PROPOSITIONAL CALCULATION FOR EXPERT SYSTEMS IN ECONOMICS

Emilia VASILE, PhD Professor
Athenaeum University, Bucharest, Romania
rector@univath.ro

Dănuţ-Octavian SIMION, PhD Associate Professor
Athenaeum University, Bucharest, Romania
danut_so@yahoo.com

George CALOTĂ, PhD Associate Professor
Athenaeum University, Bucharest, Romania
gcalota2003@yahoo.com

Abstract: *The paper presents the propositional calculation for expert systems in economics that ensure the logic for advanced programming language such as Prolog. An Expert System (ES) is a complex application (a software program) that explores a multitude of given knowledge to get new conclusions about difficult activities to examine using methods similar to human experts. An expert system can succeed in problems without a deterministic algorithmic solution. The inference engine is the one to determine all the rules that are activated, thus making the correlation between the facts base and the rule base, and then it also selects one of the rules that are activated at a given moment, which it puts into execution. Execution of a rule means the implementation of the right part of it, which may have one or more effects such as modifying the base of facts, sending messages to the operator, or transmitting signals to the outside, depending on the actions provided for in part to conclude the rule. The inference engine initiates a search in the knowledge base trying to* *solve the problem proposed by matching the left parts of the rules with the facts in the working memory and executing the rules that are enabled. ES may ask questions to the user when working when he gets stuck (he does not solve the proposed problem and cannot activate any rule) by using this dialog in the same interface. This also illustrates the difference in principle from conventional programming: the path that the inference engine will follow to reach the solution is not predetermined. It depends on the user's problem (the baseline state of the facts) and the responses the ES receives during work.*

Keywords: *Propositional calculation, expert systems, business environment, programming rules, information systems.*
JEL Classification: C23, C26, C38, C55, C81, C87

1. Introduction

The statement is any linguistic text stating anything about one or more objects. Such an affirmation may or may not have any meaning. The object or objects of a statement must belong to a well-defined domain, called the reference scope of the statement. From a syntactic point of view, every statement distinguishes: the predatory part and the subject or the subjects. The subject or subjects of a statement are the object (s) to which the statement refers.

An enunciate may have all the topics determined or may have one or more indeterminate topics. Depending on the domain of reference, the same statement may be true or false, or it can not be established a truth value, that is, it is indecisive.

states:

p: "The ball is round"

it is true if the field of reference is football, it is false if the field of reference is Rugby and undecidable if the field of reference is Sport.

A statement of all determined topics is called a proposition if it is either true or false, not one and the other simultaneously. A statement with one or more indeterminate topics is called predicate if any value given to undetermined topics becomes a sentence. Undetermined topics of a predicate are called predicate variables. Depending on the number of undetermined topics we distinguish predicates with a variable, two - binary, etc [1], [5].

This statement

p (x): "x is divisible by 2" has the reference domain number theory and the predatory part consists of the text "... is divisible by ..." The statement has two topics: “x” and “2”, one of which is undetermined. We notice that by giving x values the statement becomes true or false but not one and the other simultaneously, therefore it is a predicate with a variable.

The statement p (x, y): “x is divisible by y” is a predicate with two variables having the number theory reference domain.

2. Propositional computational elements

A sentence is a statement of all determined subjects that is true or false, not one and the other simultaneously. Such sentences we call simple sentences.

A sentence may be true or false if it corresponds to a state of fact in its field of reference. The quality of a sentence to be true or false is called the truth value of that sentence.

The simple sentences are denoted by: p, q, r, … Then true sentences are assigned the value of truth1 and false proposition the value of truth 0. Starting from one or more simple propositions by applying a finite number of operators - logical connectors we get other sentences called composite sentences.

There are four basic logical operators:

¬ (read non), ∧ (read and), ∨ (read or) and → (read implies).

Let it be a proposition. The sentence obtained by placing the particle “no” in front of the predatory part of the sentence p is called the negation of the proposition p and is marked by the symbol ¬p. The negation of the proposition p is true only when p is false.

Example: “2 is the number” with the reference range the set of natural numbers.

The negation of p is ¬p: “2 is not a number” p is a true sentence, while ¬p is a false proposition. It is easy to use a truth table to verify the relationships between the truth values of the sentences [2], [3].

The truth table for ¬p depending on the truth value of p is:

p	$\neg p$
1	0
0	1

Let p, q be sentences. The sentence obtained by putting the particle "and" between the predatory parts is called the conjunction of the propositions p, q and is denoted by the symbol p ∧ q. The proposition p ∧ q is true only when both sentences are true.

Example: The sentence "2 is a par and a 3 is a number" with the reference range The set of natural numbers is the conjunction of the sentences:

p: "2 is the number"

q: "3 is the number", p is true and q is false.

The truth value of the proposition p ∧ q according to the truth values of the sentences p, q is given in the table:

p	q	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0

Let p, q be sentences. The sentence that is obtained by putting the particle "or" between the predatory parts is called the disjunction of the sentences p, q and is denoted by the symbol p ∨ q. The proposition p ∨ q is true only when one of the sentences is true.

Example: The sentence "8 is multiple of 2 or 5 is integer" with the reference range The set of natural numbers is the disjunction of sentences:

p: "8 is multiple of 2"; q: "5 is integer", p, q being true.

The value of the proposition p ∨ q according to the truth values of the propositions p, q is given in the table:

p	q	$p \vee q$
1	1	1
1	0	1
0	1	1
0	0	0

Let p, q propositions. The expression "p involves q" is a new sentence, called the implication of the sentences p, q. The implication of sentences p, q is denoted p → q. The implication of the sentences p, q is a false proposition when the proposition q is false and p is a true and false proposition in other cases.

Example: The proposition "x 2 ≤ 0 involves x = 0" with the reference range the set of real numbers is the implication of the sentences:

p: "x 2 ≤ 0"; q: "x = 0" being a true sentence. The expression "p involves q" also uses:

"if p then q"

"p is sufficient for q"

"q is required for p"

The truth value of the sentence p → q according to the truth values of the propositions p, q is given in the table:

p	q	$p \to q$
1	1	1
1	0	0
0	1	1
0	0	1

The proposition q → p is called the reciprocal of the sentence p → q.

If we combine the proposition p → q with the proposition q → p, we get a new sentence which tells the equivalence of the sentences p, q. Equivalence of sentences is noted p ↔ q.

Let p, q be sentences. Equivalence p ↔q is that sentence which is true when p, q have the same value of truth and is false for the rest. The proposition p ↔q reads "p if and only if q" [4], [5].

The proposition "$x^3 \leq 0$ if and only if x ≤ 0" with the reference range the set of real numbers is the equivalence of sentences:

p: "$x^3 \leq 0$ involves x ≤ 0"; q: "x ≤0 implies $x^3 \leq 0$" being a true proposition. Note that the equivalence p ↔ q is true when p → q and q → p are true.

For the expression "p if and only if q" is still used:

"p is necessary and sufficient for q"

"if p then q, and vice versa"

The truth value of the proposition p ↔q according to the truth values of the sentences p, q is given in the table:

p	q	$p \leftrightarrow q$
1	1	1
1	0	0
0	1	0
0	0	1

We will write by A or A (p, q, r, ...), B or B (p, q, r, ...) the composite sentences or logical expressions obtained by applying different logical operators to simple propositions p, q, r, ...

A (p, q, r) :(p ∨ q) → r ...

The sentence computation studies the logical expressions of truth or false in relation to the logical values of the simple sentences that compose them.

A logical expression A (p, q, r, ...) that is true regardless of the truth values of p, q, r, ... is called tautology. (P, q, r, ...) → B (p, q, r, ...) is a tautology then write A (p, ...). If A (p, q, r, ...) ↔B (p, q, r, ...) is a tautology write A (p, q, r, ...)) Notes:

The symbol → is an operation from which we deduce the truth value of the proposition A → B while ⇒ indicates the connection between the propositions A (p, q, r, ..), B (p, q, r, ...).

The sign ↔ is an operation from which we deduce the truth value of sentence A ↔B while ⇔ indicates the relation between propositions A (p, q, r, ...), B (p, q, r, ...).

Examples of tautologies:

- the law of the third excluded: p ∨ (¬p).
- the law of syllogism: [(p → q) ∧ (q → r)] → (p → r).
- the reflexivity law: p ↔p.
- the idempotential law of the conjunction: p ∧p ↔p.
- the law of idempotency of disjunction: p ∨p ↔p.
- the law of double negation: ¬ (¬p) ↔p.
- the comutative law of the conjunction: (p ∧q) ↔ (q ∧p).
- the comutative law of the disjunction: (p ∨q) ↔ (q ∨p).
- the law of associativity of the conjunct:
- [(p ∧ q) ∧r] ↔ [p ∧ (q ∧r)]
- the law of associativity of the disjunct: [(p ∨q) ∨r] ↔ [p ∨ (q ∨r)]

De Morgan's laws:

¬ (p ∧ q) ↔ (¬p) ∨ (¬q) ¬ (p ∨q) ↔ (¬p) ∧ (¬q)
Conjunctivality Distribution Law:
- with respect to the disjunction: [p ∧ (q ∨r)] ↔ [(p ∧q) ∨ (p ∧r)]

Distributive Distribution Law:

- in relation to the conjunction: $[p \vee (q \wedge r)] \leftrightarrow [(p \vee q) \wedge (p \vee r)]$
$(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$
Demo 1:

p	$\neg p$	$p \vee (\neg p)$
1	0	1
0	1	1

Demo 2:

p	q	r	$p \rightarrow q$	$q \rightarrow r$	$(p \rightarrow q) \wedge (q \rightarrow r)]$	$p \rightarrow r$	$[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$
1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1
1	0	1	0	1	0	1	1
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	1
0	1	0	1	0	0	1	1
0	0	1	1	1	1	1	1
1	1	0	0	1	0	0	1

3. The elements that define predicates for expert systems in economic systems

By predicate is meant a statement with one or more indeterminate subjects that depends on a variable or several variables and has the property that for any values given to the variables a true prophecy or a false proposition is obtained.

Observation. Whenever a predicate is defined, it must also indicate the sets in which variables take values.

Example.

The predicate p (x): “x <4, x R”

has the reference range the set of real numbers and the predatory part consists of the text “... is smaller than ...”

The predicate has two subjects: “x” and “4”, of which an undetermined one is therefore a predicate with a variable [2], [6].

Example:

Let p (x, y) be the predicate “2x + y = 2”. What are the truth values of the prophecies p (2,0) and p (1,0)?

The proposition p (2,0) obtained by assigning x, y the values x = 2, y = 0 is a false proposition, while the proposition p (1,0) obtained by assigning x, y values x = 1, y = 0 is a true proposition.

Exercise.

Let p (x) be the predicate “x <4, x R”. What are the truths of p (5) and p (4)?

Generally, a predicate with n variables x1, x2,. . , xn is denoted by p (x1, x2, ..., xn).

Let p (x), q (x) single predicates. Using logical operators we construct other united predicates, namely:

(X), p (x) → q (x), p (x) ∧ q (x), p (x)

For example, for the predicate p (x), ¬p (x) is the predicate of which for each value x = a corresponds to the sentence ¬p (a). Legated with the notion of predicate, the notion of quantifier appears. We distinguish the following types of quantifiers:

The universal proposition of p (x) is the proposition “p (x) is a true sentence for any value of x in the reference range.” Note (x) p (x) denotes the universal proposition of p (x). The sign is called universal quantifier. For the universal proposition of p (x) we also use the expressions:

“for all x, p (x)”

“for each x, p (x)”.

Example. Any 10th grade student knows the number of natural numbers. The predicate is p (x): “Every student knows the number of natural numbers.” The set in which p (x) is true is the X grade class. If x (x) → q (x)) is true then we use the notation p (x) ⇒ q (x) and read: “p (x) involves q (x). It is also said in this case that the predicate q (x) is a logical consequence of the predicate p (x).

Example. Considering predicates

p (x): “x = 1”

and

q (x): “x^3-1 = 0, x ∈ R”

with the set of real numbers, we have p (x) ⇒ q (x). If the sentence (x) (p (x) ↔q (x)) is true, then we will use the notation p (x) ⇔q (x). It is also said in this case that the predicates p (x), q (x) are logically equivalent.

Example. Considering predicates

p (x): “$x > 0, x \in R$”
and
q (x): “$x^3 > 0, x \in R$”
with the set of real numbers, we have p (x) $\Leftrightarrow$ q (x).

Observation. Relationships of logical consequence and logical equivalence can also be defined between n predicates, where $n \geq 2$, in a similar way.

Definition. The existential proposition of p (x) is the sentence “there is at least one x of the reference range so that p (x)”. The notation (x) p (x) denotes the existential proposition of p (x) and is a true sentence when there is at least one element x_0 of the reference range so that p (x_0) is true [2], [4].

The sign ∃ is called existential quantificator.
For the existential proposition of p (x) we also use the expression:
“there exists x, p (x)”
Example. If we consider the predicate
p (x): “x + 5 = 0, x R”
with the set of sets of integers, then the existential proposition of p (x) is true because for x = -5 (∃ x) (x + 5 = 0) the proposition
p (-5): “-5 + 5 = 0”
it is true.
We consider the predicate p (x) defined only for a finite number of values of the variable x, namely x1, x2,. . . , xn, then:

$(\forall x)\ p(x) \Leftrightarrow p(x1) \wedge p(x2) \wedge \ldots \wedge p(xn)$
and
$(\exists x)\ p(x) \Leftrightarrow p(x1) \vee p(x2) \vee \ldots \vee p(xn)$
Taking into account the laws of De Morgan, it follows

$\neg\ (\forall x)\ p(x) \Leftrightarrow \neg\ p(x1) \vee \neg\ p(x2) \vee \ldots \vee \neg\ p(xn)$
and
$\neg\ (\exists x)\ p(x) \Leftrightarrow \neg\ p(x1) \wedge \neg\ p(x2) \wedge \ldots \wedge \neg\ p(xn)$

The negative rules set out above also apply to the general case. So for any single predicate p (x) we have:
a) $\neg\ p(x) \Leftrightarrow (\exists x)(\neg p(x))$

b) $\neg (\exists x) (\neg p(x)) \Leftrightarrow (\forall x) (\neg p(x))$

Example: The predicate is considered

p (x): "($\exists x \in N$) so that x + 1 = 2" whose truth is the truth.

Her negation is the proposition

$\neg p(x)$: " ($\forall x \in N$) , avem $x+1\neq2$"

it is obviously a false proposition.

Predicate binary. Let p (x, y) be a binary predicate.

Using quantificators we can form the unified predicates:

$(\exists x)\ p(x,y)$ şi $(\forall x)p(x,y)$

where y is the variable of these two predicates. From these united predicates we can form the predicates:

"$(\exists y)(\exists x)\ p(x,y),(\forall y)(\exists x)p(x,y),(\exists y)(\forall x)p(x,y)$ şi $(\forall y)(\forall x)p(x,y)$"

Next, it we will be demonstrated how to deduct the negative laws for binary predicates, from unicast predicate denying laws.

Example:

a') $\neg ((\forall)y (\exists x) p(x ,y)) \Leftrightarrow \exists y (\neg (\exists x) p(x ,y)) \Leftrightarrow (\exists y)(\forall x)(\neg p(x , y))$

b') $\neg ((\exists)y(\exists x) p(x , y)) \Leftrightarrow \forall y (\neg (\exists x) p(x , y)) \Leftrightarrow (\forall y)(\forall x)(\neg p(x , y))$

Analogously it can be extended these results to predicates.

4. Conclusions

The propositional calculation is the base for Prolog (Programming in logic) that is a language dedicated to symbolic, non-parametric computation. Prolog programming is geared to solving problems that involve the implementation of objects and the existing relationships between them. The Prolog language is therefore useful in solving artificial intelligence and expert systems. Prolog is a high-level language. By logic programming (reasoning) it is meant to produce a program that consists of facts and relationships (logical relationships between data sets) from which conclusions are drawn [1], [3]. A Prolog program is a collection of facts and rules. Objects are represented in Prolog by symbolic names. Existing relationships between objects are defined with clauses. There are two kinds of clauses: facts and rules.y [2], [4]. A program in Prolog is a description of some existing objects and relationships between them objects. Objects are represented by symbolic names whose syntactic structure is determined by the type of objects. There are two classes of types, elementary type and complex type. The propositional calculation may help to implement the business logic and the economic rules into Prolog programming language and then into enterprise applications.

References

[1] Chatain J-N., Dussauchoy A., *Systemes Experts: methodes et outils*, Eyrolles, Paris, 2015.

[2] Delahaye J. P., *Systemes experts: organisation et programmation des bases de connaissance en calcul propositionnel*, Eyrolles, 2015.

[3] Farreny H., *Les systemes experts: Principes et examples*, Cepadues, 2016.

[4] Kowalski R. A., *Logic for Problem Solving*, Northh-Holland Pub. Co., 2015.

[5] Lloyd J. W., *Foundation of Logic Programming*, 2nd ed., Springer-Verlag, 2016.

[6] Sterling L., Shapiro E.,*The Art of the PROLOG: Advanced Programming Techniques*, MIT Press, 2015.

URI: http://www.expertsystem.com

URI: http://www.sciencedirect.com

URI: http://ecomputernotes.com

URI: http://www.sciencedirect.com