for the presence of abnormalities in the appendix and small bowel. On finding abnormalities present, the rules forward chain on two new rule classes, XRAY.APPENDIX.RULES and XRAY.SMALL.BOWEL.RULES. Each of these rule classes has associated rules which describe the Xray results associated with abnormalities in the appendix and small bowel respectively (figure 8.4). Examples of the these rules are:

```
RULE: ERECT-SMALL.BOWEL.OBST
(IF   (THE ABNORMALITY.PRESENT OF SMALL.INTESTINE IS ?X)
      (LISP (EQUAL (CADDR ?X) 'OBSTRUCTION))
THEN
      (LISP (ADD.VALUE 'SPECIAL.INVESTIGATIONS.BS 'RES.TO.BE.PRINTED
      '(Fluid levels in the intestine were demonstrated by an erect
      abdominal Xray.) 'OWN '*WORLD*)))

RULE: LOSS.OF.PRO.PERITONEAL.FAT.LINE
(IF   (LISP (EQUAL 1 (RAND 1 10)))
THEN
      (LISP (ADD.VALUE 'SPECIAL.INVESTIGATIONS.BS 'RES.TO.BE.PRINTED
      '(Loss of the pro-peritoneal fat line) 'OWN '*WORLD*)))
```

In the premise of the second rule the random function (RAND) is called to model the uncertainty associated with the demonstration of the findings given in the rule conclusion. A complete list of all the rules is given in appendix C.

8.4.1 Expansion Of Treatment Knowledge

For each of the new disease classes, a menu item has been added to the top level of the treatment menu to index the surgical treatments associated with the disease

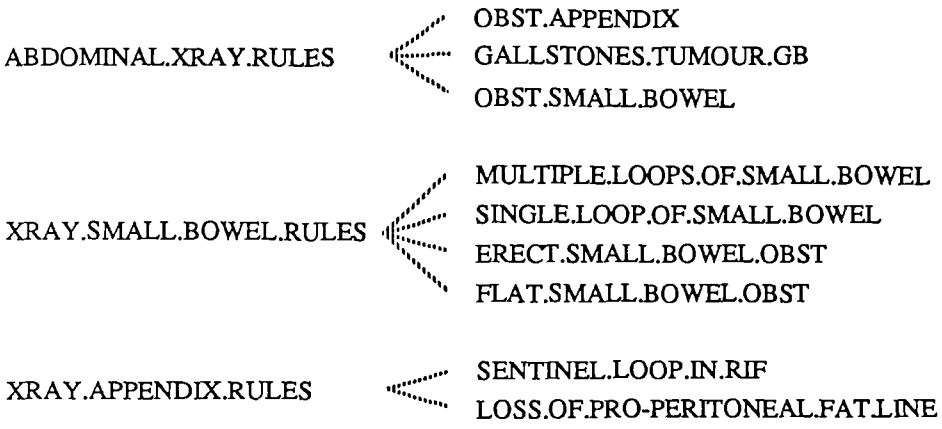


Figure 8.4: Rule Classes Associated With Xrays Of The Appendix And Small Bowel

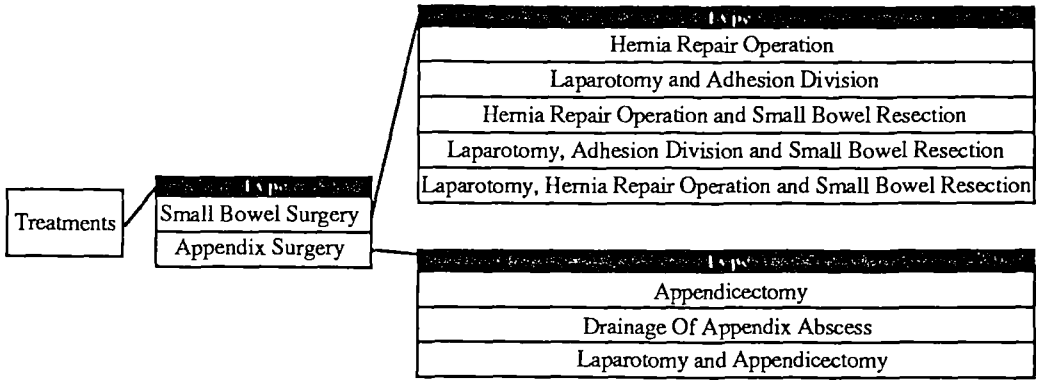


Figure 8.5: Treatment Options Available For The Correction Of Appendix And Small Bowel Diseases

class, for example 'appendix surgery'. Subitems of each of the indexing items describe the treatments required to correct all disease states associated with the class, figure 8.5. These treatments have been defined in the disease state frames, described previously. On the selection of a treatment the treatment is passed as an argument to the treatment procedure which processes it (see chapter 7).

8.5 Results

When adding more disease knowledge to the system the existing domain concepts were not augmented which suggests that the domain representation is complete. The new diseases had only to be considered at a *surface* level, since the system automatically infers from the surface specification a *deeper* knowledge representation. Using the pathological processes defined by the disease state descriptors, the system generates the physiology and clinical symptoms and signs manifested by the disease. The values

generated by the system were consistent with the real clinical situation.

Excepting the insertion of the new disease classes into the DISEASE.CLASS slot of the TUTORIAL.MANAGER frame, the number of different tutorials was increased by modifying only the domain module. Diseases were added to the disease state frame hierarchy and the special investigation and treatment knowledge was expanded. Special investigations of relevance to the new diseases were revised. The treatments applicable to the correction of the new diseases were added to the menu associated with the 'treatment' student action. The new disease knowledge supports seven new tutorials. The time taken to acquire, represent and put the new knowledge into the system was five hours given a complete understanding of the system.

8.6 Conclusion

The domain representation supports the addition of knowledge of further diseases and special investigations and is therefore extensible. When incorporating more diseases into the system they need only be considered at a surface level of detail, assuming their pathological and sign and symptom complexes are already represented by the domain model. The system derives from the surface knowledge deeper disease properties such as physiology.

The acquisition of disease knowledge, its representation and entry into the system is performed manually. A tool to assist in the acquisition of disease knowledge and its incorporation into the system could be developed. This would facilitate the entry of further domain knowledge into the system by persons unfamiliar with KEE's representation formalisms. Given a disease class the program would provide a graphics interface through which the course designer could define the diseases state transition network. The course designer would then be prompted to define each state in the network by filling in a prototype description of a disease state. This would be translated into a unit, the validity of the prototype state definition being checked using valueclass facets attached to slots representing the prototype properties. The SUCCESSOR.DIS.STATE slot value would be inferred from the disease state progression network. The unit would then be added to the disease state hierarchy. Having entered all the knowledge associated with a disease class, the 'treatment' option would be updated by inferring the treatments to be added to the treatment menu using the TREATMENT slots of the new disease states.

The disease classes were extended from one to three in five hours. The original disease class took 2 years to represent. All the medical concepts associated with the new diseases were already present in the system. The pathological process hierarchy is considered to be complete, but were any further processes identified they could easily

be added to the system using the principles described in chapter 4. New sign and symptom complexes could also be added to the domain model. Therefore the time taken to enter additional disease classes or miniworlds into the system falls as the total system development time increases. Rapid extension of the disease classes is facilitated by the mapping of the new diseases to the existing domain concepts, the simulation generation and user interface modules not requiring modification.

Ignoring bidirectional links between states in the disease state transition network but including concurrent disease states, the number of different tutorials which can be generated by the system has been expanded from 18 to 25. Traditional teaching programs can usually generate only one tutorial [75]. Comparison of the number of different tutorials which may be generated by knowledge based teaching programs and those using traditional programming representations and their development times suggests that knowledge based systems are more cost effective.

Increasing the number of different disease classes enables a more varied selection of problems for presentation in tutorials. Students' may recognise that certain management actions are applicable to many problems sharing the same domain concepts, for example the management of hypodynamic shock is the same for a perforated gallbladder as a perforated appendix. Other actions such as treatments are problem specific, for example the treatments for a perforated gallbladder and appendix are different. By taking part in the new tutorials as well as the old, the student is exposed to a wider variety of problems. The student's problem solving experience may therefore be enhanced.

Chapter 9

Conclusions and Directions For Further Research

9.1 Conclusions

An intelligent computer assisted learning system has been developed for use in the clinical environment. The system has a modular architecture comprising the domain knowledge, simulation generator, tutorial manager, user interface, simulation model and student history modules. The domain knowledge module describes a domain in general. Many different instances of problems are derived from it and used by the other system components in the generation of individual tutorials. The domain knowledge is organised about a conceptual representation which reflects its natural structure. This enables persons familiar with the domain to easily comprehend it, a feature which has been important during the development of the module and in its initial evaluation.

The system has been demonstrated in the large and complex field of acute abdominal pain. The domain module includes knowledge of gallbladder and duct diseases, anatomy, pathology, physiology, laboratory tests, special investigations and treatments. The pathological knowledge describes pathological processes in general and their effects systemically, not being restricted to single body systems like previous programs covering pathophysiology. The physiological model has knowledge of the anatomy of body fluids, pathophysiological processes associated with fluid loss and gain, and the haemodynamics associated with blood pressure and heart rate in septic and hypovolaemic shock.

In the domain module concepts of different types are organised into a multi-levelled framework. At the top level is a qualitative model of diseases, at the middle level is anatomy and pathology (also described qualitatively), and at the bottom is a combined qualitative/mathematical model of physiology. Links between diseases and pathology are explicitly stated in the specification of a disease. During the generation

of an instance of a disease, the physiological model passes qualitative knowledge of a disease and its pathology to its mathematical model of fluid balance and haemodynamics to quantify blood pressure and heart rate. The mathematical model may be rationalised using domain principles, the precise values of certain variables, for example peripheral resistance, being unknown, since they cannot be measured. To make the presentation of a disease non-deterministic and realistic, the values of patient attributes are randomised within ranges, for example the normal systolic blood pressure for a person under 50 years of age is assumed to be in the range 90 to 120 mmHg.

The teaching strategy used by the system is discovery learning. Tutorials comprise clinical scenarios in which the learner takes the role of surgical consultant and applies diagnosis and management strategies to a patient. The scenarios are selected to suit a learners clinical competence. Novice medical students may be given clinical problems which have a single underlying cause, whereas more advanced learner, for example house officers, may be presented with problems having multiple underlying causes. Traditional clinical teaching programs assume that all learners have the same problem solving capability and cannot vary a tutorial to meet an individuals needs.

The evolution of a scenario is non-deterministic, adapting to the passage of time and the reaction or lack of reaction of the learner. Students can explore the effects of time on a problem by advancing or resetting the system's clock. On resetting a scenario at some point back in time, a learner can investigate the effects of applying different management strategies. This facility increases the potential for discovery. Other teaching systems ignore time or do not let the learner have any control of it.

The system has been developed using KEE's object oriented representation. During the course of a tutorial, the tutorial manager calls the simulation generator to construct a simulation model representing a patient and their disease and builds a student history. At the end of a tutorial, the tutorial manager accesses the knowledge held in both modules to produce a tutorial summary. This shows the power provided by an object oriented representation, enabling knowledge to be used for multiple purposes. The use of object oriented representations in simulation research is ongoing. Much interest in them results from their making available to the user variables which are accessible only to the program in conventional simulation languages, a feature which has been used extensively in the prototype system.

New problems are accommodated by extending the domain module without alteration of any other system components making the development of new course material rapid. This has been demonstrated. Over five hours the domain knowledge was extended to cover appendix and small bowel diseases, the number of tutorials increasing by seven. As more problem types are added to the system the size of the domain modules conceptual representation increases, reducing the time taken to add

further problems since many of their associated concepts are already in the system. It follows that as more knowledge is added to the system the number of different tutorials resulting from it increases, reducing the development time and relative cost of a tutorial. This contrasts with teaching systems developed using traditional techniques. In these systems each tutorial demands a new program be developed. Since program components are not reuseable the development time and cost of a tutorial tends to be fixed.

The system has at this time undergone primary evaluation by exposing it to the medical experts from whom the domain knowledge was acquired. The behaviour of the system met their expectations. The domain knowledge has also been presented at various conferences [42,44,43] where it has received approval.

9.2 Further Research

Directions for further research include:

1. Secondary evaluation of the medical knowledge.
2. Carrying out field trials to assess the systems teaching performance.
3. Developing an interface to the domain module to assist a course developer in its expansion.
4. Creating a domain knowledge browser to increase the systems potential for discovery learning.
5. Adding a 'teacher-initiated' teaching strategy to the environment.

In evaluating the domain knowledge, the completeness of the pathology and physiology models would need to be determined. Diseases which have associated haemorrhagic shock would require to be modelled. To test the generality of the representation used to model sign and symptom complexes, diseases with sign and symptom complexes not already contained within the system would need to be identified. To evaluate the systems benefit to student learning it would be required to be made available to students and junior medical staff working in the clinical environment. A comparison of the clinical performance of students exposed to the system with those who had not would then need to be done.

Directions three to five above, would each require the addition of further modules into the system. The existing system components would be used by the new modules, demonstrating their potential for use in carrying out multiple tasks, a feature of their deep explicit representation.

Interfaces to the domain knowledge module could assist a course developer in extending the domain knowledge, or could provide a browsing facility. The former would use the generic representation of concepts to check that instances are consistent and complete. The browser would allow students to view medical knowledge from a general perspective, rather than in a specific context.

The teaching of diagnosis and treatment strategies is complicated by there often being multiple preferred strategies and the strategies are continuously changing as medical knowledge and aids to decision making increase. These issues would need to be addressed if 'teacher-initiated' teaching strategies were to be incorporated into the system. Research into making explicit medical decision strategies is being carried out by various workers, for example [40,39]. It is hoped that the results from such work could be incorporated into the system, and used to guide the students decision making. The system would then support two modes of learning, discovery and guided learning, increasing its potential value to clinical education.

Bibliography

[1] *Alvey-IKBS Research Workshop On Tutoring Systems: Workshop Report.* 1988.

[2] *KEE.* 1975 El Camino Real West, Mountain View, CA 94040, USA.

[3] H.H. Adelsberger, U. Pooch, R. Shannon, and G. Williams. *Rule based object oriented simulation systems*, pages 107–112. Volume 17 of *Simulation Series*, SCS, San Diego, California, 1 edition, 1986.

[4] K. Ahmed, D. Ingram, and C.J. Dickinson. *Software for Educational Computing.* MTP Press Limited, Lancaster, England, 1980.

[5] J.S. Aikins, J.C. Kunz, and E.H. Shortliffe. Puff: an expert system for interpretation of pulmonary function data. *Computers And Biomedical Research*, 16:199–208, 1983.

[6] J.R. Anderson. *Skill Acquisition; Compilation Of Weak Method Problem Solution.* Technical Report, Carnegie-Mellon University, 1985.

[7] A. Barr and E.A. Feigenbaum. *The Handbook Of Artificial Intelligence.* Volume 1, Pitman Books Limited, London, 1983.

[8] J.S. Brown, R.R. Burton, and J. De Kleer. Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman and J.S. Brown, editors, *Intelligent Tutoring Systems*, Academic Press, 1982.

[9] R.R. Burton. Diagnosing bugs in a simple procedural skill. In D. Sleeman and J.S. Brown, editors, *Intelligent Tutoring Systems*, Academic Press, 1982.

[10] J.R. Carbonell. AI in CAI: an artificial-intelligence approach to computer-assisted instruction. *IEE Transactions On Man-Machine Systems*, MMS-11(4), 1970.

[11] B. Chandrasekaran and S. Mittal. Conceptual representation of medical knowledge for diagnosis by computer: MDX and related systems. *Advances In Computers*, 22:217–293, 1983.

[12] B. Chandrasekaran and S. Mittal. Deep versus compiled knowledge approaches to diagnostic problem-solving. In *Conference Of American Association For AI, Pittsburgh, Pennyslvania, USA*, pages 349–354, 1982.

[13] W.J. Clancey. GUIDON. *Journal Of Computer-Based Instruction*, 10(1–2):8–15, 1983.

[14] W.J. Clancey. *Methodology For Building An Intelligent Tutoring System.* Memo HP-81-18, Stanford Heuristic Programming Project, Stanford University, California, USA, 1981.

[15] W.J. Clancey and R. Letsinger. Neomycin: reconfiguring a rule-based expert system for application to teaching. In *Proc. Int. Joint Conference On Artificial Intelligence-7*, 1981.

[16] G.S. Clayden and B. Wilson. *Computer-Assisted Learning In Medical Education.* ASME Office, 2 Roseangle, Dundee, DD1 4LR, Scotand, 1988.

[17] F.T. de Dombal. Acute abdominal pain – an O.M.G.E. survey. *Scandinavian Journal Of Gastroenterology*, 14:29–43, 1979.

[18] F.T. de Dombal. Computer-aided diagnosis of acute abdominal pain. The British experience. *Rev. Epidem. et Sante Publ.*, 32:50–56, 1984.

[19] F.T. de Dombal. *Diagnosis Of Acute Abdominal Pain.* Churchill Livingstone, Edinburgh, 1980.

[20] F.T. de Dombal, D.J. Leaper, J.R. Staniland, A.P. McCann, and J.C. Horrocks. Computer-aided diagnosis of acute abdominal pain. *British Medical Journal*, 2:9–13, 1972.

[21] C. Dede. A review and synthesis of recent research in intelligent computer-assisted instruction. *Int. J. Man-Machine Studies*, 24:329–353, 1986.

[22] C.J. Dickinson and D. Ingram. A review of educational applications of a series of four models of circulation, respiration, body fluids and electrolytes and drug absorption and distribution. In B. Barber, F. Gremy, K Ubola, and G. Wagner, editors, *Proceedings Medical Information Berlin*, pages 471–478, 1979.

[23] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979.

[24] P.R. Edwards. Microcomputer simulations. In W.A. Corbett, editor, *Medical Applications Of Microcomputers*, pages 69–85, John Wiley and Sons Ltd, 1987.

[25] P.R. Edwards, R. Coughlan, M.J. Taylor, and W.A. Corbett. Surgical emergencies - microcomputer-based simulations. In *Proceedings, Medical Microcomputer Applications Workshop*, Springer Verlag, London, 1986.

[26] G. Entwisle and D.R. Entwisle. The use of a digital computer as a teaching machine. *Journal Of Medical Education*, 38:803–812, 1963.

[27] P.J. Feltovich, P.E. Johnson, J.H. Moller, and D.B. Swanson. LCS: the role and development of medical knowledge in diagnostic expertise. In W.J. Clancey and E.H. Shortliffe, editors, *Readings In Medical Artificial Intelligence: The First Decade*, Addison-Wesley, 1984.

[28] T. Festa, F. Enrichens, P. Mao, G. Benedetto, A. Mauri, E. Visetti, and G. Olivero. Teaching the critically ill surgical patient evalutation by computer application. In *Medical Informatics Europe: Proceedings Of The Seventh International Congress*, pages 735–738, 1987.

[29] R.E. Filman. Reasoning with worlds and truth maintenance in a knowledge-based programming environment. *Communications of the ACM*, 31(4), 1988.

[30] L. Ford and M. Yazdani. Tutoring systems: the state-of-the-art in great britain. *Expert Systems*, 5(4):328–339, 1988.

[31] British National Formulary. *Publicaton of the British Medical Association and the Pharmaceutical Society of Great Britain.* British Medical Association and the Pharmaceutical Society of Great Britain, 1988.

[32] C. Geddes, K.E. Kendle, A.B. Selkirk, and W. Walker. Kekepi: a computer program for simulating the therapeutic responses of patients suffering from idiopathic epilepsy. *Medical Education*, 17:325–330, 1983.

[33] A.L. Glass and K.J. Holyoak. Alternative conceptions of semantic memory. *Cognition*, 3:313–339, 1975.

[34] J.C.B. Grant. *Method Of Anatomy.* E and S Livingstone Ltd, Edinburgh, 1965.

[35] J.H. Green. *Basic Clinical Physiology.* Oxford University Press, third edition, 1978.

[36] M.E. Groer and M.E. Shekleton. *Basic Pathophysiology.* The C.V. Mosby Company, St. Lousis, Missouri, 1979.

[37] J.D. Hardy, editor. *Hardy's Textbook Of Surgery.* J.B. Lippincott Company, Philadephia, 1983.

[38] J.R. Hartley and D.H. Sleeman. Towards intelligent teaching systems. *International Journal Of Man-Machine Studies*, 15(2), 1973.

[39] M. Hobsley. The nature of clinical acumen. *Theoretical Surgery*, 1(1), 1986.

[40] M. Hobsley. *Pathways In Surgical Management.* Arnold, London, 1986.

[41] J.W. Hooper and K.D. Reilly. An algorithmic analysis of simulation strategies. *International Journal Of Computer And Information Sciences*, 11(2):101–122, 1982.

[42] C.E. Johnson, P.R. Edwards, M.J. Taylor, and W.A. Corbett. Clinical instruction using a knowledge-based computer system. *European Surgical Research (Abstracts Of The 23rd Congress, Bologna 1988)*, 20(Supplement 1):68, 1988.

[43] C.E. Johnson, M.J. Taylor, and W.A. Corbett. A clinical simulation environment for medical education. In F.R. Vicary, editor, *Computers in Gastroenterology*, Springer Verlag, London, 1988.

[44] C.E. Johnson, M.J. Taylor, and W.A. Corbett. Current work on the development of an ICAI system for acute abdominal pain. In *Medical Informatics Europe: Proceedings Of The Seventh International Congress*, pages 739–746, 1987.

[45] A.M. Law and W.D. Kelton. *Simulation Modelling And Analysis.* McGraw-Hill Book Company, 1982.

[46] R.W. Lawler. Learning environments: now, then, and someday. In R.W. Lawler and M. Yazdani, editors, *Artificial Intelligence and Education Volume One*, Ablex Publishing Corporation, 1987.

[47] W.J. Long, S. Naimi, M.G. Criscitiello, and S. Kurzrok. Reasoning about therapy from a physiological model. In R. Salamon, B. Blum, and M. Jorgensen, editors, *MEDINFO 86*, pages 756–760, Elsevier Science Publishers B.V. (North Holland), 1986.

[48] F.A. Martin. Control models in computer-assisted learning. *Expert Systems*, 5(4):316–326, 1988.

[49] I. Mozetic, I. Bratko, and N. Lavrac. *The Derivation Of Medical Knowledge From A Qualitative Model Of The Heart.* Technical Report, Jozef Stafan Institute, Jamova 39, 61000 Ljubljana, Yugoslavia, 1984.

[50] T.S. Murray, W.R. Dunn, R.W. Cupples, J.H. Barber, and B.D. Scott. The potential of computer assisted learning in medical education. *Journal Of The Royal College Of Physicians*, 11:401–404, 1977.

[51] K.A. Myers, R.D. Marshall, and J. Freidin. *Principles Of Pathology In Surgery.* Blackwell Scientific Publications, 1980.

[52] G.L. Nardi and G.D. Zuidema. *Surgery: Essentials Of Clinical Practice.* Little, Brown and Company, Boston, fourth edition, 1982.

[53] R. O'Keefe. Simulation and expert systems - a taxonomy and some examples. *Simulation*, 1(46):10–16, 1986.

[54] T. O'Shea, R. Bornat, B. du Boulay, M. Eisenstad, and I. Page. Tools for creating intelligent computer tutors. In Elithor and Banerjii, editors, *Human and Artificial Intelligence*, pages 181–189, North Holland, 1984.

[55] S. Papert. Microworlds: transforming education. In R.W. Lawler and M. Yazdani, editors, *Artificial Intelligence and Education Volume One*, Ablex Publishing Corporation, 1987.

[56] R.S. Patil, P. Szolovits, and W.B. Schwartz. Modeling knowledge of the patient in acid-base and electrolye disorders. In P. Szolovits, editor, *AI In Medicine*, Westview Press, 1982.

[57] H.E. Pople. Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnostics. In P. Szolovits, editor, *AI In Medicine*, Westview Press, 1982.

[58] R. Pounder. *'Doctor, There's Something Wrong With My Guts!'.* Smith Kline and French Laboratories Limited, Welwyn Garden City, Hertfordshire AL7 1EY, 1983.

[59] C. Price and M. Lee. Applications of deep knowledge. *Artificial Intelligence In Engineering*, 3(1):12–17, 1988.

[60] M.C.A. Puntis. Computer aided learning. In W.A. Corbett, editor, *Medical Applications Of Microcomputers*, pages 45–67, John Wiley and Sons Ltd, 1987.

[61] B. Richards, J. Minshull, L. Al-Basri, and A. Campbell. A computer-aided-learning system for student nurses. In R. Salamon, B. Blum, and M. Jorgensen, editors, *MEDINFO 86*, pages 975–977, 1986.

[62] S. Rosenbaum and R.K. Skinner. A survey of heights and weights of adults in Great Britain, 1980. *Annals Of Human Biology*, 12(2), 1985.

[63] H. Schneiderman and R.L. Muller. The diagnosis game. a computer-based exercise in clinical problem solving. *J.A.M.A.*, 219:333–335, 1972.

[64] S.I. Schwartz, G.T. Shires, F.C. Spencer, and E.H. Storer, editors. *Principles Of Surgery.* McGraw-Hill Book Company, fourth edition, 1984.

[65] S.K. Shami and S. Knight. Computer-aided instruction. In F.R. Vicary, editor, *Computers in Gastroenterology*, Springer Verlag, London, 1988.

[66] E.H. Shortliffe. *Computer-Based Medical Consultations: MYCIN.* American-Elsevier, New York, 1976.

[67] B.F. Skinner. Teaching machines. *Science*, 128:969–977, 1958.

[68] J.B. Skinner, G. Knowles, R.F. Armstrong, and D. Ingram. The use of computerized learning in intensive care: an evaluation of a new teaching program. *Medical Education*, 17:49–53, 1983.

[69] D.H. Sleeman. Assessing competence in basic algebra. In D. Sleeman and J.S. Brown, editors, *Intelligent Tutoring Systems*, pages 185–199, Academic Press, 1982.

[70] F.G. Smiddy. *Tutorials In Surgery 1.* Pitman Medical Publishing Company Ltd., Tunbridge Wells, England, 1977.

[71] E.E. Smith and D.L. Medin. *Categories And Concepts. Cognitive Science Series, 4*, Harvard University Press, Cambridge, Massachusetts, 1981.

[72] R.L. Solso. *Cognitive Psychology*, chapter 7. Allyn and Bacon, Newton, Massachusetts, second edition, 1988.

[73] M. Staderini and E.M. Staderini. Simulating the effects of cardiovascular drugs: the Didapharm-de system. In *Medical Informatics Europe: Proceedings Of The Seventh International Congress*, pages 762–765, 1987.

[74] M. Stelzner, J. Dynis, and F. Cummins. *The SimKit System: Knowledge-Based Simulation and Modeling Tools in KEE.* Technical Report, Intellicorp, Inc., 1975 El Camino Real West, Mountain View, CA 94040, USA, 1987.

[75] M.J. Taylor, W.A. Corbett, P.R. Edwards, and J.R. Coughlan. System design features for clinical simulations. In *Trends In Computer Assisted Education*, Blackwells, Oxford, 1986.

[76] K.D. Tocher. Keynote address. In *1979 Winter Simulation Conference*, pages 640–654, 1979.

[77] J.B. Walter and M.S. Israel. *General Pathology.* Churchill Livingstone, fourth edition, 1974.

[78] S.M. Weiss, C.A. Kulikowski, S. Amarel, and A. Safir. A model-based method for computer-aided medical decision making. In W.J. Clancey and E.H. Shortliffe, editors, *Readings In Medical Artificial Intelligence: The First Decade*, Addison-Wesley, 1984.

[79] L.E. Widman. Representation method for dynamic causal knowledge using semi-quantitative simulation. In R. Salamon, B. Blum, and M. Jorgensen, editors, *MEDINFO 86*, pages 180–184, Elsevier Science Publishers B.V. (North-Holland), 1986.

[80] R. Wilensky. *LISPcraft.* W.W. Norton and Company, New York, 1984.

[81] B. Woolf, D. Blegen, J.H. Jansen, and A. Verloop. Teaching a complex industrial process. In R.W. Lawler and M. Yazdani, editors, *Artificial Intelligence And Education: Volume One*, Ablex Publishing Corporation, Norwood, New Jersey, 1987.

[82] M. Yazdani. Intelligent tutoring systems: an overview. In R.W. Lawler and M. Yazdani, editors, *Artificial Intelligence and Education Volume One*, Ablex Publishing, Norwood, 1987.

[83] B.P. Zeigler. Hierarchical, modular discrete-event modelling in an object-oriented environment. *Simulation*, 49(5):219–230, 1987.

Appendix A

Examples Of KEE Units Representing The Domain Knowledge

A.1 Units From The Disease Classification Hierarchy

```
(DISEASE.STATE.CLASSFN
  "Contains knowledge of disease state classification."
  ((CHRONIC.STATES ((SUBCLASS.OF #[Unit: DISEASE.STATES AB-PAIN]))
    ((Comment "Chronic disease states.")))
       (MULTIPLE.ACUTE.STATES ((SUBCLASS.OF #[Unit: DISEASE.STATES AB-PAIN]))
    ((Comment "Acute disease states which may exist concurrently.")))
    (PRIMITIVE.STATES ((SUBCLASS.OF #[Unit: DISEASE.STATES AB-PAIN]))
    ((Comment "States at the start of a disease state transition path.")))
    ))

(DISEASE.STATES
  ((CAUSES.WITH.SITES UNIQUE.VALUES (#[Unit: LIST KEEDATATYPES])
      ((Cardinality.Min (1))
       (Comment
"List of lists with 3 elements.  The first element is the entity which causes the
 anatomical abnormality, the second is a list with elements describing the patho-
 logical process resulting from the abnormality and its severity and the third is a
 list of possible anatomical sites affected."
)))
   (PAIN.SITE.AT.ONSET
            ((ONE.OF CENTRAL GENERAL LS LLQ LUQ RS RLQ RUQ UH LH ILL-LOCALISED))
     ((Cardinality.Max (1))))
   (PAIN.SITE.AT.PRESENT
((UNION (ONE.OF CENTRAL GENERAL LS RS LLQ LUQ RLQ RUQ UH LH ILL-LOCALISED)
 (MEMBERP PAIN.SITE.DSCRPTRP))))
   (STATE.DURATION (#[Unit: LIST KEEDATATYPES])
```

```
   ((Comment "The state duration in hours.")))
   (SUCCESSOR.DIS.STATES ((UNION (SUBCLASS.OF #[Unit: DISEASE.STATES AB-PAIN])
         (ONE.OF DEATH RESOLUTION))))
   (TREATMENT ((Comment
 "Treatment an expert would select in the event of experiencing a patient with the
  disease state described by this unit."
)))
   (UNDERLYING.PATH.PROCESS UNION
               ((SUBCLASS.OF #[Unit: PATHOLOGICAL.PROCESSES AB-PAIN]))
((Cardinality.Min (1))))))

(GB.DUCT.DISEASES
  (DISEASE.STATES)
  (DISEASE.STATE.CLASSFN)
  "Diseases of the gallbladder, cystic duct, common hepatic duct and bile duct."
  ((PAIN.SITE.AT.ONSET (RUQ))
   (PAIN.SITE.AT.PRESENT (RUQ)))
  ((CHRONIC.STATES (#[Unit: ASYMPTOMATIC.GALLSTONES AB-PAIN]
    #[Unit: CA.EXTRAHEPATIC.BILE.DUCTS AB-PAIN]
    #[Unit: CA.GALLBLADDER AB-PAIN]))
      (MULTIPLE.ACUTE.STATES ((ACUTE.CHOLECYSTITIS CHOLANGITIS)
   (CHRONIC.CHOLECYSTITIS CHOLANGITIS)))
   (PRIMITIVE.STATES (#[Unit: ACUTE.CHOLECYSTITIS AB-PAIN]
                      #[Unit: CHRONIC.CHOLECYSTITIS AB-PAIN]
                      #[Unit: ACUTE.PANCREATITIS AB-PAIN]
      #[Unit: CHOLANGITIS AB-PAIN] #[Unit: OBST.BT.EX.GB AB-PAIN]))
   ))

(ACUTE.CHOLECYSTITIS
  (GB.DUCT.DISEASES)
   ((ANOREXIA (PRESENT))
   (CAUSES.WITH.SITES (((GALLSTONES NIL OBSTRUCTION)
(GALLBLADDER.EX.HP HARTMANNS.POUCH CYSTIC.DUCT))))
   (MASS (PRESENT))
   (MURPHY'S.SIGN (POSITIVE)
  ((ONE.OF POSITIVE NEGATIVE))
  ((Cardinality.Min (1))
   (Cardinality.Max (1))))
   (NAUSEA (PRESENT))
   (PREVIOUS.INDIGESTION (PRESENT))
   (RADIATION (TIP.OF.SCAPULA EPIGASTRIUM CENTRE.BACK))
   (RESTRICTED.AB.MOVEMENT (WITH.RESPIRATION))
   (STATE.DURATION ((72 96)))
   (SUCCESSOR.DIS.STATES (#[Unit: EMPYEMA AB-PAIN]
                          #[Unit: CHRONIC.CHOLECYSTITIS AB-PAIN]
                          #[Unit: PERF.GB AB-PAIN]))
   (TREATMENT ((GALLSTONES ((CHOLECYSTECTOMY)))))
```

```
   (UNDERLYING.PATH.PROCESS (#[Unit: INFECTION AB-PAIN] #[Unit: INFLAMMATION AB-PAIN]))
   (VOMITING (PRESENT))))

(GALLSTONE.ILEUS
  (GB.DUCT.DISEASES)
  ((CAUSES.WITH.SITES (((GALLSTONES NIL OBSTRUCTION)
(SMALL.INTESTINE))))
   (PAIN.SITE.AT.ONSET (RUQ))
   (PAIN.SITE.AT.PRESENT (CENTRAL))
   (STATE.DURATION ((72 96)))
   (SUCCESSOR.DIS.STATES (DEATH)
      ((Comment "Assume at present the gallstone passes out through the rectum.")))
   (TREATMENT ((GALLSTONES ((CHOLECYSTECTOMY
                            REMOVAL.OF.GALLSTONES.FROM.THE.SMALL.BOWEL)))))
   (UNDERLYING.PATH.PROCESS (#[Unit: MECHANICAL.ILEUS AB-PAIN]))))

(EMPYEMA
  (GB.DUCT.DISEASES)
  ((CAUSES.WITH.SITES (((GALLSTONES NIL OBSTRUCTION)
(HARTMANNS.POUCH CYSTIC.DUCT))))
   (MASS (PRESENT))
   (MURPHY'S.SIGN (POSITIVE)
  ((ONE.OF POSITIVE NEGATIVE))
  ((Cardinality.Min (1))
   (Cardinality.Max (1))))
   (NAUSEA (PRESENT))
   (STATE.DURATION ((96 96)))
   (SUCCESSOR.DIS.STATES (PERF.GB RESOLUTION))
   (TREATMENT ((GALLSTONES ((CHOLECYSTECTOMY)))))
   (UNDERLYING.PATH.PROCESS (#[Unit: INFECT.LOC.PERITONITIS AB-PAIN]
            #[Unit: INFLAM.LOC.PERITONITIS AB-PAIN]))
   ))

(PERF.GB
  (GB.DUCT.DISEASES)
  ((CAUSES.WITH.SITES (((GALLSTONES NIL OBSTRUCTION)
(HARTMANNS.POUCH GALLBLADDER.EX.HP))))
   (PAIN.SITE.AT.PRESENT (GENERAL))
   (STATE.DURATION ((1 1)))
   (SUCCESSOR.DIS.STATES (DEATH RESOLUTION)
      ((Comment ("Next state would be paralytic.ileus but this
                  disease has not been encoded yet, so assume resoultion."
))))
   (TREATMENT ((GALLSTONES ((CHOLECYSTECTOMY)))))
   (UNDERLYING.PATH.PROCESS (#[Unit: INFECT.GEN.PERITONITIS AB-PAIN]
    #[Unit: INFLAM.GEN.PERITONITIS AB-PAIN]))
   ))
```

A.2 Units From The Pathological Processes Hierarchy

```
(PATHOLOGICAL.PROCESSES
  "Description of pathological processes underlying diseases."
  ((SHOCK.TYPE (#[Unit: SHOCK AB-PAIN])
    ((Cardinality.Min (1))
            (Comment "Type of shock associated with the pathological type."))))

(PERITONITIS
"Clinical signs associated with peritonitis.  Homeostatis attributes are attached
 to the infection/inflammation subclasses since they are dependant upon the cause
 of the peritonitis."
  ((MUSCLE.PROTECTION UNIQUE.VALUES
        ((ONE.OF INVOLUNTARY.GUARDING INCREASED.TONE RIGIDITY)))
   (REBOUND.TENDERNESS UNIQUE.VALUES)))

(LOC.PERITONITIS
  (PERITONITIS)
  "Localised pus forming an abcess."
  ((MUSCLE.PROTECTION (INVOLUNTARY.GUARDING INCREASED.TONE))
   (REBOUND.TENDERNESS (PRESENT))))

(GEN.PERITONITIS
  (PERITONITIS)
"Generalised peritonitis occurs in patients who are not very young but are
 under 70 years of age, and have pus or faeces in their peritoneum."
  ((BOWEL.SOUNDS (ABSENT))
   (MUSCLE.PROTECTION (RIGIDITY INCREASED.TONE INVOLUNTARY.GUARDING))
   (REBOUND.TENDERNESS (PRESENT))
   (TENDERNESS (PRESENT))))

(INFLAMMATION
  (PATHOLOGICAL.PROCESSES)
  ((MUSCLE.PROTECTION (LOSS.OF.FUNCTION)
((Comment "Describes the physiological precesses operating on the
 smooth or striated muscle involved (smooth muscle gives rise to involuntary
 responses, striated to voluntary)"
    )))
   (PAIN.CHARACTER (#[Unit: ACHES AB-PAIN])
((SUBCLASS.OF #[Unit: PAIN.CLASSIFICATIONS AB-PAIN]))
((Comment "All diseases of type inflammation have pain of type aches.")))
   (SHOCK.TYPE (#[Unit: PRIMARY.HYPERDYNAMIC.SHOCK AB-PAIN]))
   (TENDERNESS (PRESENT))
   (THIRD.SPACE.LOSS ((.02 .04))
     (#[Unit: LIST KEEDATATYPES])
     ((Cardinality.Min (1))
      (Cardinality.Max (1))
```

```
(Comment ( "Third space loss per hour in litres.  Daily loss of 0.5-1 litres
                    of fluid (to 2 dp.)."))))))

(INFECTION
   (PATHOLOGICAL.PROCESSES)
   ((PAIN.CHARACTER (#[Unit: ACHES AB-PAIN])
((SUBCLASS.OF #[Unit: PAIN.CLASSIFICATIONS AB-PAIN]))
((Cardinality.Min (1))
(Cardinality.Max (1))
(Comment ("All diseases of type infectin have pain of type aches."))))
   (REBOUND.TENDERNESS (PRESENT)
       ((ONE.OF PRESENT NOT.PRESENT)))
   (SHOCK.TYPE (#[Unit: PRIMARY.HYPERDYNAMIC.SHOCK AB-PAIN]))
   (TEMP.CHANGE (CONSTANT)
((ONE.OF SWINGING CONSTANT))
((Cardinality.Min (1))
 (Cardinality.Max (1))))
   (TEMPERATURE.RISE (1)
     (#[Unit: NUMBER KEEDATATYPES])
     ((Comment ("Rise in degrees centigrade."))
      (Cardinality.Max (1))))))

(INFLAM.LOC.PERITONITIS
  (LOC.PERITONITIS INFLAMMATION)
  ((MUSCLE.PROTECTION (INCREASED.TONE INVOLUNTARY.GUARDING))
   (THIRD.SPACE.LOSS ((.06 .08))
((Comment ( "Third space loss per hour in litres.  Daily loss of 0.5-2
                     litres of fluid (to 2 dp.)."))))))

(INFECT.LOC.PERITONITIS
  (LOC.PERITONITIS INFECTION)
  ((TEMP.CHANGE (SWINGING))
   (TEMPERATURE.RISE (1.5))))

(INFLAM.GEN.PERITONITIS
  (GEN.PERITONITIS INFLAMMATION)
  ((MUSCLE.PROTECTION (INCREASED.TONE INVOLUNTARY.GUARDING RIGIDITY))
   (SHOCK.TYPE (#[Unit: HYPERDYNAMIC.SHOCK AB-PAIN]))
   (THIRD.SPACE.LOSS ((.08 .1))
((Comment ( "Third space loss per hour in litres.  Daily loss of
 2-2.5 litres (to 2 dp.)."))))))

(INFECT.GEN.PERITONITIS
  (GEN.PERITONITIS INFECTION)
  ((SHOCK.TYPE (#[Unit: HYPERDYNAMIC.SHOCK AB-PAIN]))
   (TEMP.CHANGE (CONSTANT))
   (TEMPERATURE.RISE (2.5))))
```

A.3 Units From The Pain Classifications Hierarchy

```
(PAIN.CLASSIFICATIONS
  "DESCRIPTION OF THE CHARACTER OF THE PAIN."
  ((AGGRAVATING.FACTORS)
   (NAUSEA)
   (RELIEVING.FACTORS)
   (TYPE.PAIN ((ONE.OF STEADY INTERMITTENT)))
   (VOMITING)))

(ACHES
  (PAIN.CLASSIFICATIONS)
  ((AGGRAVATING.FACTORS (MOVEMENT COUGHING BREATHING))
   (RELIEVING.FACTORS (STAYING.STILL VOMITING DRUGS))
   (TYPE.OF.PAIN (STEADY))))

(COLICS
  (PAIN.CLASSIFICATIONS)
  ((NAUSEA (PRESENT))
   (TYPE.OF.PAIN (INTERMITTENT))
   (TYPE.PAIN (INTERMITTENT))
   (VOMITING (PRESENT))))
```

A.4 Units From Anatomical Parts Hierarchy

```
(ANATOMICAL.PARTS
  ((ABNORMALITY.PRESENT (#[Unit: LIST KEEDATATYPES])
((NONE))
((DELETION.MODE NIL)
 (AVUNITS ((#[Unit: CHANGE.ANATOMICAL.PATHOLOGY.MONITOR AB-PAIN] ALL NIL)
   (#[Unit: SETUP.COMPLEXES AB-PAIN] ALL NIL)))
 (Comment "Represents the cause of the pathology associated with this anatomy.
           The slot will have multiple values when multiple diseases occur which
           involve the same anatomy." )))
   (ASSOC.COMPLEXES (#[Unit: LIST KEEDATATYPES])
    ((Cardinality.Min (1))
(Comment "Tuples representing the complexes activated when abnormalities occur.
          Tuples take the form:(SEVERITY of abnormality, ABNORMALITY, COMPLEX)." )))
(CONNECTED.TO ((MEMBER.OF #[Unit: ANATOMICAL.PARTS AB-PAIN]))
((Comment "Could be used to infer the connection relationships between anatomical
           parts. Connections are considered at the deepest level of anatomical
           description.  Descriptions of connections can be made at a higher level
           by considering anatomical superparts.")))
   (SUBPARTS ((MEMBER.OF #[Unit: ANATOMICAL.PARTS AB-PAIN]))
     ((Comment "Units which are components of this unit.")))
   (SUPERPART ((MEMBER.OF #[Unit: ANATOMICAL.PARTS AB-PAIN]))
             ((Comment "Units for which this anatomical part is a component.")))))
```

```
(EXTRAHEPATIC.DUCTS
  (ANATOMICAL.PARTS )
  ((SUBPARTS (#[Unit: COMMON.BILE.DUCT AB-PAIN] #[Unit: CYSTIC.DUCT AB-PAIN]
              #[Unit: COMMON.HEPATIC.DUCT AB-PAIN]))))

(GALLBLADDER
  (ALIMENTARY.CANAL ANATOMICAL.PARTS)
  ((DAILY.VOL.SECRETION (.5))
   (NORMAL.CL.CONC (100))
   (NORMAL.HCO3.CONC (40))
   (NORMAL.K.CONC (5))
   (NORMAL.NA.CONC (145))
   (PREDECESSOR.SECRETION (#[Unit: STOMACH AB-PAIN]))
   (SUBPARTS ( #[Unit: GALLBLADDER.EX.HP AB-PAIN] #[Unit: HARTMANNS.POUCH AB-PAIN]))
   (SUCCESSOR.SECRETION (#[Unit: PANCREAS AB-PAIN]))))

(GALLBLADDER.EX.HP
  (ANATOMICAL.PARTS)
  ((ABNORMALITY.PRESENT ((AVUNITS ((#[Unit: CHANGE.ANATOMICAL.PATHOLOGY.MONITOR
AB-PAIN] ALL NIL)))))
   (CONNECTED.TO (#[Unit: HARTMANNS.POUCH AB-PAIN]))
   (SUPERPART (#[Unit: GALLBLADDER AB-PAIN]))))

(HARTMANNS.POUCH
  (ANATOMICAL.PARTS)
  ((ABNORMALITY.PRESENT ((AVUNITS ((#[Unit: CHANGE.ANATOMICAL.PATHOLOGY.MONITOR
AB-PAIN] ALL NIL)))))
   (CONNECTED.TO (#[Unit: CYSTIC.DUCT AB-PAIN] #[Unit: GALLBLADDER.EX.HP AB-PAIN]))
   (SUPERPART (#[Unit: GALLBLADDER AB-PAIN]))))

(CYSTIC.DUCT
  (ANATOMICAL.PARTS (CLASSES GENERICUNITS))
  ((ABNORMALITY.PRESENT ((AVUNITS ((#[Unit: CHANGE.ANATOMICAL.PATHOLOGY.MONITOR
AB-PAIN] ALL NIL)))))
   (CONNECTED.TO (#[Unit: HARTMANNS.POUCH AB-PAIN] #[Unit: COMMON.BILE.DUCT AB-PAIN]))
   (SUPERPART (#[Unit: EXTRAHEPATIC.DUCTS AB-PAIN]))))
```

A.5 Units From The Sign And Symptom Complexes Hierarchy

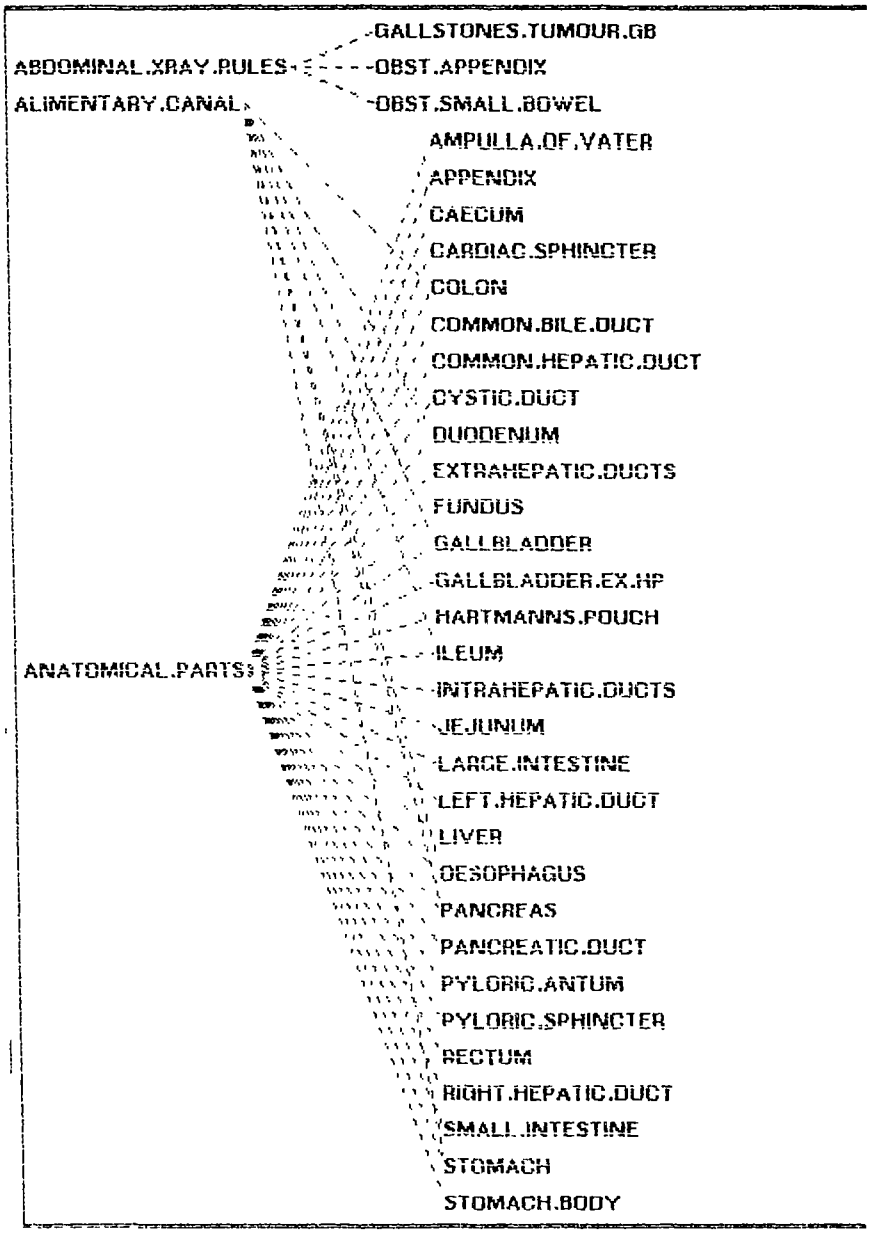
```
(SYMPTOM.SIGN.COMPLEXES
  "Description of the signs and symptoms associated with disease complexes"
  ((CLINICAL.SIGNS.SYMPTOMS (#[Unit: LIST KEEDATATYPES])
    ((Cardinality.Min (1))
     (Cardinality.Max (1))))
   (ETIOLOGY (#[Unit: LIST KEEDATATYPES])
    ((AVUNITS ((#[Unit: MONITOR.COMPLEX.ETIOLOGY AB-PAIN] ALL NIL)))
```

```
      (Comment "Pathological cause of complex.")))
    (INITIALISATION ((LAMBDA (THISUNIT SEVERITY ABNORMALITY SITE)
     PROGRAMBODY))
    METHOD
    (#[Unit: METHOD KEEDATATYPES])
    ((Comment "Sets up complex.")))
    (PROGRESSION.STATE NIL NIL ((ONE.OF ACTIVE.PROG ACTIVE.NO.PROG))
        (Comment
"If cause of complex is still active then this slot will take value T.  If this
 complex is active but the cause has been resolved then this slot will take value
 NIL."
)))
    (REGRESSIVE.PROCESSES ((Comment
"Processes associated with a complex whose cause has been resolved.")))
    (RELATED.SUBPROCESSES ((Comment
         "Processes which are secondary to that underlying the complex.")))
    (UNDERLYING.PATHOPHYSIOLOGICAL.PROCESS ((Comment
         "Process which results directly from the abnormal pathology.")))
    )
   ((ETIOLOGY)
    ))

(JAUNDICE
  (SYMPTOM.SIGN.COMPLEXES)
  ((CLINICAL.SIGNS.SYMPTOMS ((CONJUNCTIVAE.COLOUR ITCHING STOOL.COLOUR URINE.COLOUR)))
   (CONJUNCTIVAE.COLOUR (YELLOW)
(#[Unit: WORD KEEDATATYPES]))
    (ITCHING)
    (STOOL.COLOUR (#[Unit: WORD KEEDATATYPES]))
    (URINE.COLOUR (#[Unit: WORD KEEDATATYPES]))))
```

Appendix B

Graphs Of Prototype System's Knowledge Base



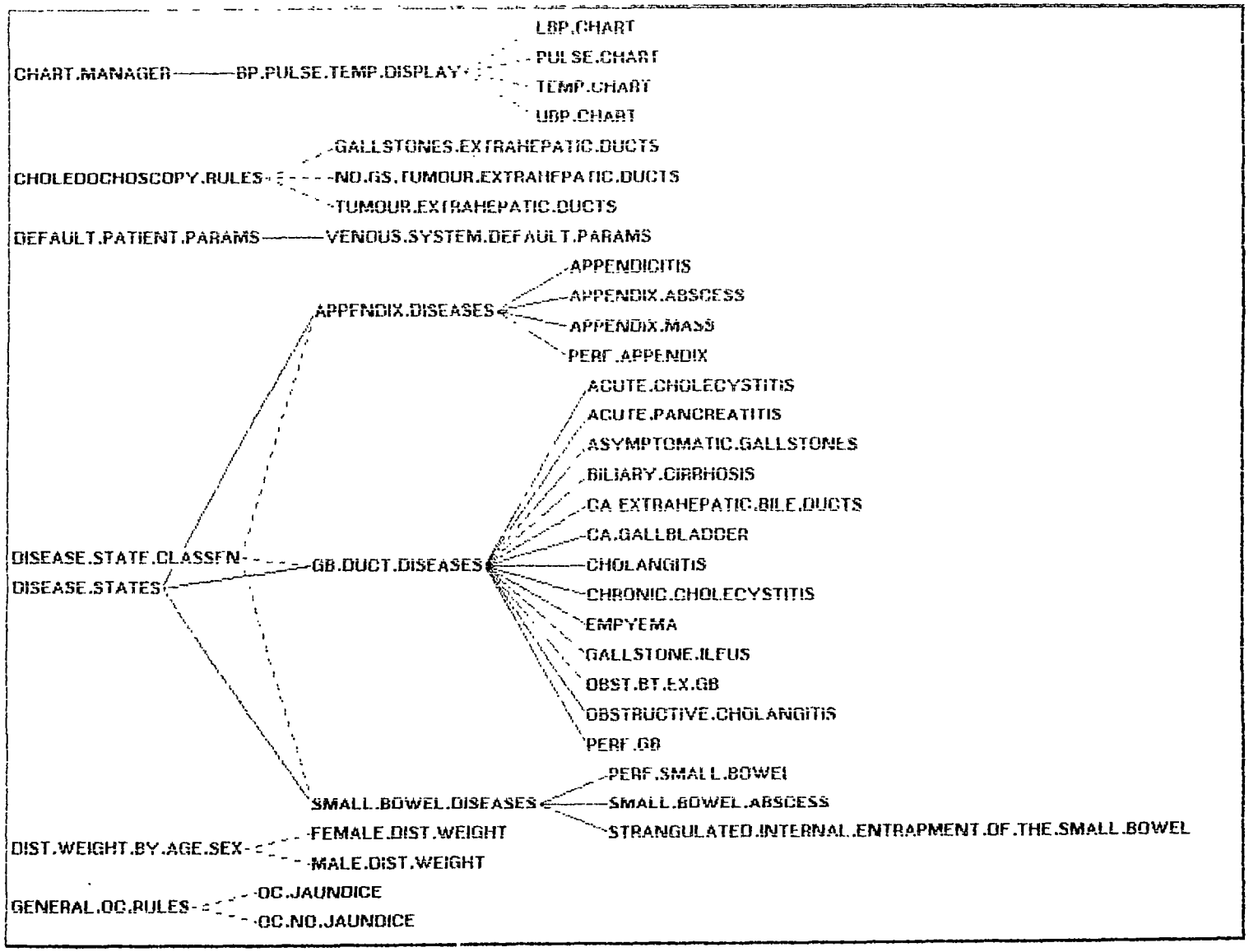
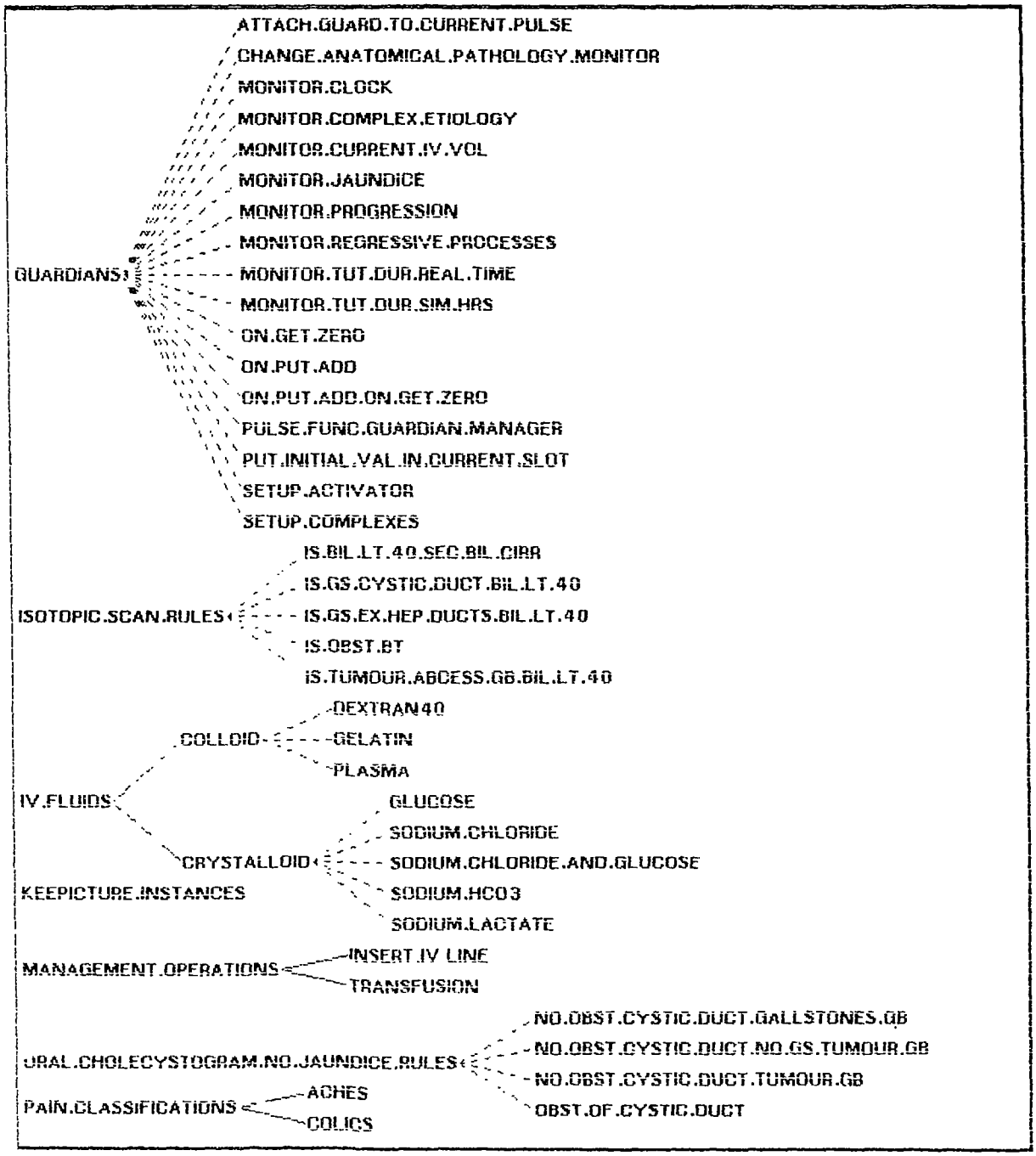
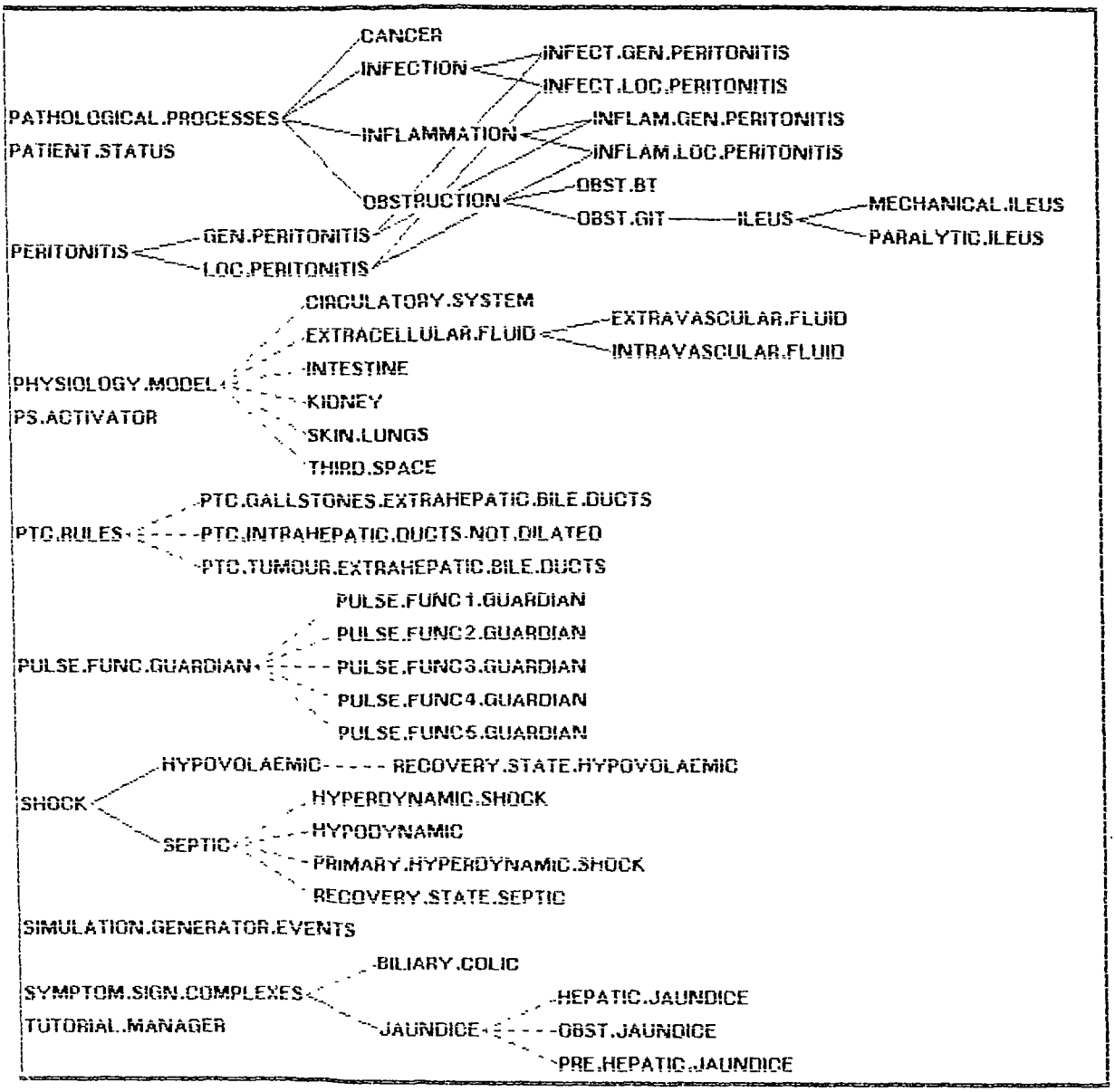
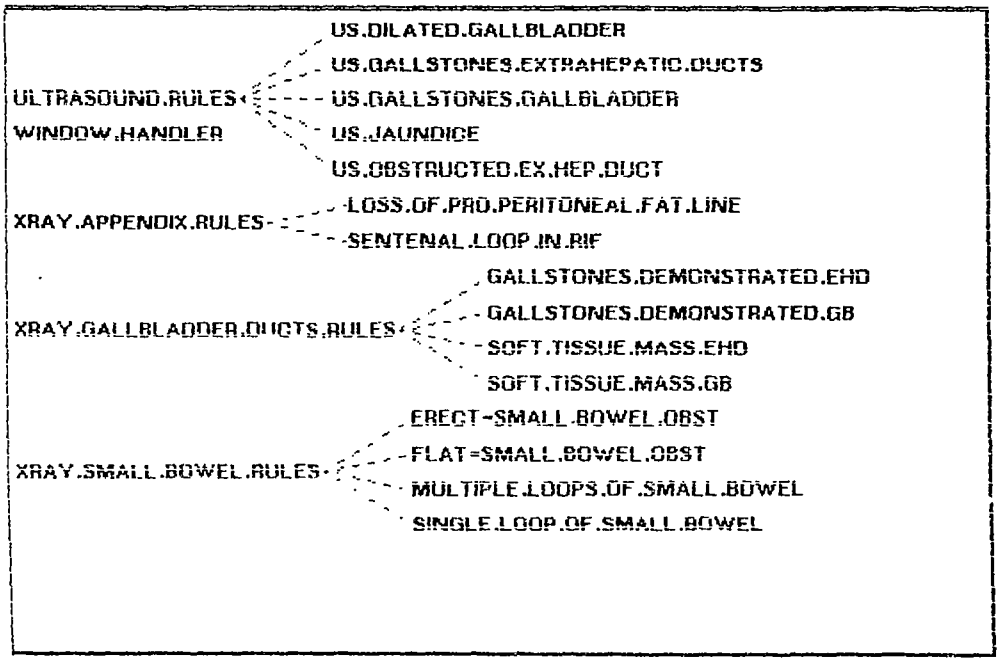


CHART.MANAGER
BP.PULSE.TEMP.DISPLAY
LBP.CHART
PULSE.CHART
TEMP.CHART
UBP.CHART
CHOLEDOCHOSCOPY.RULES
GALLSTONES.EXTRAHEPATIC.DUCTS
NO.GS.TUMOUR.EXTRAHEPATIC.DUCTS
TUMOUR.EXTRAHEPATIC.DUCTS
DEFAULT.PATIENT.PARAMS
VENOUS.SYSTEM.DEFAULT.PARAMS
APPENDIX.DISEASES
APPENDICITIS
APPENDIX.ABSCESS
APPENDIX.MASS
PERF.APPENDIX
DISEASE.STATE.CLASSFN
DISEASE.STATES
GB.DUCT.DISEASES
ACUTE.CHOLECYSTITIS
ACUTE.PANCREATITIS
ASYMPTOMATIC.GALLSTONES
BILIARY.CIRRHOSIS
CA.EXTRAHEPATIC.BILE.DUCTS
CA.GALLBLADDER
CHOLANGITIS
CHRONIC.CHOLECYSTITIS
EMPYEMA
GALLSTONE.ILEUS
OBST.BT.EX.GB
OBSTRUCTIVE.CHOLANGITIS
PERF.GB
SMALL.BOWEL.DISEASES
PERF.SMALL.BOWEL
SMALL.BOWEL.ABSCESS
STRANGULATED.INTERNAL.ENTRAPMENT.OF.THE.SMALL.BOWEL
DIST.WEIGHT.BY.AGE.SEX
FEMALE.DIST.WEIGHT
MALE.DIST.WEIGHT
GENERAL.OC.RULES
OC.JAUNDICE
OC.NO.JAUNDICE


GUARDIANS
ATTACH.GUARD.TO.CURRENT.PULSE
CHANGE.ANATOMICAL.PATHOLOGY.MONITOR
MONITOR.CLOCK
MONITOR.COMPLEX.ETIOLOGY
MONITOR.CURRENT.IV.VOL
MONITOR.JAUNDICE
MONITOR.PROGRESSION
MONITOR.REGRESSIVE.PROCESSES
MONITOR.TUT.DUR.REAL.TIME
MONITOR.TUT.DUR.SIM.HRS
ON.GET.ZERO
ON.PUT.ADD
ON.PUT.ADD.ON.GET.ZERO
PULSE.FUNC.GUARDIAN.MANAGER
PUT.INITIAL.VAL.IN.CURRENT.SLOT
SETUP.ACTIVATOR
SETUP.COMPLEXES
ISOTOPIC.SCAN.RULES
IS.BIL.LT.40.SEC.BIL.CIRR
IS.GS.CYSTIC.DUCT.BIL.LT.40
IS.GS.EX.HEP.DUCTS.BIL.LT.40
IS.OBST.BT
IS.TUMOUR.ABCESS.GB.BIL.LT.40
IV.FLUIDS
COLLOID
DEXTRAN40
GELATIN
PLASMA
CRYSTALLOID
GLUCOSE
SODIUM.CHLORIDE
SODIUM.CHLORIDE.AND.GLUCOSE
SODIUM.HCO3
SODIUM.LACTATE
KEEPICTURE.INSTANCES
MANAGEMENT.OPERATIONS
INSERT.IV.LINE
TRANSFUSION
ORAL.CHOLECYSTOGRAM.NO.JAUNDICE.RULES
NO.OBST.CYSTIC.DUCT.GALLSTONES.GB
NO.OBST.CYSTIC.DUCT.NO.GS.TUMOUR.GB
NO.OBST.CYSTIC.DUCT.TUMOUR.GB
OBST.OF.CYSTIC.DUCT
PAIN.CLASSIFICATIONS
ACHES
COLICS


CANCER
INFECTION
INFECT.GEN.PERITONITIS
INFECT.LOC.PERITONITIS
PATHOLOGICAL.PROCESSES
INFLAMMATION
INFLAM.GEN.PERITONITIS
PATIENT.STATUS
INFLAM.LOC.PERITONITIS
OBSTRUCTION
OBST.BT
OBST.GIT
ILEUS
MECHANICAL.ILEUS
PARALYTIC.ILEUS
GEN.PERITONITIS
PERITONITIS
LOC.PERITONITIS
CIRCULATORY.SYSTEM
EXTRACELLULAR.FLUID
EXTRAVASCULAR.FLUID
INTRAVASCULAR.FLUID
INTESTINE
PHYSIOLOGY.MODEL
KIDNEY
PS.ACTIVATOR
SKIN.LUNGS
THIRD.SPACE
PTC.GALLSTONES.EXTRAHEPATIC.BILE.DUCTS
PTC.RULES
PTC.INTRAHEPATIC.DUCTS.NOT.DILATED
PTC.TUMOUR.EXTRAHEPATIC.BILE.DUCTS
PULSE.FUNC1.GUARDIAN
PULSE.FUNC2.GUARDIAN
PULSE.FUNC.GUARDIAN
PULSE.FUNC3.GUARDIAN
PULSE.FUNC4.GUARDIAN
PULSE.FUNC5.GUARDIAN
HYPOVOLAEMIC
RECOVERY.STATE.HYPOVOLAEMIC
SHOCK
HYPERDYNAMIC.SHOCK
HYPODYNAMIC
SEPTIC
PRIMARY.HYPERDYNAMIC.SHOCK
RECOVERY.STATE.SEPTIC
SIMULATION.GENERATOR.EVENTS
BILIARY.COLIC
SYMPTOM.SIGN.COMPLEXES
HEPATIC.JAUNDICE
TUTORIAL.MANAGER
JAUNDICE
OBST.JAUNDICE
PRE.HEPATIC.JAUNDICE


US.DILATED.GALLBLADDER
US.GALLSTONES.EXTRAHEPATIC.DUCTS
ULTRASOUND.RULES
US.GALLSTONES.GALLBLADDER
WINDOW.HANDLER
US.JAUNDICE
US.OBSTRUCTED.EX.HEP.DUCT
XRAY.APPENDIX.RULES
LOSS.OF.PRO.PERITONEAL.FAT.LINE
SENTENAL.LOOP.IN.RIF
GALLSTONES.DEMONSTRATED.EHD
XRAY.GALLBLADDER.DUCTS.RULES
GALLSTONES.DEMONSTRATED.GB
SOFT.TISSUE.MASS.EHD
SOFT.TISSUE.MASS.GB
ERECT-SMALL.BOWEL.OBST
XRAY.SMALL.BOWEL.RULES
FLAT=SMALL.BOWEL.OBST
MULTIPLE.LOOPS.OF.SMALL.BOWEL
SINGLE.LOOP.OF.SMALL.BOWEL

Appendix C

Simulation Runs To Evaluate The Physiology Model

The physiology associated with infection progressing to localised and generalised peritonitis obtained from simulation runs of the BBC Basic program to evaluate the physiological model is as follows:

No Fluid Therapy Figures C.1 and C.2 show the extracellular fluid losses and haemodynamic response demonstrated by a 40 year old male with a normal body weight of 70 kg.

Fluid Therapy Administered Figures C.3 and C.4 show the extracellular losses and gains and the haemodynamic response displayed by a 40 year old male with a normal body weight of 70 kg. Crystalloid fluid is administered between hours 30 to 36, all other intravenous fluids being colloids.

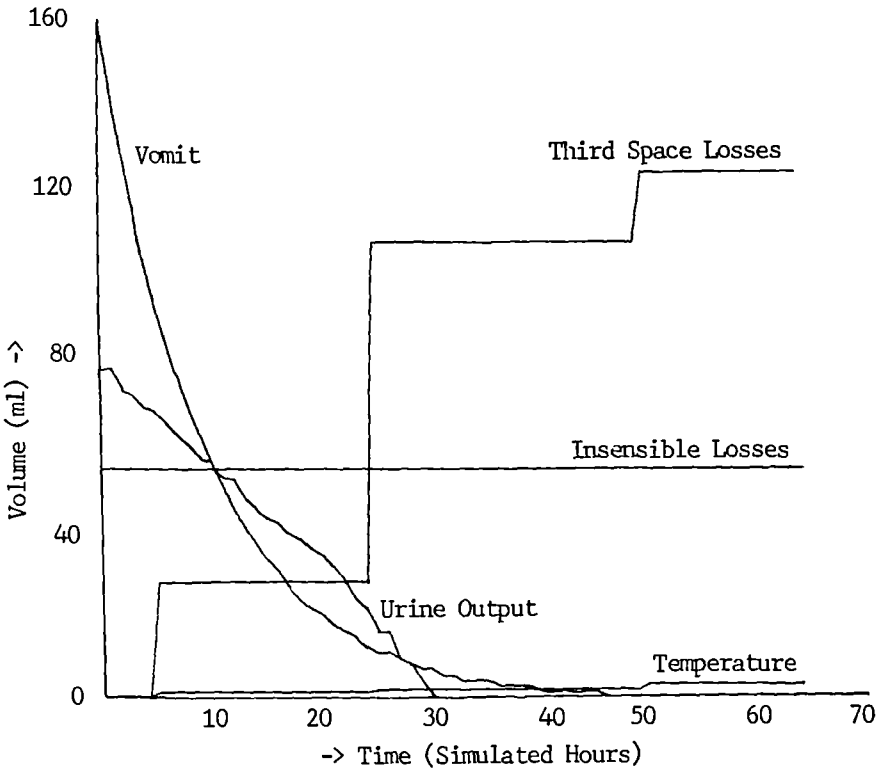


Figure C.1: Extracellular Fluid Losses Associated With The Pathological Progression Of Infection Through Localised To Generalised Peritonitis

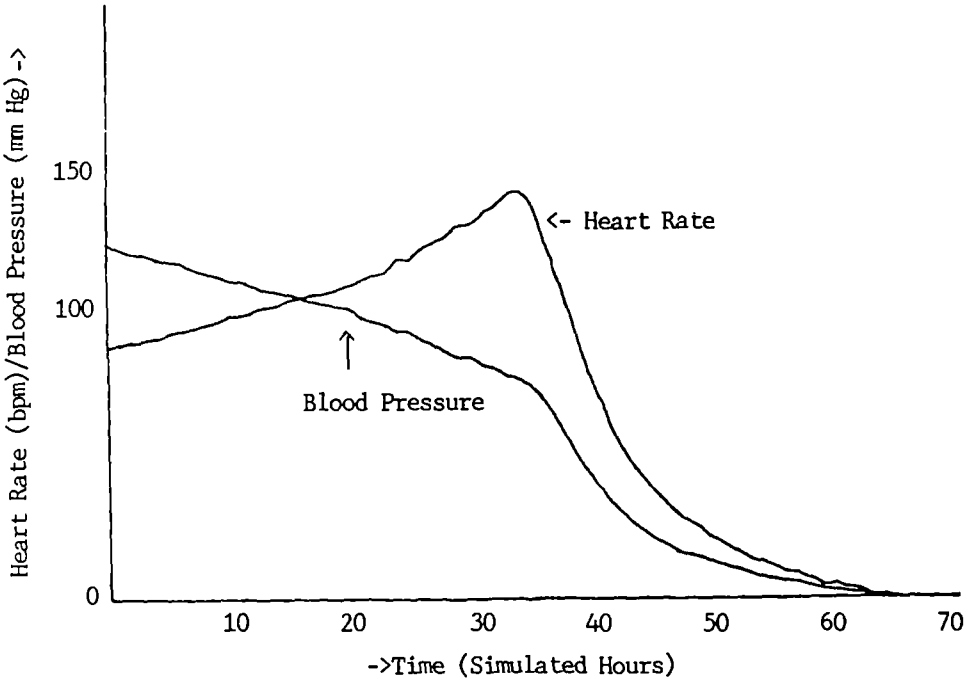


Figure C.2: Haemodynamic Changes Associated With The Pathological Progression Of Infection Through Localised To Generalised Peritonitis

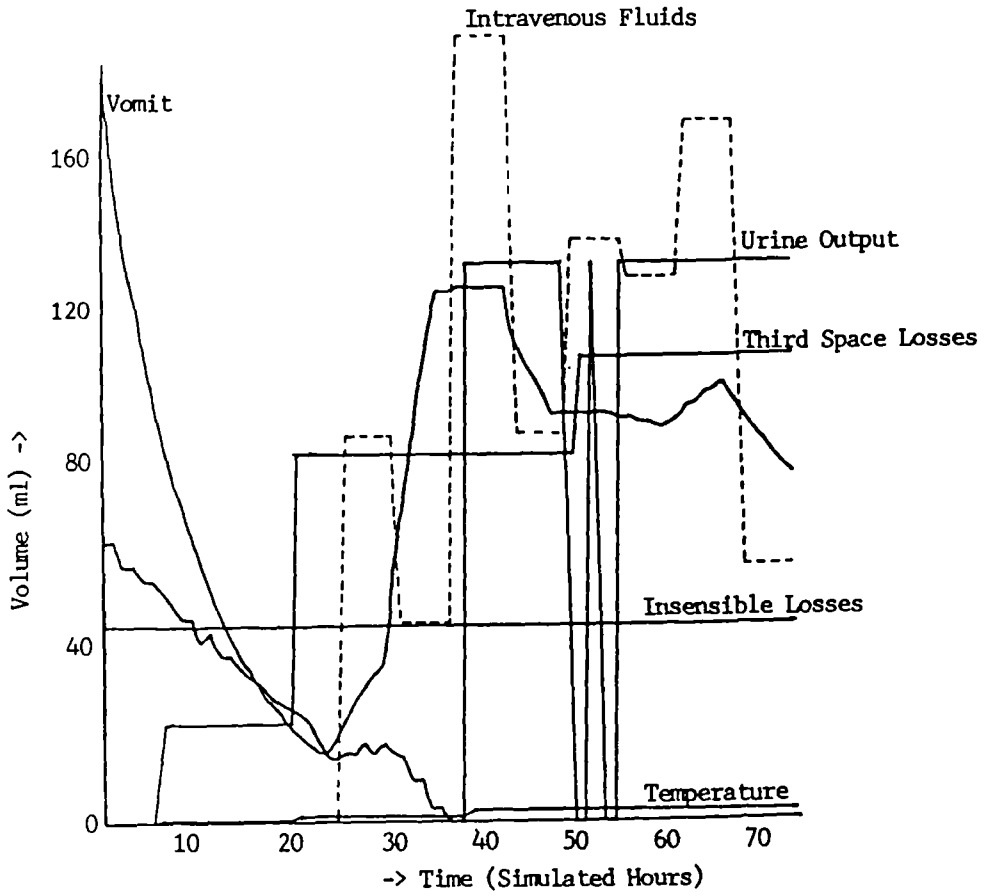


Figure C.3: Extracellular Fluid Losses Incurred By Physiological Processes And Intravenous Fluid Gains Associated With The Pathological Progression Of Infection Though Localised To Generalised Peritonitis

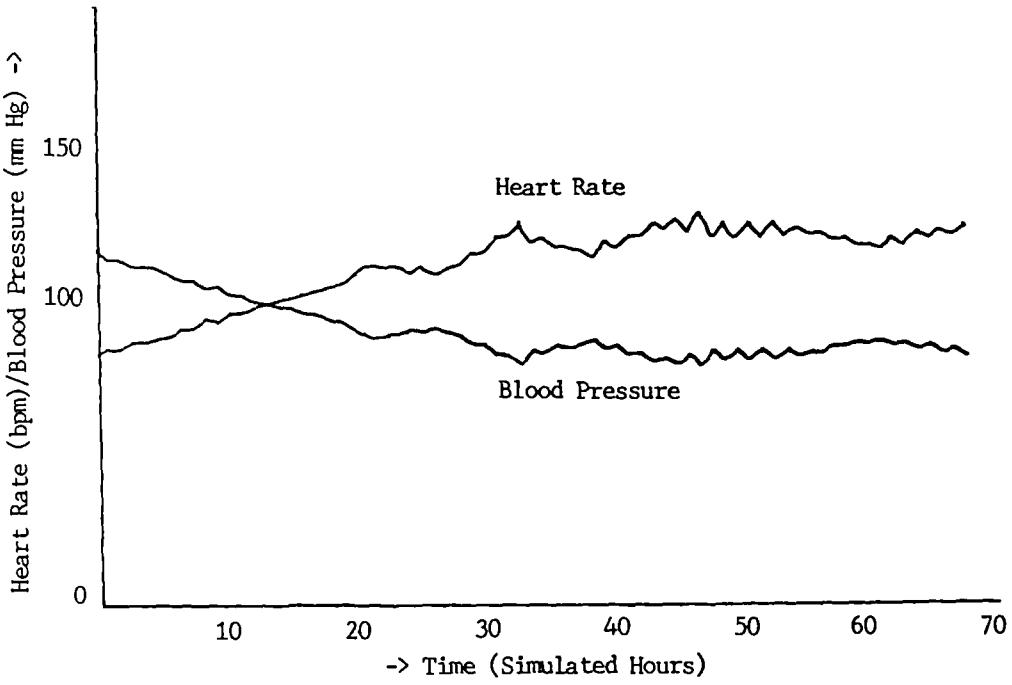


Figure C.4: Haemodynamic Changes Associated With The Pathological Progression Of Infection Through Localised To Generalised Peritonitis With Accompanying Fluid Therapy

Appendix D

Examples Of Rules Representing The Results Of Special Investigations

There follows the KEE units describing the SAME.WORLD.ACTION rules associated with the special investigation ultrasound. A graph of these units is given in appendix B.

```
(ULTRASOUND.RULES
"Investigation demonstrates gallstones, dilation of the gallbladder and ducts
 and texture changes of the liver."
   ((RES.ARRIVAL.TIME (#[Unit: NUMBER KEEDATATYPES])
     ((Comment
"If this special investigation has been requested, the time at which it will actually
 be performed is held in this slot.   After this time the results will be available."
      )))
   (WAIT.TIME (1)
      (#[Unit: NUMBER KEEDATATYPES])
      ((Comment
"Wait time (in simulated hours) for the results of a special investigation to arrive,
 having been requested."
)))
   ))

(US.OBSTRUCTED.EX.HEP.DUCT
  (ULTRASOUND.RULES)
  ((EXTERNAL.FORM
   ((IF (THE ACTIVE.COMPLEXES OF SIMULATION.GENERATOR.EVENTS IS OBST.JAUNDICE)
       (THE ABNORMALITY.PRESENT OF EXTRAHEPATIC.DUCTS IS ?X)
       (LISP (EQMEMB (CAR ?X) (QUOTE (TUMOUR GALLSTONES))))
                    THEN
       (LISP (ADD.VALUE (QUOTE SPECIAL.INVESTIGATIONS.BS)
```

```
                                        (QUOTE RES.TO.BE.PRINTED)
(APPEND (QUOTE (Dilation of))
(APPEND.PERIOD (LIST.SUBPARTS.WITH.ABNORMALITY
 (QUOTE EXTRAHEPATIC.DUCTS)
 (CAR ?X))))
(QUOTE OWN)
*WORLD*)))))
   (RULE.TYPE (SAME.WORLD.ACTION))
   ))

(US.JAUNDICE
  (ULTRASOUND.RULES)
  ((EXTERNAL.FORM
 ((IF (THE ACTIVE.COMPLEXES OF SIMULATION.GENERATOR.EVENTS IS OBST.JAUNDICE)
                        THEN
       (LISP (ADD.VALUE (QUOTE SPECIAL.INVESTIGATIONS.BS)
(QUOTE RES.TO.BE.PRINTED)
(QUOTE (Visualisation of dilated intrahepatic bile ducts.))
(QUOTE OWN)
*WORLD*)))))
   (RULE.TYPE (SAME.WORLD.ACTION))
   ))

(US.DILATED.GALLBLADDER
  (ULTRASOUND.RULES)
  ((EXTERNAL.FORM
 ((IF (LISP (UNIT.DESCENDANT.P *PATH.STATE* (QUOTE EMPYEMA)
(QUOTE MEMBER)))
       THEN
       (LISP (PRINTOUT *TTYDISPLAY.W* "Dilation of the gallbladder." .SKIP 2)))))
   (RULE.TYPE (SAME.WORLD.ACTION))
   ))


(US.GALLSTONES.EXTRAHEPATIC.DUCTS
  (ULTRASOUND.RULES)
  ((EXTERNAL.FORM
((IF (THE ABNORMALITY.PRESENT OF EXTRAHEPATIC.DUCTS IS ?X)
    (EQUAL (CAR ?X) (QUOTE GALLSTONES))
THEN
   (LISP (ADD.VALUE (QUOTE SPECIAL.INVESTIGATIONS.BS) (QUOTE RES.TO.BE.PRINTED)
(APPEND (QUOTE (Gallstones demonstrated in))
(APPEND.PERIOD (LIST.SUBPARTS.WITH.ABNORMALITY
         (QUOTE EXTRAHEPATIC.DUCTS) (QUOTE GALLSTONES))))
         (QUOTE OWN) *WORLD*)))))
   (RULE.TYPE (SAME.WORLD.ACTION))))
```

```
(US.GALLSTONES.GALLBLADDER
  (ULTRASOUND.RULES)
  ((EXTERNAL.FORM
((IF (THE ABNORMALITY.PRESENT OF GALLBLADDER IS ?X)
    (EQUAL (CAR ?X) (QUOTE GALLSTONES))
  THEN
    (LISP (ADD.VALUE (QUOTE SPECIAL.INVESTIGATIONS.BS) (QUOTE RES.TO.BE.PRINTED)
(APPEND (QUOTE (Gallstones demonstrated in))
(APPEND.PERIOD (LIST.SUBPARTS.WITH.ABNORMALITY
 (QUOTE GALLBLADDER)
 (QUOTE GALLSTONES))))
(QUOTE OWN)
     *WORLD*)))))
    (RULE.TYPE (SAME.WORLD.ACTION))
   ))
```