# K9ER (CANINE EMERGENCY ROOM): A VETERINARY PRACTICE SIMULATOR BASED ON THE INTEGRATION OF AN EXPERT SYSTEM AND A PHYSIOLOGICAL SIMULATION

by

YUNXIU XU

(Under the Direction of DONALD NUTE)

## ABSTRACT

The K9ER system was designed to provide veterinary students the opportunity to learn and practice the diagnostic and treatment processes in an emergency room. It can be operated on any personal computer with Windows95 and later. This program was written in PROLOG and C++, and was developed in the LPA WIN-PROLOG programming environment. K9ER was developed based on the integration of an expert system and a physiological simulator, QCP. QCP provides all the physiological data to display the patient responses to the treatments. The expert system consists of three parts: the knowledge base which holds the expert knowledge; the inference engine which handles the knowledge base, knowledge source, and information provided by the user to reach its conclusions; and the user interface which communicates with the user and translates the information from the user to the representations that the interface engine can understand. K9ER simulates the clinical processes, gives the realistic real-time records, evaluates the user activities, and provides the expert's suggestions. This combination of simulator and expert system is an ideal tool for interactive learning.

INDEX WORDS: Simulator, Veterinary Medicine, Clinical Processes, Emergency Room, Expert System, Physiological Simulation

# K9ER (CANINE EMERGENCY ROOM): A VETERINARY PRACTICE SIMULATOR BASED ON THE INTEGRATION OF AN EXPERT SYSTEM AND A PHYSIOLOGICAL SIMULATION

by

YUNXIU XU

BS, Anhui Normal University, China, 1986

MS, Chinese Academy of Sciences, China, 1989

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2002

# K9ER (CANINE EMERGENCY ROOM): A VETERINARY PRACTICE SIMULATOR BASED ON THE INTEGRATION OF AN EXPERT SYSTEM AND A PHYSIOLOGICAL SIMULATION

by

YUNXIU XU

| | |
|---|---|
| Major Professor: | Donald Nute |
| Committee: | Scott Brown |
| | Charles Cross |

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2002

## ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my major professor, Dr. Donald Nute, for his invaluable guidance, constant support and great patience throughout my studies at the AI Center, the University of Georgia. Without his sage advice, this work would have not been completed.

I would like to express my sincere appreciation to Dr. Scott Brown, the domain expert of this project and also a member of my thesis committee, for his sharing his valuable knowledge with me. He was always patient and helpful in the process of the knowledge acquisition.

My sincere thanks as well go to Dr. Charles Cross for his time and support and being a member of my thesis committee.

I would like to thank Tom Coleman for his great help in the development of this thesis research.

I would like to thank Jing Wang, Lei Wu, Fred Maine, and many other students at AI Center. Their friendship has made my stay at Athens very pleasurable.

Finally, I thank my husband Kai Wang and my son George Wang for their love and concerns, for their continuous support and encouragement.

## TABLE OF CONTENTS

Page

## LIST OF FIGURES

Page

# CHAPTER 1

# INTRODUCTION

The K9ER (Canine Emergency Room) system is designed for veterinary students to learn the diagnostic and treatment processes of an emergency room. This system consists of a physiological simulation and an expert system. The domain of K9ER focuses mainly on the physiological changes of mammals, especially dogs and cats, during the diagnostic and treatment processes. Dr. Scott Brown, a professor in the school of veterinary medicine at the University of Georgia, shared his expertise in veterinary physiology during the development of K9ER. In this chapter, I will describe the clinical processes of a veterinary emergency room, and outline the tasks performed by K9ER. Also the concept of expert systems will be examined, as an expert system is the main component of K9ER. Lastly I will review the literature related to the applications of simulations and expert systems in medicine. Chapter 2 will introduce the physiological simulation software QCP, another component of K9ER. Chapter 3 will focus on the implementation of K9ER. The last chapter will present conclusions concerning the effectiveness of the K9ER system, as well as a discussion of ways the software could be developed further.

Usually, in a veterinary clinical emergency, apparatus are used at first to obtain physiological data for a patient animal. Then appropriate treatments can be commenced. If inappropriate treatment is given, the animal's life may be endangered. This process of

physiological data collection can take several hours in a veterinary emergency. The cost for this is usually several hundred dollars or more.

Using K9ER, after loading a case file, the user can simulate every available diagnostic test, and choose appropriate treatments to deal with the patient. A report of results will also be given. Using this software, one can learn and practice the processes in the veterinary emergency room at any time and at any place where a computer is available. The student can experiment with different treatments without fear of harming an actual patient. This method requires little or no expenditure and is a more efficient use of time.

### 1.1 **The procedure in the emergency room**

Animals that are sent to the emergency room can generally be grouped into three categories: animals requiring extensive nursing care, animals that are physiologically stable but require intense monitoring or observation, and animals that are physiologically unstable and require constant veterinary critical care (Murtaugh, 1992).

Extensive nursing care mainly relies on calculating intake and output fluid measurements every few hours in a patient. The animals in care are stable but require a lot of nursing clinician time and effort.

Physiologically stable animals may need to be tested or observed to see whether complications from their disease may occur (e.g., seizure disorders.)

In K9ER, I focus my attention on the third group: those animals that are physiologically unstable. This group of animals demands constant care by the attending emergency clinician until the animal's physiological status becomes "stable". Frequent

physical examinations and serial monitoring of various parameters are necessary. Some of these patients become progressively more unstable with symptoms such as continual abdominal bleeding, and require changes in medical management or immediate surgical intervention.

In an emergency room, it is very important for a veterinary clinician to quickly make decisions according to the diagnostic results and to give appropriate treatments. The purpose of K9ER is to help veterinary students learn skills that are critical in successful emergency rooms.

In K9ER, I also focus on medical records. The medical record is a critically important link to the patient. Usually, medical records include the historical perspective on each patient and a chronology of the approach to the medical care which is administered to that patient, such as history, physical examinations, progress notes, laboratory reports, pathology reports, radiology reports, surgery reports, anesthesia reports and so on. In K9ER, the medical records are represented on the report screen. The user can save and print all these medical records.

The emergency room staff and equipment must perform essential emergency diagnostic procedures as well as therapeutic interventions (surgical or medical) during all hours of operation. The monitoring of arterial blood pressure, heart rate, body temperature, respiratory rate, and urinary output can be routinely performed in the veterinary critical care unit. Interpretation of the data generated gives clinicians valuable information about the patients and their diseases or injuries. It often helps clinicians choose appropriate treatments for critically ill patients and analyze the efficacy of

emergency therapy. In K9ER, the user can test these items by utilizing the **Diagnose** and **Treatment** functions.

Clinicians involved in emergency cases must realize that the diagnosis and treatment processes should maximize benefit, minimize complications, and limit unnecessary expense. The appropriate application of resources, treatment, and intervention is positive therapy. In K9ER, the user chooses a series of treatments for the patient; then the system will evaluate the outcome of these treatment regimens according to the patient's physiological status, the time spent, and the expenditure. Furthermore, the system will suggest an expert's treatment protocol.

### 1.2 The expert system in K9ER

K9ER is developed mainly based on an expert system. An expert system is a computer program that represents the knowledge of some special subject and performs reasoning over the representations of the knowledge in order to solve problems or give advice (Jackson, 1999). It simulates human reasoning about a problem domain, and focuses on imitating an expert's problem solving abilities. K9ER offers suggestions for good therapeutic protocols to veterinary students based on the emergency cases and the clinical procedure that the special expert provided.

An expert system contains three main modules: knowledge base, inference engine, and user interface. The knowledge base is usually represented in some special-purpose language and kept separate from the other parts of the program. Normally the inference engine is the code that implements the reasoning process. An expert system must also be able to explain its solutions or justify its reasoning. These functions can be performed in

natural language or graphics that are easily understood by the user. Expert systems have a wide range of users, and therefore should be designed in such a transparent way that the user can run the program easily. This crucial part is the user interface module.

To develop an expert system, the first thing one must do is acquire knowledge. Knowledge acquisition is "the transfer and transformation of potential problem-solving expertise from some knowledge source to a program" (Buchanan *et al*., 1983). This process requires a series of interviews between a knowledge engineer, who is usually a computer specialist, and a domain expert who is able to describe his expertise to some degree. To make the transfer facile, experts need to provide not only the facts or principles of a domain but also information relevant to the judgment or the ways to divide a hard problem into several small problems that can be solved independently.

After acquiring knowledge from the domain experts, the engineer needs to represent the knowledge in a special language that can communicate with other parts of the system. This kind of language normally can interpret a large body of useful information in an unambiguous way or has well-defined syntax and semantics to govern the expression of the knowledge. Among the several developed forms coding knowledge, logic programs are widely used in the world of expert systems (Kowalski, 1979). In chapter 3, I will discuss how I acquired the domain knowledge and how this knowledge is represented in K9ER.

Knowing when and how to use what one knows is also an important part of expertise. The design of expert systems involves how knowledge is accessed and applied during the search for a solution (Davis, 1980). The task of the inference engine is to interpret the knowledge stored in the knowledge base. It searches the contents of the

knowledge base and the data accumulated about the current problem and generates additional data and conclusions that are available to solve the problem (Gonzalez and Dankel, 1993). Suppose there are two columns of nodes, the left nodes representing all the signs, symptoms, characteristics, and features that are relative to the problems, and the right nodes representing the solutions to these problems. The inference engine implements the process that attempts to find connections between the features and the solutions. The inference engine might search from the features to the solutions (called forward reasoning), from the solutions to the features (backward reasoning), or from both ends simultaneously (bidirectional reasoning.) Which particular method should be used depends upon the characteristics of the problem domain and the reasoning of the expert.

An expert system is developed for users to solve their special problems. How to help a user understand and use the software is the key in the design of an expert system. To the end user, the system is simply a black box (see Figure 1.1.) The end user does not

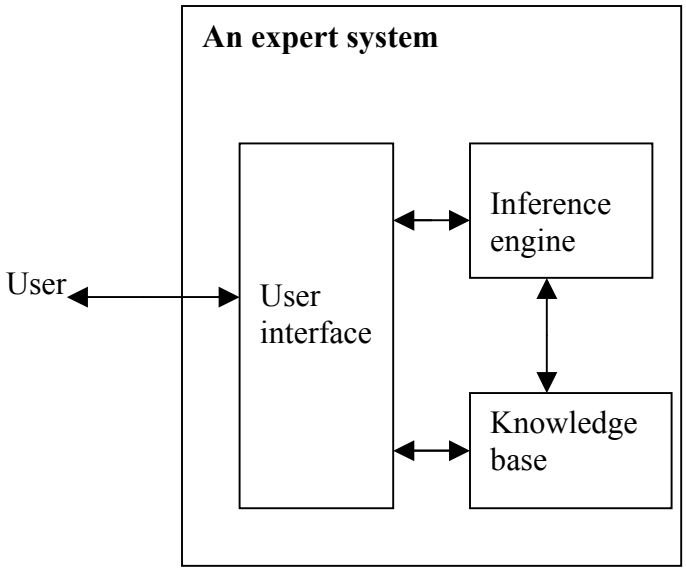



Figure 1.1: An expert system: a black box to end user

need to know how the system works. The user interface serves to provide him with a friendly means of communicating with the system. The interface can make the system post a question to the user, provide an explanation about a result, allow the user to query how a particular decision was made, and save or print the results. In K9ER, the interface includes a menu bar, buttons, and message windows for the purpose of communicating with the user.

I mentioned early in this section that logic programs are widely used in expert systems. PROLOG (which stands for PROgramming in LOGic), a logic programming language, was chosen as the primary development tool in K9ER, because it provides an automatic reasoning mechanism and a means of representing knowledge. PROLOG has a built–in inference engine that can automatically search the knowledge base in a depth-first fashion. PROLOG "represents knowledge in terms of procedure definitions, and reasoning becomes a simple process of calling the right procedure" (Convington, Nute, and Vellino, 1997). In other words, PROLOG represents the knowledge in "facts" and "rules". A fact is a true statement about the real world. For example: "grass is colored green" states a fact. In PROLOG, it is implemented as color(grass, green). A rule is the name for an implication in PROLOG. By applying rules, new facts are derived. The implication operator symbol in PROLOG is **":-"**. Thus, (A -> B) (i.e., A implies B) is represented as B :- A (i.e., B if A) in PROLOG.

K9ER was built using the LPA WIN-PROLOG environment because it allows rapid development of the user interface. The user interface is presented in windows. LPA WIN-PROLOG provides a series of predicates designed especially for creating windows and controls such as wdcreate, wccreate and window handlers. In addition, with LPA WIN-

PROLOG, any executable files can be loaded directly, which provides an environment for the communication between the expert system and the physiological simulation used by K9ER.

### 1.3 Relevant literature

Pallin and Kittell developed simulation techniques for emergency room (ER) processes. In this simulation, they focused on the quality of patient care -- the time that a patient stays in an ER ward. The purpose of this simulation is to improve scheduling practices, patient transportation systems under the fixed level of staffing, and facility resources (Pallin and Kittell, 1992). Based on this model, Duane Steward developed a veterinary practice simulator that included an expert system to process patient appointments (Steward *et al*., 1996). I will not describe this simulator in detail, because it is not used for learning and training. The emphasis of Steward's model is on the level of a patient's care, whereas the emphasis of K9ER is on the diagnosis and treatment processes in an ER. From this view in a search of the literature, it seemed that there was no simulation plus expert system for the clinical processes of an emergency room. There are, however, many simulation models and expert systems for veterinary medicine in general. Here, I would like to introduce a simulator consultant because it was intended for learning and training and it includes an expert system.

The Anesthesia Simulator Consultant (ASC) was designed to provide anesthesiologists the opportunity to practice the management of anesthesia in emergency situations. It consists of a graphical interface, physiological models, an automated record keeper and an expert system. With the graphical interface, the user administers the

general anesthetic: giving fluids and drugs, controlling the airway, and ventilating the patient. All these are controlled using the mouse. Physiological monitors display the state of the patient. Mathematical physiological models predict the patient's responses to the drugs, fluid administration, and ventilation. An automated record-keeping system prints a detailed summary of the patient's vital signs and all diagnostic and therapeutic interventions made during the simulated case. The expert system summarizes the simulated patient's preoperative problems and provides a suggested anesthetic-management plan. The coupling of the simulator, recorder, and expert system creates a unique self-study and evaluation environment (Schwid and O'Donnell, 1993). This model stimulated my interest in developing the K9ER system which combined a physiological simulation and an expert system for teaching the clinical process for the veterinary emergency room.

ASC and K9ER are both educational software. Although they have different knowledge domains, they share similar features: a graphical interface for user input, physiological models to predict the patient's responses, an automatic record system, and an expert system to interpret the patient's information. They are however two different systems. In K9ER, the record system is not a summary, but real-time, step-by-step recordings of every diagnosis and treatment. A distinguishing feature of K9ER is that the knowledge base in the expert system is not fixed, but can be expanded at any time. That means that a new case file can be added into the knowledge base after the expert system is built up. ASC does not state that the knowledge base can be expanded once the system is fixed.

# CHAPTER 2

# PHYSIOLOGICAL SIMULATION

In K9ER, QCP will be introduced as the knowledge source to provide the physiological data under normal or abnormal conditions during the ER process.

## 2.1 Introduction

QCP is a physiological simulation software system that was developed by Biological Simulators, Inc., and runs under Windows 3.1/95/98/NT. Originally, QCP focused on **q**uantitative **c**irculatory **p**hysiology. It still keeps its original name although its focus is expanded to include circulation, lungs, kidneys, body fluids, hormones, the nervous system, metabolism, acid-base balance, temperature regulation, anesthesia, pharmaceuticals, and more.

The QCP program presents a mathematical description of many important physiological functions in the human body. The user can create a disturbance, such as an arterial hemorrhage, and then observe the body's physiological reactions over time as the various physiological systems contribute to and respond to the disturbance.

The QCP program may also simulate interventions. In the case of an arterial hemorrhage, the user can compensate for the blood loss by implementing a transfusion. This kind of simulation interprets the physiological changes dynamically.

In the QCP program, simulations are controllable. The user can choose Go, Stop, Continue, or Restart on the menu bar to advance the simulation over a time interval ranging from 1 second to 1 month, stop the simulation at any time, continue the simulation then, or restart the simulation (see Figure 2.1.) Also, the user can interactively set the values of parameters. For example, to simulating an arterial hemorrhage, the user can set the volume of blood loss and the duration of bleeding. The QCP program can record all the physiological values of every variable at every time point the user sets.

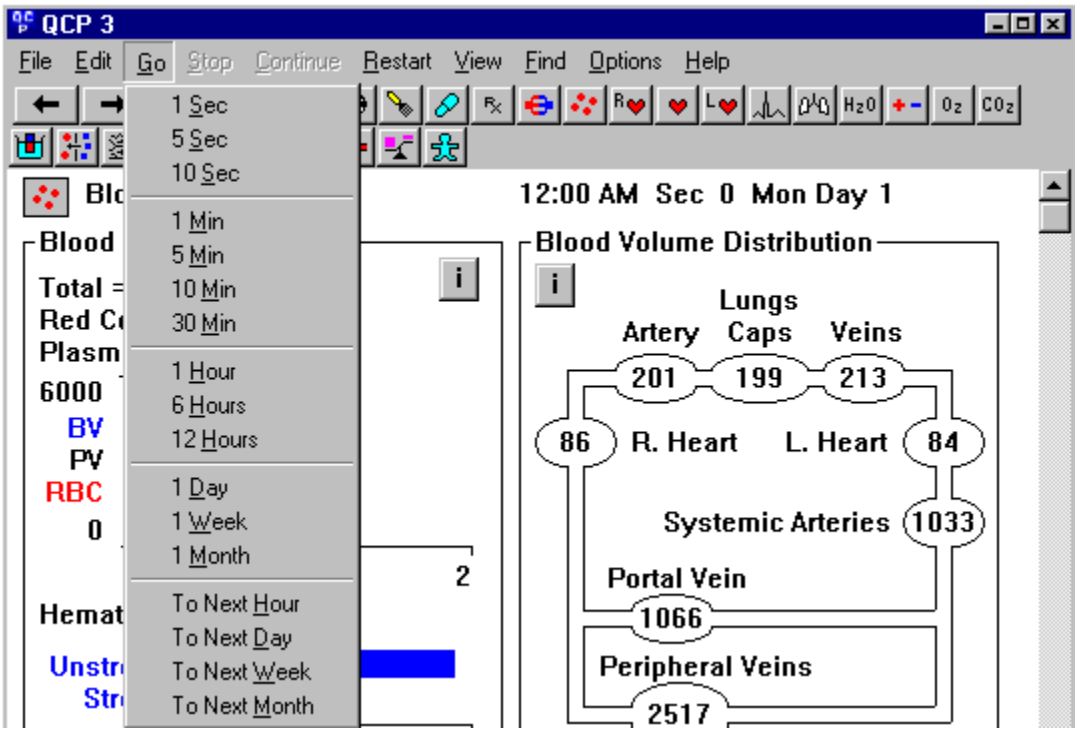



Figure 2.1: Choosing menu bar to control the simulations

In the QCP program, a simulation typically includes a control period for a series of interventions. For example, the user can observe the simulated results of 5-minutes of

hemorrhaging followed by a 5-minute transfusion (see Figure 2.2). Using the features of the QCP program, the processes of diagnosis and treatment can be simulated.

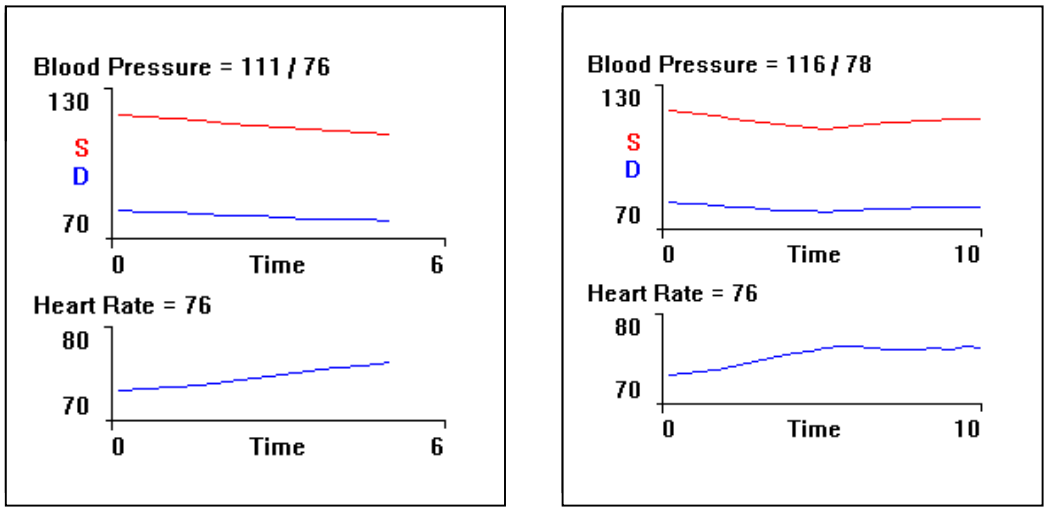



Figure 2.2: 5-minutes of hemorrhaging (left) followed by a 5-minute transfusion (right)

## 2.2 Physiological simulation in K9ER

Physiology is a basic science for medicine (Cunningham, 1997). It is the study of the normal functions of the body while medicine is the study of abnormal functions of the body. If one wants to know the mechanism of disease, he must understand the normal physiology. There is no doubt that the body's physiological situation is the sign of health and must be under observation in the process of diagnosis and treatment. This is why physiological simulation is necessary in K9ER.

Although the QCP program simulates human physiology, it still can be applied in K9ER. The K9ER system is designed for the domain of mammals, such as dogs, cats, and

horses. Humans are a specific kind of mammal. They share the same physiological changes. The difference is that rates or other aspects of some physiological changes must be scaled for body size. These differences can be presented by the kind of formula that will be illustrated in chapter 3. A simulator like QCP is very important for K9ER since our goal is to simulate physiological responses to the treatments the user administers. In the next section, I will discuss the way the link between QCP and K9ER was implemented.

### 2.3 Linkage of QCP and K9ER

In K9ER, QCP functions as a knowledge source. That means that when simulation data are needed, K9ER will send the message to QCP, and then get the data from QCP. QCP is like a black box to K9ER, and also to the user. But how does K9ER communicate with QCP while its window is hidden?

QCP was designed so that it could be run in the background and communicate with a client program using a 16-bit data structure, restricting its use in this role to a 16-bit environment (Windows 3.x) (Coleman, 1999). K9ER is developed for a 32-bit environment (Windows 95 and later.) A 32-bit version of Windows has more virtual memory than a 16-bit version of Windows. There is no problem sending a message using a 16-bit data structure to a 32-bit application in Windows 95 or later, but part of the data may be lost when sending a message to a 16-bit application using a 32-bit data structure (Boyce, 1995). Thus, K9ER cannot communicate with QCP directly. It is obvious that a bridge is needed. The bridge must not only communicate with QCP under a 16-bit environment but also be run in a 32-bit version of Windows when K9ER gives a call to it.

To bridge the gap between QCP and K9ER, I developed the C++ program BQK under Borland C++, version 3.1 (a 16-bit environment) so that BQK can communicate with QCP and also be loaded directly by K9ER. When K9ER needs physiological data from QCP, the PROLOG procedure of K9ER will generate an ASCII file that contains the commands to QCP, and then load the BQK executable file. Then BQK reads the commands in the ASCII file, launches the QCP executable file, sends the commands to QCP by a 16-bit data structure, gets outputs back from QCP, and writes these outputs to another ASCII file that the PROLOG procedure can read (see Figure 2.3.) Thus, through BQK, messages can be transferred between the PROLOG procedure of K9ER in a 32-bit environment and QCP in a 16-bit environment.

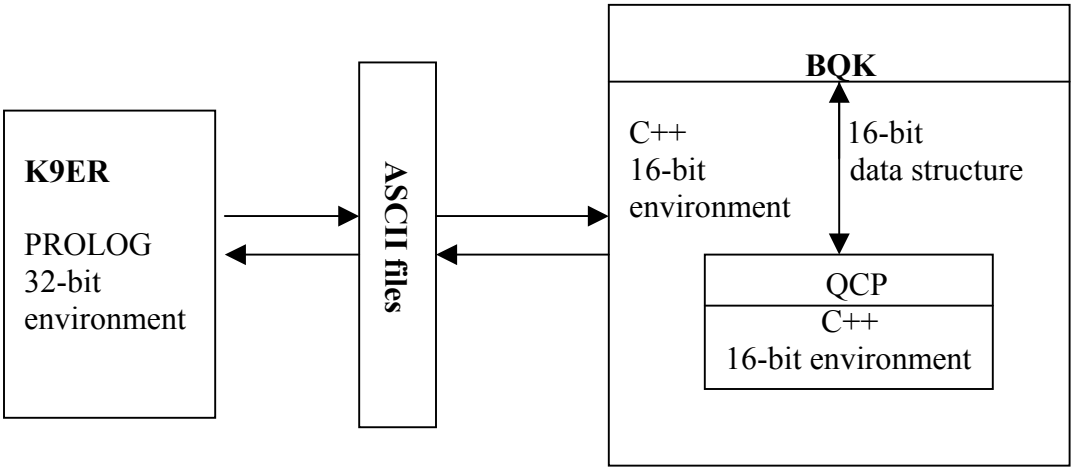



Figure 2.3: BQK transferring messages between K9ER and QCP

# CHAPTER 3

# IMPLEMENTATION

Usually a case occurs in an emergency room like this: a dog was hit by a car and has been bleeding for twenty minutes before coming into the emergency room. Clinicians will first give several diagnostic tests such as taking temperature and blood pressure and so on. After this first data collection, a series of treatments will be initiated such as whole blood transfusion. During the treatment procedure, the clinicians must keep using diagnostic tests to monitor the physiological status of the animal.

The software K9ER simulates the whole emergency room experience and provides veterinary students with a simple, swift, and convenient learning and practicing opportunity.

K9ER was developed based on the integration of an expert system and a physiological simulator, QCP. The expert system in K9ER consists of three parts: the knowledge base which holds the expert knowledge; the inference engine which handles the knowledge base, knowledge source, and information provided by the user to reach its conclusions; and the user interface which communicates with the user and translates the information from the user to the representations that the interface engine can understand.

The structure of K9ER is presented in Figure 3.1. The knowledge base of K9ER includes patient cases and clinical procedures. The user interface is graphical (a GUI.)

The inference engine handles the communication among the modules in K9ER. The simulator in K9ER is in charge of providing the physiological data.

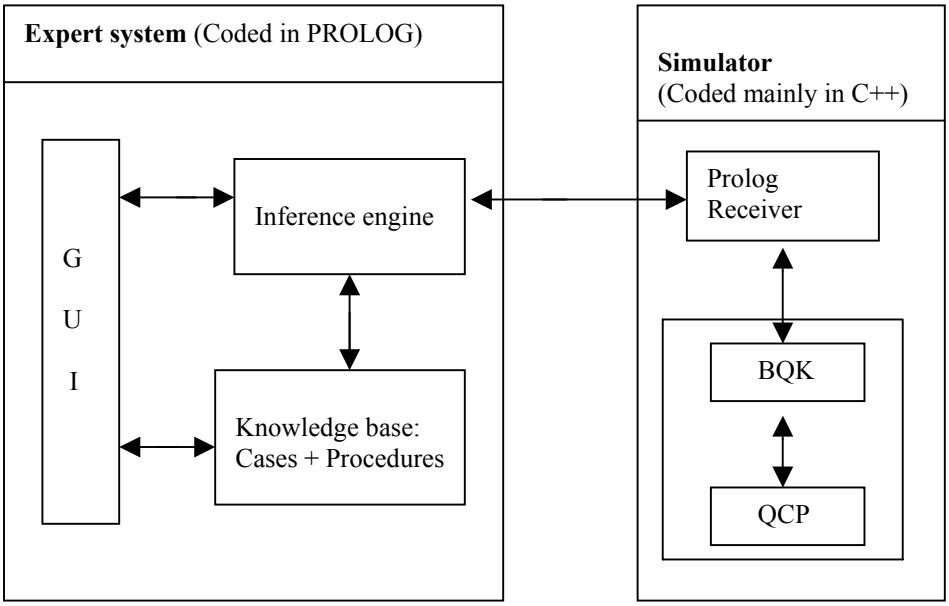



Figure 3.1: The structure of K9ER

### 3.1 The knowledge base

The building of a knowledge base involves two processes: knowledge acquisition and knowledge representation.

In K9ER, the knowledge base includes cases and procedures. A single procedures knowledge base can be used for several cases. There are two kinds of procedures: diagnostic procedures and treatment procedures. Cases relate to individual patients. Each patient has his own special case knowledge base. Dr. Scott Brown provided all the source information for the procedures and case knowledge bases.

### 3.1.1 Knowledge acquisition

Obtaining domain knowledge is the core activity of expert system development although it is a time-consuming task, because the accuracy and completeness of the system depends largely on the effective collection of the domain expertise.

Obviously, knowledge is acquired mainly from domain experts, and interviews with domain experts are the best method to obtain domain knowledge. From the interview, questions about the domain can be solved quickly, and new information can be obtained immediately. Before the interviews, I familiarized myself with the domain by reading relevant materials about the domain, such as some text books and reference books on physiology, pathology, emergency room procedure, and the regular diagnoses and treatments. This not only helped me communicate with the expert but also saved the time of the expert.

Each interview included three parts: the preparation for the interview, the interview itself, and subsequent analysis of the information gathered in the interview. Usually I interviewed the expert one hour each week for approximately a year. Before each interview, I defined my objectives and formulated questions based on the previous interview. At the beginning of each interview, the domain expert and I reviewed and confirmed the information from the previous interview. I reported the current development of the system to the domain expert and explained the design. Meanwhile, the expert tested the parts of the system that were currently functioning and gave his evaluation. I wrote down the expert's opinions for later improvement both of the knowledge base and of the GUI. After solving the current problems, we entered into new topics. The expert would describe a new case or provide treatment suggestions.

Sometimes I would ask the domain expert to offer some more information that should be present in K9ER, but was not yet included. Thus, the interviews not only helped to confirm "old" knowledge, but also provided the new ideas for building the knowledge base in K9ER. After each interview, I analyzed the information gathered, integrated it to the existing knowledge base, upgraded the program (GUI and inference engine,) and implemented the revised system to insure that it worked correctly. In the next interview, I would introduce the new "version" to the expert. Through this cycle, the system was improved in it's effectiveness and completeness. As can be seen from above, knowledge acquisition kept working throughout the entire process of the system development.

### 3.1.2 Knowledge representation

The domain knowledge acquired could be classified into several components (see Figure 3.2.) All of these needed to be implemented into facts and rules. The domain expert suggested that taking the patient history should be included in the diagnosis component; so, in K9ER, the regular menu of diagnostic exams or tests is:

*History*

*Signalment*

*Physical examination*

*Clinical pathology*

*a. Arterial blood gases*

*b. Hematology, Biochemical Profile, and Urinalysis*

*Other tests*

*a. Thoracic radiographs*

b. *Blood pressure*

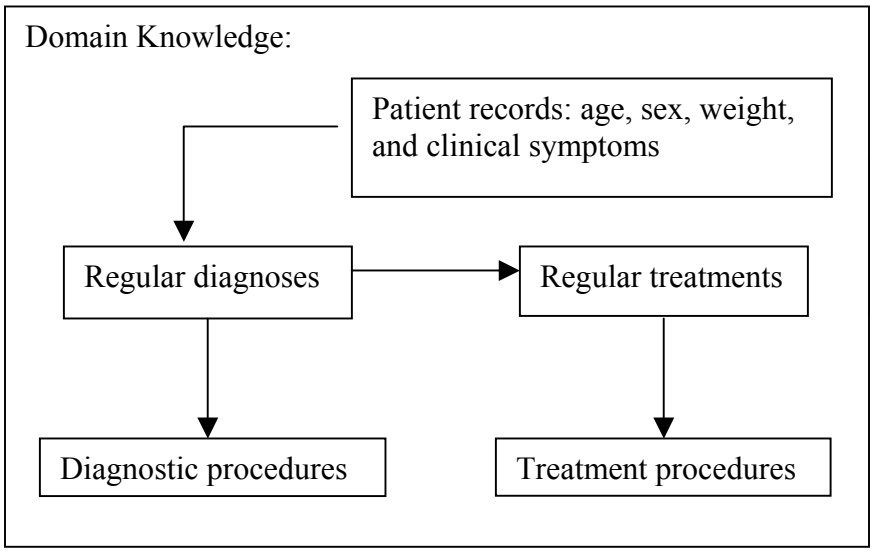
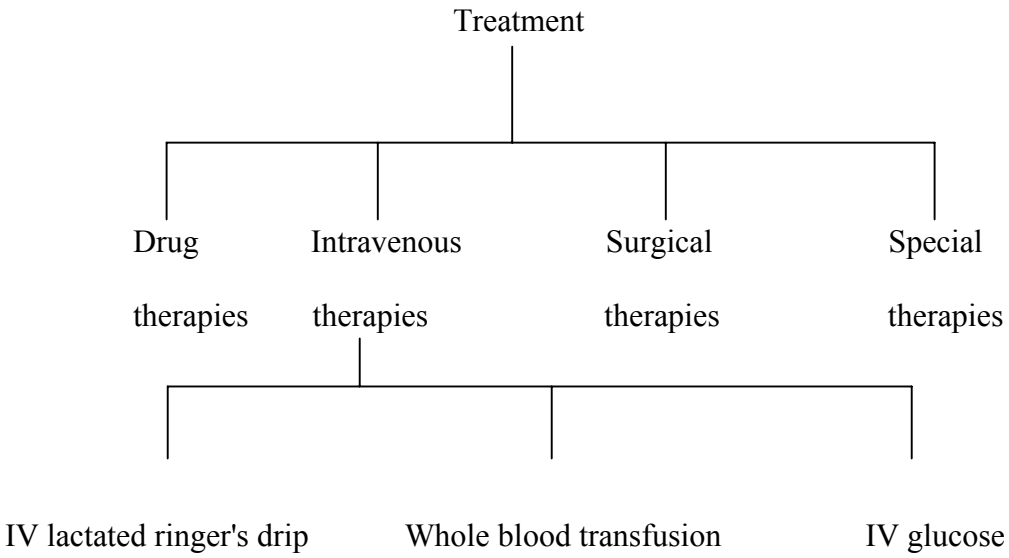



Figure 3.2: The domain knowledge

The domain expert presented the treatment items in a hierarchical structure. Part of this structure is shown below.



In the knowledge base for K9ER, each emergency case contains a set of simulation facts allowing the program to simulate a patient's physiological process. These simulation facts can be documented from the corresponding source information, such as the "Scooby Doo" case. In the "Scooby Doo" case, we were told that a car hit a dog, Scooby Doo, approximately 20 minutes before arrival at the clinic. The dog lost 1000 cc of blood in that time. In the knowledge base, this is recorded as values of parameters that must be changed in the physiological simulator (QCP):

parameter_change(`"Arterial Hemorrhage, Switch"`,1).

parameter_change(`"Arterial Hemorrhage, Final Volume"`,1000).

parameter_change(`"Arterial Hemorrhage, Duration"`,20).

Arterial Hemorrhage is a parameter for bleeding. By changing this variable's value, we can simulate bleeding by an animal. Bleeding starts by changing the Switch to 1. The value of Duration determines how many minutes the bleeding continues during the simulation. The value of Final Volume is the total volume of blood that is lost and determines the rate of hemorrhaging.

Each emergency case builds up the patient medical record. In the case of "Scooby Doo", the record will contain some additional information:

*Start:* 'An hysterical man and his son carrying a dog on a board burst through the door of your clinic screaming "Help him, help him".'

*History:* 'The dog was hit by a car approximately 20 minutes ago.'

*Signalment:* 'Two-year old great dane intact male weighing 65 kg.'

I mentioned in chapter 2 that QCP is a physiological simulation for humans. The data sent to, and obtained from, QCP should be scaled according to the dog's weight or other factors. The domain expert provided formulas to scale the data, such as:

For respiratory rate, to scale the model value (QCP output) to veterinary medicine normal, multiply by 1.8 for dogs, multiply by 2.0 for cats, multiply by 0.5 for horses.

These scaling factors are represented as simple facts the inference engine then uses. For example:

coefficient(dog, `respiratory rate`, 1.8).

coefficient(cat, `respiratory rate`, 2.0).

coefficient(horse, `respiratory rate`, 0.5).

Because these facts can be used for every case in which the patient is a dog, cat, or horse, I put them into the procedures knowledge base.

In addition to the special emergency procedures, there are routine procedures available for most cases in the knowledge base of K9ER. Procedures consist of diagnostic procedures and treatment procedures. While matching each procedure by the diagnostic item or treatment item, a medical record is built which includes the cost for the diagnosis or treatment, the time needed to conduct the test, the time required to get the results, the development of the report, and the simulation commands for the treatment. Each diagnostic procedure has associated with it the time required to conduct it and the cost in a PROLOG fact. An example is:

diag_proc(`Signalment`,30,1,0,[case(signalment)]).

In this predicate, the first parameter is an item in the diagnostic procedures list. The second parameter is the cost, the third is the time to conduct the test, the fourth is the time

to get the diagnosis result, and the last tells us which predicate is used to store the result of the test in the case file.

Here is a more complex example:

```
diag_proc(`Physical examination`,0,10,0,
        [`Temperature is `,
         'Core Heat, Temp (F)',
         ` F, pulse rate is `,
         'Heart Rate, Rate',
         ` beats/min, and the repiratory rate is `,
         'Resp Center, Respiration Rate',
         ` breaths/min.`,
         case(exam)]).
```

In this case, the last parameter includes some simulator parameters such as Core Heat, Temp (F). When this is encountered, the inference engine will go to another module to obtain the data. I will introduce the inference engine in detail later.

Treatments are stored as clauses for the PROLOG predicate treat_proc. treat_proc has five parameters: the treatment name, cost, processing time, medical report, and simulator information for this treatment. Here are examples:

```
treat_proc(`Place on Oxygen`,75,1,[`More Oxygen given.`],
                                  [(`"Air Supply, Source Of Mixture"`,1),
                                   (`"Gas O2 Tank Valve, Percent"`,100),
                                   (`"Gas N2 Tank Valve, Percent"`,1)
                                  ]).
```

This example means that the treatment ‘Place the patient on Oxygen’ costs $75.00, needs 1 minute to process, is reported as “More Oxygen given”, and a test of parameters in the simulator should be set to the default values given.

```
treat_proc(`Place animal on ventilator`,75,30,[`More air given.`,
                                                `The ventilator rate is `, Rate,
                                                `. The tidal volume is `, Volume],
                [(`"Ventilator, Switch"`,SettingN),
                 (`"Ventilator, Rate"`,RateN),
                 (`"Ventilator, Tidal Volume"`,VolumeN)]) :-
                    treatment_dialog([`Set switch of Ventilator to 1`,
                                      `Set the rate of Ventilator (1-50):`,
                                      `Set the tidal volume of Ventilator (50-2000):`],
                                      [Setting,Rate,Volume]),
                    number_string(SettingN,Setting),
                    number_string(RateN,Rate),
                    number_string(VolumeN,Volume).
```

In this example, the values of simulator parameters are not default values. The user sets the values through a dialog window. The routine treatment_dialog is used to realize this function. When the system executes treatment_dialog, a dialog window will pop up (see Figure 3.3.) The user can enter the data into appropriate fields.

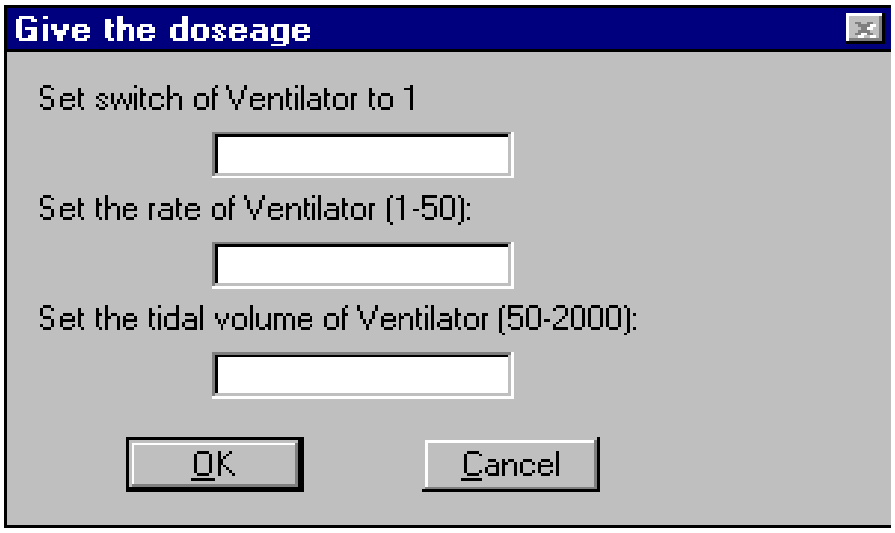



Figure 3.3: Treatment dialog window

For convenience, I put all the procedure predicates into one file (procedure.pl.) It is available for all individual cases and can be upgraded even after the completion of K9ER. Each emergency case is contained in a unique case file (such as 'scoobydoo.case'.) Each case file records a patient's unique information. Additional case files can be added into the knowledge base at any time.

### 3.2 The user interface

The user interface is a very important component of an expert system because it handles the communication between the user and the inference engine. In K9ER, the user interface includes menu bars, buttons, dialog windows and so on. In a dialog window, the user may need to enter the data to set a value or to select appropriate diagnostic and/or treatment procedures from a list. Once the input is finished, the message that the user entered is converted into a format that can be interpreted by the inference engine. Also,

after the inference engine reaches a conclusion, the user interface can display it to the user in English.

In K9ER, there are many "panes" in the interface window to implement communication between the user and the computer, such as the procedures pane, the report pane, the set-clock pane, and the information pane. Among these panes, some solicit data from the user, some support extended interaction with the user, and some only give messages to the user. Here, I introduce several main panes to describe the K9ER user interface.

The main dialog window in K9ER pops up when the user launches this program (see Figure 3.4.) On the main dialog window, there are many action buttons available for the user to handle different activities: selecting diagnostic tests or treatments, advancing the clock, stopping treatments, and so on. At the bottom of this window, there are a clock pane and a cost pane. These remain dormant until an emergency case is loaded by clicking **File** on the menu bar.

After an emergency case is loaded, the clock starts at 00:00:00 and the cost field shows the basic clinical fee. If the user clicks **Diagnose**, a list of diagnostic tests will appear in the left side of the screen (the procedures pane.) Choose one of them, then click button **Apply**, and the system starts to process the diagnostic procedure that was just chosen. The right side of the window is the reporting pane. The conclusion that the inference engine reaches is fed back to the user in this pane. Using this screen, the user plays the role of a clinician involved in an emergency case. He gives the patient diagnostic tests, checks the patient's physiological status in the report pane, chooses appropriate treatments and then checks again to see whether the patient improves after

the treatments. Each step of the activity is recorded in the report area. The user can then save and print the record of the session.

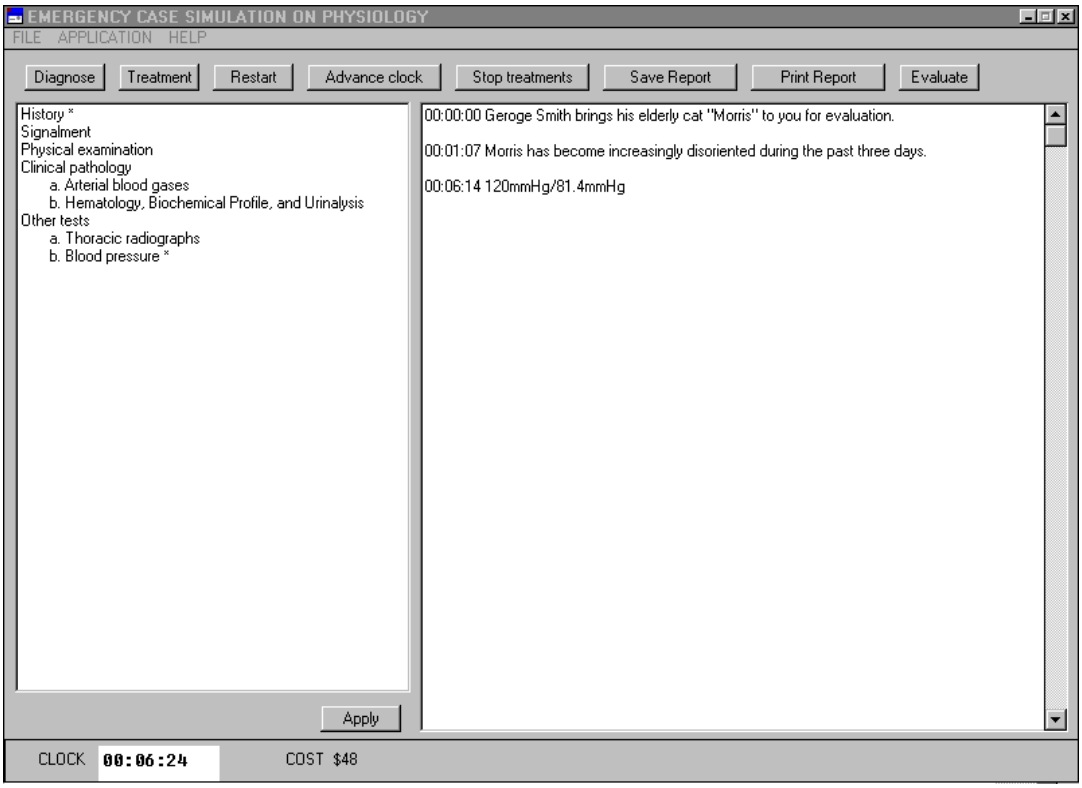



Figure 3.4: The main dialog window of K9ER

The user can set the clock ahead at any time in order to see the treatment results. Clicking **Advance clock** causes a dialog window (see Figure 3.5) to pop up, and the cursor will remind the user to enter the data.

When the user clicks **Stop treatments**, a window pops up to show any treatments that can be stopped. If there are none, a message window will pop up to tell the user. Otherwise, the user will have to determine which treatment needs to be stopped (see Figure 3.6.)

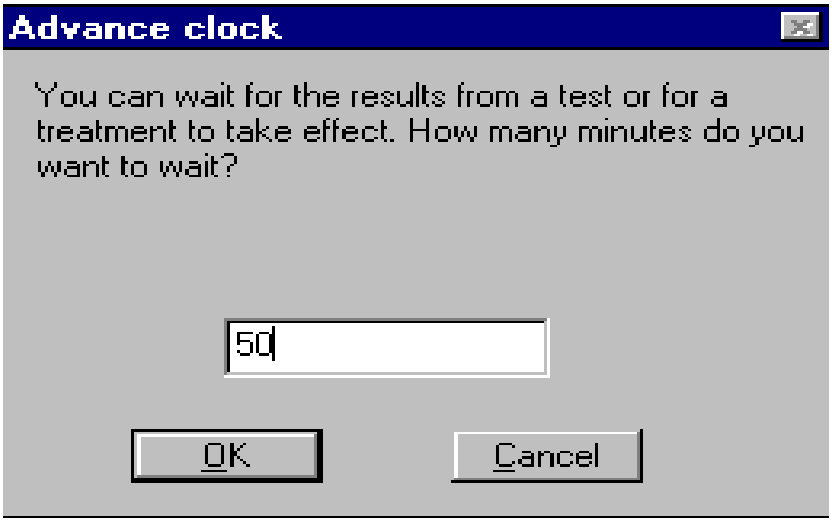



Figure 3.5: A dialog window for user to advance the clock

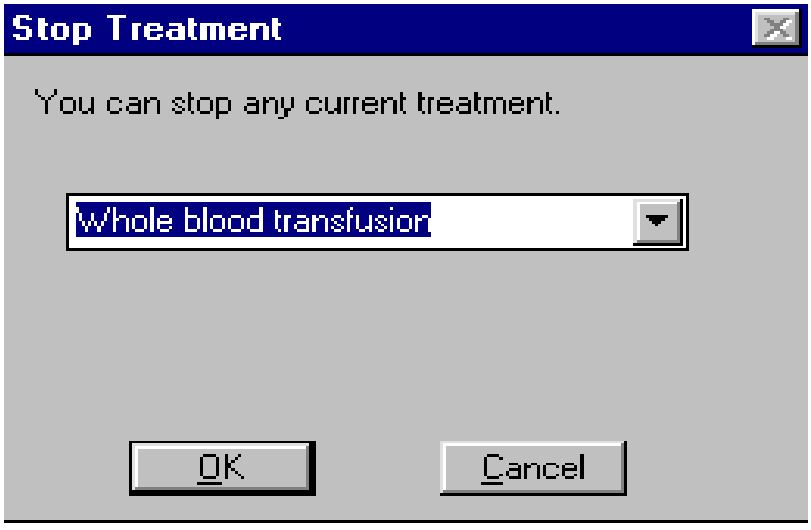



Figure 3.6: Stopping the current treatment

There is another dialog box in the user interface that appears as needed to tell the user what to do next, or to explain when the window is disabled while waiting for some procedure to finish (see Figure 3.7.)

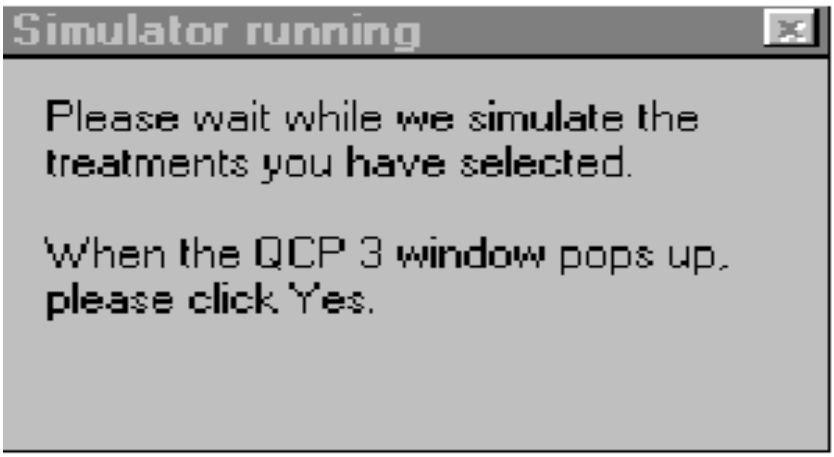



Figure 3.7: A message window

### 3.3 The simulator

Although a simulator is not a necessary component of an expert system, the simulator QCP plays an important role in K9ER. It produces physiological data and then feeds this data back to K9ER. Because this data is dynamically generated, it belongs to the working database, not to the knowledge database. The simulator includes three parts (see Figure 3.8): the receiver, BQK and QCP.

The receiver component of the simulator is a procedure written in PROLOG. When the simulator receives a request from the expert system for a simulation to be run, the receiver writes a command file for QCP according to the message received. Then the receiver calls BQK. BQK reads the command file, starts QCP, uses 16-bit data structures to send parameter settings and commands to QCP, receives results back from QCP, and

writes these results in a file where the receiver can read them. This process was described in more detail in Chapter 2.

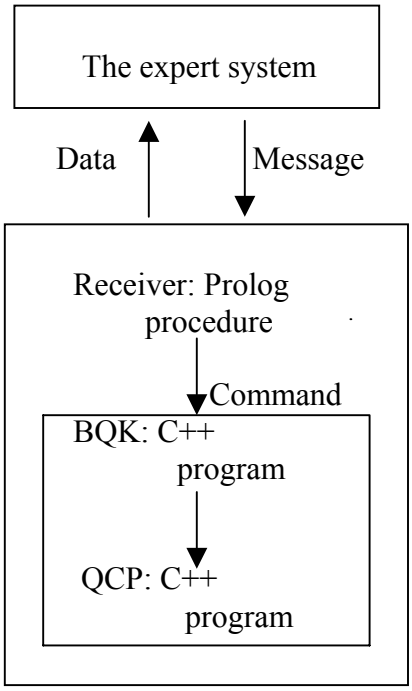



Figure 3.8: The structure of the simulator

### 3.4 The inference engine

By backtracking, which is built into LPA WIN-PROLOG, the inference engine searches the knowledge base for a rule that can satisfy a given goal. Once the search finds a matching rule, it sequentially goes through the facts or rules in the knowledge base and/or the working data base to determine whether the conditions of the rule are satisfied or not. If not, the inference engine will continue to look for another matching rule until it satisfies the goal or runs out of rules. For example, consider the simple rule

```
get_treat_result(Item, Result) :-
        treat_proc(Item,Cost,Duration,Report,List),
        Result = (Cost,Time,Report,List).
```

This rule states that the inference engine will search for the treatment result in the knowledge base by using the chosen treatment Item. If Item is not matched, the inference engine will backtrack until it finds the solution.

K9ER also provides evaluations of the user's activity. After the implementation of treatment protocols, the user can choose the **Evaluate** function to see whether the protocols improve the patient's physiological situation. The system compares the current physiological values with the normal physiological values and initial values, and then gives the evaluations back to the user. For example,

```
evaluate(Animal,Parameter,Initial_Value,Current_Value,good) :-
   normal_range(Animal,Parameter,Low,High),
   (Initial_Value =< Low;
    Initial_Value >= High),
   Current_Value >= Low,
   Current_Value =< High,
   !.
```

The system searches the normal range of Parameter (such as body temperature) for the kind of Animal in the procedures knowledge base. If the Initial Value is out of the normal range but Current Value is in the normal range, the treatment protocol is good. The system also provides the expert's treatment protocols for each case. All of these functions are implemented by the inference engine.

# CHAPER 4

# CONCLUSION

K9ER is designed for veterinary students to learn and practice the clinical processes of an emergency room. It can operate on any personal computer with Windows 95/98/2000/2002. The program was written in PROLOG and C++, and was developed in the LPA WIN-PROLOG programming environment. This environment provides the means to build a friendly user interface, syntax for building a declarative knowledge base, and an inference engine.

As a coupling of a simulator and an expert system, K9ER not only provides the student an opportunity to implement diagnostic and treatment processes, but also gives realistic real time responses to the student's actions. The combination of simulation and expert system is an ideal tool for interactive learning.

The distinguishing feature of K9ER is its flexible knowledge base. This means the knowledge base is expandable after the expert system is built up. New case files can be added into the knowledge base of the K9ER system at any time when it is needed in the teaching process. These new case files expand the kinds of clinical situations the students can learn to handle by using the system.

K9ER can complement live animal labs in medical learning and training and provide a study tool with effective cost, unlimited repetition, and scheduling convenience.

Although K9ER is a completed system, there are several ways in which it could be improved and enhanced. As a simulator, K9ER could provide physiological monitors to display the state of the patient. More graphics could be incorporated into the system to render the software even more user-friendly and realistic, and to make the learning process more engaging. For example, a video clip could take the place of the text that currently presents the patient to the student. In addition, it would be exciting to develop the current help file into reference components that make other resources, such as class notes, available to students. Eventually it will be most effective to combine K9ER with other expert systems, such as the Anesthesia Expert System (Deng, 2000), in order to form an integrated system.

# BIBLIOGRAPHY

Boyce J., Sanna P.J., Tidrow R., Chau J.J., Fuller S., Pagan K., Jacobs R., and Ruehlin R.J., 1995. *Inside Windows 95*. New Riders.

Buchanan B.G., Barstow D., Bechtel R., Bennet J., Clancey W., Kulikowski C., Mitchell T. M., and Waterman D.A., 1983. *Constructing an expert system*. Addison-Wesly.

Coleman T., 1999. *QCP DESolver: Solver Control Manual.* Biological Simulators, Inc..

Convington M.A., Nute D., and Vellino A., 1997. *Prolog Programming in Depth.* Prentice-Hall.

Cunningham J., 1997. *Textbook of Veterinary Physiology*. W.B. Saunders Company.

Davis R., 1980. Meta-rules: reasoning about Control. *Artificial Intelligence*. 15:179-222.

Deng X., 2000. AneSoft: Anesthesia Expert System. unpublished.

Gonzalez A.J. and Dankel D.D., 1993. *The Engineering of Knowledge-based systems: theory and practice*. Prentice-Hall.

Jackson P., 1999. *Introduction to Expert Systems*. Addison-Wesley.

Kowalski R.A., 1979. *Logic for Problem Solving*. North-Holland.

Murtaugh R.J. and Kaplan P.M., 1992. *Veterinary Emergency and Critical Care Medicine.* Mosby Year Book.

Pallin A. and Kittell R.P., 1992. Mercy Hospital: Simulation Techniques for ER Processes. *Industrial Engineering*. 24:35-37.

Schwid H.A. and O'Donnell D., 1993. The Anesthesia Simulator Consultant: simulation plus expert system. *Anesthesiology Review*. 20:185-189.

Steward D. and Standridge C.R., 1996. A Veterinary Practice Simulator Based on the Integration of Expert System and Process modeling. *Simulation*. 66:143-159.

# APPENDIX

## DIRECTIONS FOR USING K9ER

Extract all the files from the compacted file vv.zip under the same directory, then click vv.exe to run the K9ER system. The following is how to use this software after the main dialog window pops up. The main dialog window contains a left child window, a right child window, a menu bar, and many functions.

1. Click **File** on the menu bar and select one of the available patient cases.
2. After the case file is loaded, you can begin your diagnostic tests and treatments.
3. Select different diagnostic tests or treatments by clicking **Diagnose** or **Treatments** under the menu bar and choosing items of diagnostic tests and treatments.
4. Click an item and move the mouse to the button **Apply** under the left child window. Click again to start processing.
5. Follow the directions provided by the pop-up message, wait for the results to be reported in the right child window, and then take the next action. But if there is a message which states that the result will be back after a period of time, you don't need to wait for the results to be reported.
6. If implementing advancing time, stopping current treatment, or viewing the evaluation of your exercises, click the corresponding buttons.

7. If loading another patient case, click the **Restart** button under the menu bar, then repeat steps 1 – 6.
8. When finishing your exercises, save or print the reports, then close the window.