Automated Decision Support System Based on Ordered Sources

SYLVIA ENCHEVA
Stord/Haugesund University College
Department Haugesund
Bjørnsonsg. 45, 5528 Haugesund
NORWAY
sbe@hsh.no

SHARIL TUMIN
University of Bergen
IT-Dept.
P. O. Box 7800, 5020 Bergen
NORWAY
edpst@it.uib.no

Abstract: Making a decision based on comparing responses coming from different sources has allways been difficult. In this paper we propose application of a decision support system for selecting a shipyard based on the quality of services provided by a number of shipyards.

Key–Words: Quality of services, selection, Web-based systems

1 Introduction

Shipping companies are aiming at constant development of their competitive abilities while increasing the importance of their role in international trade. Shipping enterprises have to make business decisions during the process of choosing investment directions by ordering a new vessel or purchasing one.

Carefully chosen decisions can prevent companies from unintended outcomes of their future actions. The problem of how globally ineffective outcomes may arise out locally effective decisions is discussed in [21].

Making a decision based on comparing responses coming from different sources has allways been difficult. In this paper we propose application of a decision support system for selecting a shipyard based on the quality of services provided by a number of shipyards in case a shipping company decides to order a new ship. This can be particularly useful for researchers and system developers.

The rest of the paper is organized as follows. Related work and statements from many-valued logic may be found in Section 2 and Section 3 respectively. Application various truth values is presented in Section 4. The system architecture is described in Section 5. The conclusion is placed in Section 6.

2 Background

According to the Aristotle's low of the excluded middle and the low of non-contradiction for any proposition σ, the proposition σ or $\neg\sigma$ is true by logical necessity, and the proposition σ and $\neg\sigma$ cannot both be true. At the same time Aristotle has mentioned the possibility of future contingencies.

There are three categories in Hegel's logic 'being', 'nothing', and 'becoming', [9]. The third one has aspects of both 'being' and 'nothing' and corresponds to what something is when it is in the state between nothing and being.

Apart from philosophical problems modern databases have to deal with incomplete and contradictory information. These problems that cannot be solved following the lows of excluded middle and/or non-contradiction.

Lukasiewicz has devised a three-valued calculus whose third value, $\frac{1}{2}$, is attached to propositions referring to future contingencies [20]. The third truth value can be construed as 'intermediate' or 'neutral' or 'indeterminate' [24], [22], and [23].

The semantic characterization of a four-valued logic for expressing practical deductive processes is presented in [2]. In most information systems the management of databases is not considered to include neither explicit nor hidden inconsistencies. In real life situation information often come from different contradicting sources. Thus different sources can provide inconsistent data while deductive reasoning may result in hidden inconsistencies. The idea in Belnap's approach is to develop a logic that is not that dependable of inconsistencies. The Belnap's logic has four truth values 'T, F, Both, None'. The meaning of these values can be described as follows:

- an atomic sentence is stated to be true only (T),
- an atomic sentence is stated to be false only (F),
- an atomic sentence is stated to be both true and false, for instance, by different sources, or in different points of time (Both), and

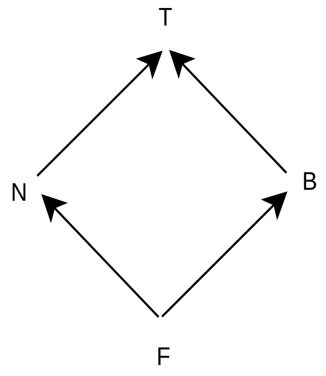

Figure 1: Logical lattice

- an atomic sentences status is unknown. That is, neither true, nor false (None).

The four truth values are arranged in a logical lattice [2] in Fig. 1. A logical conjunction and logical disjunction are related to the meet operation and to the join operation respectively.

The lattice in Fig. 1 is presented in terms of subsets of $\{0, 1\}$ in Fig. 2, [8].

Extensions of Belnap's logic are discussed in [10] and [19].

Interesting examples of poor decision making can be found in [26].

Two kinds of negation, weak and strong negation are discussed in [27]. Weak negation or negation-as-failure refers to cases when it cannot be proved that a sentence is true. Strong negation or constructable falsity is used when the falsity of a sentence is directly established.

Logic in preference modeling is discussed in [4], [18], and [22]. In [13] it is shown that additional reasoning power can be obtained without sacrificing performance, by building a prototype software model-checker using Belnap logic.

Python applications are known for increasing overall efficiency in the maritime industry [16].

LAMP is a collective name for the tools of Linux, Apache web server, MySQL database application, PHP scripting language, Perl programming language, and Python programming language. They have the advantage of being freely available, easily configured, and robust. They are a subject of constant development and improvement and are well known to be easily deployed, fully configured, and maintained with minimal efforts. The LAMP tools assist developers to do creative work without being bothered by administrative details.

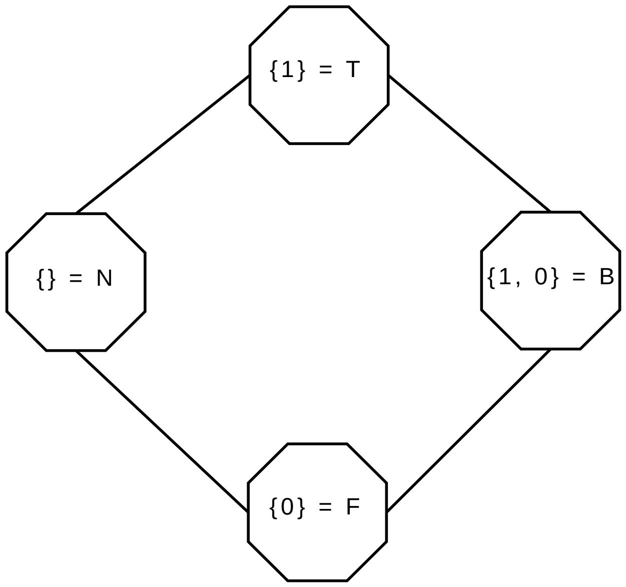

Figure 2: Subsets of $\{0, 1\}$

3 Lattices

Lattice theory have been investigated by many researchers like [3], [7], [11], [12],

A lattice is a partially ordered set [5], closed under least upper and greatest lower bounds:

- the least upper bound of α and β is called the join of α and β, and is sometimes written as $\alpha + \beta$,
- the greatest lower bound is called the meet and is sometimes written as $\alpha\dot{\beta}$.

A billatice is a set equipped with two partial orderings $\leq_t$ and $\leq_k$.

- The t partial ordering $\leq_t$ means that if two truth values ϕ, ψ are related as $\phi \leq_t \psi$ then ψ is at least as true ϕ.
- The k partial ordering $\leq_k$ means that if two truth values ϕ, ψ are related as $\phi \leq_k \psi$ then ψ labels a sentence about which we have more knowledge than a sentence labeled with ϕ.

The generalized constructive truth-value space has as a base a set $\mathcal{I} = \langle T, F, t, f \rangle$ containing the initial truth values

- *T* - a sentence is constructively proven
- *F* - a sentence is constructively refuted
- *t* - a sentence is acceptable
- *f* - a sentence is rejectable

The power set of $\mathcal{I}$ gives sixteen generalized truth values:

- { } - empty multivalue,
- { T } - a sentence is constructively proven,
- { F } - a sentence is constructively refuted,
- { t} - a sentence is acceptable,
- { f} - a sentence is rejectable,
- { T, F } - a sentence is constructively proven and constructively refuted,
- { T, t} - a sentence is constructively proven and acceptable
- { T, f} - a sentence is constructively proven and rejectable
- { F, t} - a sentence is constructively refuted and acceptable
- { F, f} - a sentence is constructively refuted and rejectable
- { t, f} - a sentence is acceptable and rejectable
- { T, F , t} - a sentence is constructively proven, a sentence is constructively refuted, and acceptable
- { T, F , f} - a sentence is constructively proven, a sentence is constructively refuted, , and rejectable
- { T, t , f} - a sentence is constructively proven, acceptable, and rejectable
- { F, t , f} - a sentence is constructively refuted, acceptable, and rejectable
- { T, F , t, f} - a sentence is constructively proven, constructively refuted, acceptable, and rejectable

The empty multivalue is denoted by N and A represents the set that contains the initial truth values T, F, t, f.

The truth table for the 16 truth values is Table 2.

A trilattice is a structure

$$\mathcal{T} = (S, \leq_i, \leq_t, \leq_c)$$

such that S is a nonempty set and $(S, \leq_i)$, $(S, \leq_t)$ and $(S, \leq_c)$ are complete lattices.

The three partial orderings $\leq_i, \leq_t, \leq_c$ arrange elements according to the possessed degree of information, truth and constructivity respectively. The bounds relative to the three partial orderings are shown in Table 1. Accomplished constructions (proofs and disproofs) are presented by constructive values. Nonconstructive truth values do not imply any completed construction.

The sixteen truth values can be arranged in a trilattice with five information levels, five logical levels and five levels of constructivity [25].

4 Service Quality

Suppose a ship company is planning on building a new vessel. Once the vessel type is determined the company has to decide which shipyard is going to undertake the job. One of the many challenges in this process is related to service quality provided by various shipyards.

In order to facilitate the process we propose application of a Web-based system involving intelligent agents. Shipyards interested to accept jobs from international ship companies send detailed information about the services they provide to available databases. These databases register information about the quality of their services including references from former customers.

When a company employee sends a quarry about service quality of several shipyards an intelligent agent is collecting proper data from all accessible databases. Another intelligent agent is arranging the collected by the first agent shipyards in a lattice structure as shown in Fig. 3 ([25]) according to the employee's request.

Ordering	Bounds	Most and least elements in the power set of $\mathcal{I}$
$\leq_i$	the initial truth values, empty multivalue	informative
$\leq_t$	constructively proven and acceptable, constructively refuted and rejectable	true
$\leq_c$	constructively proven and constructively refuted, acceptable and rejectable	constructive

Table 1: The bounds relative to the three partial orderings

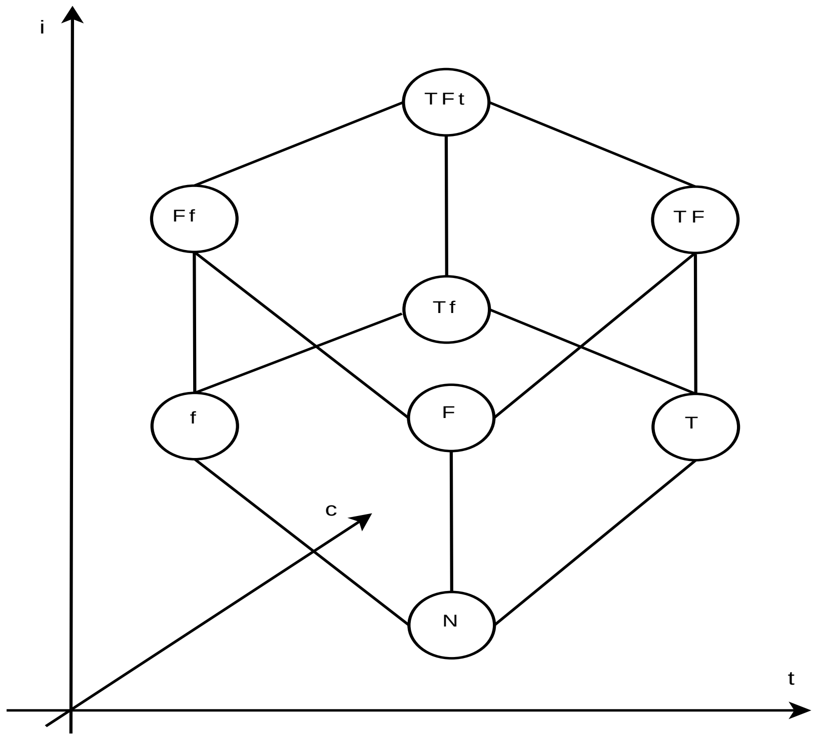

Figure 3: Sublattice of a trilattice

$\wedge$	T	F	t	f	TF	Tt	Tf	Ft	Ff	tf	TFt	TFf	Ttf	Ftf	TFtf	N
T	T	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
F	N	F	N	N	N	N	N	N	N	N	N	N	N	N	N	N
t	N	N	t	N	N	N	N	N	N	N	N	N	N	N	N	N
f	N	N	N	f	N	N	N	N	N	N	N	N	N	N	N	N
TF	N	N	N	N	TF	T	T	F	F	N	T	F	T	F	N	N
Tt	N	N	N	N	T	Tt	T	t	N	t	t	T	T	t	t	N
Tf	N	N	N	N	T	T	Tf	N	f	f	T	f	T	f	T	N
Ft	N	N	N	N	F	T	N	Ft	F	N	t	t	t	F	F	N
Ff	N	N	N	N	F	N	f	F	Ff	f	F	f	f	f	f	N
tf	N	N	N	N	N	t	f	N	f	tf	t	f	t	f	N	N
TFt	N	N	N	N	T	t	T	t	F	t	TFt	TF	Tt	N	Ft	N
TFf	N	N	N	N	F	T	f	t	f	f	TF	TFf	Tf	Ff	Ff	N
Ttf	N	N	N	N	T	T	T	t	f	t	Tt	Tf	Ttf	ft	ft	N
Ftf	N	N	N	N	F	t	f	F	f	f	N	Ff	tf	Fft	ft	N
TFtf	N	N	N	N	N	t	T	F	f	N	Ft	Ff	tf	tf	TFtf	N
N	N	N	N	N		N	N	N	N	N	N	N	N	N	N	N

Table 2: Truth table

The initial truth values *T, F, t, f* are assigned as answers to questions like 'Does company X provide good service quality in terms of flexibility?'. The truth values have the following meaning

- *T* - 'a sentence is constructively proven' means that there is a customer with positive opinion who has actually being using the services of that shipyards
- *F* - 'a sentence is constructively refuted' means that there is a customer with negative opinion who has actually being using the services of that shipyards
- *t* - 'a sentence is acceptable' means that there is a customer with positive opinion who has not actually being using the services of that shipyards
- *f* - 'a sentence is rejectable' means that there is a customer with negative opinion who has not actually being using the services of that shipyards

The relations between these truth values are presented in Table 2, [25].

5 System Architecture

The prototype system was implemented using the so-called LAMP Web application infrastructure and deployment paradigm. The architectural framework was build with a combination of free software tools on a Linux operating system of an Apache Web server, a database server and a programming environment using scripting language.

The system implementers can choose and mix these tools freely. This in contrast to commercial Web application platforms like for example, ASP.net from [14], JavaServer from Sun [15], and WebSphere from IBM [17].

Our prototype system is composed of two loosely couple Web servers deployment:
- a Web server for users interface to the system and
- a Web server for agents management.
The deployment in our two servers prototype system is based on

- an Apache front end Web server,
- a Python run-time environment for middleware application for dynamic content, data integration and users' administration, and
- a backend database based on lightweight SQLite database engine.

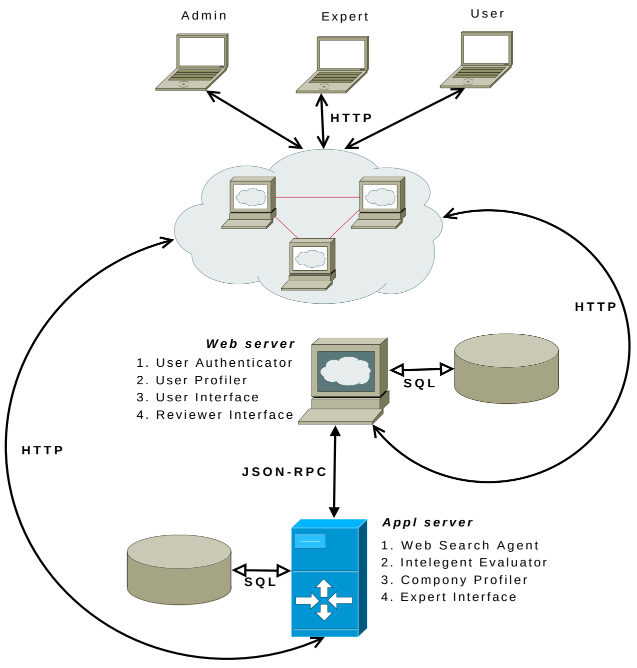

Figure 4: System architecture

HTTP server from the Apache Software Foundation is a robust and extendable Web server. Apache has been the most popular web server on the Internet since 1996. Web server from Apache is composed of the core which provides the basic functionalities and can be extended using modules. In our implementation, the Web server is extended with a Python interpreter by using 'mod_python' module.

Python is a dynamic object-oriented programming language and can be installed and run on most operating systems and machine platforms. Python offers strong integration with operating system environment, other languages and software tools by utilizing its extensive standard libraries. Python syntax is simple and clear which helps to increase productivity of the system implementers. Incorporating Python interpreter directly within Apache Web server by using 'mod_python' increases the power of the Web server by providing a dynamic programmable run-time environment.

SQLite database engine is

- self-contained: It is written in ANSI-C and it makes minimal use of the standard C library or system calls.
- serverless: Database access during read or write is done directly from or to files. There is no need for intermediate server process. SQLite does not

administer its own user and access control, it uses an operating system file protection mechanism.

- zero-configuration: There is no separate process to be initialized and stared, and there is no need for administrators and users.
- transactional: All transactions are atomic, consistent, isolated and durable.
- multiple databases: Data can be stored in multiple files that can be attached together as one database.

The two Web servers deployment are joined together with a communication framework based on JSON remote procedure call (JSON-RPC over HTTP) written in Python. JSON stands for JavaScript Object Notation and it is a lightweight data-interchange format. It is more compact then XML without sacrificing expressiveness. JSON structure is perfect for packaging and sending data in RPC request and reply messages.

JSON-RPC supports three type of massage format: request, response, and notification.

1. A request message contains four elements

- version: required since JSON-RPC 1.1
- method: a string providing the name of the remote procedure to be invoked by the server
- params: an array of objects to pass as arguments to the procedure
- id: the request identifier of any type which will be used to match request\response messages

An example of request message

```
POST /JRPC HTTP/1.1\\
User-Agent: test/1.0\\
Host: siam.uib.no\\
Content-Type: application/json\\
Content-Length: 201\\
Accept: application/json\\

{
    "version" : "1.0",
    "method"  : "sum",
    "params"  : [ 10, 20 ],
    "id"  : "rpc#1-test@siam"
}
```

2. A response message contains three elements

- version: required since JSON-RPC 1.1
- result: an object as the result from the remote procedure invocation (or) error: an error object if there was an error invoking the remote procedure
- id: this must be the same identifier as the request it is responding to.

An example of response message

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 53
Content-Type: application/json
Date: Sat, 08 Jul 2006 12:04:08 GMT
Server: Apache+Mod_python/2.1

{
    "version" : "1.0",
    "result"  : 30,
    "id"  : "rpc#1-test@siam"
}
```

3. A notification message. It is a special request which does not have a response. It has the same format as the request message except that the ”id” must be null.

Besides providing interface to the system for expert users, the application server also provides search and intelligent evaluation services. The application server acts as agents’ manager for long running programs related to search and analysis of data gathered. The separation of these two units made it possible to modularly design and implement the system as loosely coupled independent sub-systems. The purpose of the search agent is to search for different reviews from independent reviewers about a particular shipyard. This process will eventually build a database of shipyards’ service qualities reviewed by different expert reviewers. By providing an expert users Web interface, the system invites expert reviewers to submit their reviews of shipyards they have had experience working with.

The Web server provides user interface to the system. The user authenticator and user profiler modules play an important role in controlling every particular client, expert or administrator authenticity. Only valid expert reviewers can submit reviews and only registered clients can make use of the system. The administrator can approve the results of a search agent before the data is submitted to the database. The purpose of the intelligent evaluator is to rank the shipyards’ service qualities at any one time in response to users’ queries.

The Web server's middleware and the application server's software agents can run in parallel, independently of each other. As such, they can be situated on different servers. The middleware implements the Web user interface side of the system while the software agents implement the evaluation side of decision process. The two sub-system can also be implemented on the same server running only one Web server. In that case the Web server will listen on port 80 while the application server runs on different port, for example port 88.

6 Conclusion

Application of such a system can certainly shorten the process of vessel building. The main challenges are related to a creation and maintenance of reliable databases.

References:

[1] N.J. Belnap, How a computer should think. In Contemporary Aspects of Philosophy. *Proceedings of the Oxford International Sumps*, Oxford, GB, pp. 30-56, 1975.

[2] N.J. Belnap, A useful four.valued logic, *Modern uses of multiple-valued logic*, J.M. Dunn and G. Epstein (eds), D. Redial Publishing Co., Utrecht, pp. 8-37, 1977.

[3] G. Birkhoff, *Lattice Theory*, 3rd ed. Vol. 25 of AMS Colloquium Publications. American Mathematical Society, 1967.

[4] D. S. Bridges and G. B. Meta, Representations of preference orderings. Springer-Verla, Berlin, 1995.

[5] B. A. Davey and H. A. Priestley, *Introduction to lattices and order*, Cambridge University Press, Cambridge, 2005.

[6] R. P. Dilworth and P. Crawley, *Algebraic Theory of Lattices*, Prentice-Hall, 1973.

[7] T. Donnellan, *Lattice Theory* Pergamon, 1968.

[8] J. M. Dunn, ”Partiality and its Dual”, Partiality and Modality, eds. E.Thickos, F.Le page & H.Wansing, special issue of *Studio Logic*, Vol.66, pp. 5-40, 2000.

[9] A. Ferrari, *Hegel and Aristotle*, Cambridge, Cambridge University Press, 2001.

[10] J. M. Font and M. Moussaka, ”Note on a six valued extension of three valued logic's”, *Journal of Applied Non-Classical Logic's*, Vol. 3, pp. 173-187, 1993.

[11] R. Freese, J. Jezek, and J. B. Nation, ”Free Lattices”, *Mathematical Surveys and Monographs*, Vol. 42, Mathematical Association of America, 1985.

[12] G. Grtzer, *Lattice Theory: First Concepts and Distributive Lattices*, San Francisco, CA: W. H. Freeman, 1971.

[13] A. Garfunkel and M. Check, ”Yams: Model-Checking Software with Belnap Logic”, Technical Report 470, University of Toronto, April, 2005.

[14] http://msdn2.microsoft.com/en-us/asp.net/default.aspx

[15] http://java.sun.com/products/jsp/

[16] http://www.python.org/about/success/tribon/

[17] http://www-306.ibm.com/software/websphere/

[18] J. Gacrux and M. Rubens, ”Non Conventional Preference Relations in Decision Making”, *Lecture Notes in Economics and Mathematical Systems (LN MES)*, Vol. 301, Springer Verla, Berlin, 1988.

[19] Y. Glazing and A. Y. Maurits, ”A knowledge representation based on the Belnap's four valued logic”, *Journal of Applied Non-Classical Logic's*, Vol. 3, pp. 189-203, 1993.

[20] J. Lukasiewicz, ”On Three-Valued Logic”, *Rich Filofax*, Vol. 5, (1920), English translation in Bork's, L. (ed.) 1970. Jan Lukasiewicz: Selected Works. Amsterdam: North Holland, 1920.

[21] J.D. W. Morecroft, ”Strategy support models”, *Strategic Management Journal*, Vol. 5, No. 3, 1984.

[22] P. Penny and A. Souks, ”On the continuous extension of a four valued logic for preference modelling”, *In Proceedings of the Information Processing and Management of Uncertainty (IMP) conference*, Paris, pp. 302-309, 1998.

[23] G. Priest, *An Introduction to Non-Classical Logic*, Cambridge, 2001.

[24] K. M. Sim, ”Bi lattices and Reasoning in Artificial Intelligence: Concepts and Foundations”, *Artificial Intelligence Review*, Vol. 15, No. 3, pp. 219-240, 2001

[25] Y. Shrank, J. M. Dunn and T. Takenaka, ”The Trilattice of Constructive Truth Values”, *Journal of Logic Computation*, Vol. 11, No.6, pp. 761-788, 2001.

[26] J. D. Steersman, ”Perceptions of feedback in dynamic decision making”, *Organizational Behavior and Human Decisis on Processes*, Vol. 43, No. 3, pp. 301-335, 1989.

[27] G. Wagner, ”Vivid Logic: Knowledge Based reasoning with two kinds of negation”, *Lecture Notes in Artificial Intelligence*, Vol. 764, Springer-Verla, Berlin Heidelberg New York, 1994.