Rule- and Model-Based Expert Systems
How to deal with negations?

A Talk to PhD Students
at the IX International PhD Students Workshop,
Wisła-Głębce, October 21, 2007

Antoni Niederlinski

Silesian University of Technology
Gliwice, Poland

Antoni.Niederlinski@polsl.pl
Details may be found

on my Expert Systems Website:

http://www.rmes.pl
Welcome to Expert Systems by Professor Niederlinski

You may:
- download four expert system shells,
- download four pdf presentation files,
- have a look at four Handbooks based on HTML HELP CHM files (use WINDOWS\hs.exe for opening),
- download some publications

Prof. zw. dr hab. inż. Antoni Niederlinski Antoni.Niederlinski@poliba.pl is active at the Institute of Automatic Control at the Silesian University of Technology in Gliwice and at the International School of Business, Design and Technology in Belsko-Biala. His long-time control-engineering activities resulted among other in designing a measure of interaction in multivariable control plants (known as the Niederlinski Index) and in generalizing the Ziegler-Nichols method for tuning multivariable controllers. He authored the following books (in Polish):
- "Multivariable Control Systems" ("Układy wielowymiarowe automatyki"), WNT, 1974
- 2-volumes of "Digital Industrial Control Systems" ("Systemy cyfrowe automatyki przemysłowej"), WNT, 1977
- "Systems and Control: An Introduction to Control and Technical Cybernetics" ("Systemy i sterowanie. Wstęp do automatyki i cybernetyki technicznej") PWN 1983
- 2-volumes of "Industrial Computer Control Systems" ("Systemy komputerowe automatyki przemysłowej"), WNT 1984,1985
- "Adaptive Control" ("Regulacja adaptacyjna"), PWN 1995 - with J. Mończak and Z. Ogórowski
- "Multi-Edip, A Multivariable System and Signal Analyzer" ("Multi-Edip Analiza wielowymiarowych sygnałów i obiektów") Pol.], 1997 - with J. Kasprzyk and J. Figwer
- "Rule-based Expert Systems" ("Regułowe systemy ekspertowe") PKUS 2000
- "Rule- and Model Based Expert Systems rmes" ("Regułowo-modelowe systemy ekspertowe rmes") PKUS 2005
Rule- and Model-Based Expert Systems

Polish website: http://www.rmse.pl

Antoni Niederliński
Regułowo - modelowe systemy ekspertowe rmse

Polish book
PKJS, Gliwice, 2006,
450 pp., 49 zł
Goal

To discuss techniques of dealing with negations in automated reasoning.

To present some examples.
Introduction to Exact Expert Systems

Let’s start with some jokes!
What is an expert?

Expert - a man who knows everything about something and nothing about anything else.

Ambrose Bierce,

“The Devils Dictionary”
What is an expert?

Expert - a man who knows more and more about less and less.

Nicholas Butler
What is an expert?

Expert - a man who has made all the mistakes which can be made in a very narrow field.

Niels Bohr
What is an expert?

Expert - a man who knows some of the worst mistakes that can be made in his subject, and how to avoid them.

Werner Heisenberg
Expert systems share with human experts an important feature:

The narrowness of their expertise
Basic definition

**Expert systems** - tools for modelling expert decision making, structured as:

\[
\text{Expert System} = \text{Inference System} + \text{Knowledge Base}
\]

- **Inference System** - an executable program
- **Knowledge Base** - a set of text files
Remaining expert system elements

- **Knowledge base editor** – to create, modify and delete knowledge bases
- **User interface** – to communicate with the expert system
- **Dynamic data base** - a relational data base for storing user replies and results of inference
Software Structure of Expert Systems

User Interface

Knowledge Base Editor

Knowledge Base

Inference System

Dynamic Data Base

Text File

Exe file (Expert System Shell)
The essence of expert systems

The knowledge base is a text file:
- to be created,
- to be read,
- to be modified

using the knowledge base editor, without jeopardizing the integrity of the inference system.
The essence of expert systems

This:

• gives quick insight into the domain knowledge used by the expert system,

• contributes towards transparency of the domain knowledge

• contributes towards modifiability of the domain knowledge
Why do we use expert systems?

Not because they are "intelligent"!

But because:

• they allow sophisticated programs ("applications") to be efficiently and cost-effectively designed by "domain experts"

• their knowledge base may be understood by "domain experts"

• their knowledge base may be easily modified by "domain experts"
The talk is based on:

rmes

Rule and Model-based Expert Systems

A family of four expert system shells for a broad range of applications, developed by A. Niederlinski
KNOWLEDGE BASE STRUCTURE FOR

rmes

Knowledge base

- Rule base
- Constraint base
- Advice files

- Model base
- Advice base
KNOWLEDGE BASE STRUCTURE FOR *rmes*

- Rule base – logical domain knowledge
- Constraint base – logical domain knowledge
- Model base – mathematical domain knowledge
- Advice base
  - Advice file
  - Complementary and explanatory knowledge
KNOWLEDGE BASE STRUCTURE FOR rmepes

- Rule base
- Constraint base
- Model base
- Advice base

consists of clauses with variables
Variables used in rmes shells

String variables only!

String variable:
any string of characters beginning and ending with upper quotation marks "."
String variables

String variable = "Inclusion"

Inclusion is inside the quotation marks
String variables used in rmes shells

String variables may be:

- logical
- real
- integer
- uncertain
String variables used in rmes shells

Logical string variables:

"Name" is a logical string variable, if Name is instantiated with elements of the set of logical constants \{True, False\} only.
Real string variables:

"Value" is a real string variable, if Value is instantiated with elements of the set of real numbers only.
String variables used in rmes shells

Integer string variables:

"Value" is an integer string variable, if 
Value is instantiated with elements of the set of integer numbers only.
String variables used in rmes shells

Uncertain string variables:

"Value" is an uncertain string variable, if Value is instantiated with reals from the continuum [-1,..., 1] only.
Instantiation of variable:

Assigning a particular value to the variable
Advantages of string variables:

Readability:

1) Inclusions (names of conditions and conclusions) need not be restricted to single atoms, e.g.:
   "Excellent financial standing"

2) Inclusions (names of arguments and results) need not be restricted to single atoms. e.g.:
   "financial indicator"

3) Inclusions may start with large or small letters or digits.
Rule Base

Rules = general domain knowledge

Facts = particular domain knowledge

Any Knowledge Base must contain a Rule Base.
Rule Bases are indispensable!
Rule example in natural language:

If
"student x scored all subjects in time"
and
"student x passed all exams in time",
then
"student x is registered for next semester"

Fact example in natural language:
"student J.Kowalski passed all exams in time"
**Rule Base**

**Rules**: conditional clauses, e.g.
- in natural language:
  
  A is true if B and C and D are true

- in logic:
  
  \[ B \land C \land D \Rightarrow A \]

- in Prolog:
  
  \[ A \leftarrow B, C, D \]

B, C, D – rule **conditions**
A – rule **conclusion**
\[ \Rightarrow - \text{implication in logic} \]
\[ \leftarrow - \text{implication in Prolog} \]

*Do not mistake an implication for an input-output relationship!*
RULE BASE

Logic knows nothing about „inputs” and „outputs”

Example: the following three clauses are logically equivalent:

\[
A \land B \text{ is true } \\
A \text{ is true } \land A \implies B \\
B \text{ is true } \land B \implies A
\]

Do not mistake an implication for an input-output relationship!
RULE BASE

Clauses of exact rule bases in *rmes*:

```
rule(Rule_Number,
    "Conclusion",
    List_of_conditions,
    Display_semaphore)
```

```
fact("Askable condition")
```
List_of_conditions

["Condition_1",..., "Condition_n"]

"Condition_i":

Logical string variable, representing the name of condition_i
Display_semaphore

= 0 - no message is displayed while the rule is being applied

= 1 - a message is displayed with details about the rule being applied
RULE BASE

The meaning of an exact rule:

rule(Rule_Number, "Conclusion", List_of_conditions, Display_semaphore)

is:

If all conditions are true, then Conclusion is true.
RULE BASE

What happens if some conditions of the rule:

\[
\text{rule}(	ext{Rule\_Number},
\text{"Conclusion"},
\text{List\_of\_conditions},
\text{Display\_semaphore})
\]

are false?
Dealing with negation

This is a major headache!

Well known to the Ancient Romans:

**Negativa non sunt probanda!**

Negations should not be proven!

Generally, there is no strict logical means of verifying that a fact is NOT given.

A Latin legal maxim (a paremia)
1. It is impossible to prove that somebody was not a secret collaborator of the Communist Security Services.

Negativa non sunt probanda!

To prove this, a massive exhaustive search had to be done!
2. It is possible to prove that somebody was a secret collaborator of the Communist Security Services.

Negativa non sunt probanda!

A single file with proper documents suffices to prove this.
1. However, this *paremia* is not an unquestionable command!

2. It is possible – in a number of technical and business applications - to run an exhaustive search over the entire decisions space and settle that something is not true.

**Negativa non sunt probanda!**
Dealing with negation

Implication of logic: $q \Rightarrow p$

<table>
<thead>
<tr>
<th>$q$</th>
<th>$p$</th>
<th>$q \Rightarrow p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

If the condition is false, nothing can be said about the conclusion.

Therefore we cannot claim anything.

This indeterminism is entirely unacceptable for expert systems!
Two way of dealing with negation in expert systems:

1. The Open World Assumption
2. The Closed World Assumption
First solution: Open World Assumption

Implication of logic: \( q \Rightarrow p \)

<table>
<thead>
<tr>
<th>q</th>
<th>p</th>
<th>( q \Rightarrow p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

Implication of expert systems: \( q \rightarrow p \)

<table>
<thead>
<tr>
<th>q</th>
<th>p</th>
<th>( q \rightarrow p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>Undetermined</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

If the condition is false, nothing can be said about the conclusion.

The open world assumption
„Open World Assumption”

If something cannot be inferred from:

the elementary rule base;
the constraint base;
the model base;
facts and arguments declared by the user,

it is considered to be undetermined (unknown).
„Open World Assumption“

We are leaving the door open for some other, yet unknown rule to appear, which might settle the question of whether the conclusion is or is not true.
Expert system shells using the open world assumption

Nesting of rules with negation is not feasible:

\[
\text{rule}(N, "Conclusion\_N", [..., "notABCD", ...], 1)
\]

\[
\text{rule}(M, "ABCD", \text{Lista\_of\_conditions\_M}, 1)
\]
Now it's time for some "Open World" examples to be run under \textit{rmes\_EE}
Second solution: Closed World Assumption

Implication of logic: $q \Rightarrow p$

<table>
<thead>
<tr>
<th>$q$</th>
<th>$p$</th>
<th>$q \Rightarrow p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

Implication of expert systems: $q \rightarrow p$

<table>
<thead>
<tr>
<th>$q$</th>
<th>$p$</th>
<th>$q \rightarrow p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

Taking into account the entire domain knowledge makes a false condition result in a false conclusion.

The closed world assumption.
Closed World Assumption

It is assumed, that the augmented knowledge base is complete.

If something cannot be inferred from:

- the rule base;
- the constraint base;
- the model base;
- facts and arguments declared by the user,

it is considered to be false.
Another name for the "Closed World Assumption"

Negation as failure
Closed world assumption =

„in dubio pro reo”

„in doubt favor the accused”

If a man cannot be proved to be guilty, he is considered innocent
Expert system shells using the Closed World Assumption

Nesting of rules with negation is feasible:

\[ \text{rule}(N, "Conclusion\_N", [..., "nABCD", ....], 1) \]

\[ \text{rule}(M, "ABCD", \text{Lista\_of\_conditions\_M}, 1) \]
Now its time for some "Closed World" examples
to be run under rmes_AE
Thank you for your attention!