The inference mechanism for expert system categorizing airborne objects

Jakub Novak[a]

[a] Ing. Jakub Novak, Faculty of Military Technology, University of Defence, Ph.D. student, Brno, Czech Republic. E-mail: kliklong@email.cz

Abstract

This article describes the inferential mechanism designed and written as a part of an expert system for assessment of airborne objects in the assigned airspace. The paper theoretically describes different types of inferential mechanisms and associated methods utilized by them.
The design of the inferential mechanism is described by classes with their attributes and functions (source code). The inference mechanism works in two steps. I describe source code, which evaluates rules from base of rules and the process of identification.

Key words: inference, inference mechanism, expert system, NATINAMDS.

1. Introduction

As the guarantee of the sovereignty of Czech airspace, the Czech Air Force is tasked to prevent military and civil airplanes from attacking areas of interest in the Czech Republic.
This article describes the inference mechanism, which was designed for the identification system (ID system). The system's main task is to categorize airborne objects in Czech airspace. This identification is exercised as the main task of the North Atlantic Treaty Organization Integration Air and Missile Defense System (NATINAMDS). The identification system is designed as a combination of an expert and fuzzy logical system. The inference mechanism is described theoretically and practically in source code.

2. Inference

Term inference means reasoning or derivation definite statements from other statements.

In expert systems, inference presents the process of finding solution consistent with the knowledge base and source information [3]. The inferential process includes algorithms which, with utilizing the knowledge base, modify the database and seek the required solution of the problem. A typical inferential mechanism is based on the inference rule for derivation of new knowledge from the existing knowledge (the knowledge base search strategy).

The inference mechanism can also be based on a special "table type" control mechanism. In rule systems, the inference is constructed by „modus ponens" or „modus tollens" rules.

Modus ponens utilizes direct deduction. If there is a presumption E and a rule $E \rightarrow H$, then the result H is valid.

Modus tollens means backward deduction. If there is a rule $E \rightarrow H$ and the result H is false, then the premise E is also false. These rules are displayed in formula 1 and 2.

$$\frac{E \quad E \rightarrow H}{H} \qquad (1)$$

$$\frac{\neg H \quad E \longrightarrow H}{\neg E} \qquad (2)$$

The rule systems can be implemented as so-called inference nets.

An inference net can be described as reticular graph consisting of nodes and edges.

Single facts are represented by single nodes and single rules are represented by the diagram border.

Inference net implementation is easy, and that is why it is often used for problems with a small number of solutions.

We can specify two fundamental possibilities of deduction.

The first one is derived from data; this type is named "direct deduction".

The second one is derived from solution; this type is named "backward deduction".

Examples of inference methods:

- Deduction – logical deduction, the conclusions have to be consistent with presumptions.
- Induction – the progress from specific to common cases.

- Abduction – from true conclusion to the premise evoking the conclusion.
- Heuristics – „sound thinking“ based on experience.
- Generation and testing – the trial and error method
- Analogy – deduction from other similar situations
- Default inference – deduction based on universal knowledge in case that specific data are not available.
- Inmonotonous inference – correction or revoking of currently valid knowledge is possible based on new evidence.
- Intuition – hardly explainable way of reasoning, whose findings are perhaps based on unknown recognition of a certain design. This type of reasoning has not been implemented so far and is possibly similar to the reasoning in case of neuron networks.

3. The design of the inferential mechanism

The inference mechanism is an important part of an expert system. The main task of inference mechanism is to construct a conclusion of the entire expert system, assigning an identification to every object in the airspace as quickly as possible after the target is detected.

The inference mechanism evaluates information from the database in accordance with rules from knowledge base. Inference mechanism is therefore connected to both database and knowledge base and it modifies data in both bases based on the results (fig. 1).

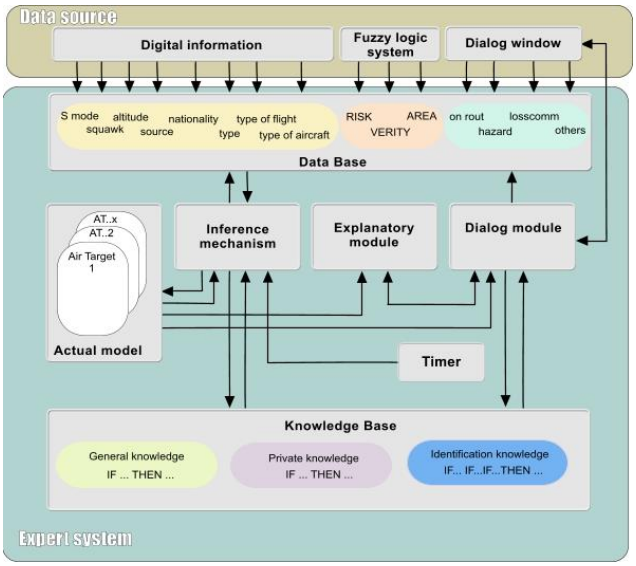

Fig. 1: ID system diagram

Furthermore, the inference mechanism generates an up-to-date model, which represents every airborne object. The inference mechanism works in two steps.

3.1. First step

In first step, rules from knowledge base are evaluated. The inference mechanism's behavior is deductive. That means that it reads every rule from the knowledge base and the rule is broken down to the „ANTECEDENT“ and the „CONSEQUENT“. If the antecedent is true, a command is executed in the consequent. You can see logical operators which can be used to create rules in table 1.

Table 1: logic operator

Logic operators	description
AND	A and B have to be true
OR	A or B have to be true
<	smaller number
>	larger number
=	equals, pertains
!=	not equal
IF	After IF follow antecedent
THEN	After THEN follow consequent
ACTION	After this operator follow commands which change knowledge base, data base
GO	After this operator follow other operator

We can use logical operators in the antecedent as well as in the consequent. The knowledge entered in the form IF antecedent THEN consequent is interpreted by the inferential mechanism. The rules are created from parameters which are present in the database, are part of the respective rules' consequents or that are entered by the operator. The individual parameters can be combined in the rules with the logical operators from (Table1).

The rule assessment is executed by a step by step breakdown to fragmentary rules down to the operator level (=, !=, <, >). Utilizing these operators, the partial rules are assessed, then the original rule is assessed as a whole and if the conditions are met, the operations in the consequent are carried out. The consequent can contain two kinds of activity. The first one is the set-up of any variable of the airborne object. The second option is to remove the defined rule or define a new rule.

The inference mechanism evaluates rules using the breadth-first search method, meaning the rule divides into two or more predecessor branches with respective operators until it gets to the point where there is no operator in any branch.

The breadth-first search method is an algorithm that searches all the vertices of a specific level. The information about the number of vertices in the level is saved in the "Brother" value (class Rootrule).

The next step is to go back from vertices and return to the next level to find out whether the ancestor value is TRUE or FALSE.

When the result of the complete rule is true, the consequent is executed. Examples of rules are in table 2.

Table 2: Demonstration of the rules from knowledge base

IFamode!=7500ANDamode!=7600AND amode!=7700THENID=Friend
IFflynot!=HCANDflynot!=VIPTHEN ID=Friend
IFcratyp!=K35RANDcratyp!=E3THENID =Friend
IFcratyp=K35RORcratyp=E3THEN Action=10
IFamode=7700ORamode=7600OR amode=7500THENAction=9
IFflynot=HCORflynot=VIPAND amode!=7700ANDflrout!=OffTHEN Action=7GOAction=8

The decomposition of rules is shown in the fig. 2. After the decomposition the inference mechanism sets parts of rule true or false.

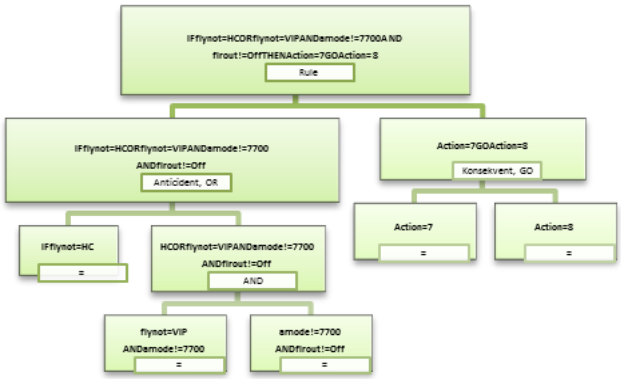

Fig. 2: Decomposition of rules

3.2. *Second step*

The second task is to assign a correct identification to every airborne object. The inference mechanism uses the breadth-first search method.

The backward chaining method is suitable because we have a limited number of hypothesis (identifications of an airborne object) and we have enough information about the airborne object. The system searches for an ID category which would be true for all rules.

The inference mechanism gradually tests every ID category and searches for one with the highest count of true values, in accordance with formula 3.

IDk is a coefficient for every ID category. The most suitable ID category has the IDk coefficient nearing 1.

$$ID_k = \frac{\Sigma r_1}{r_1(True)} \qquad (3)$$

The symbol rI means the number of rules applying to the currently assessed ID category and rI(True) means the number of true rules for the currently assessed ID category.

4. Realization

For the realization of the ID system, I used the C# objective orientation programming language. Attached is the source code, which is used in the inference mechanism of the ID system.

4.1. *Class Rootrule*

This class describes the first step of the inference mechanism. It may be necessary to divide the rules to a logical state as they can be quite complicated, e.g. A=B, A<B, A>B, A!=B, so that the algorithm can assess whether the condition is valid or not. The base rule properties are contained in the Rootrule class with following attributes (source code 1):

- PRE-VALUE defines a rule from which it this instance of class is derived.
- BROTHER defines a number of parts derived from the forepart, which means all the rules at the same level.
- VALUE contains rule text. After evaluation, the value reads either true or false, dependent on whether the condition is fulfilled or not.
- The information about operator, which was used for the rule separation, is saved in attribute TYPE
- SYMBOL defines the operator used for evaluation of this part of rule.

```
    class Rootrule
    {
      private int prevaule;
      private int brother;
      private string value;
      private string typ;
      private string symbol;
      // constructor
          public Rootrule (int prevaule, int
        brother, string value, string typ,string
        symbol)
      public Rootrule()
      //function
      public string WritteRules()
      public string[] Seperate(string typ)
      public string Retvaule()
      public string Retsymbol()
      public string Rettyp()
      public int Retbrother()
      public int Retprevaule()
      public void Setvaule(string vaule)
      public void Setsymbol()
      public void Evaluate(string a, string b,
string symbol)
    }
```

Source code 1: Class Rootrule

We can construct the instant of class either empty or with all defined attributes. For this purpose, I defined two constructors listed in table 3 and in source code 1.

Table 3: Function of class Rootrule

WritteRules	return string “ancestor”; ”Brother”; “value“; “type“; “symbol“
Seperate	divided value by the input parameter type
Retvaule	return value
Retsymbol	return symbol
Rettyp	return type
Retbrother	return number of brother (brother is mean rules in same level)
Retprevaule	return ancestor
Setvaule	set value for instance of this class
Setsymbol	defines a symbol of value for instance of this class
Evaluate	this function evaluate rule and set result false or true

4.2. Class Frontrule

The class instance consists of ANTECEDENT or CONSEQUENT. The rule list contains the rule’s breakdown to the single parts using the different operators. Items in the list are instances of Rootrule class and the parts are indexed. The class is described in fig. 2.

```
class Frontrule
  {
    List<Rootrule> front;
    // constructor
    public Frontrule()
//funkce
    public void Clear()

        public   void   Addrulelogic(int
      prevaule,  int  brother,  string
      vaule, string typ, string priznak)

    public int Retcountrule()
    public pravidlo Retrule(int i)
  }
```

Source code 2: Class Frontrule

I created a procedure Addrulelogic to insert parts of separated rules. The functions of class are described in table 4.

Table 3: Function of class Rootrule

Clear	Clears the contents of the list rules.
Addrulelogic	Add rule to the list rules
Retcountrule	Returns the number of rules in the list, which is mean number Root-
	rule instances in which the rule was distributed.
Retrule	Returns rule of sheet rules according to the index.

Main task of this class is to enable a successive assessment of the rule, which is already divided to the consequent and the antecedent. Antecedent is divided to rules which are assessed or further separated using the logical operation.

5. Main part of inference mechanism and step 2

In the previous text, I described source code, which evaluates rules from base of rules. In this part, I will describe the source code responsible for determination of the resulting identification category (second step of inference mechanism) and all others steps that are carried out by the inference mechanism.

The process of identification runs in a circle to the point when all airborne objects are categorized. This is done by the “for” command (see Source Code 3). After that, an assessment of fuzzy sets (*CountFuzzSetParam()*) is carried out, the results are cleared (*listBox1.Items.Clear()*) and the rules are reloaded (*allrules.Read()*).

The rules are loaded from main window used for identification system control. The program checks the rule format. After the load, the identification is performed.

Function *Identification()* evaluates all rules in relation to data about one specific airborne object. The rules are separated and defined as true or false. When the rule is true, operations contained by the consequent are realized on the next step (*KonsekventAction()*). The operation in consequent change information in data base or it can change rules in knowledge base.

The last main function is *Strategie()*. This function performs backward chaining assessment, loading a possible identification for every airborne object. Additionally, it selects the rules that apply in the specific situation and sets the coefficient to the respective identification. The coefficient is saved and then the process is repeated with another identification. If the latter has its coefficient nearer to 1, the properties of the airborne objects’ identification are changed.

```
   private void btnStart_Click(object sender,
EventArgs e)// will evaluate all the objec-
tives and print the resulting identification
         {
              for (int i = 0; i < Grouptar-
get.Count(); i++)
              {
                  NewTarget = Grouptar-
get.Rettarget(i); //set value and parameters
in data base
                  CountFuzzSetParam();
// evaluate fuzzy sets
                  listBox1.Items.Clear();
// clean results
                  allrules.Read();
// load all rules
                  Identification(allrules);
// evaluate input
                  KonsekventAction();
// removes rules that are inconsistent
                  Strategie();
// for evaluate use method backward chaining
              }

              listTarget.Items.Clear();
              listTar-
get.Items.AddRange(Grouptarget.Rettasklist())
;
         }
```

Source code 3: Main function

6. Conclusion

This article describes the inferential mechanism designed and written as a part of an expert system for assessment of airborne objects in the assigned airspace. The paper theoretically describes different types of inferential mechanisms and associated methods utilized by them.
Furthermore, it describes the functional principle of the inferential mechanism based on its source code. The inferential mechanism works in two steps. In the first step, it works deductively while assessing rules and in the second step, the backward chaining method is used for determining the final identification. The main asset is the description of a practical realization which serves for airborne object identification purposes as a part of a dissertation thesis.

6.1. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have already been defined in the abstract. Abbreviations such as SI, ac, and dc do not have to be defined. Abbreviations that incorporate periods should not have spaces: write “C.N.R.S.,” not “C. N. R. S.” Do not use abbreviations in the title unless they are unavoidable (for example, “CCD” in the title of this article).

6.2. PDF Creation and The Article File Name

The posted article has to be in PDF form. If there is DOC file you can send it too. If you are using other formats, convert it to PDF and send PDF file.

As far as possible, use standard PDF conversion tools Adobe Acrobat give best results. It is important that all fonts be embedded/subsetted in the resulting PDF.

References

[1] KRUPKA, Z., ŘEŘUCHA, V., ŠTEFEK, A. *Automatické řízení 3. díl(B)*. University of Defence, Brno, 2007.

[2] ŘLP ČR, s.p., LETECKOU INFORMAČNÍ SLUŽBOU, *Air information* publication(AIP), URL< http://lis.rlp.cz/ais_data/www_main_control/frm_cz_aip.htm>

[3] Petr Faruzel, *Webový průvodce světem expertních systémů*<]http://faruzel.borec.cz/400.html> [cit. 2014-10-09.]

[4] MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J. A kol.: *Umělá inteligence (3)*, Akademie věd ČR, Praha, 2001.

[5] KOUTSAJANNIS, C., HATZILYGEROUDIS, I., FESMI: *A Fuzzy Expert System for Diagnosis and Treatment of Male Impotence*, KES 2004, LNAI 3214, page 1106–1113, 2004.

[6] NOVÁK, J.: *Využití inteligentních systémů při identifikaci vzdušných objektů*. Písemná zpráva ke státní doktorské zkoušce, University of Defence, Brno, 2011.

[7] NOVÁK, J.: *The use of intelligent systems for the identification of air targets*, International Conference on Military Technologies, Brno, 2011, page 438-444, ISBN 978-80-7231-787-5.

[8] UTTENDORFSKÁ, M.: *Identification automation in the air forces environment*, International Conference on Military Technologies, Brno, 2011, page 823-828, ISBN 978-80-7231-787-5.