ST Algorithm for Medical Diagnostic Reasoning

Irosh Fernando and Frans A. Henskens

***Abstract*—The authors have previously described an approach for medical diagnostic reasoning based on the ST (Select and Test) model introduced by Ramoni and Stefanelli et al. This paper extends the previous approach by introducing the required algorithm for medical expert system development. The algorithm involves a bottom-up and recursive process using logical inferences, abduction, deduction, and induction. Pseudocode for the algorithm, and the data structures involved, are described, and the algorithm's implementation using a small sample knowledgebase and programmed in Java is included in appendixes. Implementation of a successful expert system is a challenging process; development of the necessary algorithm for its inference engine, and definition of a knowledgebase structure that models expert diagnostic reasoning and knowledge, only fulfils the initial step. Challenges associated with the remaining steps of the development process can be identified and dealt with using the CLAP software process model.**

***Index Terms*—Medical diagnostic reasoning, medical expert systems, ST model.**

I. Introduction

REALISATION of medical expert systems has been one of the earliest goals of the AI community. Unfortunately, attempts by major projects such as INTERNIST-I and CADUCEUS have not been successful [1].

One of the reasons for this failure can be understood in relation to the lack of models that capture the depth and complexity of expert medical diagnostic reasoning. Models previously proposed for medical diagnostic reasoning include: scheme-inductive reasoning [2]; hypothetico-deductive reasoning [3]; backward and forward reasoning [4]; pattern recognition [5]; Parsimonious Covering Theory [6]; Information Processing Approach [7]; Process Model for diagnostic reasoning [8]; Certainty Factor model [9]; models based on Bayes Theorem [10-12]; and models based on Fuzzy logic [13-15]. The authors have previously described the limitations of some of these approaches, and proposed an approximate reasoning model for medical diagnostic reasoning [16]. This previously proposed model was based on the epistemological framework (also known as Select and Test (ST) model) for medical diagnostic reasoning proposed by Ramoni and Stefanelli et al. [17].

This paper complements the authors' previous approach by introducing the required algorithm for diagnostic inference. The previously proposed reasoning model requires of at least three layers of knowledgebase entities, namely diagnoses, symptoms and symptom attributes, together with mathematical functions to quantify those entities. In order to improve readability, the algorithm described in this paper has been deliberately simplified by restricting its application to the first two layers only. Extension of the algorithm to incorporate the full model is explained in the Discussion section of this paper.

The remainder of the paper begins with an introduction to the ST Model followed by formalisation of the knowledgebase as a graph consisting of symptoms and diagnoses. Then, the algorithm's pseudocode and the data structures, and its implementation, are described using a sample knowledgebase. Before the paper is concluded, other challenges that are faced in developing successful medical expert systems are briefly outlined, and the CLAP software process model [18] is described as a framework for addressing these challenges in a systematic manner.

II. Select And Test (ST) Model

The ST Model describes a cyclical process (Fig. 1), which uses the logical inferences, abduction, deduction, and induction that were described by Charles Peirce [19]. Usually, diagnostic reasoning in clinical contexts begins when a patient reports a symptom or symptoms to their clinician. Whilst these symptoms are well-defined entities in the clinician's mind, patients may use various descriptive terms to describe their symptoms. For example, a patient may use the descriptive term 'a dark cloud over me' to describe the symptom 'low mood'.

The process of mapping these descriptive terms understood by patients onto well-defined symptom entities used in the knowledgebase is known as abstraction. The next step, known as abduction, involves determining all likely diagnoses related to the reported symptoms. Then, for each likely diagnosis, it is necessary to determine if the patient is experiencing other expected symptoms. This is known as deduction. These three steps repeat cyclically until all the required symptoms and diagnoses have been explored. Once this cycle is ended, the final step, induction, occurs.

Manuscript received August 6, 2013; accepted for publication on September 30, 2013.

D. A. I. P. Fernando is with the School of Electrical Engineering and Computer Science, University of Newcastle, NSW 2308, Australia (phone: +61 423 281 664; e-mail: irosh.fernando@uon.edu.au).

F. A. Henskens is with the School of Electrical Engineering and Computer Science, University of Newcastle, NSW 2308, Australia (e-mail: frans.henskens@newcastle.edu.au).

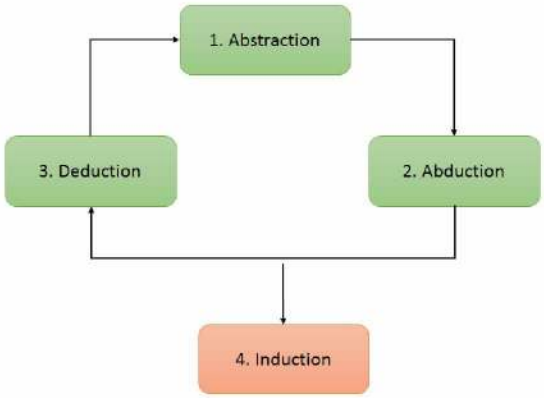

Fig. 1. The ST Model.

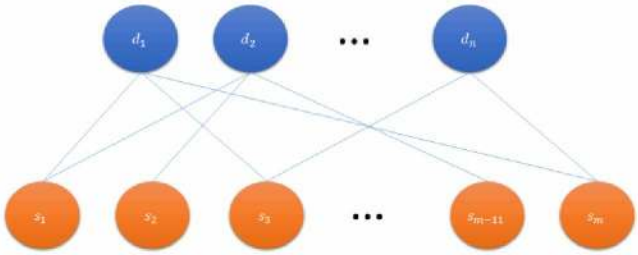

Fig. 2. Simplified knowledgebase representing diagnoses and symptoms only.

Induction involves matching the elicited symptoms with the expected symptoms for each likely diagnosis, thus determining whether the patient is suffering from any of the likely diagnoses. More details of the ST model can be found in the paper published by Ramoni and Stefanelli et al [17].

III. Formal Model For Knowledgebase

By way of formalising the process described above, let us represent all the diagnoses and symptoms in our knowledgebase as sets $D = \{d_1, d_2, \ldots, d_n\}$ and $S = \{s_1, s_2, \ldots s_m\}$ respectively. The relationship representing 'given a symptom s_i how likely is diagnosis d_j' is represented as a two-layer graph (Fig. 2), in which each arc is associated with a value θ_{ij} representing the likelihood (L) that s_i implies d_j; note $0 \leq \theta_{ij} \leq 1$. This can also be represented using the notation $L(d_j \mid s_i) = \theta_{ij}$. By way of example, in Fig. 2 the arc connecting d_1 and s_3 would have associated likelihood θ_{31}. The knowledgebase consisting of the two layers, symptoms and diagnoses, can be represented as a matrix $[\theta_{ij}]$.

IV. Select And Test Algorithm

Medical diagnostic reasoning involves two main steps:

1. search for symptoms,
2. arrive at diagnoses based on the symptoms found in the previous step.

Because of the vastness of the knowledgebase, one of the most challenging aspects of diagnostic reasoning is the symptom search process. It is therefore not uncommon that even an experienced clinician can at times miss a diagnosis because of failure to elicit a key symptom that would have provided an important clue to a diagnosis. If all the symptoms are known, arriving at a diagnosis is relatively easy computationally, depending on the commonly agreed or established diagnostic criteria used in different medical specialities. For example, in psychiatry, if all the symptoms are known, the second step involves matching the elicited symptoms with the diagnostic criteria described in a standard diagnostic manual such as DSM-V [20]. In the ST algorithm, abstraction, abduction and deduction are involved in the first step, and induction is involved in the second step.

The proposed algorithm uses five dynamic data structures, namely *symptomsFound*, *symptomsToBeElicited*, *symptomsAlreadyElicited*, *diagnosesToBeElicited* and *diagnosesAlreadyElicited*, which are implemented as linked lists. Also, in order to describe how the algorithm works, a static data structure *patientProfile*, which an artificial entity that encapsulates all the symptoms actually present in a patient, is used. The nature of the real world diagnostic problem is that the symptoms a patient actually has are initially unknown to the clinician. Symptom searching (the first step) in real world diagnostic reasoning can be conceptualised as an endeavour to find all the content, or at least all the clinically important symptoms, stored in *patientProfile*. In real world situations *patientProfile* is a virtual entity because it represents the patient's actual symptoms, which need to be discovered by the clinician when the patient is interviewed.

Details of the abstraction step have also been simplified in this paper. Whilst, in the real world setting, abstraction involves mapping the patient's symptom descriptions to defined knowledgebase entities, this largely mechanical matching process is omitted. Rather, it is assumed that the symptom descriptions stored in *patientProfile* correspond to the symptom descriptions used in the knowledgebase.

Implementation of abstraction in a real world application would require, for example, the patient informing symptoms by answering closed-ended questions using check boxes in a very basic human computer interface, or via an actual dialog between patient and expert system using natural language capabilities.

The data structures used in a computer-based implementation of the algorithm are described in Fig. 3, and the algorithm itself is shown in Fig. 6.

The ST Algorithm starts when a patient reports a set of initial symptoms that are stored in *symptomsFound* and *diagnosesAlreadyElicited*.

Abduction then begins, returning all the diagnoses connected to each symptom stored in *symptomsFound*. A threshold value *likelihoodThreshold* in relation to the connection strength between any symptom and related diagnosis can be used to determine which diagnoses are to be retrieved.

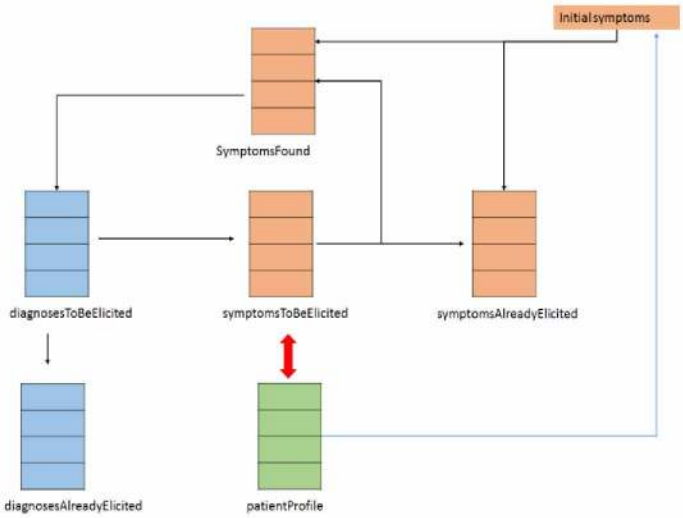

Fig. 3. Data structures used in the ST algorithm.

Accordingly, for any given symptom s_i the system will retrieve all the diagnoses d_j for which the likelihood that the symptom is caused by the diagnosis is in accordance with $\theta_{ij} >$ *likelihoodThreshold*.

These diagnoses are stored in the linked list *diagnosesToBeElicited*; before storing each diagnosis the system checks if it has already been stored in *diagnosesAlreadyElicited* because of association with a previous symptom, thus avoiding the possibility of duplicate diagnoses.

Next, deduction begins by returning all the expected symptoms connected with each diagnosis stored in *diagnosesToBeElicited*, after which the diagnosis is removed from *diagnosesToBeElicited* and transferred into *diagnosesAlreadyElicited*. All the expected symptoms that are returned for each diagnosis are transferred into *diagnosesToBeElicited* unless they are already stored in *diagnosesAlreadyElicited*.

Abstraction then commences, eliciting each symptom stored in *symptomsToBeElicited* by searching *patientProfile*. Then the elicited symptom is removed from *symptomsToBeElicited* and transferred into *symptomsAlreadyElicited* . If the elicited symptom is found in *patientProfile* it is stored in *symptomsFound*.

Finally, induction involves matching diagnostic criteria (i.e. symptoms expected for each diagnosis in *diagnosesAlreadyElicited*) with the symptoms stored in *symptomsFound*. If the diagnostic criteria are met, depending on the expected symptoms and the symptoms in *symptomsFound* then the respective diagnosis is accepted. Otherwise the respective diagnosis is excluded.

V. An Example And Its Implementation

In order to elaborate the proposed algorithm, let us consider a small knowledge base consisting of twelve symptoms and six diagnoses as described in Fig. 4. The relationships between these diagnoses and symptoms (i.e. θ_{ij} as described previously) are described in the matrix shown in Fig. 5. Java implementations of this knowledgebase and the algorithm are presented in Appendices 1 and 2, respectively.

s_1	Depressed mood
s_2	Loss of motivation
s_3	Weight loss
s_4	Fatigue
s_5	Chest discomfort
s_6	Worrying thoughts
s_7	Low self-esteem
s_8	Headache
s_9	Loss of appetite
s_{10}	Hand tremors
s_{11}	Hypertension
s_{12}	Gastrointestinal bleeding

d_1	Major Depression	0.5
d_2	Generalized Anxiety Disorder	0.2
d_3	Hyperthyroidism	0.6
d_4	Pheochromocytoma	0.7
d_5	Anaemia	0.7
d_6	Ischaemic Heart Disease	0.9

Fig. 4. Symptoms and diagnoses to be included in sample knowledgebase.

Details of the induction step are omitted in the implementation; the implementation of this step depends on the diagnostic criteria that are used to match the elicited symptoms with the expected symptoms of diagnoses. In its simplest form, the induction step can be implemented by one to one matching of the expected symptoms with the elicited symptoms. Nonetheless, depending on the diagnosis, it may not be necessary to have all the expected symptoms to make that diagnosis. In such situations, logical expression constructed using *AND* and *OR* operators can be used to formulate diagnostic rules by connecting different combination of symptoms. These diagnostic rules can be enhanced by allowing quantification of the severity of the symptoms elicited in the patient, as described elsewhere [16].

	d_1	d_2	d_3	d_4	d_5	d_6
s_1	0.9	0	0.3	0	0.3	0.3
s_2	0.9	0	0	0	0.3	0
s_3	0.6	0	0.7	0	0	0
s_4	0.6	0.7	0.6	0	0.8	0.3
s_5	0	0.6	0	0	0	0.8
s_6	0	0.9	0	0	0	0
s_7	0.6	0.4	0	0	0	0
s_8	0	0.6	0	0.5	0	0
s_9	0.7	0	0	0	0	0
s_{10}	0	0.6	0.8	0	0	0
s_{11}	0	0	0	0.9	0	0.4
s_{12}	0	0.4	0.3	0	0.6	0.6

Fig. 5. Representation of the knowledgebase as a matrix, *[θij]*.

```
1.  // Declare dynamic data structures as linked lists
2.  symptomsFound;
3.   symptomsToBeElicited;
4.  symptomsAlreadyElicited;
5. diagnosesToBeElicited;
6.  diagnosesAlreadyElicited;

7.   // Declare the threshold value for the likelihood of diagnoses
8.  likelihoodThreshold;
9.  BEGIN
10.
11. Store the symptoms initially reported in symptomsFound;
12.
13.      // ABDUCTION: get all the diagnoses related to symptoms
14.      //-------------------------------------------------------------------------------
15.    FOR EACH  symptom in symptomsFound  DO
16.          declare diagnoses as a temporary linked list
17.          Get all the diagnoses related to symptom above the likelihoodThreshold  and store in diagnoses;
18.           FOR EACH diagnosis in diagnoses DO
19.                IF diagnosis is NOT already in (diagnosesToBeElicited OR  diagnosesAlreadyElicited) THEN
20.                      Store diagnosis in diagnosesToBeElicited
21.                END-IF
22.            END-FOR
23.
24.           // DEDUCTION: get all the symptoms related to diagnoses
25.           //-------------------------------------------------------------------------------
26.          WHILE diagnosesToBeElicited is NOT empty   DO
27.             declare symptoms as a temporary linked list
28.             Get all the symptoms related to the current diagnosis in diagnosesToBeElicited above the likelihoodThreshold  and store in symptoms;
29.               FOR EACH symptom in symptoms  DO
30.                    IF symptom is NOT already in ( symptomsFound OR symptomsAlreadyElicited) THEN
31.                         Store symptom in symptomsToBeElicited;
32.                    END-IF
33.               END-FOR
34.               Remove the current diagnosis from diagnosesToBeElicited and store it in diagnosesAlreadyElicited;
35.               Next diagnosis in diagnosesToBeElicited becomes the current diagnosis
36.            END-WHILE
37.
38.          // ABSTRACTION:  Check if the expected symptoms in likely diagnoses are found in patient
39.          //-------------------------------------------------------------------------------------------
40.          WHILE symptomsToBeElicited is NOT empty   DO
41.               IF the current symptom in symptomsToBeElicited is found in patientProfile THEN
42.                    store  the current symptoms  in symptomsFound;
43.               END-IF
44.                  Store the current symptom in symptomsAlreadyElicited;
45.                 Remove the current symptom from symptomsToBeElicited;
46.                 Next symptom in symptomsToBeElicited becomes the current symptom;
47.          END-WHILE
48.
49.
50.    END-FOR EACH
51.
52.    // INDUCTION:  Check if the likely diagnoses meet their diagnostic criteria
53.    //-------------------------------------------------------------------------------
54.     FOR EACH  diagnosis in diagnosesAlreadyElicited DO
55.          IF  the diagnostic criteria of diagnosis are met based on the symptoms stored in symptomsFound THEN
56.                    diagnosis is included;
57.          ELSE
58.                    diagnosis is excluded;
59.           END-IF
60.       END-FOR
61.
62. END
```

Fig. 6. ST algorithm.

For example, consider the diagnosis d_1 = *Major Depression*, s_1 = *Depressed Mood*, and the related symptoms s_1 = *Depressed Mood*, s_2 = *Loss of Motivation*, s_3 = *Weight Loss*, and s_7 = *Low Self Esteem*.

Suppose we have a patient who presents with the above symptoms, each with a different level of severity. Let us assume that the severity of these symptoms (i.e. quantification) corresponds to q_1, q_2, q_3 and q_7 respectively. Using threshold values t_{11}, t_{12}, t_{13} and t_{17} respectively for each of these symptoms in relation to d_1, an example of a diagnostic rule is as follows:

$$IF(q_1>t_{11}\ AND\ q_2>t_{12}\ AND\ q_3>t_{13}\ AND\ q_7>t_{17})\ THEN\ accepted(d_1) = TRUE$$

where $accepted(d_1)$ indicates whether the diagnostic criteria for d_1 is met, resulting in its acceptance (or rejection) as a diagnosis. It may require several such diagnostic rules for each diagnosis, and some of the rules may also require the logical operator OR in addition to AND. Developers may have to consult standard diagnostic manuals (for example, DSM V [20] in psychiatry) when formulating the diagnostic rules.

VI. Discussion

The knowledgebase model and algorithm presented above represent a simplified version of what it is required for effective diagnostic inference in real world settings. Nonetheless, they encapsulate the essential basic characteristics of the reasoning process. This basic structure can be extended and customised according to the characteristics of clinical knowledge in various medical subspecialties (i.e. subdomains). For example, in psychiatry, the knowledgebase may require addition of an extra layer known as *clinical phenomenon* between the symptoms and diagnoses layers [21]. Also, an extra layer of symptom attributes can be added below the symptoms layer, and each symptom can be quantified using the values associated with the related symptom's attributes using mathematical functions that approximate their relationships, as described elsewhere [16].

In addition to searching for diagnoses related to a given symptom based on likelihood, as described in the algorithm, diagnostic reasoning in the real world setting also involves searching for more critical (i.e. associated with relatively worse consequences if undetected) diagnoses even though their likelihoods seem low based on the patient's reported symptoms. The ST algorithm does an exhaustive search, and therefore can be useful in ruling out more critical diagnoses that can present with rather atypical symptoms. It is possible to enhance ST by introducing a critical value δ_j associated with each diagnosis d_j that determines the level of criticality of the diagnosis. Similarly to the *likelihoodThreshold* described previously, a threshold value *criticalityThreshold* can be used to select diagnoses for which δ_j>*criticalityThreshold*.

The next significant challenge to implementing the algorithm in a practically useful expert system is developing and maintaining a sufficiently large knowledgebase. Because of the vastness of the knowledgebase and the amount of manpower and commitment required to develop and maintain it, a sufficient database has been very difficult to achieve using traditional development methods [22]. For example, despite expending nearly 25-30 person years of work, it has still not been able to complete the knowledgebase of INTERNIST-I, an expert diagnostic system in Internal Medicine [1].

Even if the required knowledgebase were implemented, there are yet more challenges. An important challenge is engaging clinicians, who may often feel threatened by medical expert systems on the grounds they may be intended to duplicate and replace some of their skills [23]. The authors have previously discussed these challenges, and introduced a software process model known as a Collaborative and Layered Approach (CLAP) as a strategy to deal with these challenging issues [18].

The main layers and the activities within each layer of CLAP are shown in Fig. 7, and the reader is encouraged to refer to the main paper on this model for more details [18]. The form of ST algorithm introduced in this paper can be considered as the main product of the conceptual layer, which primarily deals with conceptualising the expert medical reasoning process and the knowledgebase, and then translating into a formal model. The societal layer then deals with engaging clinicians in a collaborative development process, and defining the role of the under-development expert system within the complex modern day organisational structure of healthcare services in which it will be used. Finally, the computational layer deals with software and hardware implementation of the expert system. As an important strategy to overcome the difficulty of developing and maintaining the knowledgebase, the CLAP model suggests use of an online collaborative approach [24], which can be realised due to advancement of Internet-based social networking platforms.

VII. Conclusion

Whilst acknowledging the challenges in developing successful medical expert systems, this paper introduced a simplified version of the algorithm and data structures required for implementing an inference engine and knowledgebase, based on a previously introduced diagnostic reasoning model [16]. Even though there are many diagnostic reasoning models that have been previously introduced, the authors claim that the reasoning model on which the algorithm introduced in ST this paper is designed, is more comprehensive in relation to the overall expert diagnostic reasoning process.

Furthermore, the algorithm closely models the recursive steps that are involved in real world diagnostic reasoning, using logical inferences. Because of the complexity and the space required to describe the full algorithm and its implementation, it was necessary in this paper to simplify the

algorithm and knowledgebase described. However, the paper still provides the core structure on which, the full model can be built. As a means to identifying and resolving other challenges associated with the development process, the authors suggest use of the CLAP software process model for developing medical expert systems [18].

APPENDICES

Appendix 1. Java representation of the knowledgebase

```
package diagnosticalgorithm;
import java.util.ArrayList;

/**
 * @author Irosh Fernando
 */
public class Knowledgebase {

   // Declare one-dimensional array of symptoms
    static String symptoms[]= {
      "Depressed mood",      /* 1 */
      "Loss of motivation",  /* 2 */
      "Weight loss",         /* 3 */
      "Fatigue" ,            /* 4 */
      "Chest discomfort",    /* 5 */
      "Worrying thoughts",   /* 6 */
      "Low self-esteem",     /* 7 */
      "Headache",            /* 8 */
      "Loss of appetite",    /* 9 */
      "Hand tremors",        /* 10 */
      "Hypertension",        /* 11 */
      "Dizzinesse"           /* 12 */
     };

   // Declare one dimensional array of diagnoses
    static String diagnoses[]= {
      "Major Depression",               /* 1 */
      "Generalised Anxiety Disorder",   /* 2 */
      "Hyperthyroidism",                /* 3 */
      "Paechromocytoma",                /* 4 */
      "Anaemia",                        /* 5 */
      "Ischaemic Heart Disease",        /* 6 */
     };

  // Declare the knowledgebase as a two dimensional array

    static double diag_symp[][]={
/* symptoms index:  1,   2,   3,   4,   5,   6,   7,   8,   9,   10,  11,  12 */
/* diagnosis index: 1 */ { 0.9, 0.9, 0.6, 0.6, 0.0, 0.0, 0.6, 0.0, 0.7, 0.0, 0.0, 0.0 },
/* diagnosis index: 2 */ { 0.0, 0.0, 0.0, 0.7, 0.6, 0.9, 0.4, 0.6, 0.0, 0.6, 0.0, 0.4 },
/* diagnosis index: 3 */ { 0.3, 0.0, 0.7, 0.6, 0.0, 0.0, 0.0, 0.0, 0.0, 0.8, 0.0, 0.3 },
/* diagnosis index: 4 */ { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.0, 0.0, 0.9, 0.0 },
/* diagnosis index: 5 */ { 0.0, 0.3, 0.0, 0.8, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.6 },
/* diagnosis index: 6 */ { 0.3, 0.0, 0.0, 0.3, 0.8, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.6 },
};

   // return the index of a given symptom
   static public int getSymptomIndex(String symptom){
      // return -1 if not found
      int index=-1;
      for( int i=0; i<symptoms.length; i++){
         if( symptoms[i].equalsIgnoreCase(symptom))
              index=i;
      }
      return index;
   }

  // return the index of a given diagnosis
   static public int getDiagnosisIndex(String diagnosis){
     // return -1 if not found
      int index=-1;
      for( int i=0; i<diagnoses.length; i++){
         if( diagnoses[i].equalsIgnoreCase(diagnosis))
              index=i;
      }
      return index;
   }

   // return all diagnoses related a given symptom above a given threshold
   static public ArrayList getDiagnoses(String symptom, double threshold){
      ArrayList<String> diagnosesList = new ArrayList<>();
      int index= getSymptomIndex(symptom);
      for( int i=0; i< diagnoses.length; i++){
         if(diag_symp[i][index]> threshold )
            diagnosesList.add(diagnoses[i]);
      }
      return diagnosesList;
   }

    // return all symptoms related a given diagnosis above a given threshold
   static public ArrayList getSymptoms(String diagnosis, double threshold){
      ArrayList<String> symptomList = new ArrayList<>();
      int index= getDiagnosisIndex(diagnosis);
      for( int i=0; i< symptoms.length; i++){
         if(diag_symp[index][i]> threshold )
            symptomList.add(symptoms[i]);
      }
      return symptomList;
   }
}
```

Appendix 2. Java representation of the algorithm

```
package diagnosticalgorithm;
import java.util.ArrayList;
import java.util.List;

/**
* @author Irosh Fernando
 * @Date 30th of June 2013
 */
public class STAlgorithm {
   static Knowledgebase KB;
   static PatientProfiles Patient;

   static List<String> symptomsFound = new ArrayList<>();
   static List<String> symptomsToBeElicited = new ArrayList<>();

   // To store both symptoms found and not found
   static List<String> symptomsAlreadyElicited = new ArrayList<>();

   // store diagnose of which symptoms are to be explored
   static List<String> diagnosesToBeElicited = new ArrayList<>();

   // Store diagnoses of which symptoms have already been explored
   static List<String> diagnosesAlreadyElicited = new ArrayList<>();

   // set the likelihood threshold
   static double likelihoodThreshold=0.5;

   // Initialise the symptoms reported by patient at the beginning
   static private void initialise(){
      symptomsFound.add("Depressed Mood");
```

```
    symptomsAlreadyElicited.add("Depressed Mood");
    //...add more symptoms as necessary
}

// Abduction
  static private void doAbduction(){
    for(int i=0; i<symptomsFound.size();i++){
      ArrayList<String> diagList;
      diagList = KB.getDiagnoses(symptomsFound.get(i),
                 likelihoodThreshold);
      // insert each diagnosis into likelyDiagnoses if not already in
      for(int j=0; j< diagList.size();j++){
        if (!diagnosesAlreadyElicited.contains(diagList.get(j)) )
          diagnosesToBeElicited.add(diagList.get(j));
      }
      doDeduction();
      doAbstraction();
    }
  }

  // Deduction
  static private void doDeduction(){
    for(int i=0; i<diagnosesToBeElicited.size();i++){
      ArrayList<String> sympList;
      sympList = KB.getSymptoms(diagnosesToBeElicited.get(i),
                 likelihoodThreshold);
// insert each expected symptom into symptomsToBeElicited if not already in
      for(int j=0; j< sympList.size();j++){
        if (!symptomsAlreadyElicited.contains(sympList.get(j)))
          symptomsToBeElicited.add(sympList.get(j));
      }
      // store alrady found symptoms in symptomsAlreadyElicited
      diagnosesAlreadyElicited.add(diagnosesToBeElicited.get(i));
    }
    // Empty the diagnosesToBeElicited after eliciting all the diagnoses
    diagnosesToBeElicited.clear();
  }

    // Abstraction
  static private void doAbstraction(){
    for(int i=0; i<symptomsToBeElicited.size();i++){

 if( !symptomsAlreadyElicited.contains(symptomsToBeElicited.get(i)) ){
        if(Patient.symptomPresent(symptomsToBeElicited.get(i))){
          symptomsFound.add(symptomsToBeElicited.get(i));
        }
        symptomsAlreadyElicited.add(symptomsToBeElicited.get(i));
      }
  }
// Empty the symptomsToBeElicited after eliciting all the expected symptoms
  symptomsToBeElicited.clear();
}
```

REFERENCES

[1] D. A. Wolfram, "An appraisal of INTERNIST-I," *Artificial Intelligence in Medicine,* vol. 7, pp. 93-116, 1995.

[2] H. Mandin, A. Jones, W. Woloschuk, and P. Harasym, "Helping students learn to think like experts when solving clinical problems," *Academic Medicine,* vol. 72, pp. 173-179, 1997.

[3] A. S. Elstein, L. S. Shulman, and S. A. Sprafka, *Medical Problem-Solving: an Analysis of Clinical Reasoning*: Cambridge, MA: Harvard University Press 1978.

[4] E. Hunt, "Cognitive Science: Definition, Status, and Questions " *Annual Review of psychology,* vol. 40, pp. 603-629 1989.

[5] G. R. Norman, C. L. Coblentz, L. R. Brooks, and C. J. Babcook, "Expertise in visual diagnosis - a review of the literature.," *Academic Medicine,* vol. 66(suppl), pp. s78-s83, 1992.

[6] J. A. Reggia and Y. Peng, "Modeling diagnostic reasoning: a summary of parsimonious covering theory," *Computer Methods and Programs in Biomedicine,* vol. 25, pp. 125-134, 1987.

[7] P. M. Wortman, "Medical Diagnosis: An Information-Processing Approach," *Computers and Biomedical Research,* vol. 5, pp. 315-328, 1972.

[8] J. Stausberg and M. Person, "A process model of diagnostic reasoning in medicine," *International Journal of Medical Informatics,* vol. 54, pp. 9-23, 1999.

[9] E. H. Shortliffe and B. G. Buchanan, "A model of inexact reasoning in medicine," *Mathematical Biosciences,* vol. 23, pp. 351-379, 1975.

[10] S. Andreassen, F. V. Jensen, and K. G. Olesen, "Medical expert systems based on causal probabilistic networks," *International Journal of Bio-Medical Computing,* vol. 28, pp. 1-30, 1991.

[11] T. Chard and E. M. Rubenstein, "A model-based system to determine the relative value of different variables in a diagnostic system using Bayes theorem," *International Journal of Bio-Medical Computing,* vol. 24, pp. 133-142, 1989.

[12] B. S. Todd, R. Stamper, and P. Macpherson, "A probabilistic rule-based expert system," *International Journal of Bio-Medical Computing,* vol. 33, pp. 129-148, 1993.

[13] K. Boegl, K. P. Adlassnig, Y. Hayashi, T. E. Rothenfluh, and H. Leitich, "Knowledge acquisition in the fuzzy knowledge representation framework of a medical consultation system," *Artificial Intelligence in Medicine,* vol. 30, pp. 1-26, 2004.

[14] L. Godo, R. L. de Mántaras, J. Puyol-Gruart, and C. Sierra, "Renoir, Pneumon-IA and Terap-IA: three medical applications based on fuzzy logic," *Artificial Intelligence in Medicine,* vol. 21, pp. 153-162, 2001.

[15] T. Vetterlein and A. Ciabattoni, "On the (fuzzy) logical content of CADIAG-2," *Fuzzy Sets and Systems,* vol. 161, pp. 1941-1958, 2010.

[16] I. Fernando, F. Henskens, and M. Cohen, "An Approximate Reasoning Model for Medical Diagnosis," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. vol. 492, R. Lee, Ed., ed: Springer International Publishing, 2013, pp. 11-24.

[17] M. Ramoni, M. Stefanelli, L. Magnani, and G. Barosi, "An epistemological framework for medical knowledge-based systems " *IEEE Transactions on Systems, Man and Cybernetics,* vol. 22, pp. 1361-1375, 1992.

[18] I. Fernando, F. Henskens, and M. Cohen, "A Collaborative and Layered Approach (CLAP) for Medical Expert System Development: A Software Process Model," in *IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS12)*, 2012, pp. 497-502.

[19] C. S. Peirce, "Illustrations of the logic of science, sixth paper-deduction, induction, hypothesis," *The Popular Science Monthly,* vol. 1, pp. 470-482, 1878.

[20] American Psychiatric Association, *Diagnostic and Statistical Manual of Mental Disorders: DSM-5*: American Psychiatric Publishing Incorporated, 2013.

[21] I. Fernando, M. Cohen, and F. Henskens, "A systematic approach to clinical reasoning in psychiatry," *Australasian Psychiatry,* vol. 21, pp. 224-230, 2013.

[22] E. L. Kinney, "Medical Expert Systems - Who needs them ?," *CHEST,* vol. 91, pp. 3-4, 1987.

[23] A. K. Das, "Computers in Psychiatry: A Review of Past Programs and an Analysis of Historical Trends," *Psychiatric Quarterly,* vol. 73, pp. 351-365, 2002.

[24] D. Richards, "Collaborative Knowledge Engineering: Socialising Expert Systems," in *11th International Conference on Computer Supported Cooperative Work in Design*, 2007.