Remote Diagnosis of Hypertension through HTML-based Backward Inference

Yong Uk Song[a], Young Moon Chae[b], Kyoung Won Cho[c], Seung Hee Ho[b]

[a]*Department of MIS, Yonsei University Wonju Campus*
234, Maji, Wonju, Gangwon 220-710, Korea
Tel: +82-33-760-2340, Fax: +82-33-763-4324, E-mail: yusong@dragon.yonsei.ac.kr

[b]*Graduate School of Health Policy and Administration, Yonsei University*
134, Shinchon-dong, Seodaemoon, Seoul 120-752, Korea
Tel: +82-2-361-5048, Fax: +82-2-392-8996, E-mail: ymchae@yumc.yonsei.ac.kr
Tel: +82-2-361-5048, Fax: +82-2-392-8996, E-mail: arose@yumc.yonsei.ac.kr

[c]*Vivendi Universal Publishing MEDIMEDIA KOREA*
Clifford Bldg., 1338-23, Seocho-dong, Seoch, Seoul 137-070, Korea
Tel: +82-2-522-6100, Fax: +82-2-522-5907, E-mail: kwcho@kimsonline.co.kr

Abstract

An expert system for the diagnosis and indication of hypertension is implemented through HTML-based backward inference. HTML-based backward inference is performed using the hypertext function of HTML, and many HTML files, which are hyperlinked to each other based on the backward rules, should be prepared beforehand. The development and maintenance of the HTML files are conducted automatically using the decision graph. Still, the drawing and input of the decision graph is a time consuming and tedious job if it is done manually. So, automatic generator of the decision graph for the diagnosis and indication of hypertension was implemented. The HTML-based backward inference ensures accessibility, multimedia facilities, fast response, stability, easiness, and platform independency of the expert system. So, this research reveals that HTML-based inference approach can be used for many Web-based intelligent site with fast and stable performance.

Keywords:

Diagnosis, Backward inference; Decision Graph; HTML

Introduction

Along with the explosive increase of information services using the World Wide Web (WWW), there has been a rapid succession of presentations of practical applications of cases applied to expert systems. (Song and Lee, 2000) Accessibility and easiness of WWW offer a good opportunity for remote diagnosis and indication in medical domains. Traditionally, diagnosis has been a major application domain of backward chaining inference.

Lots of inference engines, which were used for stand-alone environment, have been adapted for WWW environment. The Web-based inference engines uses some up-to-date technologies such as multi-thread, Java, etc., but they suffers from the problems of heavy burden to Web servers and high price. HTML-based inference approach, which was introduced by Song and Lee (2000), offers a good opportunity to Web based inference system of fast response and stability. We developed a Web-based system for diagnosis and indication of hypertension using the HTML-based inference methodology.

The HTML-based approach, however, has the problem of difficulty in development and maintenance. To cope with the problem, we developed two systems: graph generator and HTML generator. Graph generator is a system that draws a graph – decision graph – based on the domain knowledge, and HTML generator is a system that converts the graph to a set of HTML files which will be uploaded to Web servers and used for inference service toward the cyberspace. This paper is organized as follows. In chapter 2, the motivation and methodology of HTML-based approach are introduced. The domain knowledge for the diagnosis and indication of hypertension is described in chapter 3, and the issues and methodology of implementing the system are explained in chapter 4. Finally, the conclusion is addressed in chapter 5.

Web-based Expert Systems

Song and Lee (2000) introduced an HTML-based backward inference mechanism. The development of the diagnosis and indication of hypertension system is based on the approach. So, we explain the HTML-based backward

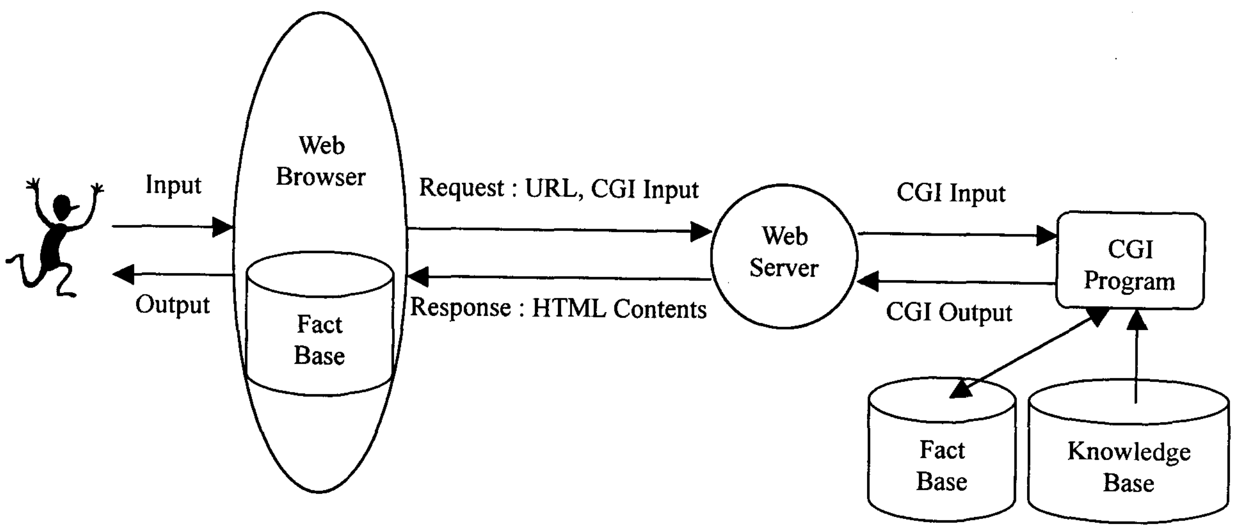


Figure 1 – The case of implementing the inference engine as CGI

inference methodology with the necessity of the approach in this chapter.

Structural Comparison of the Web based Expert Systems

P. C. Kim (1996) divided the integration structures of the Web and the databases and compared the advantages and disadvantages. In this chapter, which expounds on and applies the theory, there is a comparison of the advantages and disadvantages between the structures of the Web-based expert systems. The expert systems in the Web environment generally are classified into that which the inference engine is located in the server side and that which the inference engine in the client side.

The structure where the inference engine is located on the server side

Inference engine in the server side situation can be further divided into three parts. First is the case of using CGI, second is the case of the Web server itself internalizing the inference engine, and third is the case of using HTML as this thesis asserts.

In the case (Figure 1) of using CGI (Dwight, Erwin, and Niles, 1997), the user's input is handled by using an HTML form tag and CGI rules. This method can be used without changing existing technologies, such as Web servers, Web browsers, URL, HTTP, HTML, and CGI. Advantages are that system development and testing are easy, and expanding the system later on, maintenance and support are also easy. The problem with this method is that because of the connectionless and the stateless characteristics of the Web, a separate means must be made to remember the content of the continuing dialogue with the user. Currently, the typical methods in use to do this are the method using a Hidden Form tag control[1] from among the HTML tags, the method that uses a cookie[2], and the method that saves and uses each user's dialogue content in a database. Moreover, when the expert system formed by CGI has the inference engine and internalized knowledge base, and the process gets larger, it leads to using up system resources and lowering performance as the number of users increase and demand service simultaneously. In the case of large-scale service especially, it can be terminal. To overcome this, a method can be used that changes the inference engine to Demon[3], which is not a CGI program (Figure 2). In this method, the lightweight dispatcher process becomes driven by the CGI of the Web server, and in this dispatcher[4] process, a service request is made from the client to the server inference engine, and the result is sent to the user through the Web server.

The CGI program dispatcher in this situation involves no lowered performance due to the large-scale service as the result of the lightweight process. However, to implement

[1] The hidden control is not shown on the monitor in the Web browser unlike other controls, such as the text, button, or check box, etc., of the HTML form tag, but is forwarded to the Web server at the time the values are submitted like other controls. The Web server using this function allows the Web browser to save specified information and then can present it to the Web server when needed.

[2] When a Web browser receives HTML information called a cookie from a Web server, it is information received and stored separately at the Web server's request. The stored cookie is there so the Web browser can present the HTML request to the Web server with the cookie when HTML information is required to the Web server.

[3] Demon is the server program that stands by while awaiting the service request of the client at the server computer. Demon can regulate the system so that it does not fall into an extreme deficiency of resources situation by controlling the service requests of the client based on its subordinate terms.

[4] The dispatcher is demon's client. The dispatcher ensures that system load is not exceeded even though numerous dispatchers are done simultaneously to make the implemented file sizes small because it is a program set up to dialogue with demon.

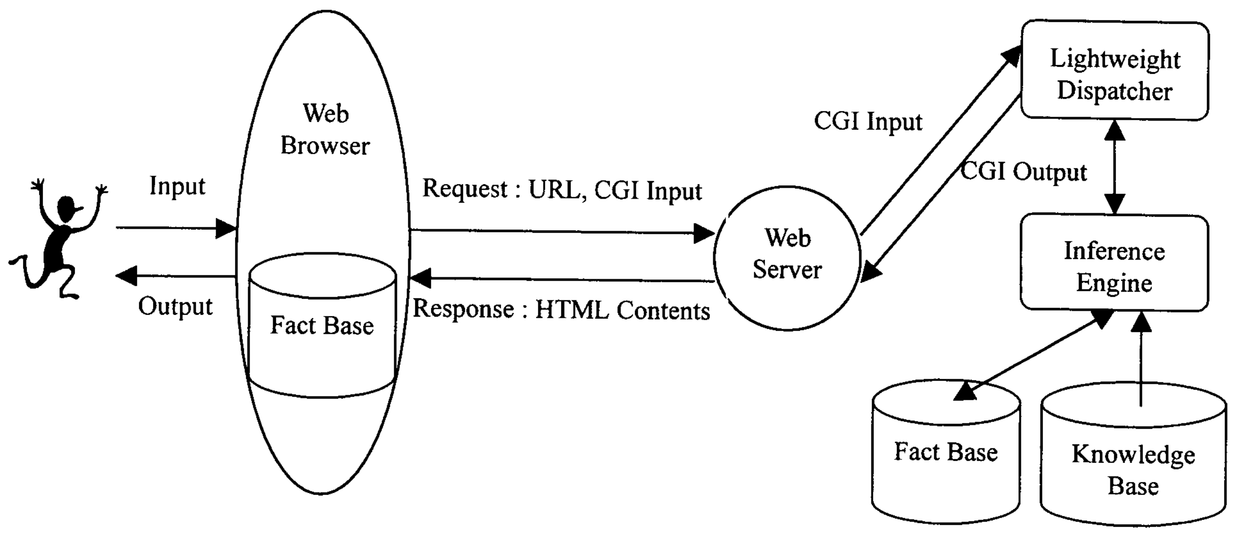


Figure 2 – The case of implementing the inference engine as CGI (large-scale service)

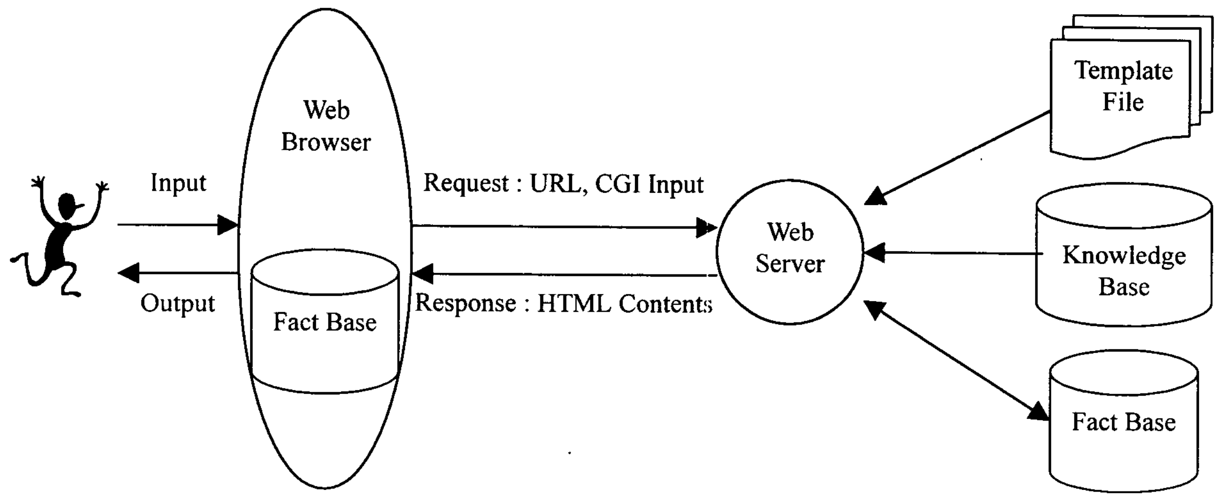


Figure 3 – The case of the Web server containing the inference engine

this method, there must be additional protocols set up between the dispatcher and the server's inference engine, and an additional methodology and program that schedules and distributes resources of the inference engine.

Two methods are possible in the case of the Web server containing the inference engine (Figure 3). One is developing an exclusive Web server for expert systems containing the inference engine, and the second is adding the inference engine to the Web server by using expanding application program interface (API), for example Netscape's NSAPI, etc which is supported by existing Web servers. Because these methods have an inference engine contained within the Web server, development, testing, system expansion, maintenance and repair are not simple. Furthermore, until now, there has not been any research on and nor attempts to develop a method using the inference mechanism contained in a Web server. In the case of the exclusive-use Web server especially, flexibility leaves much to be desired when it comes to future expansion or relocation because it is dependent upon a specific Web server.

Only in the case of using HTML (Figure 4) are such things as rudimentary development, testing, system expansion, maintenance and repair easy because basic HTML technology only is used. The problem of system subordination during large-scale service that occurs when using CGI does not exist, either. The only problem that arises with this method is the extent to which an ordinary expert system can be formed by using HTML only. Following the third chapter of this paper, there is an explanation of structuring backward inference using HTML only.

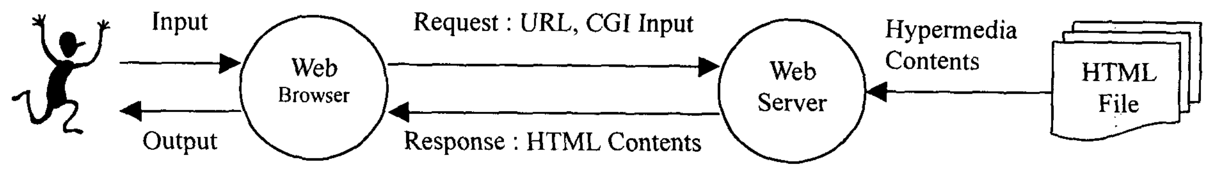


Figure 4 – HTML based expert systems

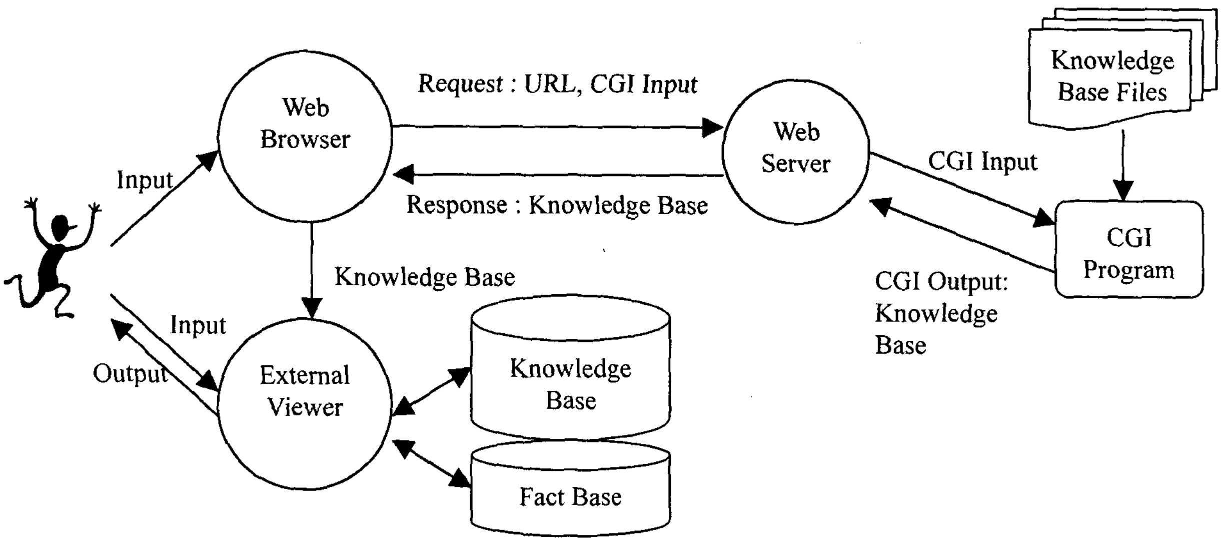


Figure 5 – Structure of an expert system using an external viewer

The structure where the inference engine is located on the client's side

In the case of the inference engine located on the client's side, there are two divisions that can be made. The first is the situation using the external viewer and the second is the situation that includes the inference engine in the Web browser.

The Web browser supports so that an external application program can be driven when the Content-type header[5] value of the message received from the Web server cannot be handled on its own. This external application program is called an external viewer. When using this method, the external viewer has an inference engine, and works when the knowledge base is delivered from the Web server. If additional knowledge is required during the inference, the external viewer connects to a knowledge base server through a separate network, and can acquire the knowledge needed. In the case of this method (Figure 5), the inference engine does not add a burden on the server nor the network because it is completely carried out by one of the application programs resident on the user's computer. Therefore, faster inference results can be attained when the inference engine is resident on the server. Furthermore, no problems occur from the connectionless and stateless characteristics of the Web. However, the problem with this method is that the user must be competent in using the functions of the separate software called "external viewer," and must also download and install the external viewer in advance in order to use it. In particular, various versions of the external viewer must be made to match and operate on specific system platforms. Moreover, system expansion, maintenance and repair are difficult whenever an external viewer is changed and all users should download and install modified software again.

Including the inference engine in the Web browser does not mean programming the inference engine in the Web browser as in containing it in the Web server. This means using Java Applet (Lemay and Cadenhead, 1998). In other words, since there is already an interpreter carrying out Java bytecodes[6] in the Web browser, after the inference

[5] The HTTP response consists of the header and the entity body. Several kinds of information are contained in the header portion, such as type of the content, length, error status, etc. The type of the content—HTML, picture, audio, word processor files, etc.—are represented in the Content-type header. In the case of HTML files or graphic files such as GIF and JPG, they are shown on the window of Web browser after being processed by the Web browser itself, but in the case of the other types, the Web browser does not handle them and must be processed by external application programs.

[6] A characteristic of JavaScript is that it directly executes a resultant file that has been compiled without need of new compiling regardless of what computer is being used. However, because the machine languages are different in individual computers, compiling is done based on a binary code for efficient interpretation by Java interpreter, not a binary machine language that can be executed directly by each computer as in other

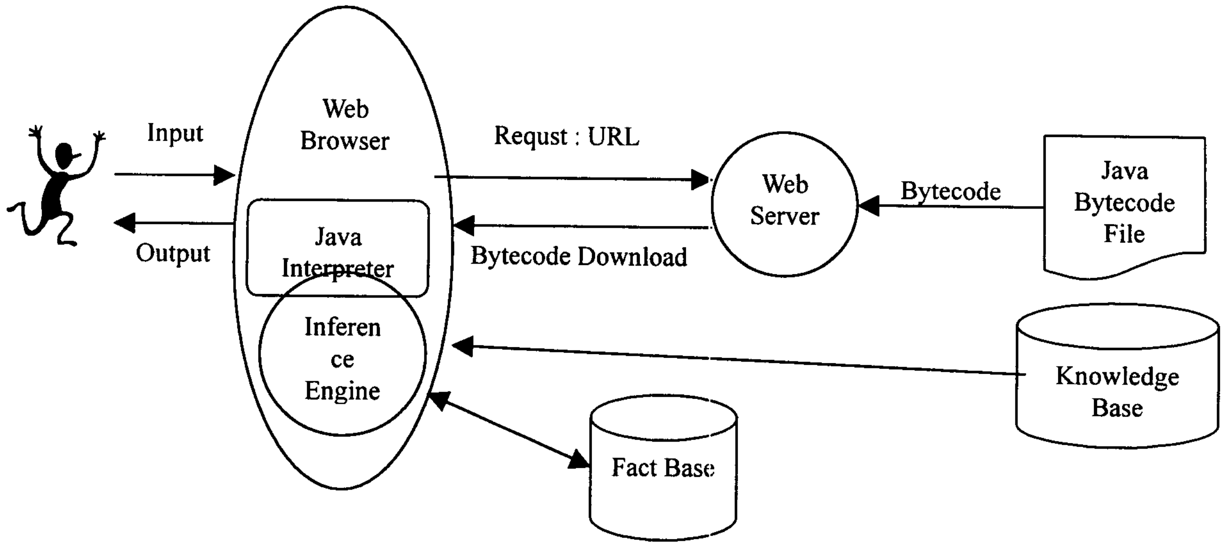


Figure 6 – Structure of an expert system using Java Applet

engine is implemented using the Java Applet, and the Java bytecodes and knowledge base are sent from the Web server to the browser, it is carried out by the Web browser's Java interpreter. (Figure 6) This method solves all of the problems, such as system subordination, installations, etc., occurring in the external viewer. Therefore, developing an inference engine, testing, expansion and even maintenance and repair are easy. At the same time, all of the advantages of the external viewer continue to stay as they are. The disadvantage of this method is that it takes longer to receive the Java bytecodes from the Web server. Moreover, it should be pointed out that because the ability to program in high-level Java language rather than HTML is required, it is comparatively disadvantageous since novice programmers will find it difficult. A comparison of the advantages and disadvantages of each method explained thus far follows in Table 1.

Table 1 – Comparison of Web-based Expert System Structures

	CGI	Web Server	HTML	External Viewer	Java Applet
Development	∇	×	O	∇	∇
Testing	O	×	O	O	O
Scalability	∇	×	O	×	O
Portability	∇	×	O	×	O
Maintenance	O	×	O	∇	O
Large-scale service	×	O	O	O	O
Session	×	×	O	O	O
Simplicity	O	O	O	O	O
Installation	O	O	O	×	O
Starting speed	O	O	O	O	×
Execution speed	×	×	O	∇	O
Rule expressivity	O	O	×	O	O

(Legend) O - Good ∇ - Satisfactory × - Bad

The many general methods now in use are the CGI method, HTML, and Java Applet. Since the development is comparatively simple compared to using Java Applet, and there is no initial download time, the CGI method, which is comparatively superior under the currently slow Internet speeds, is in wide use. (EXSYS) However, when the network speed improves and the initial download time sufficiently shortens, most expert systems will likely operate using the Java Applet method. It appears that interest in the HTML method has fallen off comparatively in contrast with the deepening interest in CGI and Java Applet. As the last line in Table 1 shows, the reason for this is that HTML only is thought to be very difficult when it comes to making an ordinary expert system because of the limitations in representing its rules. Be that as it may, when

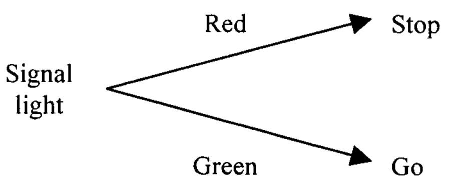
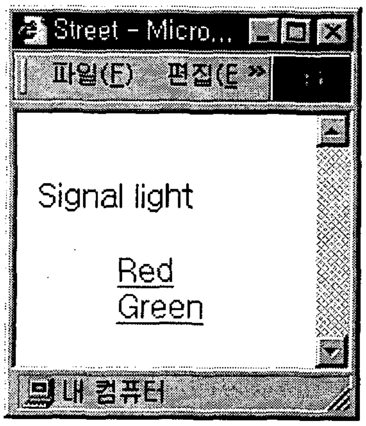
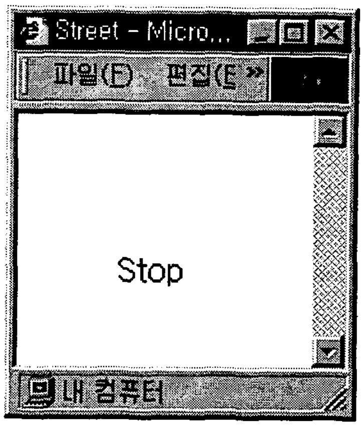


Figure 7 – An Example of a Decision Tree

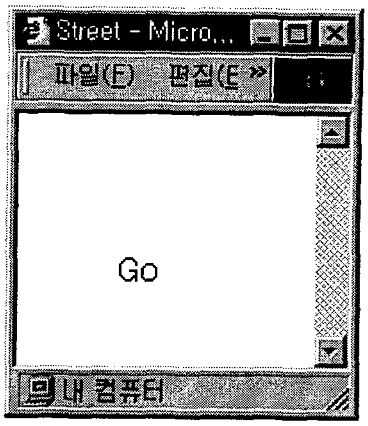
programming languages.



<Figure 9.a>



<Figure 9.b>



<Figure 9.c>

Figure 9 – Decision Tree Formed on the WWW

it comes to focusing on the point that it is possible to form a decision tree and backward inference by the HTML hyperlink function, at a minimum, the HTML expert systems can be set up with regard to such problems as diagnosis and classification using backward inference and such systems have the advantages listed in Table 1.

File: a.html

```
<HTML>
<HEAD> <TITLE> Decision Tree </TITLE> </HEAD>
<BODY>
<BR> Signal light <BR><BR>
<BLOCKQUOTE>
<A HREF="b.html"> Red </A> <BR>
<A HREF="c.html"> Green </A>
</BLOCKQUOTE>
</BODY> </HTML>
```

File: b.html

```
<HTML> <HEAD>
<TITLE> Decision Tree </TITLE></HEAD>
<BODY>
<BR><BR><BR>
<BLOCKQUOTE>
Stop
</BLOCKQUOTE>
</BODY> </HTML>
```

File: c.html

```
<HTML> <HEAD>
<TITLE> Decision Tree </TITLE></HEAD>
<BODY>
<BR><BR><BR>
<BLOCKQUOTE>
Go
</BLOCKQUOTE>
</BODY> </HTML>
```

Figure 8 – HTML File for the Decision Tree

Therefore, beginning in chapter 3, which follows, an explanation of implementing backward inference using HTML is presented.

HTML and Decision Trees

It is possible to construct a decision tree when using an HTML hyperlink. An example of a decision tree is depicted in Figure 7.

The above decision tree can be formed on the Web as shown in Figure 9 when a Web site is set up with the three HTML files shown in Figure 8. The color "Red" hypertext in Figure 9.a becomes the hyperlink in Figure 9.b, and the color "Green" hypertext becomes the hyperlink in Figure 9.c.

Moreover, HTML files making up the decision making process do not necessarily have to take the structure of a tree, but can have the structure of a graph as well. When taking advantage of this characteristic, backward inference can be attained through arranging HTML files as hyperlinks. This is explained more fully in chapter 4.

Backward Inference Based on HTML

Considering the rules in the knowledge base used for the backward inference from the chaining structure point of view, they can be understood using a decision graph[7]. As shown in Figure 10, rules having several conditions that are connected according to AND, OR, and NOT, etc., can be converted into a decision graph. Therefore, as was discussed in chapter 3 above, it is possible to form backward inference by merely arranging the hyperlinks as HTML files. The decision tree shown in Figure 7, for example, expresses the same rules shown in Figure 11, and can be structured by the HTML files shown in Figure 8. Note that each node of the decision graph corresponds to each individual HTML file. Additionally, to express this, each node of decision graph is expressed as an image of the flow chart as shown in Figure 10.

[7] Decision graph is the graph form of decision tree.

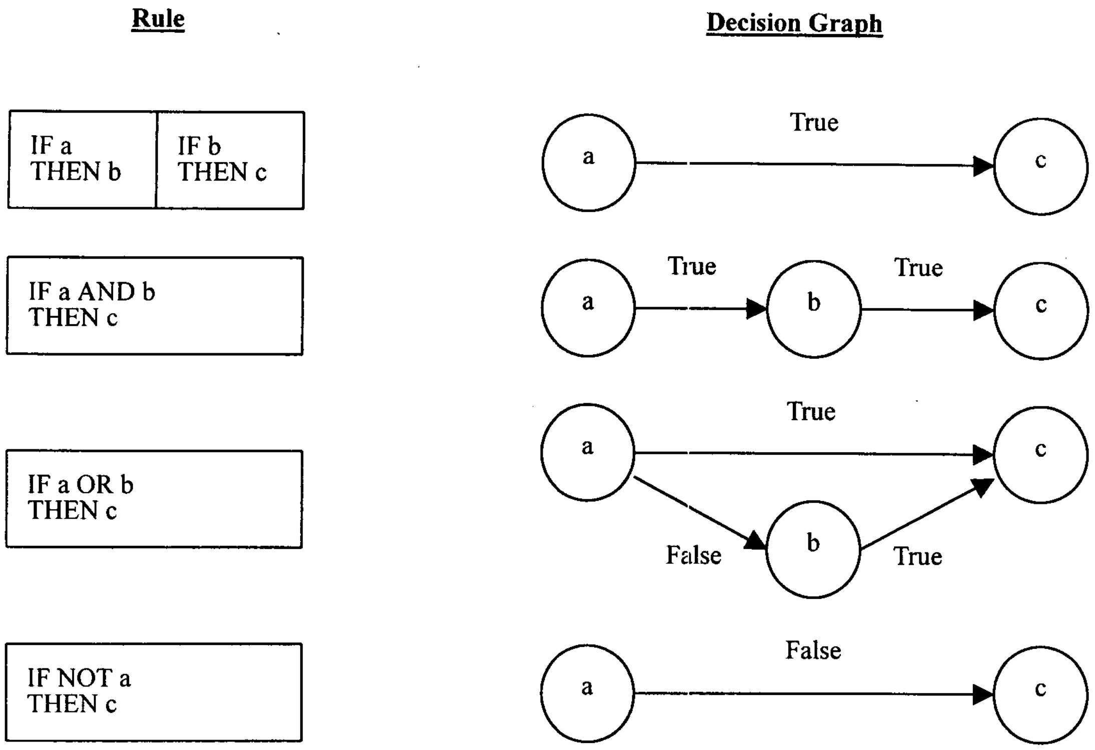


Figure 10 – Decision Process Chart: Rule Conversion

```
If signal-light IS red
THEN stop

If signal-light IS green
THEN go
```

Figure 11 – Example of the Rule Base

To implement backward inference, such things as fact input, chaining based on subgoals, and output of inference results has to be supported. Chaining based on subgoals can be formed by the hyperlink as discussed above, and inference output results will work if the Web browser is made to display the HTML files. Therefore, the remaining problem is a method that supports fact input. The value is saved to a variable in backward inference. The format of the variable from backward inference can be of three types: fact type, object-attribute-value type (OAV), and numeric type. The variable "signal light" in the rule base in Figure 11 is the OAV type variable having a value of "Red" or "Green," and this is formed as in Figure 9.a. This also works in the case of the fact type variable as well when designated "true" or "false." For example, the first screen image in Figure 13 depicts the condition of the rules that the fact type variables shown below generate.

```
IF "Does the animal have hair?" IS TRUE
THEN ...
IF "Does the animal have hair?" IS FALSE
THEN ...
```

The numeric type is not solved by a simple enumeration of the hypertext because calculation must occur after the input vale is received. One way is structuring the CGI to handle numeric type variables, but a more simple and effective way is using such client-side script language[8] as JavaScript (Danesh, 1996) or VBScript (Jerke et al., 1997). The following is an example of those rules.

```
IF total_income >= 0.2 * threshold
THEN pay_tax
IF total_income > 0.2 * threshold
THEN do_not_pay_tax
```

[8] Script language is a programming language that directly operates while interpretation takes place by the interpreter, not the compiler. In the Web environment, there is a script language that interprets and operates in the Web server and a script language that interprets and operates in the Web browser. JavaScript and VBScript, which are the typical client-side script languages, operate via the interpret function within the Web browser and show the results in the Web browser.

```
<HTML>
<HEAD>
<TITLE> Expression Type </TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function verifyValue(form)
{
if (form.total_income.value >= 0.2 * form.threshold.value)
     location="http://iis.kaist.ac.kr/tax.html"
else
     location="http://iis.kaist.ac.kr/notax.html"
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="exprform">
Please input followings. <BR>
total_income:
<INPUT TYPE="text" NAME="total_income"> <BR>
threshold:
<INPUT TYPE="text" NAME="threshold"> <BR>
<INPUT TYPE="button" VALUE="OK" onClick=
"verifyValue(this.form)">
</FORM>
</BODY>
</HTML>
```

Figure 12 – Implementation of Numerical-type variables using JavaScript

The variable "total_income" and "threshold" in the above two rules are numeric type variables. HTML files using JavaScript as in Figure 12 can set up these two rules. As shown in Figure 12, the value of all variables occurring in the conditions set by the rules are received and input by the Form tag, and the hyperlink, based on the evaluation of a numerical expression that includes the input variables, is set within the script tag. Another advantage of this method is that the numerical expression of the conditions using JavaScript, one of the programming languages, can be all kind of numerical expressions. This is shown in the third screen image in Figure 13 as an example of forming an inquiry screen for each variable.

One limitation from a functional aspect in handling the numerical-type variable using JavaScript is that variables used in the condition part of rules cannot be used again in the conclusion part of the rule. This also means that the value of the variable received for input in the condition part of a rule cannot be delivered to the condition part of another rules. Consider the following example.

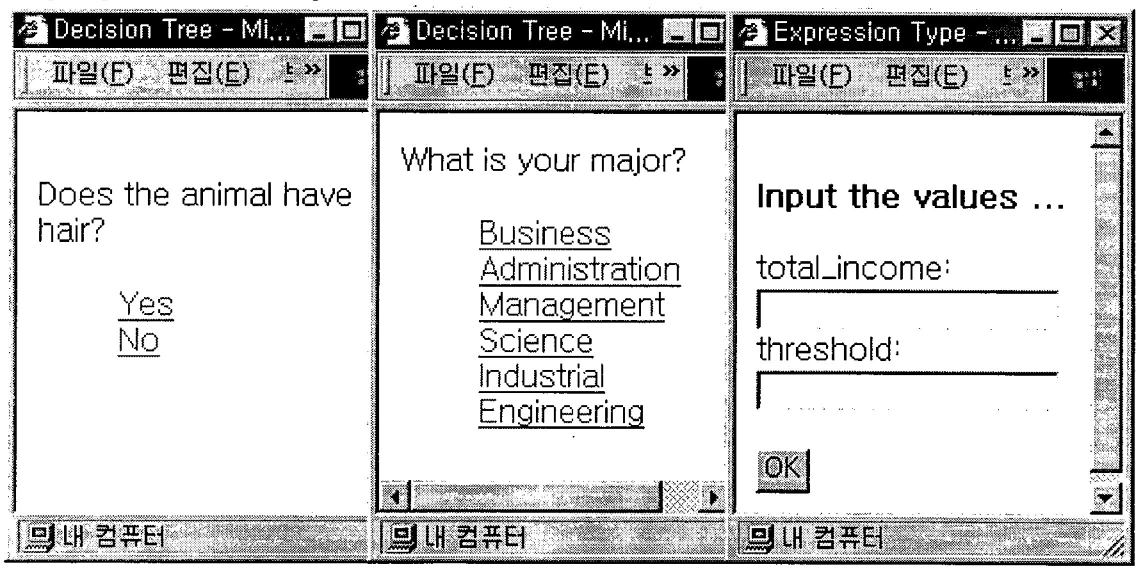
IF total_income < 0.2 * threshold
THEN DISPLAY "TOTAL INCOME must be larger than" + 0.2 * threshold

In the above example, a message is displayed for the user after the value of the "total_income" and "threshold" variables are entered, enumerated and compared. The "threshold" variable in the message is to be noted. Rules such as in the above example cannot be formed because the condition and conclusion parts are handled as a separate HTML file, and there is no method to allow sending the value of the "threshold" variable between the two files. Only when using CGI is this possible, but once again the disadvantages of using CGI apply.

Diagnosis and Indication of Hypertension

For the diagnosis and indication of hypertension disease, we use the guideline for hypertension indication. The guideline is described in Table 2.

The first column is the name of the symptom, and the



Does the animal have hair?

Yes
No

What is your major?

Business Administration
Management Science
Industrial Engineering

Input the values ...

total_income:

threshold:

Figure 13 – Inquiry screens by variable type

second column is the criteria to determine the symptom. The third column is the most preferable indication, and the fourth column is the second best indication. The contra indication is the indication that should not be taken.

For example, the first line of the guideline says that:

(1) *If SBP value of a patient is larger than or equal to 140 and DBP value is less than 90, the symptom is the systolic hypertension.*

(2) *If a patient shows the symptom of systolic hypertension, then the most preferable indication is the use of the Diuretics and Ca.*

However, a hypertension patient may have several symptoms. So, there are another rules for showing several symptoms simultaneously. The rules are summarized as follows:

Single condition

If the patient shows a single symptom, use the first or second indication as explained above. The choice is dependent on the doctor.

Complex condition

If the patient shows several symptoms simultaneously, follow the next rules:

① *In the case where there are common drugs among the first or second indications.*

If there are common drugs among the first indications, then use the common drugs.

If there are not common drugs among the first indications but there are common drugs among the first and second indications, then use the common drugs.

② *Otherwise, use all the drugs in the first and second indications except the drugs in the contra indications.*

(b) In the case where there are no common drugs among the first or second indications.

If there are no common drugs among the first or second indications, use all the drugs in the first and second indications except the drugs in the contra indications.

No complication

If the patient shows no symptom, use the first indication at the last line of the table.

Pre-generation Approach and Implementation of the System

The domain knowledge explained in chapter 3 can be easily represented by rules for backward chaining inference. For example, if a patient shows a single symptom, systolic hypertension, the rule for the case could be made as follows:

```
IF SBP ≥ 140 and DBP < 90
AND not(Age > 70)
AND not(Chest PA = cardiomegaly and EF ≤ 50)
AND not(LVH IS TRUE)
AND not(Protein(U/A) ≥ 1)
AND not(Glucose(AC) > 140)
AND not(EKG = ischemia pattern)
AND not(T.chol > 250)
AND not(Phx = VD or Complication = VD)
AND not(K ≥ 5.5)
AND not(Phx = Gout or Complication = Gout)
THEN
DISPLAY "1st Indication : Diuretics, Ca; 2nd Indication :
        -; Contra Indication : -"
```

However, there are some problems from the implementation point of view. The following characteristics are required for the remote diagnosis and indication system:

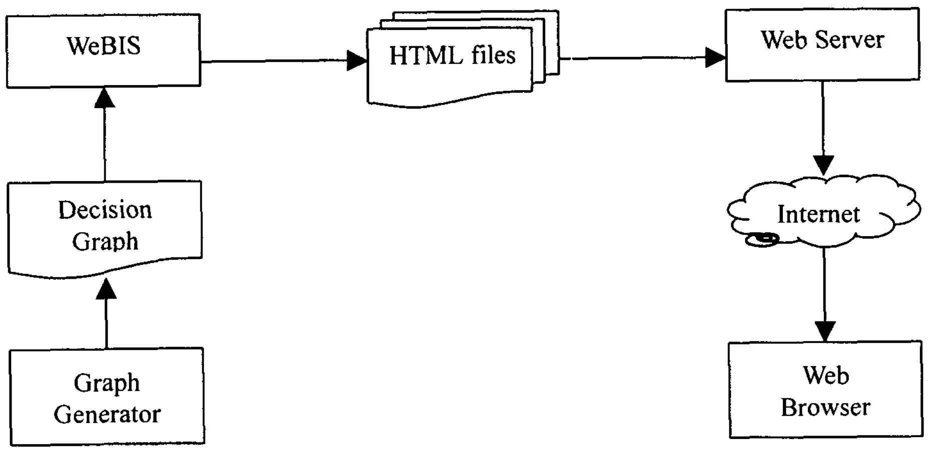


Figure 14 – Overall Architecture of the System for Diagnosis and Indication of Hypertension

Accessibility: it is accessible from anywhere and at anytime

Easiness: it is easy to use

Multimedia facilities: it supports multimedia

Platform independency: it is independent of platform

Fast response: response time is short

Stability: it is stable and free from system down

The first four characteristics – accessibility, easiness, multimedia, and platform independency – can be achieved by using WWW. Almost all computers have Web browsers and are connected to Internet today. Web browsers are familiar to all users and so easy to use, support multimedia contents, and are independent of platform. The remaining characteristics – fast response and stability – are dependent of Web servers. We have already compared the response time and stability among different structures of Web-based expert systems. As we mentioned above, HTML-based approach is the most preferable approach if the rules can be represented by HTML files. But, as in the above example about the systolic hypertension rule, which will be represented by eleven HTML files for question and one file for conclusion, the rules for the diagnosis and indication of hypertension can be represented by HTML files.

Another problem in implementing the system is the size of the rule base. Since there are eleven symptoms and a patient may show the combination of the eleven symptoms, the rule base consists of 2048 ($= 2^{11}$) rules. Hence, the system would be made of 4095 ($= 2^{12} - 1$) HTML files including file for question. So development and maintenance of the files would be very time-consuming and tedious job.

So we developed another system that supports development and maintenance of the HTML files for the diagnosis and indication of hypertension. The overall structure of our approach is depicted in Figure 14.

In Figure 14, WeBIS (Web-based Inference System) generates HTML files from a decision graph, and the HTML files are delivered to the Web browsers by the Web servers through the Internet. Since the HTML files are hyperlinked to each other based on the decision graph, the users who are using the Web browsers are conducting the inference for the diagnosis and indication of hypertension.

The problem of development and maintenance of large number of HTML files is solved by the Graph Generator in Figure 14. The Graph Generate generates the decision graph automatically, and the WeBIS generates HTML files from the decision graph in turn. Figure 15 shows the decision graph generated by the Graph Generator, and Figure 16 shows the WeBIS generating HTML files.

Uploading the HTML files generated by WeBIS, we can service through the Internet the Web site for the diagnosis and indication of hypertension. Figure 17 and Figure 18 shows the screen images of the Web site. Figure 17 shows an screen to input an answer for a question, and Figure 18 shows an screen for the concluded indication.

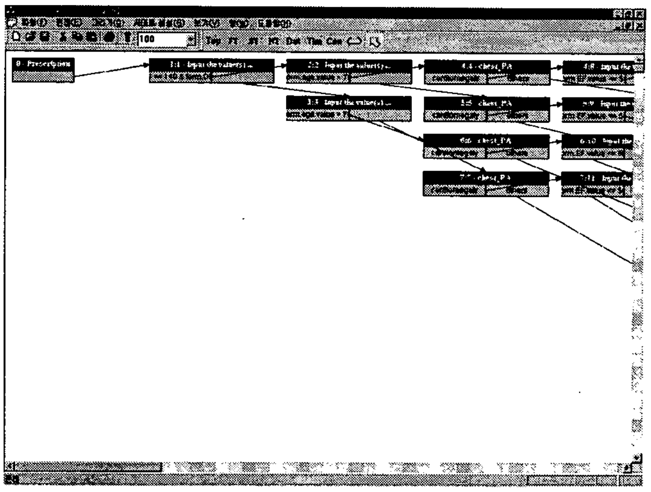


Figure 15 – Generated Decision Graph

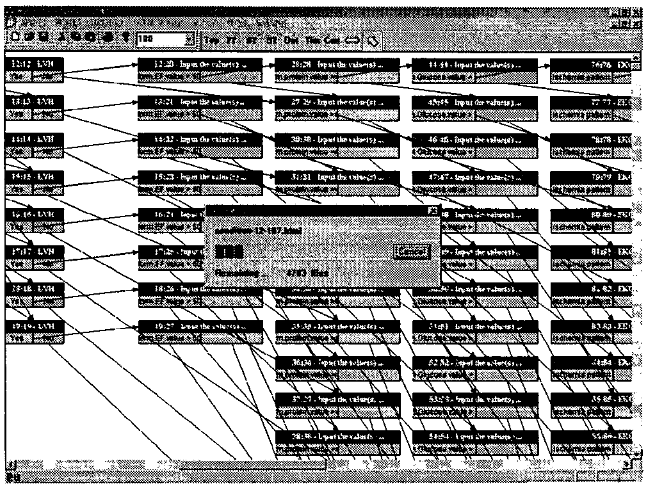
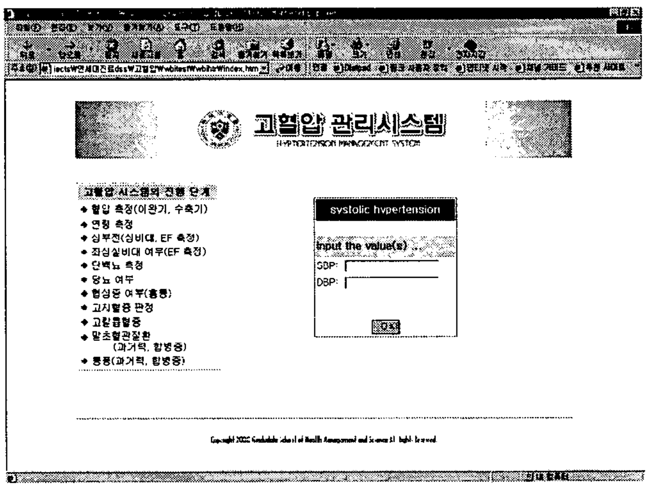


Figure 16 – WeBIS Generating HTML Files


고혈압 관리시스템

HYPERTENSION MANAGEMENT SYSTEM

고혈압 시스템의 진행 단계

- 혈압 측정(이완기, 수축기)
- 연령 측정
- 심부전(심비대, EF 측정)
- 좌심실비대 여부(EF 측정)
- 단백뇨 측정
- 당뇨 여부
- 협심증 여부(흉통)
- 고지혈증 판정
- 고칼륨혈증
- 말초혈관질환 (과거력, 합병증)
- 통풍(과거력, 합병증)

systolic hypertension

Input the value(s) ..

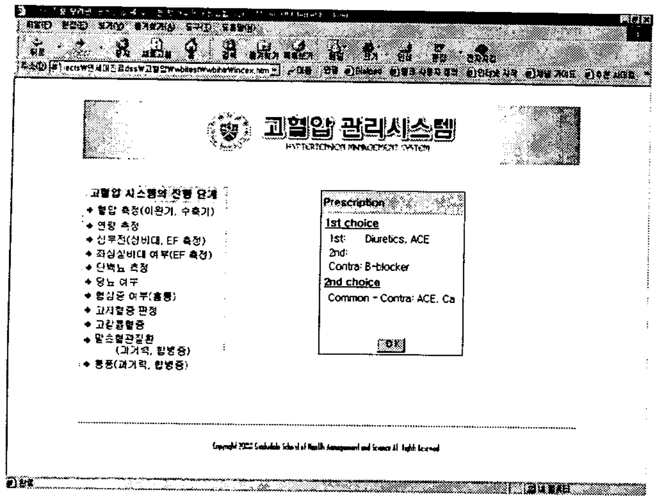
SBP:

DBP:

OK

Copyright 2002 Graduate School of Health Management and Science All rights reserved.

Figure 17 – Screen for Input



Prescription

1st choice

1st: Diuretics, ACE

2nd:

Contra: B-blocker

2nd choice

Common – Contra: ACE, Ca

Figure 18 – Screen of Indication

Conclusion

An expert system for the diagnosis and indication of hypertension is implemented through HTML-based backward inference. HTML-based backward inference is performed using the hypertext function of HTML, and many HTML files, which are hyperlinked to each other based on the backward rules, should be prepared beforehand. The development and maintenance of the HTML files are conducted automatically using the decision graph. Still, the drawing and input of the decision graph is a time consuming and tedious job if it is done manually. So, automatic generator of the decision graph for the diagnosis and indication of hypertension was implemented. The HTML-based backward inference ensures accessibility, multimedia facilities, fast response, stability, easiness, and platform independency of the expert system. So, this research reveals that HTML-based inference approach can be used for many Web-based intelligent site with fast and stable performance.

References

[1] Danesh, Arman, Teach Yourself JavaScript in a Week, Sams.net Publishing, 1996.

[2] Dwight, Jeffry, Michael Erwin, and Robert Niles, Special Edition Using CGI, Second Edition, QUE, 1997.

[3] Eriksson, Henrik, "Expert Systems as Knowledge Servers," IEEE Expert, pp. 14-19, June 1996.

[4] EXSYS, Inc., Moving an EXSYS Application to the EXSYS Web Runtime Engine (WREN).

[5] Far, Behrouz H. and Zenya Koono, "Ex-W-Pert System: A Web-Based Distributed Expert System for Groupware Design," Expert Systems With Applications, Vol. 11, No. 4, pp. 475-480, 1996.

[6] Fielding, R., J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, Hypertext Transfer Protocol -- HTTP/1.1, INTERNET-DRAFT <draft-ietf-http-v11-spec-rev-05>, Internet Engineering Task Force, 11 September 1998.

[7] Giarratano, Joseph and Gary Riley, Expert Systems: Principles and Programming, 2nd Edition, PWS Publishing Company, 1994.

[8] "Intelligent Internet Systems: Tools and Applications – Part I," Intelligent Software Strategies, December 1996.

[9] "Intelligent Internet Systems: Tools and Applications – Part II," Intelligent Software Strategies, Winter 1997.

[10]"Java Expert Systems Tools," Intelligent Software Strategies, Summer 1997.

[11]Jerke, N., M. Hatmaker, J. Anderson, VBScript Interactive Course, Waite Group Press, 1997.

[12]Kim, P. C., "System for the Classification of Database Paths on the Internet (in Korean)," WWW 96-1, pp. 50-66, 1996.

[13]Lee, J. K., et al., Development of an Expert System for Accepting Automobile Applicants (in Korean), Dongbu Insurance Co, November, 1995.

[14]Lee, J. K., H. R. Choi, H. S. Kim, M. S. Suh, S. C. Chu, and W. C. Jhee, Principle and Development of the Expert System (in Korean), Bobyoungsa, 1996B.

[15]Lee, J. K., I. K. Lee, and H. R. Choi, "Automatic rule generation by the transformation of Expert's Diagram: LIFT," Int. J. Man-Machine Studies, 32, pp. 275-292, 1990.

[16]Lee, J. K., Y. U. Song, S. B. Kwon, W. Kim, and M. Y. Kim, Development of an Expert System Using UNIK (in Korean), Bobyoungsa, 1996A.

[17]Lemay, Laura and Rogers Cadenhead, Teach Yourself JAVA 1.2 in 21 Days, SAMS Publishing, 1998.

[18]Lim, K. K., J. Y. Kang, and J. K. Lee, "Structure and Analysis of Web-based Expert Systems (in Korean)," Proceedings of the '97 Fall Conference of the Korea Expert Systems Society, pp 63-73, 1997.

[19]O'Leary, Daniel E., "The Internet, Intranets, and the AI Renaissance," IEEE Computer, January 1997, pp. 71-78.

[20]Raggett, Dave, Arnaud Le Hors, and Ian Jacobs(ed.), HTML 4.0 Specification REC-html40-19980424, W3C, 24 April 1998.

[21]"SELECTICA: Java-based Configuration for Internet and Electronic Commerce Applications," Intelligent Software Strategies, October 1996.

[22]Song, Y. U. and J. K. Lee, "Automatic Generation of Web-based Expert Systems (in Korean)," Journal of Intelligent Information Systems, Vol.6, No.1, pp. 1-16, 2000.

[23]Amzi! URL → http://www.amzi.com/

[24]Applied Logic Systems URL → http://www.als.com/

[25]Attar Software URL → http://www.attar.com/

[26]Bowne Internet URL → http://www.bowneinternet.com/

[27]Brightware URL → http://www.brightware.com/

[28]Gensym URL → http://www.gensym.com/

[29]ILOG Inc. URL → http://www.ilog.com/

[30]Inference URL → http://www.inference.com/

[31]IntelliSystems URL → http://www.intellisystems.com/

[32]JESS URL → http://herzberg1.ca.sandia.gov/jess/

[33]MultiLogic URL → http://www.multilogic.com/

[34]Neuron Data URL → http://www.neurondata.com/

[35]RadNet URL → http://www.radnet.com/

[36]Selectica URL → http://www.selectica.com/

[37]The Molloy Group URL → http://www.molloy.com/home.html