Eugene ROVENȚA
Computer Science and Engineering Department
York University, Toronto, ON
Office: CSEB 3026
Voice: 416-736-2100 ext. 33928
E-mail: roventa@yorku.ca

George ROȘU
„Aurel Vlaicu" University of Arad, Engineering Faculty
Bd. Revoluției nr. 77, 310130, Arad, Romania,
E-mail: george_rosu_s@yahoo.com

PROLOG EXPERT SYSTEM: THE DIAGNOSIS OF KIDNEY DISEASES

NOTE: This paper was presented at the International Symposium "Research and Education in an Innovation Era", Section III, November 16-18, 2006, "Aurel Vlaicu" University of Arad, Romania.

ABSTRACT

A Medical Expert System made in Visual Prolog is proposed. This Expert System makes a differential diagnosis among the main kidney diseases. The diagnosis is made taking into account the clinical exam (the symptoms that can be seen and felt) and the paraclinical exam (the results of laboratory tests). This system is designed to give help to a medical expert in making the appropriate diagnosis of a patient. Why would it be needed in helping a physician? Because the kidney diseases have a lot of common symptoms and many of them are very much alike, fact that makes it very difficult even for a kidney specialist to put a right diagnosis. The proposed Expert System can address that. It contains in its knowledge base twenty-seven kidney diseases from nine different categories.

KEYWORDS:

Expert System, Differential diagnosis, Clinical and paraclinical exam,

INTRODUCTION

1. MEDICAL EXPERT SYSTEMS

Artificial Intelligence is defined as intelligence exhibited by an artificial entity. AI programs that achieve expert-level competence in solving problems in task areas by bringing to bear a body of knowledge about specific tasks are called *knowledge-based* or *expert systems.* A lot of Expert Systems are build in medical domain. Their purpose is the diagnosis and treatment of certain diseases. A Medical Expert System is made out of a group of programs and a medical knowledge base with which one can have a dialogue with a computer. The information obtained from the computer is similar to the information given by an expert doctor in that certain area.

1.1. The Proposed Medical Expert System

The proposed system has in its knowledge base twenty seven kidney diseases from nine different categories. The user is asked to answer with **Yes** or **No** if a certain symptom appears or not. In the end, based on the user's answers, the name of the disease is posted up on the screen. A "minus" of this system (and usually of any other Expert System) is that only the symptoms put in the knowledge base by the programmer are available. It doesn't think and doesn't learn by itself; but the knowledge base can be updated anytime with new symptoms and new diseases.

1.2 The Sections of the Program

The „facts" section

In this section we have declared two facts:

xpositive(symbol,symbol) – for a positive answer and
xnegative(symbol,symbol) – for a negative answer

which will be used in defining the rules for the predicates „positive" and „negative", like this:

```
positive(X,Y):-
        xpositive(X,Y),!.
  positive(X,Y):-
        not(xnegative(X,Y)),
        question(X,Y,yes).
```
- if the answer to the question is affirmative;

```
  negative(X,Y):-
        xnegative(X,Y),!.
  negative(X,Y):-
        not(xpositive(X,Y)),
        question(X,Y,no).
```
- if the answer to the question is negative.

Of course that the negation of a negation is an affirmation and the negation of an affirmation is a negation.

The „predicates" section

In this section we have declared the following predicates:

```
  disease(symbol) - nondeterm (o)
  is_disease(symbol) - nondeterm (i)
  question(symbol,symbol,symbol)-    determ (i,i,i)
  remember(symbol,symbol,symbol)- determ (i,i,i)
  positive(symbol,symbol) - determ (i,i)
  negative(symbol,symbol) - determ (i,i)
```

```
clear_facts - determ ()
run - determ ()
```

The predicate „disease" will have as a parameter the name of the disease in the **clauses** section. The predicate „is_disease" will have as a parameter the category of diseases which the certain disease takes part of. This category is defined and recursively appealed each time it is met in the program.

The predicate „question" is the predicate of a „no" or "yes" answer. The clauses (the rules) for this predicate will be shown in the section „clauses".

The predicate „remember" is used by the program for remembering the answer given to a fact, before adding more facts while running the program, through the pre-defined predicate „assertz", like this:

```
remember(X,Y,yes):-
        assertz(xpositive(X,Y)).
remember(X,Y,no):-
        assertz(xnegative(X,Y)).
```

Through the predicates „positive" and „negative" we introduce the symptoms of the disease: the symptom is put as an argument at „positive" if it is available for that certain disease, respectively as an argument at "negative" if it is not available.

The predicate „clear_facts" is used for stopping the compiling of the program, and the predicate „run" is used for running the program.

The „clauses" (rules) section

In this section we introduced all the rules that define the diseases, using the predicates „disease", „is_disease", „positive" and „negative", defined in the predicates section.

For a better understanding we illustrate here the rules for one disease:

```
disease(sindromul_Goodpasture):-
       positive(se_semnaleaza,hemoragii_pulmonare),
        is_disease(glomerulonefrita_rapid_progresiva),
          positive(apare_o_infectie_a,cailor_respiratorii_superioare),
          positive(simptome_respiratorii,tuse_dispnee),
positive(anemia_este,variabila),
          positive(hematuria_este,microscopica_si_moderata),
positive(proteinuria_este,moderata),
         positive(hipertensiunea_arteriala,este_usoara),
          positive(apar_fenomene_articulare,artralgii_variabile),
          negative(complexele_imune_circulante,sunt_crescute),
positive(fractia_C3,este_normala),
          positive(ureea_sanguina_si_acidul_uric,inregistreaza_valori_crescute),
          positive(se_evidentiaza_prezenta,unor_infiltrate_pulmonare_bazale),
          positive(imunofluorescenta_evidentiaza,depozite_liniare_de_imunoglobuline_IgG).
```

In the following we present the common clause for the category "Glomerulonefrite rapid progresive" (in this case), which is called in the rules from all the diseases in this category.

```
is_disease(glomerulonefrita_rapid_progresiva):-
       positive(se_semnaleaza_inflamatie_glomerulara,si_oligoanurie),
positive(debutul_e_lent_progresiv,rar_acut),
  positive(evolutie_spre_insuficienta_renala,in_perioada_de_cateva_zile_la_cateva_luni),
       positive(leziunea_predominenta_o_reprezinta,glomerulita_extracapilara).
```

We also have the clause for „question", using the predicate „remember":

```
question(X,Y,yes):-!,
            write(X," ",Y,'\n'),
            readln(Reply),nl,
            frontchar(Reply,'y',_),
            remember(X,Y,yes).
   question(X,Y,no):-!,
            write(X," ",Y,'\n'),
            readln(Reply),nl,
            frontchar(Reply,'n',_),
            remember(X,Y,no).
```

Here is the clause for the predicate „clear_facts”:

```
clear_facts:-
            write("\n\nPress <<space>> to exit. \n"),
            retractall(_,dbasedom),
            readchar(_).
```

„Jumps” out of the program when pressing „space”: „retractall” (deletes facts while running) and „readchar”(reads a “char” variable) – these are pre-defined predicates in Prolog.

The clause for the predicate „run”:

```
 run:-
            disease(X),!,
            write("\nThe disease having all these symptoms is:  ",X),
            nl,
            clear_facts.
   run:-
            write("\nThis disease cannot be determined. \n\n"),
            clear_facts.
```

The program calls the predicate „disease” and posts up on the screen the name of the disease (read from the argument), and in case it cannot find all the symptoms or the disease cannot be identified it gives a certain message.

The „goal” section

Here the “run” predicate is called and also the program posts up on the screen information which the user will be able to read when running the program (the name of the system, who made it, how to use it etc.).

2. HOW THE PROGRAM WORKS

We consider for example a certain disease from a certain category. The program “thinks” like this: if it has a positive answer to a symptom, it goes on with the symptoms from that disease. If only one symptom from the disease is negative it “jumps” to the first symptom from the next disease. Of course that in takes into accounts the category symptoms also. If at least one symptom from the category is negative, the program goes to the next disease. If all the category symptoms are affirmative, it goes on to the symptoms which make the difference between this disease and the other diseases from this category.

3. FUTURE IMPROVEMENTS

The knowledge base can be improved with new diseases and even new symptoms.

An important way of improving this Expert System is using fuzzy techniques. In this case the system would establish the degree that the diagnosis is close to the reality.

Here we present some ideas for fuzzifying the Prolog rules:

p(X,a) :- q(X,u), r(X,Y,v). where:

a = degree to which „X” satisfies „p”;
u = degree to which „X” satisfies „q”;
v = degree to which„(X,Y)” satisfies „r”.

The degree is a number between 1 and 0.

$$\mathbf{a = sup(u \wedge v)}$$

Example:

big(X,a) :- tall(X,u), heavy(X,v). which means:

X is big (with a degree a) if X is tall (with a degree u) and X is heavy (with a degree v).

In this way we could fuzzify all the rules in this program to obtain the degrees in diagnosis of the kidney diseases.

CONCLUSIONS

Taking into account the fact that we are dealing with a person's health and we have to put an approximate diagnosis on a certain disease, this system used in practice implies a great risk. In reality there are more kidney diseases than we have in this system's knowledge base. Therefore, our knowledge base is not complete, but we can update and improve it anytime with new symptoms and new diseases.

On the other hand, it is possible that the symptoms already present are not 100% right, because different experts have different opinions and there are a lot of anomalies in Medicine.

REFERENCES

[1] Bostaca, I. (1999) *Cheile diagnosticului în clinica medicală*, Editura Polirom.
[2] Gluhovschi, G. (2004) *Curs de Nefrologie*, Lito U.M.F.T.
[3] Romoşan, I. (1999) *Rinichiul. Ghid diagnostic şi terapeutic*, Editura Medicală.
[4] Luger, G.L. (2002) *Artificial Intelligence. Structures and Strategies for Complex Problem Solving,* 4ed., Addison-Wesley.
[5] Negnevitski, M. (2002) *Artificial Intelligence Guide to Intelligent Systems*, Addison-Wesley.

[6] Rovența, E. (2000) *Elements de logique pour l'Informatique*, GREF Toronto

[7] Rovența, E., Spircu, T. (in publication) *Management of Knowledge Imperfection in Developing Intelligent Systems*, Springer-Verlag

[8] Russell, S., Norvig P. (1995) *Artificial Intelligence. A Modern Approach*, Prentice Hall.

[9] Rowe, N.C. (1988) *Artificial Intelligence through Prolog*, Prentice Hall.

[10] Bratko, I. (2000) *Prolog Programming for Artificial Intelligence*, Addison - Wesley.

[11] Nilsson, U., Maluszynski, J. (2000) *Logic Programming and Prolog*(2ed), John Wiley & Sons Ltd.

[12] Reghiş, M., Rovența, E. (1998) *Classical and Fuzzy Concepts in Mathematical Logic and Applications*, CRC Press New York