Dynamic Causal Mining

A thesis submitted to the Cardiff University

For the degree of

Doctor of Philosophy

By

Yi Wang

Manufacturing Engineering Centre

Cardiff University

United Kingdom

2008

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Causality plays a central role in human reasoning, in particular, in common human decision-making, by providing a basis for strategy selection. The main aim of the research reported in this thesis is to develop a new way to identify dynamic causal relationships between attributes of a system.

The first part of the thesis introduces the development of a new data mining algorithm, called Dynamic Causal Mining (*DCM*), which extracts rules from data sets based on simultaneous time stamps. The rules derived can be combined into policies, which can simulate the future behaviour of systems. New rules can be added to the policies depending on the degree of accuracy. In addition, facilities to process categorical or numerical attributes directly and approaches to prune the rule set efficiently are implemented in the *DCM* algorithm.

The second part of the thesis discusses how to improve the *DCM* algorithm in order to identify delay and feedback relationships. Fuzzy logic is applied to manage the rules and policies flexibly and accurately during the learning process and help the algorithm to find feasible solutions.

The third part of the thesis describes the application of the suggested algorithm to a problem in the game-theoretic domain. This part concludes with the suggestion to use concept lattices as a method to represent and structure the discovered knowledge.

Acknowledgements

I would like to express my deepest gratitude to Professor D. T. Pham, my supervisor, for creating the opportunity of my studying in the UK. I am grateful for his invaluable guidance and for his consistent encouragement during the past four years.

The Manufacturing Engineering Centre (MEC), Cardiff University, is a good place to study and work. I thank all its members for their friendship and help.

I would especially like to thank my family for their support. Thanks also go to my parents who have supported me all my life and my wife who has contributed and sacrificed much for me to accomplish this thesis.

Contents

List of Figures

List of Tables

Abbreviations

A- Antipathetic

BN- Bayesian Network

CD – Causal diagram

D- Database

DAG - Directed acyclic graph

DCGM-Dynamic causal game mining

DCM – Dynamic causal mining

FDCM- Fuzzified DCM

HA- High antipathetic

Hs-High sympathetic

n- Neutral

p- Polarity

S- Sympathetic

TID – Transaction ID

List of Symbols

$\Delta a_{m,t_i}$ - the *m*th record of dynamic action Δa with time stamp t and i= 1,2,3.....

ΔA_i- the *i*th dynamic attribute in an example, where i= 1,2,3.....

$\Delta A_{m,t_i}$ - the *m*th record of dynamic attribute ΔA with time stamp t and i= 1,2,3.....

Δt_i- *dynamic time stamp* i= 1,2,3.....

μ — *fuzzy membership function*

a_{m,t_i} - the *m*th record of action a with time stamp t and i= 1,2,3.....

A_i - the *i*th attribute in an example, where i= 1,2,3.....

A_{m,t_i} - the *m*th record of attribute A with time stamp t and i= 1,2,3.....

$\overline{C_k}$ - candidate sets where k is number of the itemsets

D_{new} –*dynamic database*

$\tilde{f}$- *fuzzified result*

F_i— *fuzzified point*

$F(\tilde{})$- *fuzzified function*

K — *coefficient*

k_t— *coefficient based on time stamp t*

L — linguistic term

$\overline{L_k}$ - large *itemsets* where k is number of the itemsets

$S_{m,\Delta t_i}$ - the *m*th record of strategy S with dynamic time stamp Δt and i= 1,2,3.....

$S_{player\ A}$- *Group of strategy belong to player A*

t_i - *time stamp* i= 1,2,3.....

$t_{\Delta a_2}$ - time stamp belong to dynamic action Δa where i= 1,2,3.....

Chapter 1 Introduction

1.1 Background

Since the dawn of civilisation, humankind has been attempting to understand causality. Ancient Egyptians tried for thousands of years to identify the cause of rain in order to make predictions for the harvest. The Chinese were interested in the causal relationships between the lines on the palm of the hand and destiny. Many Greek philosophers have debated and discussed what causality is. Aristotle stated: "*All causes of things are beginnings; that we have scientific knowledge when we know the cause; that to know a thing's existence is to know the reason why it* is" (Mure, 2007). Ancient Hindu scriptures and commentaries describe causal relationships as: "*Cause is the effect concealed, effect is the cause revealed*" (Vivekananda & Vashishta, 1902). In modern days, many scientific methodologies have been developed to determine the physical and philosophical properties in the universe.

The research reported in this thesis concentrates on the development of one type of algorithms, namely inductive association learning algorithms. An important characteristic of inductive learning is that the induced model structure is readily understood by humans. Because of this structure, the developed algorithms have become more popular for causal analysis. However, the rules mined using inductive association learning indicate only association relationships among variables in a system. They do not specify the essential underlying mechanism of the system that describes causal relationships.

1.2 Research hypotheses and objectives

Data Mining is the discovery of hidden information found in databases and can be viewed as a step in the knowledge discovery process (Chen, 1996 and Fayyad, 1996). Data mining functions include clustering, classification, and associations. One of the most important data mining applications is that of mining association rules. Association rules are becoming one of the most researched areas of data mining and have received much attention from the database community. They have proven to be quite useful in the marketing and retail communities as well as other more diverse fields. Measures such as causality have been used to infer rules of the form "the existence of one attribute implies the existence of another". However, such rules indicate only a statistical correlation between and they do not specify the dynamic nature of the causality: whether a change of value of one attribute will lead to change of another attribute. In applications, knowing such causal relationships is extremely useful for enhancing understanding and effecting change. Deductive Causality reasoning based on experts' input have been researched in System dynamics community (Sterman, 2000 and Forrester, 1994). However, their works are dependent on the user knowledge and lack the ability to identify hidden knowledge.

The overall aim of this research was therefore to design and develop new dynamic causal mining algorithms suitable for data mining applications where causal relationships are to be discovered. These algorithms should be able to handle a large amount of data in an efficient and effective way. Moreover, they should be able to deal properly with both categorical and numerical attributes which are the two kinds of attributes found in real systems. Finally, the structure of these models should be easy to understand by human experts. They should achieve good accuracy, be

compact, and have short execution times. To accomplish the overall aim of the research, the following sub-objectives were set.

To perform a detailed analysis of existing techniques of *Association Mining*, *System Dynamics* and *Game Theory*, with particular emphasis on causality analysis, and to assess their appropriateness for data mining applications. This analysis was performed through a review of the published literature on these subjects.

To develop computationally efficient causal mining algorithms that can be applied to solve large and complex problems. This was achieved initially by focusing on events that happen in a time window. The methodology adopted was to combine concepts of *System Thinking*, *System Dynamics* and *Association Mining.*

To produce a causal mining algorithm suited to extracting dynamic relationships. This was performed by identifying delay and feedback relationships between attributes.

To develop techniques for dynamic algorithms that can increase execution speeds and accuracies. This was achieved through employing effective pruning methods to reduce the size of the rule sets.

Three hypotheses will be investigated and proved in this research:

- *Association Mining* can be combined with *System Dynamics* to discover hidden causal knowledge.

- Improved such the combination with higher accuracy and faster execution speed.
- The new method can provide inventive conflict resolution in solving game-theoretical problems.

1.3 Thesis structure

Chapter 2 briefly reviews the basic principles and methodologies of *causal analysis* and discusses some of the main existing algorithms for causal analysis.

Chapter 3 presents a new algorithm called *DCM* (*Dynamic Causal Mining*), which combines *Association Mining* and *System Dynamics* concept. The algorithm achieves good accuracy for the rules extracted by focusing on the measurement of the polarity which is derived from the attributes. The advantage is that there is no need for expert knowledge to develop mental models. A new pruning method especially designed for *DCM* is proposed for pruning away redundant attributes.

Chapter 4 introduces an improved rule pruning approach and a more accurate rule extraction algorithm. This chapter focuses on extracting delay and feedback relationships from data sets. The algorithm uses fuzzy logic theory to improve the capability of extracting accurate rules including delay and feedback relationships among attributes.

Chapter 5 describes the application of *DCM* to *Game Theory*. The chapter focuses on the development of a structural representation of the dynamic causal relationships between players in a game using a conceptual lattice.

Chapter 6 summarises the thesis and proposes directions for further research.

Appendix A describes the data sets used to test the performance of the developed algorithms.

Appendix B gives a general description of *Game Theory.*

Appendix C gives a general description of formal concept theory.

Chapter 2 Literature review

2.1 Preliminaries

Causality is a fundamentally interesting area for analysts from different disciplines, with applications in the fields of philosophy, psychology, engineering, chemistry, physics, law and humanity. Figure 2.1 shows the classification tree for the causality analysis. Causality analysis is divided into scientific, religious and philosophical. Religious and philosophical causal analysis starts as early as the dawn of civilization. The scientific analysis comes much later, and can be divided into hypothesis-based analysis and simulation-based analysis. Hypothesis-based analysis focuses on developing tools to identify the causal rules and causal knowledge. Simulation-based analysis focuses more on the possible performance or behaviour patterns created by the causal rules. This thesis builds a bridge between hypothesis-based and simulation-based analysis.

This chapter reviews three areas of causality analysis:

1. *Statistical focused causal analysis.* In a Statistical context, given two attributes A_1 and A_2,, if A_1 causes A_2, then A_1 must *always* be followed by A_2. Informally, A_1 causes A_2 if and only if A_1's occurrence increases the probability of A_2. Generally, the statistical algorithms search through the possible causal structures among the attributes and remove ones which are strongly incompatible with the observed correlations.

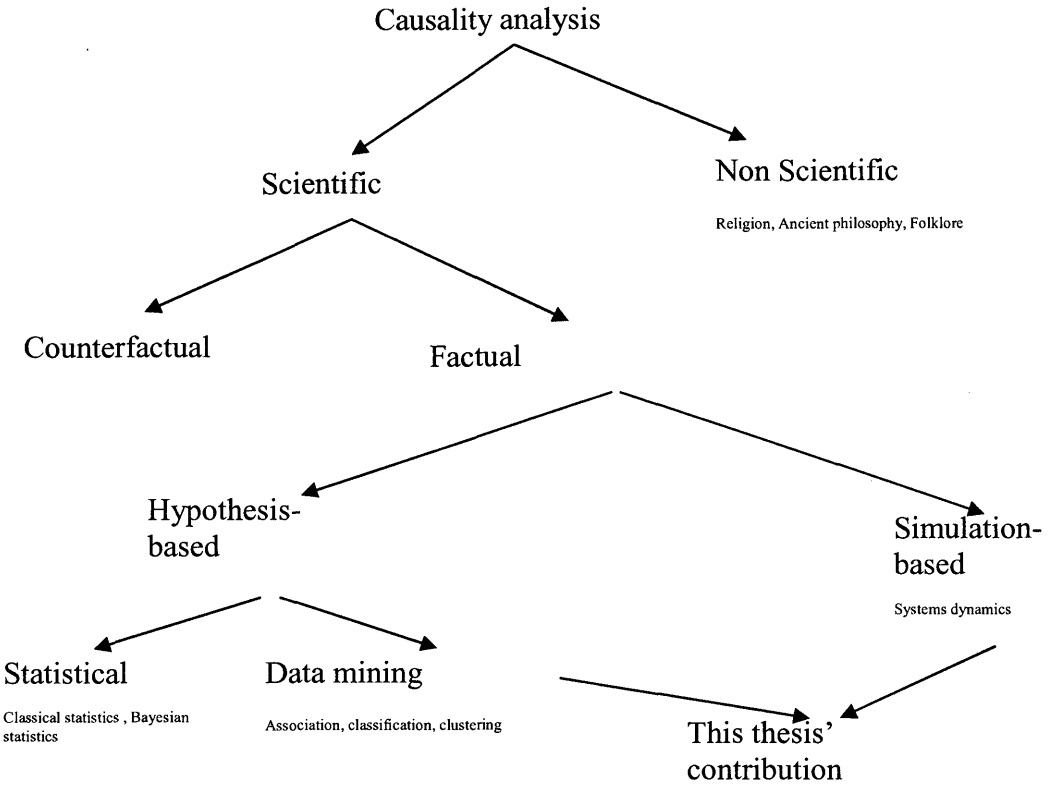


Figure 2.1 Classification of causal analysis

2. *Data mining focused causal analysis.* In a data mining context, given two attributes A_1 and $A_{2,,}$ a causal rule between these two attributes captures the relationship that the presence of A_1 causes the appearance of A_2. The most frequently used algorithm to identify causal relationships is *Association Mining*. *Association Mining* describes co-occurrence among the attributes of the input relation. Such co-occurrence can give valuable information on the patterns hidden in the data. With the explosive growth of data stored in an electronic form, *Association Mining* has become a popular tool in searching for nontrivial, implicit, previously unknown and potentially useful information from a huge amount of data. The *Association Mining* technique has been focused mostly on identifying all of the frequent attribute sets which satisfy a minimum support threshold.

3. *Simulation-based causal analysis*. The major contribution in this area was made by John Sterman (Sterman, 2000) and Peter Senge (Senge, 1990). The methodology is called *System Dynamics*. *System Dynamics* is used for modelling causal dynamic relations among entities to assist in decision-making and future-planning. It involves studying and managing complex feedback systems, such as business and social systems. *System Dynamics* explains the complexity of systems by using feedback loops. A causal loop diagram is used to describe how entities in a system are connected by feedback loops, which create the nonlinearity found frequently in modern daily problems.

Data mining and statistical analysis are closely related research areas with different research goals. Statistical analysis focuses more on the solution while data mining focuses more on the calculation speed and system performance. This chapter describes:

(1) The fundamentals of basic statistical causal analysis.

(2) Some of the most prominent *Association Mining* algorithms.

(3) The fundamental modelling techniques of *System Dynamics.*

2.2 General causal analysis

The first well-established theory of causality was developed by David Hume. He claimed that "*The cause has to be temporally and spatially contiguous to the effect and there is a time succession between the cause and the effect*" (Hume, 1740). *Contiguity, succession,* and *constant conjunction* are three basic elements of Hume's *regularity* theory of causality. Hume gives two separate definitions for the generic cause and its effect.

Many famous scientists and philosophers, such as Thomas Hobbes, Immanuel Kant and John Stuart Mill, found that Hume's theory was insufficient since Hume defined the cause/effect relation as deterministic.

Based on the work of Hume, two new theories were developed: counterfactual and probabilistic.

2.2.1 Counterfactual theory

The counterfactual approach, focused on causation, has been used to analyze causation in terms of *counterfactual conditionals* (Pearl, 1994), based on the assumption that all statements about causality can be understood as counterfactual statements (Lewis, 1973). A counterfactual condition is a conditional statement indicating that if an event A caused an event B and if A had not occurred, B would not have occurred. For example, if Yi Wang is in China, the following two conditionals have both a false antecedent and a false consequent.

(1) If Yi Wang were in Norway then he would be in Africa.

(2) If Yi Wang were in Norway then he would be in Europe.

Indeed, if Yi Wang is in China, then all three conditions "Yi Wang is in Norway", "Yi Wang is in Africa", and "Yi Wang is in Europe" are false. However, (1) is obviously false, while (2) is true.

2.2.2 Probabilistic theory

The major work in probabilistic causal analysis was undertaken by Suppes. He intended to construct a theory, which incorporated indeterminism into Hume's theory (Suppes, 1970).

Suppes' definition of a causal relationship is the probability of the effect compared to the case where the cause is absent. Let A, B, C, represent events. Let P be a probability function such that P(A) represents the probability that event A occurs. Let P($B \mid A$)

represent the *conditional probability* of B, given A. Formally, conditional probability (also called Bayesian probability) is defined as a certain ratio of probabilities:

$$P(B \mid A) = P(A \text{ and } B)/P(A). \qquad (2.1)$$

If $P(A)$ is 0, then the ratio in the definition of conditional probability is undefined. Assume A raises the probability of B is that $P(B \mid A) > P(B \mid \text{not-}A)$. Thus a first attempt at a probabilistic theory of causation would be:

PR: A causes B if and only if $P(B \mid A) > P(B \mid \text{not-}A)$. (2.2)

This formulation is labelled *PR* for "Probability-Raising." When $P(A)$ is strictly between 0 and 1, the inequality in PR turns out to be equivalent to $P(B \mid A) > P(A)$ and also to $P(A\text{and}B) > P(A)P(B)$. When this last relation holds, A and B are said to be *positively correlated.* If the inequality is reversed, they are *negatively correlated.* If A and B are either positively or negatively correlated, they are said to be *probabilistically dependent.* If equality holds, then A and B are *probabilistically independent* or *uncorrelated.*

PR addresses the problems of imperfect regularities and indeterminism, discussed above. But it does not address the other two problems discussed in section 1 above. First, probability-raising is symmetric: if $P(B \mid A) > P(B \mid \text{not-}A)$, then $P(A \mid B) > P(A \mid \text{not-}B)$. The causal relation, however, is typically *asymmetric.*

Second, PR has trouble with *spurious correlations.* If A and B are both caused by some third factor, C, then it may be that $P(B \mid A) > P(B \mid \text{not-}A)$ even though A does not cause B. For example, let A be an individual having yellow-stained fingers, and B that individual

having lung cancer. Then the expected result would be P(B | A) > P(B | not-A). The reason that individuals with yellow-stained fingers are more likely to suffer from lung cancer is that smoking tends to cause both the yellow stains and lung cancer. Because they are more likely to be smokers, they are also more likely to suffer from lung cancer. Intuitively, the way to address this problem is to require that causes raise the probabilities of their effects *ceteris paribus**.

Pearl (Pearl, 1997) and Peter Spirites (Spirites, et al., 1993) developed algorithms that can make causal inference automatically. To conduct causality tests on non-stationary time series can sometimes lead to wrong result (Engle and Granger, 1988). The two Nobel laureates developed a technique for determining whether one time series is useful in forecasting another. A time series X is said to cause Y if it can be shown that those X values provide statistically significant information on future values of Y.

2.2.3 Bayesian network

A Bayesian Network (BN) is a graphical representation of a joint probability distribution function. Given a set of three attributes A_1, A_2 and A_3, the joint probability distribution $P(A_1, A_2, A_3)$ can be written, by the chain rule, as a factorisation:

$$P(A_1, A_2, A_3) = P(A_1, | A_2, A_3)\, P(A_2| A_3)\, P(A_3) \qquad (2.3)$$

Given a *directed acyclic graph* (*DAG*) distributed over these attributes as shown in Figure 2.2., the joint distribution can be rewritten as;

* A Latin phrase, literally translated as "with other things [being] the same", and usually rendered in English as "all other things being equal".

$$P(A_1, A_2, A_3) = P(A_1)P(A_2, \mid A_1)\, P(A_3 \mid A_1) \tag{2.4}$$

This follows from the conditional independence statements encoded in the *DAG* (Pearl, 1988). Given a set of attributes, A_i, *where* $i = 1, \ldots, n$ and a *DAG* associated with this set of attributes, the joint distribution can be written as:

$$P(A_1, A_2, \ldots, A_n) = \prod_{i=1}^{n} P(A_i \mid pa(A_i)) \tag{2.5}$$

where $pa(A_i)$ denotes the set of nodes parent to node A_i. Thus Bayesian networks are able to store large distribution functions, implicitly in terms of small ones and using the factorization above as a key (Pearl, 1988).

There has been significant work in using various forms of Bayesian networks for causal discovery. A *Bayesian network* is a combination of a probability distribution and a structural model that is a directed acyclic graph where the nodes represent the attributes and the edges represent probabilistic dependence. In a *causal Bayesian network* the predecessors of a node are interpreted as directly causing the attribute associated with a node. However, Bayesian networks can be computationally expensive, since a *complete* causal Bayesian network involving thousands of attributes is essentially impossible to establish.

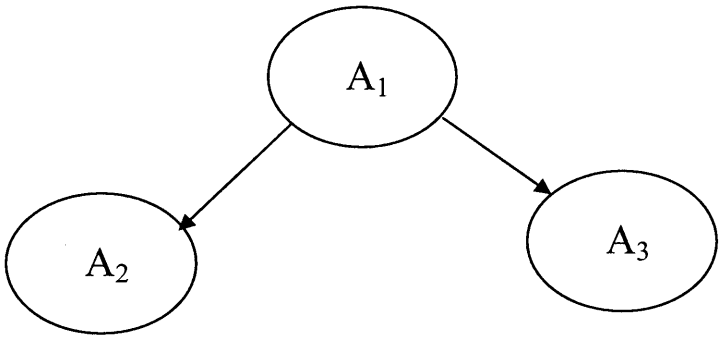


Figure 2.2 Directed acyclic graph (DAG)

2.2.4 Causal diagram

Causal diagrams constitute a second modality of representation in causal modelling. They differ from *Bayesian networks* in that "causal relationships are expressed in the form of deterministic and functional equations, and probabilities are introduced through the assumption that certain attributes in the equations are unobserved" (Pearl, 2000) in contrast to *Bayesian networks* where all relationships are inherently stochastic. *CD*s follow from structural equation models expressed as:

$$X_i = f_i(PA_i, Uu_i), \quad i = 1, \ldots, n \qquad (2.6)$$

where PA_i stands for the set of attributes that are the immediate causes of A_i and the error terms U_i are unobserved quantities through which probabilities are introduced. Each equation represents an autonomous mechanism that determines the value of just one distinct attribute, i.e., the model is causal (Dowe, 2000). Given such a set of structural equations, the corresponding causal diagram is constructed by drawing an arrow from each member of PA_i towards X_i. Such models satisfy the causal Markov condition (Pearl, 2000). Each attribute A_i is independent of all of its non-descendents, given its parent set PA_i.

2.3 Systems Thinking and System Dynamics

The *Systems Thinking* approach assists managers of complex organizations to improve their decision-making abilities by identifying the basic concepts and knowledge of a target system.

System Dynamics is an approach to identifying the behaviour of complex systems over time. It deals with internal feedback loops and time delays that affect the behaviour of the entire system. *System Dynamics* is very similar to systems thinking and constructs the same causal loop diagrams of systems with feedback. However, *System Dynamics* utilises simulation to study the behaviour of systems and the impact of alternative policies.

2.3.1 Systems thinking

A system can be defined as "*a set of interrelated components working together toward some common objective or purpose*" (Blanchard, et. al., 1998). Based on this definition, *Systems Thinking* can be defined as "*a conceptual framework, a body of knowledge and the tools that have been developed over the past fifty years ...*" (Senge, 1990), or as "*the way we can discern some rules, some sense of patterns and events, so we can prepare for the future and gain some influence over the future*" (O'Connor, et al., 1997), or as "*simply a way of thinking about these total systems and their components*" (Churchman, 1968).

There are five aspects of systems (Churchman, 1968) that need to be considered.

(1) The objective of the system and the system performance metrics.

(2) The environment of the system – those things which affect the system but which the system cannot influence very much.

(3) The resources inside the system that it uses to accomplish the objective.

(4) The system's components, their activities and performance.

(5) The management of the system.

Systems Thinking is used as the foundation of a *learning organization – "an organization that is continually expanding its capacity to create its future"* (Senge, 1990), by designing structure into an organization to shape growth, thinking, and communication.

Different *Systems Thinking skills* should be used separately in order to prevent the "cognitive overload" experienced by people trying to learn them all at once (Richmond, 1993) and the following seven skills need special attention.

(1) Dynamic thinking: the ability to focus on patterns of behaviour over time rather than on specific events.

(2) Closed-loop Thinking: observing a system as an interdependent spider web of components rather than as an organizational chart of singular cause and effect relationships.

(3) Generic thinking: attributing events to the structure of the system, not to specific individuals.

(4) Structural thinking: recognizing and appreciating the dimensions of the stocks and flows in a system.

(5) Operational thinking: integrating realistically the essential elements required to make a system operate.

(6) Continuum thinking: replacing the polarizing if-then-else approach to decision-making with acceptance of multiple possibilities along a continuum of alternatives.

(7) Scientific thinking: quantifying attributes and using them to test hypotheses.

2.3.2 System Dynamics

System Dynamics can be defined as "*a qualitative and quantitative approach to describe the model and design structures for managed systems in order to understand how delays, feedback, and interrelationships among attributes influence the behaviour of the systems over time*" (Coyle, 1977), or "*the purpose of the model is to solve a problem, not simply to model a system. The model must simplify the system to a point where the model replicates a specific problem.*" (Sterman, 1994).

The modelling process of *system dynamics* consists of five essential steps (Sterman, 1994).

(1) Problem articulation. To define what the problem is and why it is a problem, identify key attributes, determine the time horizon, and develop reference modes.

(2) Formulation of dynamic hypothesis. To develop a dynamic hypothesis that describes the problem as a result of the relationship among the attributes and the internal structure of the system.

(3) Formulation of a simulation model. To create a computer model by specifying units, relationships, and equations and to clarify the concepts from steps 1 and 2.

(4) Testing. To check the usefulness of the model.

(5) Policy design and evaluation. To identify situations and policies, and to determine the sensitivity and interactions of policies.

With the growing interest in *System Dynamics* comes a lack of guidance for developing *System Dynamics* simulation models (Forrester, 1994). It is important to highlight the importance of starting with a simple model and building increased complexity during the iterative model development process (Ford and Sterman1998). Ford presented eight steps in a modelling process.

(1) To become acquainted with the system

(2) To specify the dynamic problem

(3) To construct the stock and flow diagram

(4) To draw the causal loop diagram

(5) To estimate the parameter values

(6) To test the model

(7) To conduct a sensitivity analysis

(8) To test the impact of policies

The eight steps include all the activities involved from creating to testing *System Dynamics* models, and getting to know the individual or group with the dynamic problem is also recommended. The process also determines if there is a dynamic problem.

2.3.3 System dynamics in practice

System Dynamics (Sterman, 2000) uses causal loop diagrams to describe how entities in a system are connected by feedback loops. Computer software is then used to simulate a *System Dynamics* model of the situation being studied. Currently available *System Dynamics* software, such as Powersim or Vensim, makes it possible for modellers to make a smooth transition through the mechanical aspects of diagramming.

Causal loops diagram

Causal loops diagram are used to get an overview of the causal relationships by identifying possible characteristic behaviour. Figure 2.3 illustrates an example of a causal loops diagram. Arrows symbolise causal relationships between two attributes. Attributes must be formulated in such a way that it makes sense to talk about an increase or decrease of the attribute. The causal relationship must be unidirectional. The polarities must be positive or negative, not both.

Feedback Loops

Feedback Loops can enhance or reduce changes that occur in a system. Positive feedback loops enhance changes; this tends to move a system away from its equilibrium state and make it more unstable. Negative feedback loops tend to reduce changes; this tends to hold a system to some equilibrium state, making it more stable.

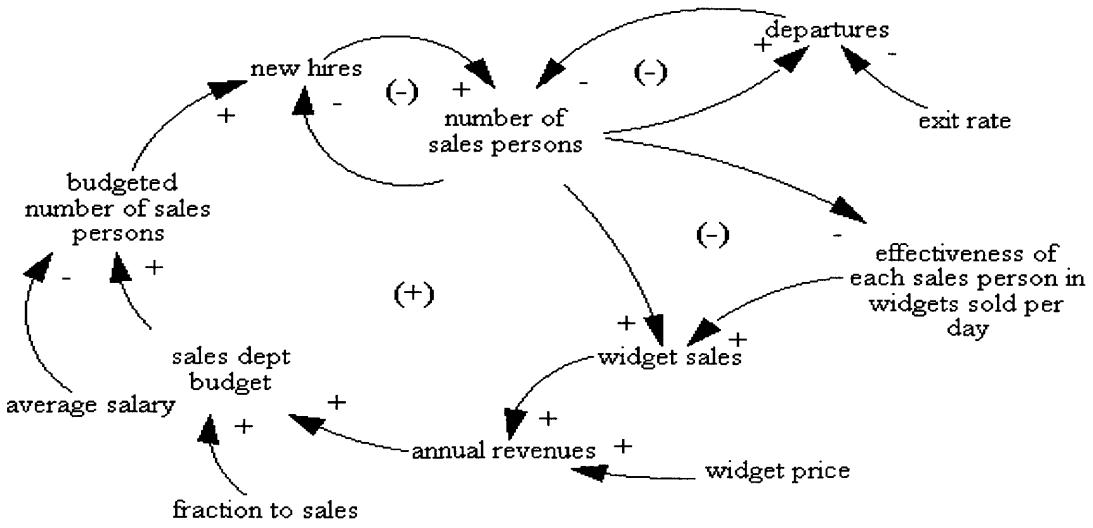


Figure 2.3 Causal loops diagram

A positive connection is one in which a change (increase or decrease) in some attribute results in the same type of change (increase or decrease) in a second attribute. A negative connection is one in which a change (increase or decrease) in some attribute results in the opposite change (decrease or increase) in a second attribute. When these two connections are combined, the positive connection multiplied by the negative connection gives a negative loop feedback effect.

2.3.4 Fundamental modes of dynamic behaviour

There are three fundamental modes of behaviour and three derived modes of behaviour. Each of these modes is generated from a particular type of system structure, which includes a positive loop, a negative loop, and balancing loops with delays. Figure 2.4 shows the causal dynamic behaviour.

Exponential Growth

Exponential growth is caused by an increase or decrease in the output of a target system. It reinforces a change with more change in the same direction. This can lead to rapid growth, e.g. in a virus population, which could be difficult to stop.

Examples of exponential growth behaviour can be found in manufacturing. For instance, as funds are invested to increase the capacity of a plant, more products will be manufactured which will generate more funds which can be again invested to create more capacity.

Goal seeking

Goal seeking represents an adjustment to achieve a certain goal or objective. It indicates a system attempting to change from its current state to a goal state.

This implies that if the current state is above the goal state, then the system forces it down. If the current state is below a goal state, the system pushes it up. Goal seeking behaviour provides useful stability but resists external changes.

When there is a difference between the goal state and the current state, a gap is created. A feedback signal is generated that tends to reduce that difference, the larger the gap the larger the feedback signal. The signal will continue to exist as long as the difference is non zero.

Oscillatory behaviour arises when significant time delays exist in a *fully antipathetic* relationship. Time delays cause feedback to continue after the goal has been attained, which leads to overcorrection.

There are many basic modes of dynamic behaviour and combining *fully sympathetic* and *fully antipathetic* rules creates even more modes. One of the most commonly observed behaviour patterns in complex and dynamic systems is S-shaped growth. S-shaped growth is the result of interaction between an *exponential growth* and a *goal seeking* behaviour. After a start-up period, the growth is rapid but it gradually slows down, as shown in Figure 2.4.

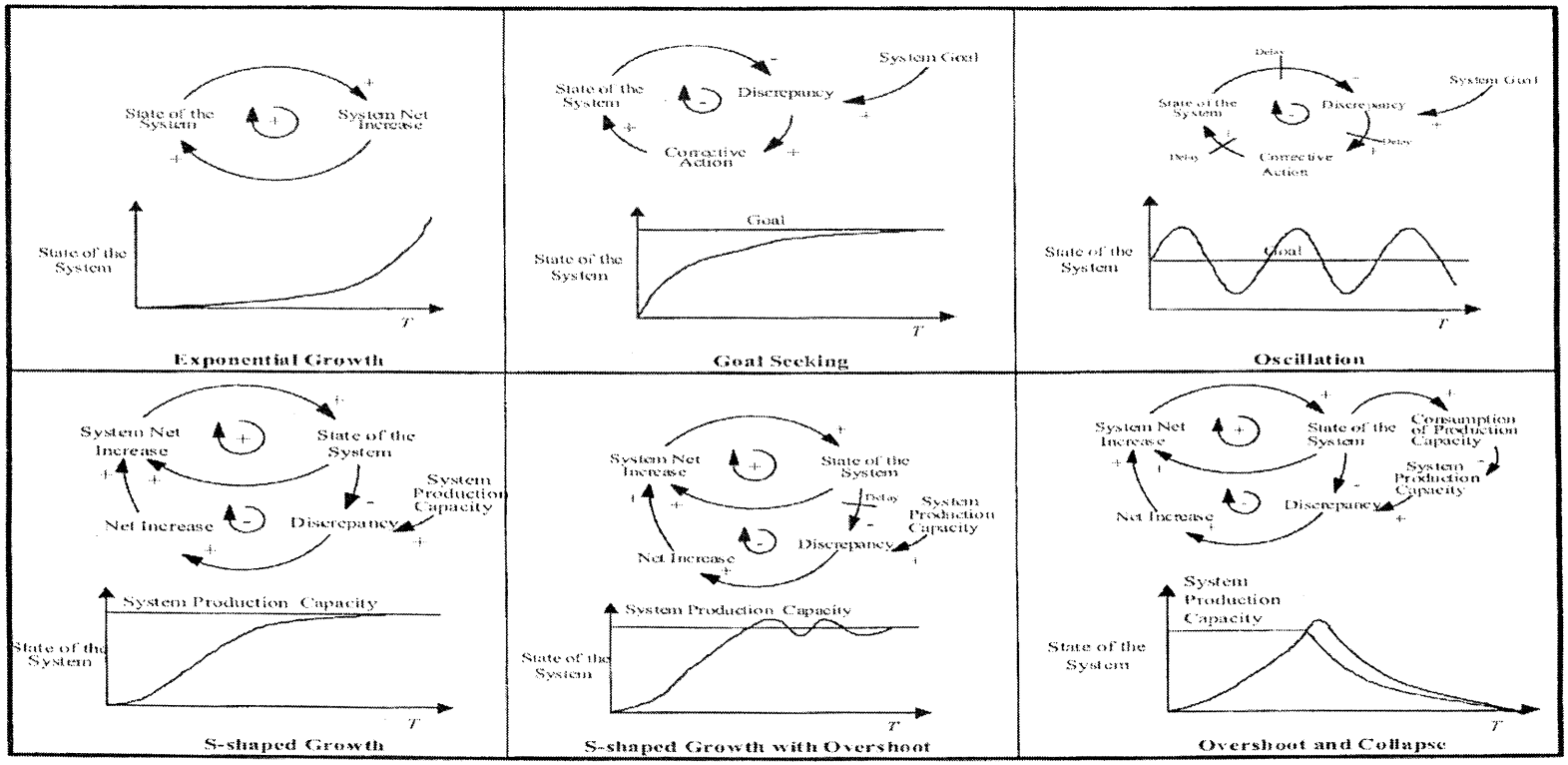


Figure 2.4 Causal dynamic behaviour (Sterman, 2000)

Recent research in *System dynamic* and *Systems Thinking* is mainly focused on the expanding the application areas of both techniques. *Systems Thinking* have been applied for the problems of rural development and community development, where complexity of problems, number of different actors and interests make this approach very useful (Spruill et al., 2001; King, 2000 and Yuri Kondratenko, 2003). *System dynamics* has been applied, for example, in education (Nuhoglu and Nuhoglu, 2007 and Kim et, al., 2007), Policy development in Chinese steel industry (Zheng, 2007), model the Governance Small –to-medium enterprises (Li et. el., 2008) and exploiting the information of dynamic business behaviour to develop business services (An, et. al., 2006).

2.4 Data mining

Data mining extracts hidden rules from very large databases. These rules express causal or predictive relationships between attributes. Sometimes, attributes may or may not have a deterministic relationship; for example, if a customer purchases a hammer, this may causally affect the purchase of nails, since many customers use a hammer with nails (the customer might need to nail a wooden plank). However, at other times the purchase *of a hammer* may not always cause the purchase *of nails* (the customer may just replace a broken hammer). So, there is a conditional primary effect (*hammer with nail*) linked with a secondary effect (needs of nail). One of the main tasks of data mining is to identify the primary and secondary causal relation between attributes.

The discovery association rules are one of the most researched areas of causal analysis and have recently received much attention from chemistry, manufacturing, bio-science, etc (Esposito, et al., 1997). They have proved to be quite useful in the marketing and retail communities as well as in other more diverse fields.

Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of m attributes. Let D be a database, where each record T has a unique identifier, and contains a set of items such that $T \subseteq I$. An *association rule* is an implication of the form $X \Rightarrow Y$, where $X, Y \subset I$, are sets of items called *itemsets*, and $X \cap Y = \phi$. Here, X is called antecedent, and Y consequent (Agrawal, et al., 1993 and Cheung, et. al, 1996).

Two important measures are applied for association rules, *support* and *confidence*. The *support* of an association rule is the ratio (usually as a percentage) of the records that contain $X \cup Y$ to the total number of records in the database. Therefore, if the support of a rule is 5% then it means that 5% of the total records contain $X \cup Y$. *Support* is the statistical significance of an association rule.

For a given number of records, *confidence* is the ratio (usually as a percentage) of the number of records that contain $X \cup Y$ to the number of records that contain X. Thus, if a rule has a *confidence* of 85%, it means that 85% of the records containing X also contain Y. The *confidence* of a rule indicates the degree of correlation in the dataset between X and Y. *Confidence* is a measure of a rule's strength.

Association mining consists of finding all rules that meet the user-specified threshold *support* and *confidence*. The problem of mining association rules can be decomposed into the following three sub-problems (Agrawal and Srikant, 1994):

(1) Finding all sets of items, which occur with a frequency that is greater than or equal to the user-specified threshold *support.*

(2) Generating the desired rules using the large *itemsets*, which have user-specified threshold *confidence.*

(3) For every subset, obtaining a rule of the form $x \Rightarrow (l - x)$ if the ratio of the frequency of occurrence of l to that of x is greater than or equal to the threshold *confidence.*

The first step finds large or frequent *itemsets. Itemsets* other than those are referred to as *redundant itemsets*. Here an *itemset* is a subset of the total set of items of interest from the database. An interesting observation about large *itemsets* is that:

> If an *itemset* X is small, any superset of X is also small.

The second step finds association rules using the large *itemsets* obtained in the first step. The identification of the large *itemsets* is computationally expensive (Agrawal and Srikant, 1994). Since finding large *itemsets* in a huge database is very expensive and dominates the overall cost of mining association rules, most research has been focused on developing efficient algorithms to solve step 1 (Agrawal and Srikant, 1994; Cheung, et al., 1996 and Klemettinen, et al., 1994). The following section provides an overview of these algorithms.

2.4.1 Basic algorithms

Associative items set (AIS)

The Associative items set (AIS) algorithm was the first published algorithm developed to generate all large *itemsets* in a transaction database (Agrawal, et al., 1993). It focuses on the enhancement of databases with the necessary functionality to process decision support queries. This algorithm is targeted to discover qualitative rules.

The AIS algorithm makes multiple passes over the entire database. During each pass, it scans all transactions. In the first pass, it counts the *support* of individual items and determines which of them is large or frequent in the database. Large *itemsets* of each pass are extended to generate candidate *itemsets*.

After scanning a transaction, the common *itemsets* between the large *itemsets* of the previous pass and items of this transaction are determined. These common *itemsets* are extended with other items in the transaction to generate new candidate *itemsets*. A large itemset *l* is extended with only those items in the transaction that are large and occur in the lexicographic ordering of items later than any of the items in *l*.

If an *itemset* is absent in the whole database, it can never become a candidate for measurement of large *itemsets* in the subsequent pass. To avoid replication of an *itemset*, items are kept in lexicographic order.

Set oriented mining (SETM)

The Set oriented mining (SETM) algorithm is motivated by the desire to use SQL to calculate large *itemsets* (Srikant and Agrawal, 1996 and Houtsma and Swami, 1995). In this algorithm each member of the frequent *itemsets*, $\overline{L_k}$, is in the form $\langle TID, itemset \rangle$ where *TID* is the unique identifier of a transaction. Similarly, each member of the set of candidate *itemsets*, $\overline{C_k}$, is in the form $\langle TID, itemset \rangle$ <TID, itemset>.

The SETM algorithm also makes multiple passes over the database. In the first pass, it counts the *support* of individual items and determines which of them is large or frequent in the database. Then, it generates the candidate *itemsets* by extending large *itemsets* of the previous pass. In addition, the SETM remembers the *TIDs* of the generating transactions with the candidate *itemsets*. The relational merge-join operation can be used to generate candidate *itemsets* (Srikant and Agrawal, 1996).

Apriori

Apriori generates the candidate *itemsets* by joining the large *itemsets* of the previous pass and deleting those subsets which are small in the previous pass without considering the transactions in the database (Agrawal and Srikant, 1994 and Cheung, et. al., 1996b). By only considering large *itemsets* of the previous pass, the number of candidate large *itemsets* is significantly reduced.

Figure 2.5 shows the pseudo code for the Apriori algorithm. The apriori_gen() function consists of two steps. During the first step, L_{k-1} is joined with itself to obtain C_k. In the

second step, apriori_gen() deletes all *itemsets* from the join result, which have some (*k-1*)–subset that is not in L_{k-1}. Then, it returns the remaining large *k-itemsets*.

In the first pass, the *itemsets* with only one item are counted. The discovered large *itemsets* of the first pass are used to generate the candidate sets of the second pass using the apriori_gen() function. Once the candidate *itemsets* are found, their *supports* are counted to discover the pairs of large *itemsets* by scanning the database. In the third pass, the large *itemsets* of the second pass are considered as the candidate sets to discover large *itemsets* of this pass.

Counting *support* of candidates is a time-consuming step in the algorithm (Cengiz, 1997). To reduce the number of candidates that need to be checked for a given transaction, candidate *itemsets* C_k are stored in a hash tree. When an *itemset* is inserted, it is required to start from the root and go down the tree until a leaf is reached. Furthermore, L_k are stored in a hash table to make the pruning step faster (Srikant and Agrawal, 1995).

2.4.2 Partition algorithm

The general idea is to reduce the number of database scans to *2* (Savasere, et al., 1995). It divides the database into small partitions such that each partition can be handled in the main memory. Since each partition can fit in the main memory, there will be no additional disk *I/O* for each partition after loading the partition into the main memory. Then the unions of the local large *itemsets* found in each partition are used as the candidates and are counted through the whole database to find all the large *itemsets*.

Apriori_gen() [Agrawal, 1994]
Input: set of all large (k-1)-itemsets L_{k-1}
Output: A superset of the set of all large k-itemsets

I_i = Items I //Join step
insert into C_k
Select $p.I_1, p.I_2, \ldots\ldots, p.I_{k-1}, q.I_{k-1}$
From L_{k-1} is p, L_{k-1} is q
Where $p.I_1 = q.I_1$ and ..... and $p.I_{k-2} = q.I_{k-2}$ and $p.I_{k-1} < q.I_{k-1}$.

forall itemsets $c \in C_k$ **do** //pruning step
    forall (k-1)-subsets s of c **do**
        If $(s \notin L_{k-1})$ **then**
    delete c from C_k

Function count(C: a set of itemsets, D: database)
    begin
        for each transaction $T \in D = \bigcup D^i$ **do begin**
            forall subsets $x \subseteq T$ **do**
                if $x \in C$ **then**
                    x.count++;
            end
        end

Apriori [Agrawal, 1994]
Input:
I, D, s
Output:
L
$C_1 := I$;
for $(k = 2; L_{k-1} \neq \phi; k++)$ **do begin**
    C_k = apriori-gen(L_{k-1});
    Count (C_k, D)
    $L_k = \{c \in C_k \mid c.count \geq minsup\}$
    end
$L := \bigcup_k L_k$

Figure 2.5 Apriori algorithm

The partition algorithm has been applied widely in data mining analysis. Figure 2.6 shows an example of the partition algorithm's pseudo code.

2.4.3 Parallel algorithm

The parallel algorithms can be classified as *data parallelism* and *task parallelism* (Chattratichat, et al., 1997). The two paradigms differ as to whether the candidate set is distributed across the processors. In the task parallelism paradigm, the candidate set is partitioned and distributed across the processors, and each node counts a different set of candidate (Agrawal and Shafer, 1996, Park, et al., 1995, Cheung, et al., 1996 and Zaki, et al., 1997). In the data parallelism paradigm, each node counts the same set of candidates (Agrawal and Shafer, 1996; Han, et al., 1997 and Shintani and Kitsuregawa,1998).

The data parallelism paradigm has simpler communication and thus less communication overhead. It only needs to exchange the local counts of all candidates in each iteration. The basic count distribution algorithm can be further improved by using hash techniques, candidate pruning techniques and short-circuited counting. However, the data parallelism paradigm requires that all the candidates fit into the main memory of each processor.

2.4.4 Sampling algorithm

The sampling method, reduces the number of database scans to one in the best case and two in the worst (Toivonen, 1996). The pseudo code is shown in Figure 2.7. A sample which can fit in the main memory is first drawn from the database. The set of large itemsets in the sample is then found from this sample by using Apriori.

Algorithm PARTITION (Savasere, 1995)
Input:
I- items, s-support , D^1-data base partition 1, D^2-data base partition 2, ..., D^p-data base partition P
Output:
L – Large itemsets
Algorithm: //scan one computes the local large itemsets in each partition
for i from 1 to p **do**
L^i = Apriori(I,D^i,s); //L^i are all local large itemsets(all sizes) in D^i
//scan two counts the union of the local large itemsets in all partitions
$C = \bigcup_i L^i$;
count(C, D) = $\bigcup D^i$;
return $L = \{x \mid x \in C, x.count \geq s \times |D|\}$;

Figure 2.6 Partition algorithm

Let the set of large itemsets in the sample be PL. The candidates are generated by applying the negative border function, BD^-, to PL. The negative border of a set of itemsets PL is the minimal set of itemsets which are not in PL. The negative border function is a generalization of the apriori_gen function in Apriori. The negative border can be applied to a set of itemsets of different sizes, while the function apriori_gen() only applies to a single size. After the candidates are generated, the whole database is scanned once to determine the counts of the candidates.

2.4.5 Fuzzy association algorithm

Real world data is often filled with imprecise data that has to be normalized into well-defined and unambiguous data in order to be handled by the standard relational data model. Several extensions to the classical relational model have been proposed (Agrewal and Srikant, 1996) to support quantitative data.

The fuzzy association rule is of the form "If X is F_X then Y is F_Y". As in the binary association rule, "X is F_X " is called the antecedent of the rule while "Y is F_Y" is called the consequent of the rule. X, Y are sets of attributes of database and F_X, F_Y are sets containing fuzzy sets which characterize X and Y respectively.

The fuzzy approach represents a more robust solution for the lack of flexibility. This approach not only obtains more human-understandable knowledge from the database data but also provides more compact and robust representations not weakened by "patched" data types based on a strong theoretical model (Zaki, et al., 1997).

Algorithm Sampling [Toivonen, 1996]
Input:
I- items, s-support, D-database
Output:
L –large itemset
Algorithm: //draw a sample and find the local large itemsets in the sample
Ds = a random sample drawn from D;
PL = Apriori(I,Ds,s);//first scan counts the candidates generated from PL

$C = PL \cup BD^{-}(PL)$;
count(C, D);//second scan counts additional candidates if there are misses in $BD^{-}(PL)$
$ML = \{x \mid x \in BD^{-}(PL), x.count \geq s \times |D|\}$; //ML are the misses
if $ML \neq \varnothing$
then //MC are the new candidates generated from the misses
$MC = \{x \mid x \in C, x.count \geq s \times |D|\}$;
repeat $MC = MC \cup BD^{-}(MC)$;
until MC doesn't grow;

MC = MC - C); //itemsets in C have already been counted in scan one

count(MC, D);
return $L = \{x \mid x \in C \cup MC, x.count \geq s \times |D|\}$;

Figure 2.7 Sampling algorithm

Additionally, mining association rules based on fuzzy sets can handle quantitative data and create smoother transition boundaries between partitions for numerical values, constituting a perfect solution for both well-defined and imprecise data.

In order to introduce fuzziness into the relational data model, structural modifications have to be introduced to represent and manage the quantitative data. Two major approaches have been proposed: the proximity relation model (Brin, et al., 1982) and the model based on probability distribution (Agrawal, et al., 1993).

The Boolean association rules have been adapted to handle quantitative data based on the fuzzy layout (Kuok, et.al., 1998; Fu, et al., 1998 and Hong, et al., 2000), reusing all the previous research and algorithms without the need to discover new techniques.

In (Raju and Majumdar, 1988) a deep analysis of different relations in the fuzzy domain was established, however, most of the mining algorithms used only the fuzzy association rule or functional dependence to extract the relations.

The fuzzy mining algorithms relieved the fuzzy sets of the quantitative attributes creation and the membership functions to an external entity, usually to the end user or an expert (Fu, et.al., 1998). The quality of the results produced by the algorithm relies quite crucially on the appropriateness of the fuzzy sets to the given data.

2.4.6 Formal concept algorithm

Formal Concept Analysis (FCA) provides a strong theory for improving both performance and results of association rule mining algorithms. The use of FCA allows not only an efficient computation but also a drastic reduction in the number of rules that have to be presented to the user, without any information loss.

Early algorithms for computing the concept lattice described a minimal set of *implications* (exact rules) from which all rules can be derived (Fay, 1973 and Ganter, 1984).

Based on the previous work, the connection between *Data mining* and *Formal Concept Analysis* was discovered (Pasquier, et al., 2001; Zaki and Hsiao, 1999 and Stumme, 1999). Since then, the attention on Formal Concept Analysis has increased largely within the data mining community and many researchers have joined the field.

Inspired by the observation that the frequent closed itemsets are sufficient to derive all frequent itemsets and their supports, the search for other representations began. The key itemsets can be classified as *disjunction-free sets* (Bykowski and Rigotti, 2001) or as *disjunction free generators* (Kryszkiewicz, 2001). The frequent disjunction-free sets together with their support allow computing the support of all frequent itemsets. This approach is further extended into *non-derivable itemsets*, where the previous equation is extended from the two elements to an arbitrary number of elements (Calder and Goethals, 2002).

A second trend is the analysis of *approximate representations,* where itemsets have `almost no dependencies relation' (Boulicaut and Bykowski, 2000). The support of any frequent itemset can be computed up to a certain error.

2.4.7 Other algorithms

Apriori-TID

As mentioned earlier, Apriori scans the entire database in each pass to count *support.* Scanning of the entire database may not be needed in all passes. Based on this conjecture, Agrawal (Agrawal and Srikant, 1994) proposed another algorithm called Apriori-*TID.* Similar to Apriori, Apriori-*TID* uses the Apriori's candidate generating function to determine candidate *itemsets* before the beginning of a pass. The main difference from Apriori is that it does not use the database for counting *support* after the first pass. Rather, it uses an encoding of the candidate *itemsets* used in the previous pass denoted by $\overline{C_k}$.

Apriori-Hybrid

Apriori has better performance in earlier passes and Apriori-*TID* outperforms Apriori in later passes (Agrawal and Srikant, 1994). Based on this experimental observation, the Apriori-Hybrid technique uses Apriori in the initial passes and switches to Apriori-*TID.* It is observed that Apriori-Hybrid performs better than Apriori except in the case when the switching occurs at the very end of the passes (Srikant and Agrawal, 1996).

Dynamic Itemset Counting

Dynamic Itemset Counting tries to generate and count the *itemsets* earlier, thus reducing the number of database scans (Brin, et al., 1997). The database is viewed as intervals of transactions and the intervals are scanned sequentially. The major idea is that when reaching the end of the database, it rewinds the database to the beginning and counts the *itemsets*. The actual number of database scans depends on the interval size. If the interval is small enough, all *itemsets* will be generated in the first scan and fully counted in the second scan. It also favours a homogeneous distribution as does the partition algorithm.

Multiple Min-supports Association Rules

If the threshold support is set too high, rules involving rare items will not be found. If the threshold is set too low, this may produce too many redundant rules (Liu, et al., 1999). Two strategies are suggested to solve this problem.

(a) split the data into a few blocks according to the supports of the items and then discover association rules in each block with a different threshold support.

(b) group a number of related rare items together into an abstract item so that this abstract item is more frequent. Then apply the algorithm for finding association rules in numerical interval data (Han and Fu, 1995 and Liu, et al.,1999).

Continuous Association Rule Mining Algorithm (CARMA)

CARMA brings the computation of large *itemsets* online (Hidber, 1999). CARMA shows the current association rules to the user and allows the user to change the parameters,

minimum support and minimum confidence, at any transaction during the first scan of the database.

SQL Extensions

There have been several proposed SQL extensions to facilitate the generation of association rules (Houtsma and Swami, 1995 and Meo, et al., 1996). The query is submitted with the select statement and is terminated with an *extracting rule* clause which includes the support and confidence values requested.

Recent research

Most recent analysis in data mining is based on combination of the existing algorithms and exploring the new area of mining application. Nanopoulos (Nanopoulos, et. al., 2007) suggests a Parallel Multi-pass algorithm that partitions the domain of items according to their correlations. Parallel Multipass algorithm is combined with Inverted Hashing and Pruning for mining association rules between words in text databases, which was shown to be more efficient than other existing algorithms in the context of mining text databases (Holt and Chung, 2007). Another similar framework for parallel mining frequent itemsets and sequential patterns based on sampling technique was suggested by Cong (Cong et.al. 2005).

Constrained parallel mining, which was run using a 32 process cluster on a retail database containing one billion transactions, took less than an hour and a half to complete the process (Mohammad and Osamar, 2006). Chen (Chen, et. al., 2008) treated products'

prices as an important decision variable in mining retail knowledge. Moustakidesa and Verykios (Moustakidesa and Verykios, 2008) suggest an approach applying the max-min criterion upon the support theory of frequent itemsets.

2.5 Summary

This chapter gives some fundamental insight into *general causal analysis*, *System Dynamics,* and *Association Mining*.

System Dynamics is a modelling method that simulates complex systems to design more effective policies. During the simulation, space and time can be compressed and slowed so the long-term side effects of decisions can be experienced. *A System Dynamics* model is created by identifying dynamic causal relations based on the *dynamic hypothesis*. Real data is then entered to allow the simulation to run in order to evaluate the result (Sterman, 2000). The modelling process is based on individual or group understanding of the system. *System Thinking* is applied to understand the target system by studying larger numbers of interactions inside and around the system (Sterman, 2000). This understanding might take a very long time to develop and is dependent upon the number of entities related to the system. Also, as each individual has their own view of the system when working in a group, this can lead to conflict in model development.

Association rule mining finds interesting associations and/or correlation relationships among large sets of data items. Association rules show attributes value conditions that

occur frequently together in a given dataset. A typical and widely-used example of association rule mining is Market Basket Analysis. Association rules provide information of this type in the form of "if-then" statements. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature.

Finding causality in the context of *Association Mining* can be less time consuming than *System Dynamics.* But the increasing size of the data still poses a challenge and *Association Mining* is used mainly for exploratory analysis. Although *Association Mining* does allow for some specializations and optimizations of causal algorithms, it discovers only causal relationships of a static nature.

Chapter 3 Dynamic causal mining (DCM)

3.1 Preliminaries

Causality plays a central role in human reasoning and decision-making by making the observation and recognition of the causality. A common scientific approach to recognizing causality is by manipulating attributes by experimentation. However, real world events are often affected by a large number of potential factors and many of these factors are hidden. For example, in manufacturing, many factors such as cost, labour, weather, social events, etc., can all affect the final product. Some of the factors, such as cost and labour, have a clear causality with the final product. Other factors, such as weather and social events may not have a clear causality with the final product.

This chapter suggests an integration of *System Dynamics* and *Association Mining* for identifying causality and expanding the application area of both techniques. This gives an improved description of the target system represented by a database; it can also improve strategy selection and other forms of decision making. Such a combination extracts important dynamic causality. This type of causality is very common in daily life. For example, "an increase of productivity in a factory might cause an increase of pollution in the environment" and "the increasing pollution will cause a decreasing level of human health and welfare". In the real world, an occurrence of an event is often affected by a large number of potential factors. The aim is to identify causal factors hidden in the data and discover the underlying causality between the observed data.

3.2 Dynamic causal mining

This section provides a general framework for *DCM*. It is an iterative and continual process of mining rules, formulating policies, testing, and revision of the models. The stages are depicted in Figure 3.1.

Stage 1: Problem definition. In this phase, the problem is identified and the key variables are given. Also, the time horizon is defined so that the cause and effects can be identified.

Stage 2: Data preparation. Data are collected from various sources and a homogeneous data source is created to eliminate the representation and encoding differences.

Stage 3: Data mining. This stage involves transforming data into rules. This thesis suggests using *DCM* as a data mining tool. The details of *DCM* are explained in the following sections and next chapter.

Stage 4: Policy formulation. Policies are groups of the rules extracted by mining techniques. Policies improve the understanding of the system. The interactions of different policies must also be considered since the impact of combined policies is usually not the sum of their impacts alone. These interactions may reinforce each other or have an opposite effect. The policy can be used for behaviour simulation to predict the future outcome.

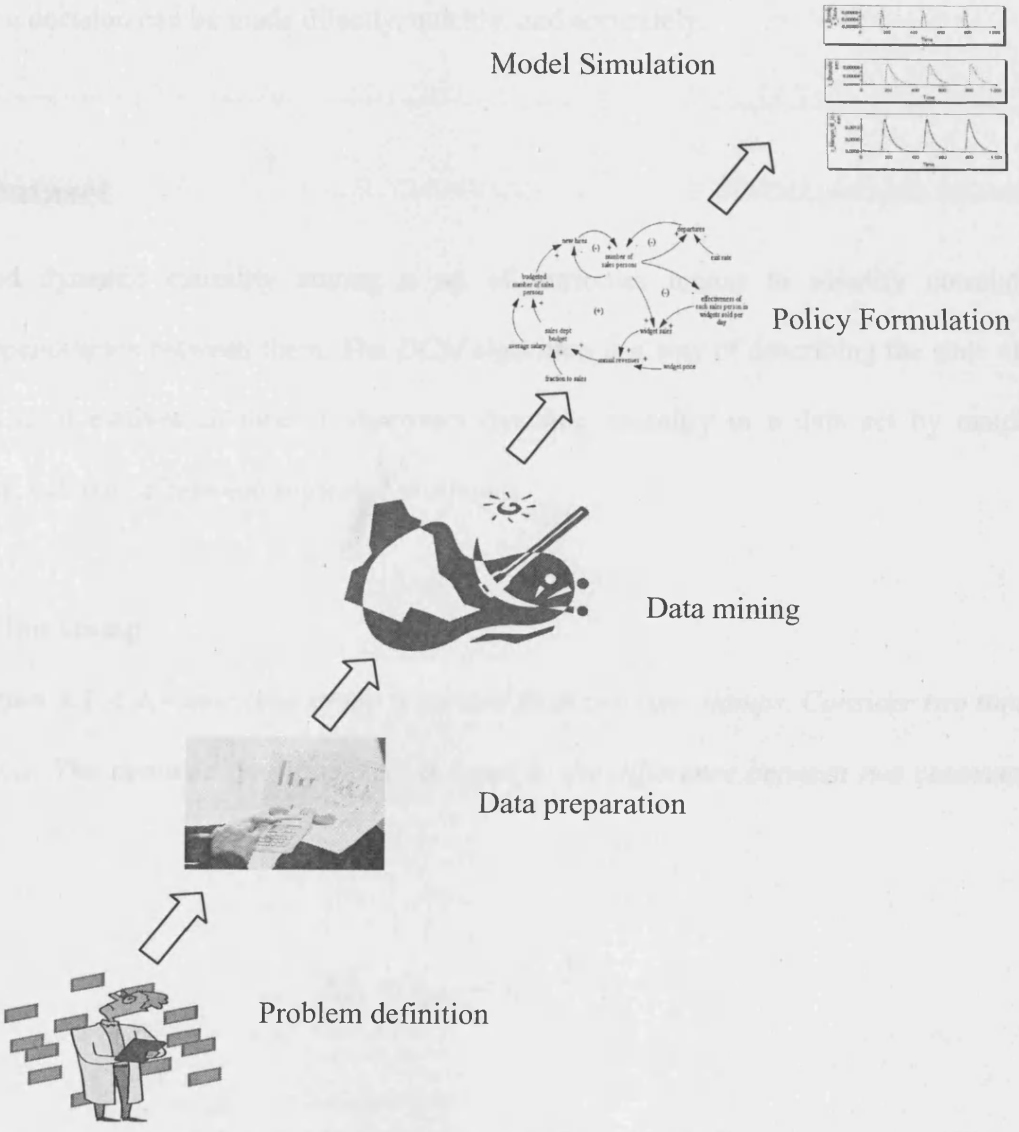


Figure 3.1 *DCM* process

Stage 5: Model Simulation. This stage tests the accuracy of the policies. The policies will predict results for new cases so the managers can alter the policy to improve future behaviour of the system. It is necessary to capture the appropriate data and generate a prediction in real time, so that a decision can be made directly, quickly, and accurately.

3.3 Dataset

To find dynamic causality among a set of attributes means to identify correlation and interdependencies between them. The *DCM* algorithm is a way of describing the state of a target system as it evolves in time. It discovers dynamic causality in a data set by matching the dynamic behaviour between separated attributes.

3.3.1 Time stamp

Definition 3.1 *A dynamic time stamp is created from two time stamps. Consider two time stamps* t_i *and* t_{i+1}. *The dynamic time stamp* Δt_i *is equal to the difference between two consecutive time stamps.*

$$\Delta t_i = t_{i+1} - t_i \tag{3.1}$$

Time stamps are used for identifying the range of variables. The size of each time stamp is selected by the specific need and may vary in different situations. For instance, a time stamp for an increase in production may be in the order of months, while for a change in a cell may be in

the order of milliseconds. The time stamp also can help to determine how detailed the variables need to be. The attribute may increase or decrease dramatically if the time stamp is in the order of seconds, however it may be assumed to be constant if the time stamp is in the order of years. All the time stamps should be of uniform length. In order to carry out *DCM*, the time stamps are summarised or partitioned into equal-sized time stamps. A time stamp is useful for describing and prescribing changes to the systems and objects.

3.3.2 Data

Definition 3.2 A *dynamic attribute is the change or the difference between two attribute values with consecutive time stamps. The two types of value do not have the same nature. Let D denote a data set which contains a set of n records with attributes* $\{A_1, A_2, A_3, \ldots A_m\}$, *where each attribute is of a unique type (for example, sale price, production volume, inventory volume, etc). Each attribute is associated with a time stamp* t_i, *where* $i = \{1,2,3,\ldots n\}$. *Let* D_{new} *be a new database constructed from D such that dynamic attribute* $\Delta A_{m,\Delta t_i}$ *in* D_{new} *is given by:*

$$\Delta A_{m,\Delta t_i} = A_{m,t_{i+1}} - A_{m,t_i} \tag{3.2}$$

where m identifies the attribute of interest. Table 3.1 shows the original database D with two attributes. Table 3.2 shows the derived database D_{new} *from Table 3.1.*

Time	A_1	A_2
1	9	2
2	17	3
3	10	12
4	4	16
5	7	24

Table 3.1 Original database D

Δt	ΔA_1	ΔA_2
Δt_1	+8	+1
Δt_2	-7	+9
Δt_3	-6	+4
Δt_4	+3	+8

Table 3.2 Derived database D_{new}

The classical *Association Mining* algorithms can be applied only to data in the original form (attribute form), e.g. in the market basket problem (Agrawal, et. al., 1993) the focus is on the items of each purchase. On the other hand, *DCM* is interested in the dynamic changes between data. To apply *DCM*, the records are arranged in a temporal sequence ($t = 1, 2, ..., n$). Definition 3.2 is only for numerical attributes and the causality between categorical attributes in D can be identified by examining the differences of corresponding changes in attribute values.

Definition 3.3 *In the case of categorical attributes, the dynamic causal attributes can be identified by joining the polarities of corresponding changes in attribute values. Let* D_{new} *be a new data set constructed from D such that attribute* $\Delta A_{m,\Delta t_i}$ *in* D_{new} *is given by:*

$$\Delta A_{m,\Delta t_i} = join(A_{m,t_{i+1}}, A_{m,t_i}) \tag{3.3}$$

where join is a function combining $A_{m,t_{i+1}}$ *and* A_{m,t_i}. *For example, (* $t=1$, $A_{m,t_{i+1}}$ = *Red) and (*$t=2$, A_{m,t_i} = *Blue) then (*$\Delta t = 1$, $\Delta A_{m,\Delta t_i}$ = *RedBlue).*

The *attribute* is gathered first and the *dynamic attribute* is derived from the *attribute*. The *dynamic attribute* identifies the significant relationship between the dynamics of the *attribute*.

3.4 Measurements

Since the input of the *DCM* algorithm can be quite large, it is important to prune away the redundant attributes.

Definition 3.4 *A polarity indicates the direction of a change of an attribute. There are three types of polarity (+, -, 0); where + indicates an increase, - indicates a decrease, and 0 indicates neutrality, i.e. no change at all.*

Definition 3.5 *A polarity combination is a joint set of two or more polarities. The simultaneous presence of combinations (+,+) and (-,-) indicates sympathetic changes and will produce a sympathetic rule. The simultaneous presence of combinations (+,-) and (-,+) indicates antipathetic changes and produces antipathetic rules. There are four different combinations of polarity (+,-) (antipathetic negative), (+,+) (sympathetic positive), (-,+) (antipathetic positive) and (-,-) (sympathetic positive) to indicate the degree of causality.*

This differs from the classical causal loops relation which has only + and -, due to a simultaneous increase of an attribute set not automatically leading to a simultaneous decrease of the same set.

Definition 3.6 A *support is the ratio of records of a certain polarity combination over the total number of records in the dynamic attributes. Three supports are applied in DCM; sympathetic support, antipathetic support, and single support. For data set* D_{new} *and any two attributes* $\Delta A_{1,\Delta t_i}$ *and* $\Delta A_{2,\Delta t_i}$ *the three kinds of supports are defined as follows.*

$$\textbf{\textit{Sympathetic Support}}\ (\Delta A_{1,\Delta t_i}, \Delta A_{2,\Delta t_i}) = \frac{freq\,(+,+)}{n} \qquad (3.4a)$$

or $$\frac{freq\,(-,-)}{n} \qquad (3.4b)$$

$$\textbf{\textit{Antipathetic Support}}\ (\Delta A_{1,\Delta t_i}, \Delta A_{2,\Delta t_i}) = \frac{freq\,(+,-)}{n} \qquad (3.5a)$$

or $$\frac{freq\,(-,+)}{n} \qquad (3.5b)$$

$$\textbf{\textit{Single Attribute Support}}\ (\Delta A_{m,\Delta t_i}) = \frac{freq(+)}{n} \qquad (3.6a)$$

or $$\frac{freq(-)}{n} \qquad (3.6b)$$

or $$\frac{freq(0)}{n} \qquad (3.6c)$$

where freq (+,+) is a function counting the number of times where an increase in $\Delta A_{1,\Delta t_i}$ *is associated with a simultaneous increase in* $\Delta A_{2,\Delta t_i}$, *freq (-,-) is a function counting the number of times where an decrease in* $\Delta A_{1,\Delta t_i}$ *is associated with a simultaneous decrease in* $\Delta A_{2,\Delta t_i}$, *freq (-,+) is a function counting the number of times where an decrease in* $\Delta A_{1,\Delta t_i}$ *is associated with a simultaneous increase in* $\Delta A_{2,\Delta t_i}$, *freq (+,-) is a function counting the number of times where an increase in* $\Delta A_{1,\Delta t_i}$ *is associated with a simultaneous decrease in* $\Delta A_{2,\Delta t_i}$, *etc.*

All *supports* relate to the frequencies of the occurring patterns. For a given user specified *support*, the problem of *DCM* is to find all rules where the support is greater than the user defined *support*. The *support* is the frequency of occurrences of attribute sets that support a rule.

Figure 3.2 shows the results of the comparison between two dynamic attributes. The *X*-axis represents the time stamp and the *Y*-axis represents the value of the dynamic attributes. Figure 3.3 shows the graphical representation of two dynamic attributes, where the polarity combination is indicated.

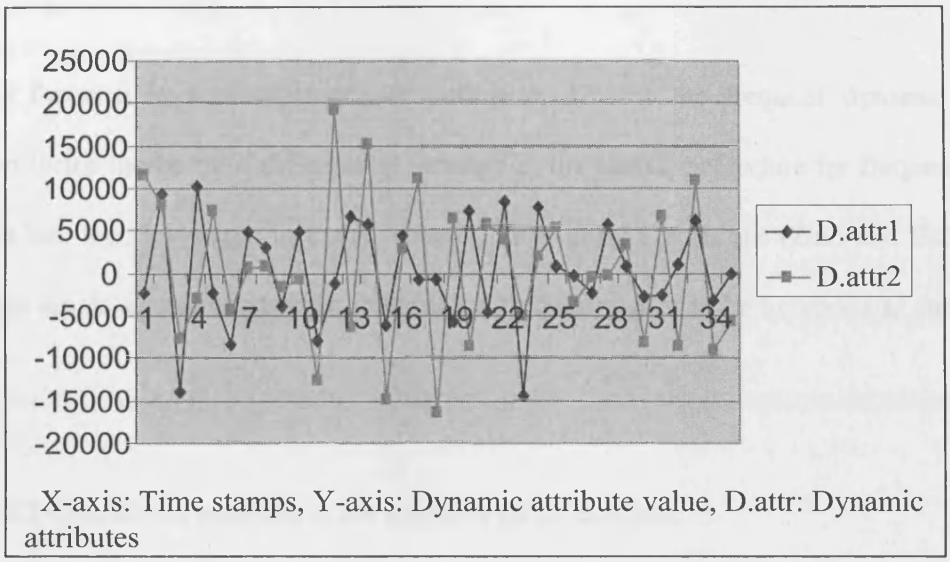
Definition 3.7 A support level is the value threshold for each dynamic attribute. Every record in a dynamic attribute must have an absolute value larger or equal to the support level in order to be considered as candidate for the dynamic rule.

For a given *support level*, a positive and negative value of the *support level* can be then drawn as shown in Figure 3.3. The support is the occurrence of the polarity combination above the value of the support level. Table 3.3 illustrates the support counting direction between two dynamic attributes. Table 3.4 shows the counting results.

Definition 3.8 *A frequent dynamic set is a pair of dynamic attributes which contain a polarity combination with frequency occurrence above a user-defined support threshold.*

Theorem 3.1 *If a pair of dynamic attributes* ($\Delta A_1, \Delta A_2$) *is infrequent, then either one individiual dynamic attribute is infrequent or both dynamic attributes are infrequent.*

Proof : *If a dynamic attribute set is frequent, then this indicates that both the dynamic attributes are above the user-defined threshhold.*



X-axis: Time stamps, Y-axis: Dynamic attribute value, D.attr: Dynamic attributes

Figure 3.2 Graph of two dynamic attributes

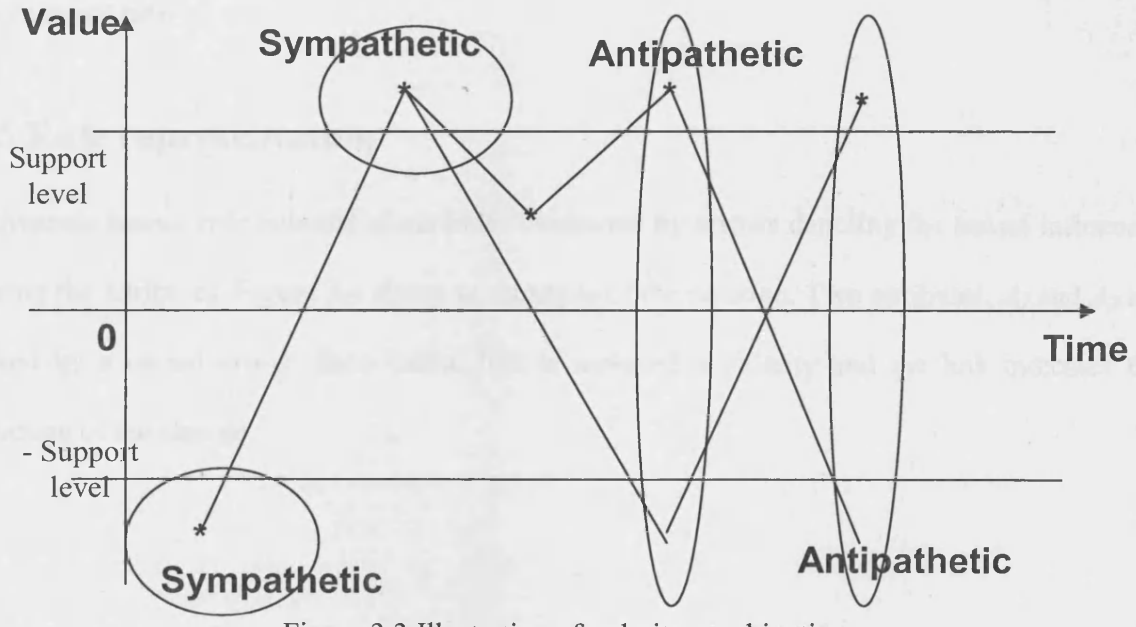


Figure 3.3 Illustration of polarity combination

The above theorem is a consequence of definition 3.7 for the frequent dynamic set. This observation forms the basis of the pruning strategy in the search procedure for frequent dynamic sets, which has been leveraged in many Association Mining algorithms (Zaki and Hsiao, 1999), that only the single dynamic attribute found to be frequent needs to be extended as candidate for the rule.

Theorem 3.2 *Confidence measure is not useful in DCM analysis.*

Proof: *The confidence measure is not used here because the total numbers of records in an attribute is equal to the total number of time stamps. Thus it makes the confidence equal to the support. Instead, multiple supports are introduced to further reduce the running time and size of the relevant rules.*

3.5 Rule representation

A dynamic causal rule consists of variables connected by arrows denoting the causal influences among the attributes. Figure 3.4 shows an example of the notation. Two attributes, A_1 and A_2 are linked by a causal arrow. Each causal link is assigned a polarity and the link indicates the direction of the change.

Δt	ΔA_1	ΔA_2
Δt_1	+8 →	+1
Δt_2	-7 →	+9
Δt_3	-6 →	+4
Δt_4	+3 →	+8

Table 3.3 Derived database D_{new} with arrows indicating *support* counting direction

	(+,+)	(-,-)	(+,-,)	(-,+,)
Supports($\Delta A_1, \Delta A_2$)	2/4	0	0	2/4

Table 3.4 Counting result

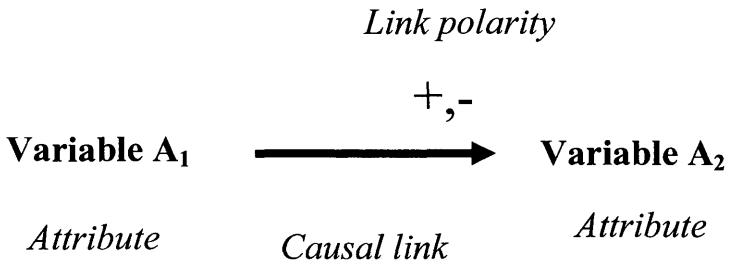


Figure 3.4 Notation of a dynamic causal rule

In *System Dynamics,* a symbol $x \rightarrow^{+} y$ can be interpreted as $\delta y/\delta x > 0$ and $x \rightarrow^{-} y$ can be interpreted as $\delta y/\delta x < 0$. This analogy is applied in *DCM* and the *dynamic causal rules* produced by *DCM* can form causal diagrams, which will be used to simulate future behaviour.

Definition 3.9 *A dynamic causal rule is derived from a frequent dynamic attribute set. A dynamic causal rule can be either strong or weak. A weak rule is a set of attributes with polarity that partially fulfils equation (3.4), (3.5), or (3.6). A strong rule is a set of attributes with polarity that completely fulfils equation (3.4), (3.5), or (3.6). There are two types of strong rule, sympathetic and antipathetic.*

Figure 3.4 shows that variables A_1 and A_2 are causally dependent. If *any* two variables A_1 and A_2 are truly causally related, then a change of A_1 causes a change of A_2. This chapter focuses on the discovery of the causality with no time delay, which means that attributes are in same time period or interval. This implies that attributes should occur within the same time stamp.

Theorem 3.4 *The support for a strong rule is less than or equal to the half of the total time stamps.*

Proof: *Each dynamic attribute can have only one type of polarity at one time stamp. If the occurrence of one polarity is huge, the occurrence of the other two polarities will diminish. Following the definition of strong rules, the occurrence of the polarities + and – have both to be highly frequent. Since support indicates the times each polarity pair occurred over the total time stamp, the support cannot be more than ½ of the total time stamps.*

A *sympathetic* rule causes an increase or decrease in the output of a target system. It reinforces a change with more change in the same direction. An *antipathetic* rule represents an adjustment to achieve a certain goal or objective. It indicates a system attempting to change from its current state to a goal state. This implies that if the current state is above the goal state, then the system forces it down. If the current state is below the goal state, the system pushes it up. An *antipathetic* rule provides useful stability but resists external changes.

3.6 Dynamic policy

Definition 3.10 *A dynamic policy consists of one or more dynamic causal rules. In a dynamic policy with several dynamic causal rules, each rule should share at least one dynamic attribute with other rules.*

Definition 3.11 *A dynamic policy is single if it consists only of one rule and if that rule does not share any common attribute with other rules in a dynamic policy.*

Figure 3.5 illustrates the possible states of a dynamic policy. Given a *single dynamic policy* where an attribute A_1 is dynamically causally related to another attribute A_2, then such a rule can be represented as the following function.

$$\Delta A_2 = k\Delta A_1 \tag{3.8}$$

where

$$k_t = \frac{\Delta A_{2,t}}{\Delta A_{1,t}} \tag{3.9}$$

Given A_1^{new} indicating new values added in A_1, a ΔA_1^{new} can be calculated based on A_1^{new}, and based on equation 3.8 a ΔA_2^{new} can be calculated. If given a value $A_{2,t=0}$ *a new set of* A_2 can be derived, where $A_{2,t=2} = A_{2,t=1} + \Delta A^{new}_{2,t=1}$, $A_{2,t=3} = A_{2,t=2} + \Delta A^{new}_{2,t=2}$, *etc.*

Definition 3.12 *A dynamic policy is serial if each rule shares only one attribute with one other rule in a dynamic policy. A serial dynamic policy is open if each attribute in the policy has only one causal link, in other words, if there is a start and an ending attribute. A serial dynamic policy is closed if there is no start or ending attribute.*

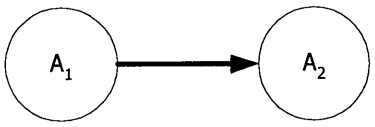
Given an *open serial dynamic policy* where an attribute A_1 is dynamically causally related to another attribute A_2, and A_2 is dynamically causally related to another attribute A_3, then such a rule can be represented as the following function.

$$A_{2,T} = k_{1,T} A_{1,T} \text{ and } A_{3,T} = k_{2,T} A_{2,T} = k_{2,T}\, k_{1,T} A_{1,T} \tag{3.10}$$

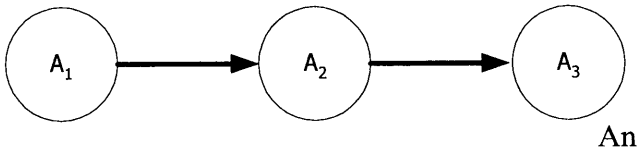
where

$$\text{T} = (1, 2, 3, \ldots.\text{t}),\ k_{1,t} = \frac{A_{2,t}}{A_{1,t}} \text{ and } k_{2,t} = \frac{A_{3,t}}{A_{2,t}} \tag{3.11}$$

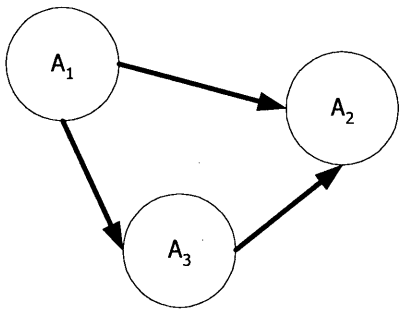
Definition 3.13 *A dynamic policy is complex if the policy consists of a mixture of open and closed serial dynamic policies.*



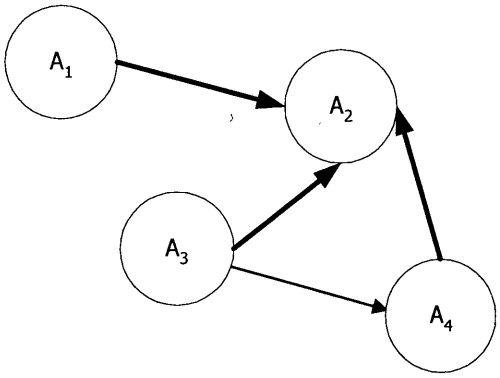
A singe dynamic policy



An open serial dynamic policy



A closed serial dynamic policy



A complex dynamic policy

Figure 3.5 Dynamic policy

For a complex or closed dynamic policy there are attributes which are connected with more than one attribute. For instance, A_1 is connected with A_2, A_3, and A_4. A_2, A_3 and A_4 are not connected. Then A_1 can be represented with:

$$A_{2,T} * A_{3,T} * A_{4,T} = C_T \tag{3.12}$$

where

$$T = (1, 2, 3, \ldots t) \text{ and } C_t = A_{2,t} * A_{3,t} * A_{4,t}$$

3.7 Illustrative example

Stage 1: To establish a dynamic data set. The dynamic set is calculated based on the dataset illustrated in Table 3.5 and Equation 3.1. Table 3.6 shows the dynamic set after the subtraction.

Stage 2: To prune the dynamic data set based on the specified *support*. Pruning is carried out to remove columns (attributes) where the level of support is below the minimum set. In this example, the *support* is set to 2, which means columns with two or more 0s are removed (value 0 indicates no dynamics in the attribute at the corresponding time stamp and more occasionally the set value of 0 indicates the attribute is not dynamic). Table 3.7 illustrates the 'pruned' dynamic data set.

Stage 3: To create dynamic rules. The dynamic *supports* are counted and calculated according to Equations (3.3), (3.4), and (3.5). Table 3.8 shows the dynamic supports for the pairs of attributes

in Table 3.7. In this example, the user-specified support is set to 0.3, which means any attribute pair with dynamic support with value larger than or equal to 0.3 is a dynamic rule. Table 3.9 shows how the *DCM* rules are generated. Note that (F1&F7) is the only strong sympathetic rule because when one of the attributes increases its value, the other will automatically increase its value and vice versa.

Table 3.10 shows the attribute pair A_1 and A_7 and dynamic attributes ΔA_1 and ΔA_7.The strokes indicate redundant attributes and redundant *dynamic attributes.* Given a new attribute $A_{1,new}$ and $A_{2,new}$ as shown in Tables 3.11 and 3.12, from it $\Delta A_{1,new}$ and $\Delta A_{2,new}$ can be calculated as shown in Table 3.13. Figure 3.6 shows the plots of $A_{1,new}$ and $A_{2,new}$ and it is clear that the two plots are causally related.

3.8 Mining algorithm

3.8.1 Problem formulation

Let D denote a database which contains a set of n records with attributes $\{A_1, A_2, A_3, \ldots A_n.\}$, where each attribute is of a unique type (sale price, production quantity, inventory volume, etc). Each attribute is linked to a time stamp t. To apply *DCM*, the records are arranged in a temporal sequence ($t = 1, 2, \ldots, n$). The causality between attributes in D can be identified by examining

Time	A_1	A_2	A_3	A_4	A_5	A_6	A_7
1	9	2	10	22	20	100	13
2	17	3	10	28	20	100	13
3	10	12	10	27	20	100	6
4	4	16	10	22	20	100	2
5	7	24	10	14	20	3	10
6	6	18	10	11	20	100	5
7	6	18	10	5	20	100	4
8	11	21	10	5	20	100	9
9	20	13	10	1	20	100	2
10	21	8	10	8	5	100	8

Table 3.5 Input dataset

ΔA_1	ΔA_2	ΔA_3	ΔA_4	ΔA_5	ΔA_6	ΔA_7
+8	+1	0	+6	0	0	0
-7	+9	0	-1	0	0	-7
-6	+4	0	-5	0	0	-4
+3	+8	0	-8	0	-97	+8
-1	-6	0	-3	0	97	-5
0	0	0	-6	0	0	-1
+5	+3	0	0	0	0	+5
+9	-8	0	-4	0	0	-7
+1	-5	0	+7	-15	0	+6

Table 3.6 Dynamic dataset

ΔA_1	ΔA_2	ΔA_4	ΔA_7
+8	+1	+6	0
-7	+9	-1	-7
-6	+4	-5	-4
+3	+8	-8	+8
-1	-6	-3	-5
0	0	-6	-1
+5	+3	0	+5
+9	-9	-4	-7
+1	-5	+7	+6

Table 3.7 Pruned dataset

	(+,+)	(-,-)	(-,+)	(+,-)
F1&F2	0.3	0.1	0.2	0.2
F1&F4	0.2	0.3	0	0.2
F1&F7	0.3	0.3	0	0.1
F2&F4	0.1	0.2	0.1	0.3
F2&F7	0.2	0.2	0.1	0.2
F4&F7	0.1	0.5	0.1	0

Table 3.8 Result

(+,+)	(F1&F2), (F1&F7)
(-,-)	(F1&F4), (F1&F7), (F4&F7)
(+,-)	(F2&F4)

Table 3.9 The rules generated

A_1	A_7
9	13
17	13
10	6
4	2
7	10
6	5
6	4
11	9
20	2
21	8

ΔA_1	ΔA_7
+8	0
-7	-7
-6	-4
+3	+8
-1	-5
0	-1
+5	+5
+9	-7
+1	+6

Table 3.10 The attribute pair A_1 and A_7, and dynamic attribute ΔA_1 and ΔA_7

$A_{1,new}$
2
5
4
7
6
12
5

Table 3.11 The new attribute $A_{1,new}$

$\Delta A_{1,new}$	$\Delta A_{2,new}$
3	3
-1	-0.66667
3	8
-1	-5
6	6
-7	-42

Table 3.12 The new attribute pair $\Delta A_{1,new}$ and $\Delta A_{2,new}$

$A_{2,new}$
12
15
14.33333
22.33333
17.33333
23.33333
-18.6667

Table 3.13 The new dynamic attributes $A_{2,new}$

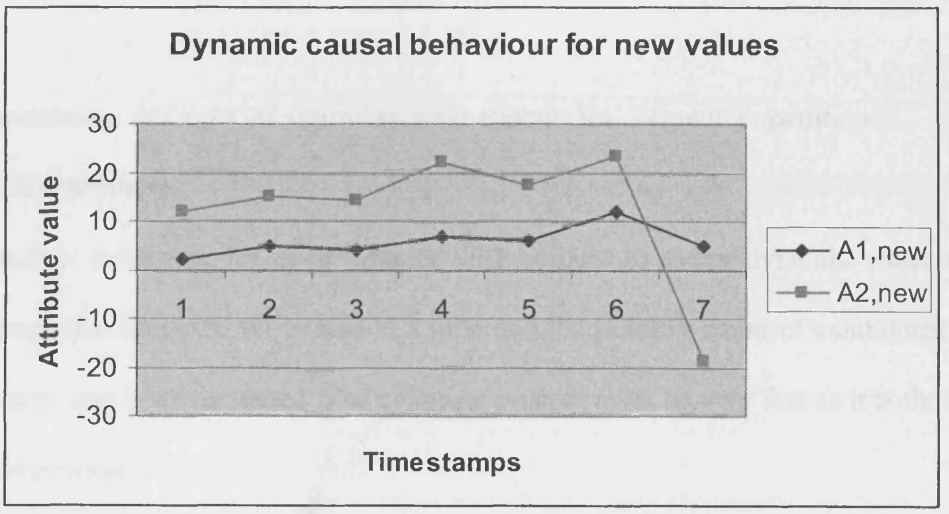


Figure 3.6 Plot for $A_{1,new}$ and $A_{2,new}$

the polarities of corresponding changes in attribute values. Let D_{new} be a new data set constructed from D. A generalized dynamic association rule is an implication of the form $A_1 \rightarrow^p A_2$, where $A_1 \subset D$, $A_2 \subset D$, $A_1 \cap A_2 = \phi$ and p is the polarity.

The implementation of the *DCM* algorithm must support the following operations:

(1) To add new attributes.

(2) To maintain a counter for each polarity with respect to every dynamic value set. While making a pass, one dynamic set is read at a time and the polarity count of candidates supported by the dynamic sets is incremented. The counting process must be very fast as it is the bottleneck of the whole process.

3.8.2 Algorithm description

DCM makes two passes over the data as shown in Figures 3.7 and 3.8. In the first pass, the *support* of individual attributes is counted and the frequent attributes are determined. The dynamic values are used for generating new potentially frequent sets and the actual *support* of these sets is counted during the pass over the data. In subsequent passes, the algorithm initializes with dynamic value sets based on dynamic values found to be frequent in the previous pass. After the second of the passes, the *causal rules* are determined and they become the candidates for the dynamic policy. In the *DCM* process, the main goal is to find the strong dynamic causal rule in order to form a policy. It also represents a filtering process that prunes away static attributes, which reduces the size of the data set for further mining.

Part 1: – Preprocessing: Removal of the "least" causal data from database

Part 2: – Mining: Formation of a rule set that covers all training examples with minimum number of rules

Part 3: – Checking: Check if an attribute pair is self contradicting (sympathetic and antipathetic at the same time)

Input: The original database, the values of the pruning threshold for the neutral, sympathetic and antipathetic *supports*.

Output: Dynamic sets

Step 1: Check the nature of the attributes in the original database (numerical or categorical). Initialize a new database with dynamic attributes based on the attributes and time stamps from original database.

Step 2: Initialize a counter for each of the three polarities.

Figure 3.7 The steps of *DCM*

Input: The mined database, the values of the pruning threshold for the supports of the polarity combinations.

Output: Dynamic sets

Step 1. Check weather a rule is self-contradictory (a rule is both sympathetic and antipathetic).

Step 2. If step 1 returns true then

Retrieve the attribute pair form the preprocessed database

Step 3. Initialize a counter that includes polarity combination

Step 4. For the pair of attributes

Count the occurrence of polarity combination with two records each time.

Prune away the pairs if the counted support is below the input threshold.

Figure 3.8 The checking step of *DCM*

The process time would be n^2-n, where n is the number of attributes. This becomes a huge problem if n becomes too large. It is obvious that the task becomes much simpler if the size of n could be reduced before the search. Table 3.14 shows the pruned percentage of the total database based on the support. The support level is set to 0.

3.9 Experiment

3.9.1 Data preparation

The overall aim is to identify hidden dynamic changes. The original data was given as shown in Table 3.15. The only data of interest are the data with changes, for example sale amounts of a product, the time stamp, etc, The rest of the static data, such as the weight and the cost of the product can be removed.

After cleaning the data, the *dynamic attributes* are found as shown in Table 3.16. The *dynamic attribute* is calculated by finding the difference between sales amounts in one month and sales amounts in the previous month. In the next step, the neutral attributes are pruned. The idea of pruning is to remove redundant dynamic attributes; thus fewer sets of attributes are required when generating rules. The first pruning is based on the single attribute *support*. In this case, the single attribute *support* is defined to be 0.5, which means that if an attribute with polarity +, -, or 0 occurs in more than half of total time stamps, it will be pruned. In this case, 429 attributes remain for the rule generation.

In this experiment, dynamic sets are compared based on a simultaneous *time stamp*. Then the support of *sympathetic* and *antipathetic* rules for each dynamic set is calculated. The *support* is used as the threshold to eliminate unsatisfactory dynamic sets and to obtain the rules from the satisfactory sets.

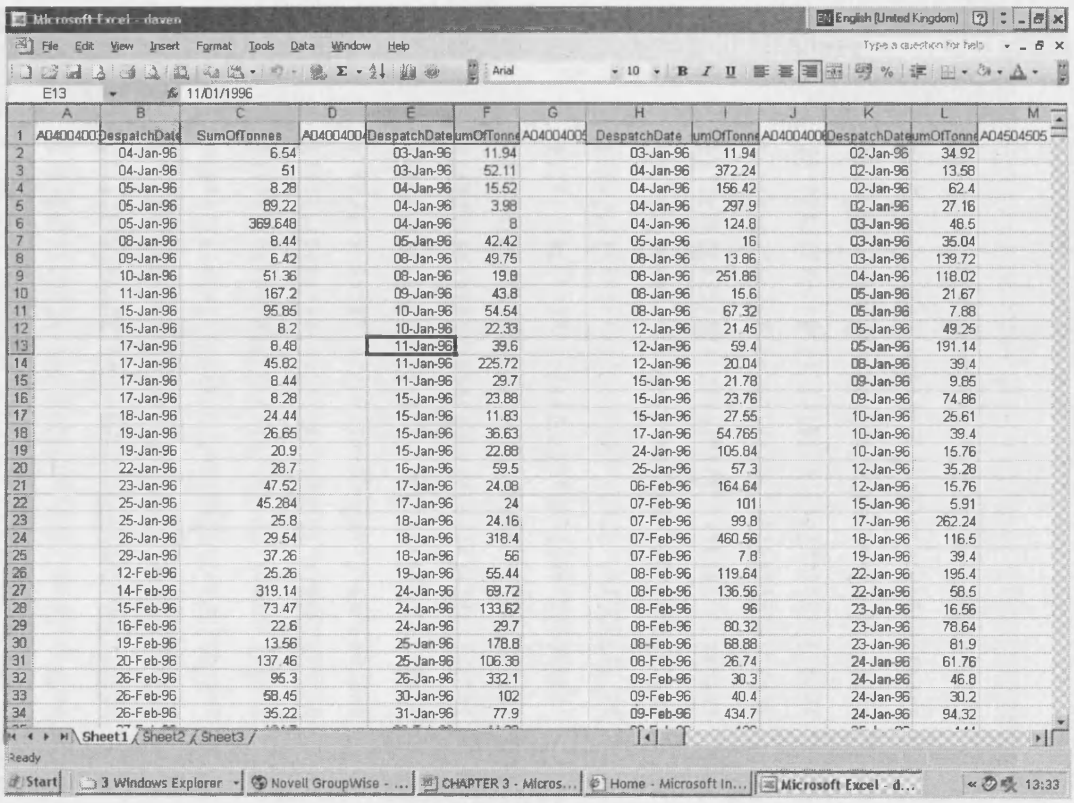
3.9.2 Evaluation and results

The algorithm was run based on the procedures described in previous sections. Figure 3.9 shows the plot of sympathetic and antipathetic support. The x –axis represents the support and the y-axis represents the number of rules. This database shows that there are more sympathetic rules than antipathetic rules. Figure 3.9 also shows that increasing support will lead to exponential growth of the rules. As the support reaches 0.05 or 5, as it indicates on the Figure 3.9, the number of rules is 630. Most of these rules are redundant and have no meaning due to the low support. Figure 3.10 shows the rule plot with support equal to average value, where the +support = the average of all positive records and –support = the average of all negative records. The number of rules has decreased to half by applying the support level.

Table 3.17 shows the extracted strong rules with support level equal to average value and support larger than 0.08. There are only dynamic pairs so there is no need to do the simulation. The items in Table 3.17 (C15276179, F030008....) represent different types of product.

Data set	Single Support						
	0.05	0.10	0.15	0.20	0.25	0.30	0.35
Adult	5%	20%	27%	74%	100%	100%	100%
Bank	11%	20%	60%	94%	100%	100%	100%
Cystine	5%	33%	70%	100%	100%	100%	100%
Market basket	6%	10%	50%	86%	100%	100%	100%
Mclosom	1%	13%	38%	72%	90%	100%	100%
ASW	1%	8%	40%	68%	70%	97%	100%
Weka-base	6%	25%	52%	86%	100%	100%	100%

Table 3.14 Pruned results



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	A0400400	DespatchDate	SumOfTonnes	A0400400	DespatchDate	umOfTonne	A0400400	DespatchDate	umOfTonne	A0400400	DespatchDate	umOfTonne	A0450450
2		04-Jan-96	6.54		03-Jan-96	11.94		03-Jan-96	11.94		02-Jan-96	34.92	
3		04-Jan-96	51		03-Jan-96	52.11		04-Jan-96	372.24		02-Jan-96	13.58	
4		05-Jan-96	8.28		04-Jan-96	15.52		04-Jan-96	156.42		02-Jan-96	62.4	
5		05-Jan-96	89.22		04-Jan-96	3.98		04-Jan-96	297.9		02-Jan-96	27.16	
6		05-Jan-96	369.648		04-Jan-96	8		04-Jan-96	124.8		03-Jan-96	46.5	
7		08-Jan-96	8.44		05-Jan-96	42.42		05-Jan-96	16		03-Jan-96	35.04	
8		09-Jan-96	6.42		08-Jan-96	49.75		08-Jan-96	13.86		03-Jan-96	139.72	
9		10-Jan-96	51.36		08-Jan-96	19.8		08-Jan-96	251.86		04-Jan-96	118.02	
10		11-Jan-96	167.2		09-Jan-96	43.8		08-Jan-96	15.6		05-Jan-96	21.67	
11		15-Jan-96	95.85		10-Jan-96	54.54		08-Jan-96	67.32		05-Jan-96	7.88	
12		15-Jan-96	8.2		10-Jan-96	22.33		12-Jan-96	21.45		05-Jan-96	49.25	
13		17-Jan-96	8.48		11-Jan-96	39.6		12-Jan-96	59.4		05-Jan-96	191.14	
14		17-Jan-96	45.82		11-Jan-96	225.72		12-Jan-96	20.04		08-Jan-96	39.4	
15		17-Jan-96	8.44		11-Jan-96	29.7		15-Jan-96	21.78		09-Jan-96	9.85	
16		17-Jan-96	8.28		15-Jan-96	23.88		15-Jan-96	23.76		09-Jan-96	74.86	
17		18-Jan-96	24.44		15-Jan-96	11.83		15-Jan-96	27.55		10-Jan-96	25.61	
18		19-Jan-96	26.65		15-Jan-96	36.63		17-Jan-96	54.765		10-Jan-96	39.4	
19		19-Jan-96	20.9		15-Jan-96	22.88		24-Jan-96	105.84		10-Jan-96	15.76	
20		22-Jan-96	28.7		16-Jan-96	59.5		25-Jan-96	57.3		12-Jan-96	35.28	
21		23-Jan-96	47.52		17-Jan-96	24.08		06-Feb-96	164.64		12-Jan-96	15.76	
22		25-Jan-96	45.284		17-Jan-96	24		07-Feb-96	101		15-Jan-96	5.91	
23		25-Jan-96	25.8		18-Jan-96	24.16		07-Feb-96	99.8		17-Jan-96	262.24	
24		26-Jan-96	29.54		18-Jan-96	318.4		07-Feb-96	460.56		18-Jan-96	116.5	
25		29-Jan-96	37.26		18-Jan-96	56		07-Feb-96	7.8		19-Jan-96	39.4	
26		12-Feb-96	25.26		19-Jan-96	55.44		08-Feb-96	119.64		22-Jan-96	195.4	
27		14-Feb-96	319.14		24-Jan-96	69.72		08-Feb-96	136.56		22-Jan-96	58.5	
28		15-Feb-96	73.47		24-Jan-96	133.62		08-Feb-96	96		23-Jan-96	16.56	
29		16-Feb-96	22.6		24-Jan-96	29.7		08-Feb-96	80.32		23-Jan-96	78.64	
30		19-Feb-96	13.56		25-Jan-96	178.8		08-Feb-96	68.88		23-Jan-96	81.9	
31		20-Feb-96	137.46		25-Jan-96	106.38		08-Feb-96	26.74		24-Jan-96	61.76	
32		26-Feb-96	95.3		26-Jan-96	332.1		09-Feb-96	30.3		24-Jan-96	46.8	
33		26-Feb-96	58.45		30-Jan-96	102		09-Feb-96	40.4		24-Jan-96	30.2	
34		26-Feb-96	36.22		31-Jan-96	77.9		09-Feb-96	434.7		24-Jan-96	94.32	

Table 3.15 Original data sets

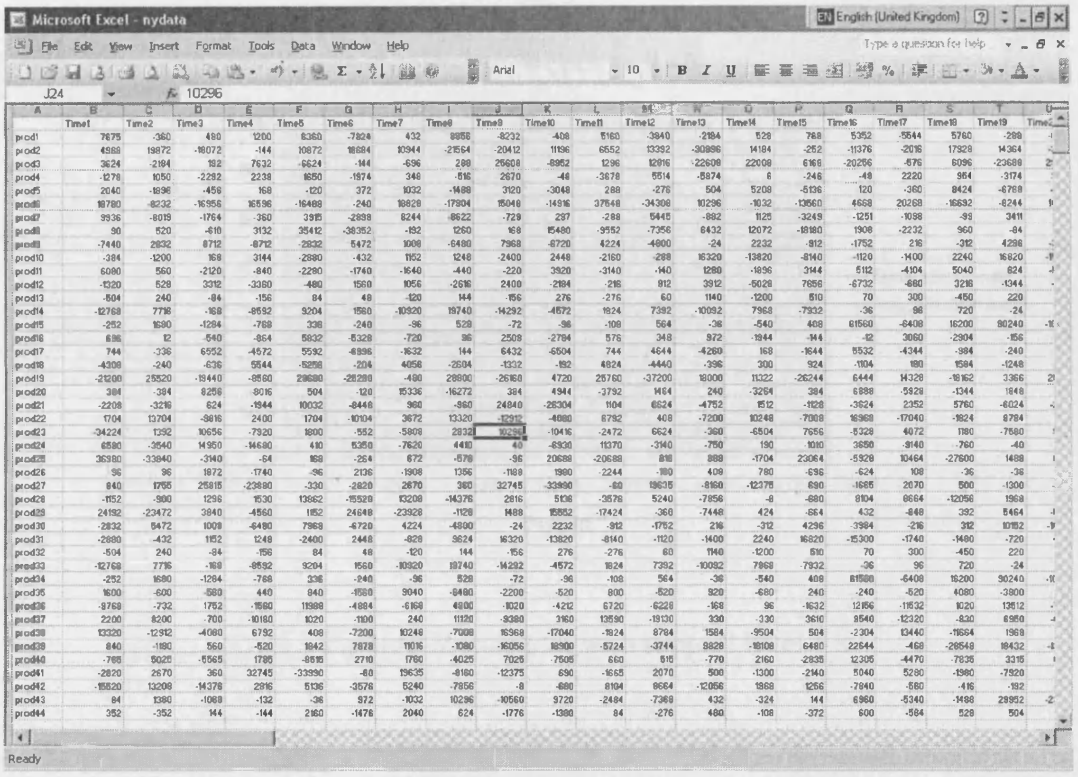


Table 3.16 Dynamic attributes

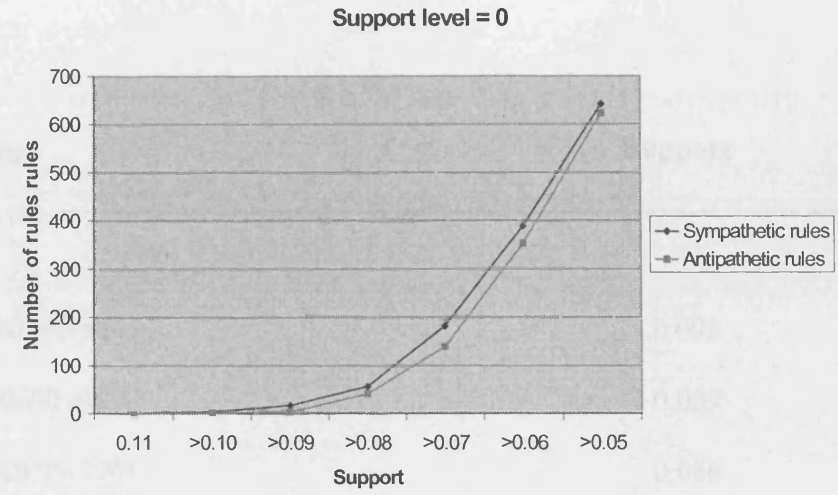


Figure 3.9 Rule plot with support level = 0

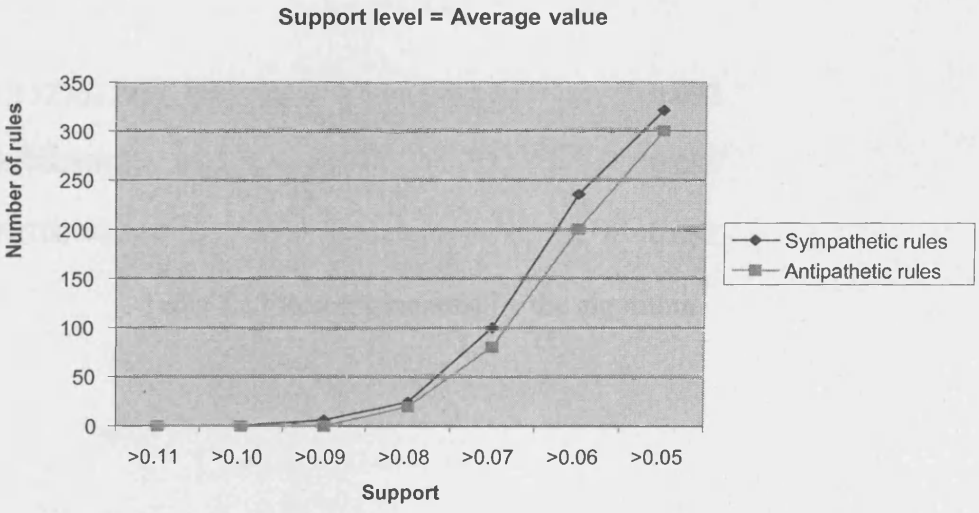


Figure 3.10 Rule plot with support level = Average value

Strong rules	**Support**
Sympathetic	
{C15276179, F030008}	0.093
{J08008008, F060010}	0.089
{A04004004, A05005005}	0.086
{A05005006, C10251104}	0.084
{A04004004, F100020}	0.082
Antipathetic	
{A05005008, C15276179}	0.092
{C10251104, F070010}	0.083
{A05005008, F030008}	0.082

Table 3.17 Result generated by the algorithm

3.10 Discussion

Apriori provides some form of causal information, i.e. suggesting a possible direction of causation between two attributes, but there is no basis to conclude that the arrow indicates direct or even indirect causation. The *DCM* algorithm, on the other hand, shows causality between attributes. Thus, where association rule generation techniques find surface associations, causal inference algorithms identify the structure underlying such associations.

Each type of relationship generated by the *DCM* algorithm provides additional information. The *DCM* algorithm finds four kinds of relationships, each of which deepens the user's understanding of their target system by constructing the possible models. For example, $A_1 \rightarrow^{+} A_2$ provides more information than $A_1 \rightarrow A_2$ because the latter indicates that A_1 coexists with A_2. The condition of the rule is not stated (whether sympathetic or antipathetic). A genuine causality such as $A_1 \rightarrow^{+} A_2$ provides useful information because it indicates that the relationship from A_1 to A_2 is strictly sympathetic causal.

The *rules* extracted by *DCM* can be simulated by using software to model the future behaviour. The *rules* extracted by association algorithm cannot be simulated.

3.11 Summary

The *DCM* algorithm presented in this chapter enables the generation of *causal dynamic rules* from data sets by integrating concepts of *Systems Thinking* and *System Dynamics* with

Association Mining. The algorithm can process data sets with both categorical and numerical attributes. Compared with other *Association Mining* algorithms, *DCM* rule sets are smaller and more dynamically focused. Rule pruning is carried out based on polarities. This reduces the size of the pruned data set and still maintains the accuracy of the generated rule sets. The rules extracted can be joined to create dynamic policy, which can be simulated through software for future decision making.

DCM provides information that can be used for prediction and indicates the structure of the relationship underlying that rule. The measures of *support level* and *support* are simple thresholds for evaluating a rule. The task of determining whether the rules generated are meaningful is left to the analysts.

The algorithm presented in this chapter investigates only events happening in the same time widow. There is no clear identification of feedback or delay, which is essential in dynamic analysis. A new algorithm, dealing with data with different time windows, is considered in the next chapter.

Chapter 4 Improvements of *DCM*

4.1 Preliminaries

Dynamic Causal Mining (DCM) assists decision-makers to control a system at decision points by converting information into policy. *DCM* searches for simultaneous *dynamic causal* relations in a database. However, it ignores delays and feedback information which play an important role in any dynamic system.

This chapter expands the *DCM* approach to discover delay and feedback relationships between attributes based on separate time stamps. This makes the algorithm more suitable for dynamic modelling and enables the discovery of hidden dynamic structures, which can be applied to predict the future behaviour of a dynamic system.

This chapter also suggests relaxing the strict separation among polarity +, polarity -, and *neutral,* and using more flexible linguistic terms like "High increase", "Increase", "High decrease", "Decrease", and "Neutral". Fuzzy sets can provide a reasonable representation using cognitive concepts in terms of natural languages. These linguistic terms use graded statements rather than ones that are strictly true or false, and thus provide an approximate but effective way to describe the dynamic causal behaviour of systems (Zadeh, 1975).

The new approach imposes problems such as accuracy and efficiency and this chapter further suggests using fuzzy sets to solve these problems. Compared to quantitative rules, fuzzy rules correspond better to sharp boundaries between neighbour sets. In

most real life applications, databases contain many other attribute values other than 0 and 1. Quantitative attributes such as production volume and income take values from an ordinal scale. One way of dealing with a quantitative attribute is to divide the range of the original attributes into partitions, such as low, medium, and high. It is more intuitive to allow attribute values to vary from the interval [0, 1] (instead of just 0 or 1), indicating the degree of belonging. Thus attributes are no longer binary but fuzzy.

Fuzzy sets are sets with boundaries that are not precise and membership in this fuzzy set is not a binary statement but rather a matter of degree. This approach obtains not only a more human-understandable knowledge from the database but also provides more compact and robust representations. The use of fuzzy partitions of the domains of quantitative attributes can avoid some undesirable threshold effects which are usually produced by crisp (non-fuzzy) partitions.

4.2 Delay and feedback

Delay and feedback play central roles in many processes and systems (Forrester, 1968; Coyle, 1977 and Mohapatra, 1994). It takes time to create a product, manage a service, execute an operation or build a facility. Delays are important parts of any system. They represent the time between a change occurring in one part of the system and the cause of change in the other part.

Definition 4.1 *A delay is the time difference between two dynamic causal events occurring in different parts of the same system. In the scope of the thesis, the dynamic causal event is represented by the dynamic causal attribute and the system characteristic is represented by the database.*

Delays play important roles in deciding the dynamic causal behaviour of systems. The modelling of systems therefore necessitates that delays are properly represented in order that real life behaviour can be replicated.

A delayed *dynamic causal* relationship between two attributes implies a change of attribute values as shown in Figure 4.1. A dynamic attribute A_1 at a time point t_1 causes a change to attribute A_2 at a time point t_2.

Definition 4.2 *A feedback is a counter effect from another source to the original source of the effect.*

Feedback deals with the control and determination of deviations from a desired state and executes corrective action regarding these deviations. Feedback refers to the method of controlling a system by reinserting the results from its past performance.

A feedback relation means that the change in A_2 at t_2, due to A_1 at t_1 would in turn cause the value of A_1 to alter at t_3.

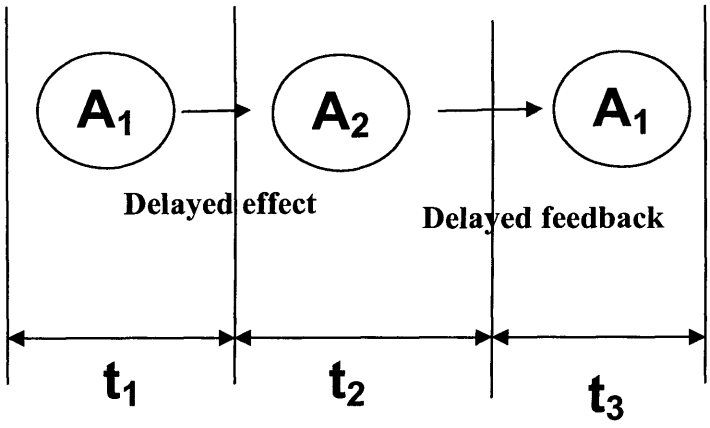


Figure 4.1 Delayed dynamic *causal* relation

4.3. Measures

Let D denote a database, which contains a set of n records with attributes $\{A_1, A_2, A_3, \ldots, A_m\}$, where each attribute is of a unique type (e.g. sale price, production volume, inventory volume, etc). Each attribute is associated with a time stamp t. The records are arranged in a temporal sequence $(t_1, t_2, \ldots, t_n)$. Table 4.1 is an example of such a database.

Since the sequence $t_1, t_2, \ldots, t_n$ is arranged in ascending numerical order, Δt_i always represents an increase. Table 4.2 illustrates the new database D_{new} derived from Table 4.1. Tables 3.1 and 3.2 are identical to Tables 4.1 and 4.2.

Definition 4.3 (modified from definition 3.6): *Dynamic support is defined as the ratio of the number of records of a given polarity combination to the total number of records in the database based on the respective time stamps.*

Five measures are used to identify *dynamic causal* rules (or relationships). These are *fully sympathetic support, fully antipathetic support, self-sympathetic support, self-antipathetic support,* and *neutral support.* For database D_{new} and any two attributes ΔA_1 and ΔA_2, the different types of *support* are defined as:

Fully sympathetic support $(\Delta A_1, \Delta A_2, \Delta A_1) =$

$$\frac{freq \quad (+_{\Delta t_i}, +_{\Delta t_{i+1}}, +_{\Delta t_{i+2}})}{n} \tag{4.1.a}$$

$$\frac{freq \quad (-_{\Delta t_i}, -_{\Delta t_{i+1}}, -_{\Delta t_{i+2}})}{n} \tag{4.1.b}$$

Time	A_1	A_2
1	9	2
2	17	3
3	10	12
4	4	16
5	7	24

Table 4.1 Original database D

Δt	ΔA_1	ΔA_2
Δt_1	+8	+1
Δt_2	-7	+9
Δt_3	-6	+4
Δt_4	+3	+8

Table 4.2 Derived database D_{new}

Fully antipathetic support $(\Delta A_1, \Delta A_2, \Delta A_1) =$

$$\frac{freq \quad (+_{\Delta t_i}, -_{\Delta t_{i+1}}, -_{\Delta t_{i+2}})}{n} \tag{4.2.a}$$

$$\frac{freq \quad (-_{\Delta t_i}, +_{\Delta t_{i+1}}, +_{\Delta t_{i+2}})}{n} \tag{4.2.b}$$

Self-sympathetic support $(\Delta A_1, \Delta A_2, \Delta A_1) =$

$$\frac{freq \quad (+_{\Delta t_i}, -_{\Delta t_{i+1}}, +_{\Delta t_{i+2}})}{n} \tag{4.3.a}$$

$$\frac{freq \quad (-_{\Delta t_i}, +_{\Delta t_{i+1}}, -_{\Delta t_{i+2}})}{n} \tag{4.3.b}$$

Self-antipathetic support $(\Delta A_1, \Delta A_2, \Delta A_1) =$

$$\frac{freq \quad (+_{\Delta t_i}, +_{\Delta t_{i+1}}, -_{\Delta t_{i+2}})}{n} \tag{4.4.a}$$

$$\frac{freq \quad (-_{\Delta t_i}, -_{\Delta t_{i+1}}, +_{\Delta t_{i+2}})}{n} \tag{4.4.b}$$

Neutral support $(\Delta A_m) = \frac{freq \quad (0)}{n}$ (4.5)

Single Support $(\Delta A_m) = \frac{freq \quad (+)}{n}$ (4.6)

Single Support $(\Delta A_m) = \frac{freq \quad (-)}{n}$ (4.7)

where n is the total number of time stamps and m identifies the attribute of interest. $freq\ (+_{\Delta t_i}, +_{\Delta t_{i+1}}, +_{\Delta t_{i+2}})$ is a function of the number of times where an increase in ΔA_1 is followed by an increase in ΔA_2 which induces another increase in ΔA_1 with

respect to the time stamps Δt_i, Δt_{i+1} and Δt_{i+2}. Similarly, *freq* $(-_{\Delta t_i}, -_{\Delta t_{i+1}}, -_{\Delta t_{i+2}})$ is a function of the number of times where a decrease in ΔA_1 is followed by a decrease in ΔA_2 which induces another decrease in ΔA_1 with respect to the time stamps Δt_i, Δt_{i+1} and Δt_{i+2}. The *neutral support* indicates the frequency of value 0 in a derived attribute. The *single support* indicates the frequency of value + or - in a derived attribute. All three *supports* are used to prune ineffectual attributes.

Table 4.3 shows the derived database D_{new} with arrows indicating the direction in which *supports* are counted. In this example, the *neutral support* is 0 since there is no record of value 0 in ΔA_1 and ΔA_2. The other *supports* are counted by following the direction of the arrows. A left-to-right arrow indicates the causal relation $\Delta A_{1,\Delta t_i} \rightarrow \Delta A_{2,\Delta t_{i+1}}$ and a right-to-left arrow indicates $\Delta A_{2,\Delta t_{i+1}} \rightarrow \Delta A_{1,\Delta t_{i+2}}$. The result is shown in Table 4.4.

Dynamic causal rules are used to predict future dynamic behaviour. Each causal rule is assigned a polarity combination with a time stamp. Each polarity is assigned a time stamp. According to equations (4.1), (4.2), (4.3), and (4.4) and as already seen in Table 4.4, there are eight polarity combinations of interest, namely, $(+_{\Delta t_i}, +_{\Delta t_{i+1}}, +_{\Delta t_{i+2}})$, $(-_{\Delta t_i}, -_{\Delta t_{i+1}}, -_{\Delta t_{i+2}})$, $(-_{\Delta t_i}, +_{\Delta t_{i+1}}, +_{\Delta t_{i+2}})$, $(+_{\Delta t_i}, -_{\Delta t_{i+1}}, -_{\Delta t_{i+2}})$, $(+_{\Delta t_i}, -_{\Delta t_{i+1}}, +_{\Delta t_{i+2}})$, $(-_{\Delta t_i}, +_{\Delta t_{i+1}}, -_{\Delta t_{i+2}})$, $(+_{\Delta t_i}, +_{\Delta t_{i+1}}, -_{\Delta t_{i+2}})$, and $(-_{\Delta t_i}, -_{\Delta t_{i+1}}, +_{\Delta t_{i+2}})$. This polarity representation differs from that used in classical causal rules (either + or -) which is too simple to model dynamic behaviours in real world systems.

Δt	ΔA_1	ΔA_2
Δt_1	+8	+1
Δt_2	-7	+9
Δt_3	-6	+4
Δt_4	+3	+8

Table 4.3 Derived database D_{new} with arrows indicating support counting direction

	(+,+,+)	(-,-,-)	(+,-,-)	(-,+,+)
Supports($\Delta A_1, \Delta A_2, \Delta A_1$)	0	0	0	1/4
	(+,-,+)	(-,+,-)	(+,+,-)	(-,-,+)
Supports($\Delta A_1, \Delta A_2, \Delta A_1$)	0	0	1/4	0

Table 4.4 Counting result

4.4 An illustrative example

Table 4.5 shows a database where the first column indicates the time instant which could be hours, weeks, or years. The numbers in the columns have the same units. They could, for example, be purchase prices or sales levels etc. The first row of Table 4.5 can therefore be interpreted as in week 1, company 1 decides to produce 9 units; company 2 to make 2 units etc.

Dynamic Causal Mining is to be applied to this data to derive any *dynamic causal* relationships between these production volumes in order to assist a company in deciding its future manufacturing strategy. Table 4.6 shows the database derived after the difference calculation using equation (3.1) from chapter 3.

Table 4.7 illustrates the 'pruned' database. Pruning is carried out to remove columns (attributes) where the level of *neutral support* is below a set minimum. In this example, columns with seven or more zeros (meaning with seven or more records with neutral polarities) are removed.

In general, when the number of zeros in a column is high with respect to the total number of entries, the corresponding attributes can be regarded as unaffected by attributes represented in the other columns. Even if a few of the remaining non-zero entries are large in magnitude, their effect on the *sympathetic/antipathetic support* counts will be small. Table 4.8 shows the *supports* for the attributes in Table 4.7 taken in pairs. The *supports* are calculated according to equations (4.1), (4.2), (4.3), (4.4), and 4.5.

Time	A_1	A_2	A_3	A_4	A_5	A_6	A_7
1	9	2	10	22	20	100	13
2	17	3	10	28	20	100	13
3	10	12	10	27	20	100	6
4	4	16	10	22	20	100	2
5	7	24	10	14	20	3	10
6	6	18	10	11	20	100	5
7	6	18	10	5	20	100	4
8	11	21	10	5	20	100	9
9	20	13	10	1	20	100	2
10	21	8	10	8	5	100	8

Table 4.5 Original database

ΔA_1	ΔA_2	ΔA_3	ΔA_4	ΔA_5	ΔA_6	ΔA_7
+8	+1	0	+6	0	0	0
-7	+9	0	-1	0	0	-7
-6	+4	0	-5	0	0	-4
+3	+8	0	-8	0	-97	+8
-1	-6	0	-3	0	97	-5
0	0	0	-6	0	0	-1
+5	+3	0	0	0	0	+5
+9	-8	0	-4	0	0	-7
+1	-5	0	+7	-15	0	+6

Table 4.6 Derived database

ΔA_1	ΔA_2	ΔA_4	ΔA_7
+8	+1	+6	0
-7	+9	-1	-7
-6	+4	-5	-4
+3	+8	-8	+8
-1	-6	-3	-5
0	0	-6	-1
+5	+3	0	+5
+9	-9	-4	-7
+1	-5	+7	+6

Table 4.7 Pruned database

	(+,+,+)	(-,-,-)	(-,+,+)	(+,-,-)
$\Delta A_1\&\Delta A_2$	0	0	0.1	0
$\Delta A_1\&\Delta A_4$	0	0.1	0	0.1
$\Delta A_1\&\Delta A_7$	0	0	0	0.2
$\Delta A_2\&\Delta A_4$	0	0	0	0.1
$\Delta A_2\&\Delta A_7$	0	0	0.1	0.1
$\Delta A_4\&\Delta A_7$	0	0.2	0	0
	(+,-,+)	(-,+,-)	(-,-,+)	(+,+,-)
$\Delta A_1\&\Delta A_2$	0.1	0.1	0	0.1
$\Delta A_1\&\Delta A_4$	0.1	0	0.2	0
$\Delta A_1\&\Delta A_7$	0.1	0.1	0.2	0
$\Delta A_2\&\Delta A_4$	0.2	0	0.1	0
$\Delta A_2\&\Delta A_7$	0.2	0	0.1	0.1
$\Delta A_4\&\Delta A_7$	0	0.2	0	0

Table 4.8 Counting results

Polarity	Rule candidates
(-,-,-)	($\Delta A_1\&\Delta A_4$), ($\Delta A_4\&\Delta A_7$)
(-,+,+)	($\Delta A_1\&\Delta A_2$), ($\Delta A_2\&\Delta A_7$)
(+,-,-)	($\Delta A_1\&\Delta A_4$), ($\Delta A_1\&\Delta A_7$), ($\Delta A_2\&\Delta A_4$), ($\Delta A_2\&\Delta A_7$)
(+,-,+)	($\Delta A_1\&\Delta A_2$), ($\Delta A_1\&\Delta A_4$), ($\Delta A_1\&\Delta A_7$), ($\Delta A_2\&\Delta A_4$), ($\Delta A_2\&\Delta A_7$)
(-,+,-)	($\Delta A_1\&\Delta A_2$), ($\Delta A_1\&\Delta A_7$), ($\Delta A_4\&\Delta A_7$)
(-,-,+)	($\Delta A_1\&\Delta A_4$), ($\Delta A_1\&\Delta A_7$), ($\Delta A_2\&\Delta A_4$) , ($\Delta A_2\&\Delta A_7$)
(+,+,-)	($\Delta A_1\&\Delta A_2$), ($\Delta A_2\&\Delta A_7$)

Table 4.9 Obtained results

Suppose that the *support* threshold is set to 0.1, which means any attribute pair with *support* larger than or equal to 0.1 is considered *dynamically causally* related. The obtained results are shown in Table 4.9. Thus, for the given database, the *strong self-sympathetic* rules are ($\Delta A_1 \& \Delta A_2$) and ($\Delta A_1 \& \Delta A_7$). The only *strong self-antipathetic* rule is ($\Delta A_2 \& \Delta A_7$). The only *strong fully antipathetic* rule is ($\Delta A_2 \& \Delta A_7$).

The derived rules reveal to decision-makers that changes in attribute A_1 will be reinforced and that changes in attribute A_2 will tend to be opposed. Such a finding would not have been possible without considering delayed and feedback relationships.

4.5 Sharp boundary problem in *DCM*

One of the possible reasons for applying the fuzzy concept in the rule is the lack of accuracy. For example, there are two rules: "If A_1 and A_2 then A_3" with *support* equal to 80%, and "If A_1 and A_2 then A_3" with *support* equal to 79.9%. If there is a user-specified threshold of 80%, then according to *DCM* the first rule is chosen. However, the second rule is as strong as the first rule. Therefore, there should be an equal choice of the two rules to find a correct result following either the first or the second rule.

This section suggests relaxing the strict separation between polarity +, polarity -, and *neutral,* and using more flexible fuzzy set terms like "High increase", "Increase", "High decrease", "Decrease", and "Neutral". Fuzzy sets can provide a reasonable representation using cognitive concepts in terms of natural language. These linguistic terms use graded statements rather than ones that are strictly true or false and thus provide an approximate but effective way to describe the dynamic causal behaviour of systems.

The aim is to use fuzzy sets in order to make a more smooth solution by allowing more values between + and – by applying a fuzzy membership function. The fuzzy membership function is a graphical representation of the magnitude of participation of each dynamic record. It associates a weighting with each of the dynamic records that are processed and defines the functional overlap between records.

In previous sections, a dynamic record of value 0.01 belongs only to set +, as shown in Figure 4.2. Obviously, this is an inaccurate definition. In Figure 4.3 the dynamic attribute is mapped by a membership function. It shows that 0.01 is 60% neutral and 30% increased. Sharp boundaries between intervals may lead to undesirable threshold effects. As a result, the elements located near the boundary will contribute to more than one interval such that some intervals may become interesting in this case. It is, however, not reasonable for an element near the boundaries to contribute the same as those located within an interval.

In previous work on fuzzy mining (Kuok, et al., 1998; Fu, et al., 1998 and Hong, et al.,2000), the required fuzzy sets and their corresponding membership functions must be provided by an expert. The quality of the results relies on the appropriateness of the fuzzy sets to the given data. However, the fuzzy sets and their corresponding membership functions provided by the experts may not always be suitable and it is unrealistic to assume that experts will always be available to provide all the necessities.

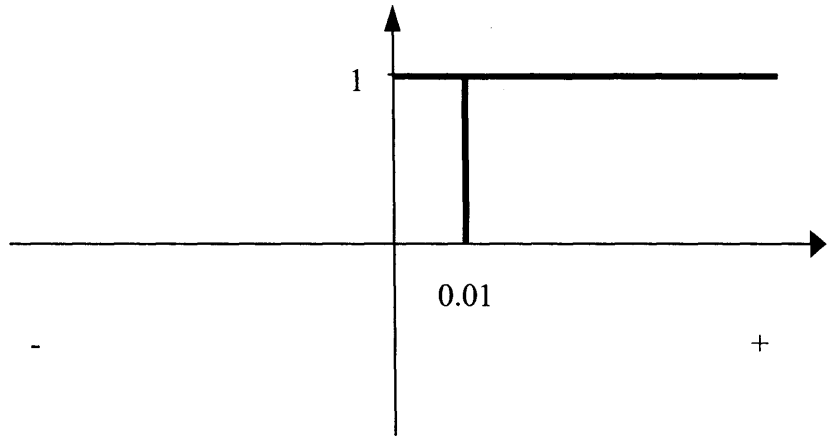


Figure 4.2 Crisp representation

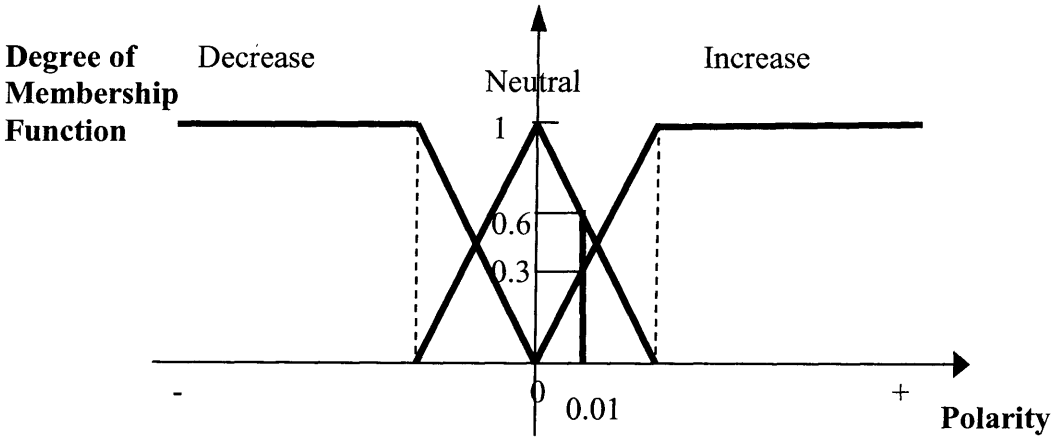


Figure 4.3 Fuzzy representation

The following section suggests a novel solution to the identification of a fuzzy membership function.

4.6 Fuzzy approach

This section gives details of how fuzzy logic is applied to *DCM*. The trapezoid membership function is used as an illustration as it is the simplest.

Definition 4.4 *A fuzzy dynamic attribute f is identified by a so-called membership function, which is a generalization of the characteristic function of a dynamic attribute* $\Delta A_{m,\Delta t_i} \subseteq D_{new}$. *For each record in* $\Delta A_{m,\Delta t_i}$, *a function specifies the degree of membership of* μ *that belongs to a linguistic term L. The membership degrees are taken from the unit interval [0, 1], i.e. a membership function is a mapping* $D_{new} \rightarrow [0, 1]$. $F(\Delta A_{m,\Delta t_i})$ *denotes the degree of membership of the dynamic attribute* $\Delta A_{m,\Delta t_i}$ *at time stamp i. It can be represented as;*

$$f = \mu\, L \qquad \text{where } \mu = m(\Delta A_{m,\Delta t_i}) \rightarrow [0, 1] \tag{4.8}$$

where $m(\Delta A_{m,\Delta t_i})$ indicates the membership function of $\Delta A_{m,\Delta t_i}$. Figure 4.4 shows a possible membership function used in *DCM*. Given an attribute $\Delta A_{m,\Delta t_i}$, F_1 = Min ($-\Delta A_{m,\Delta t_i}$), F_2 = Average($-\Delta A_{m,\Delta t_i}$), F_3 = Average($+\Delta A_{m,\Delta t_i}$), and F_4 = Max ($+\Delta A_{m,\Delta t_i}$). F_1 is the smallest value in $\Delta A_{m,\Delta t_i}$, F_2 is the average of the sum of the negative values in $\Delta A_{m,\Delta t_i}$, F_3 is the average of the sum of the positive values in $\Delta A_{m,\Delta t_i}$, and F_4 is the largest value in $\Delta A_{m,\Delta t_i}$.

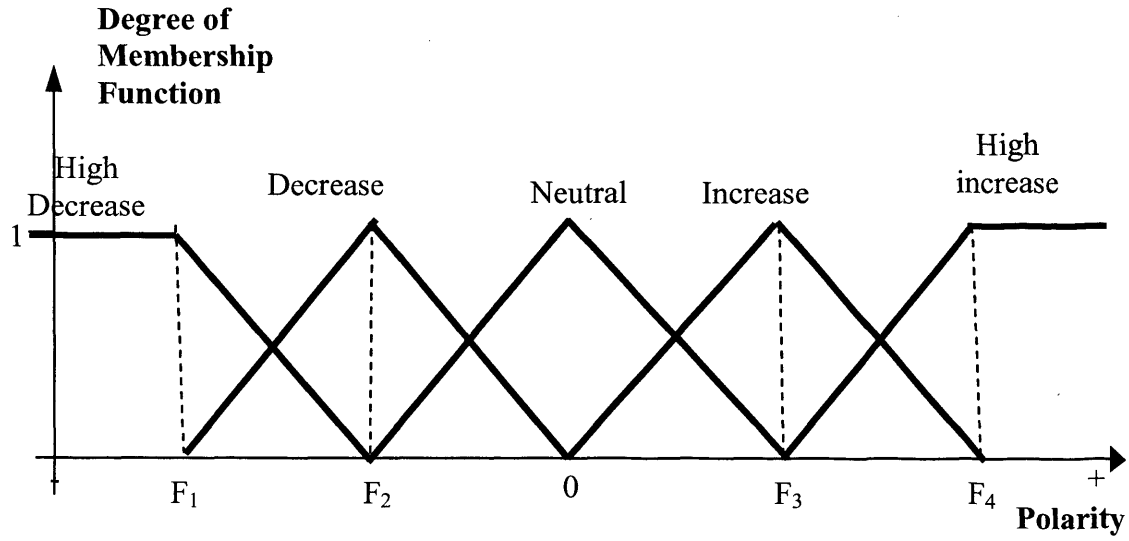


Figure 4.4 Proposed fuzzy partitions

$$F(\text{neutral}(\Delta A_{m,\Delta t_i})) = \begin{cases} 0, & if \quad \Delta A_{m,\Delta t_i} \le F_2 \\ \dfrac{\Delta A_{m,\Delta t_i} - F_2}{F_2}, & if \quad F_2 < \Delta A_{m,\Delta t_i} \le 0 \\ \dfrac{F_3 - \Delta A_{m,\Delta t_i}}{F_3}, & if \quad 0 < \Delta A_{m,\Delta t_i} \le F_3 \\ 0, & if \quad \Delta A_{m,\Delta t_i} > F_3 \end{cases} \tag{4.9}$$

$$F(\text{increase}(\Delta A_{m,\Delta t_i})) = \begin{cases} 0, & if \quad \Delta A_{m,\Delta t_i} \le 0 \\ \dfrac{\Delta A_{m,\Delta t_i} - 0}{F_3}, & if \quad 0 < \Delta A_{m,\Delta t_i} \le F_3 \\ \dfrac{F_4 - \Delta A_{m,\Delta t_i}}{F_4 - F_3}, & if \quad F_3 < \Delta A_{m,\Delta t_i} \le F_4 \\ 0, & if \quad \Delta A_{m,\Delta t_i} > F_4 \end{cases} \tag{4.10}$$

$$F(\text{high increase}(\Delta A_{m,\Delta t_i})) = \begin{cases} 0, & if \quad \Delta A_{m,\Delta t_i} \le F_3 \\ \dfrac{\Delta A_{m,\Delta t_i} - F_3}{F_4 - F_3}, & if \quad \Delta A_{m,\Delta t_i} > F_3 \end{cases} \tag{4.11}$$

$$F(\text{decrease}(\Delta A_{m,\Delta t_i})) = \begin{cases} 0, & if \quad \Delta A_{m,\Delta t_i} > 0 \\ \dfrac{0 - \Delta A_{m,\Delta t_i}}{F_2}, & if \quad F_2 < \Delta A_{m,\Delta t_i} \le 0 \\ \dfrac{\Delta A_{m,\Delta t_i} - F_1}{F_1 - F_2}, & if \quad F_1 < \Delta A_{m,\Delta t_i} \le F_2 \\ 0, & if \quad \Delta A_{m,\Delta t_i} \le F_1 \end{cases} \tag{4.12}$$

$$F(\text{high decrease}(\Delta A_{m,\Delta t_i})) = \begin{cases} 0, & if \quad \Delta A_{m,\Delta t_i} \ge F_2 \\ \dfrac{\Delta A_{m,\Delta t_i} - F_2}{F_1 - F_2}, & if \quad F_2 > \Delta A_{m,\Delta t_i} \end{cases} \tag{4.13}$$

Based on equations 4.9-4.13, the fuzzy database can be developed as shown in Table 4.10. The results in Table 4.11 are obtained by identifying the fuzzy dynamic attribute with the highest degree of membership in Table 4.10.

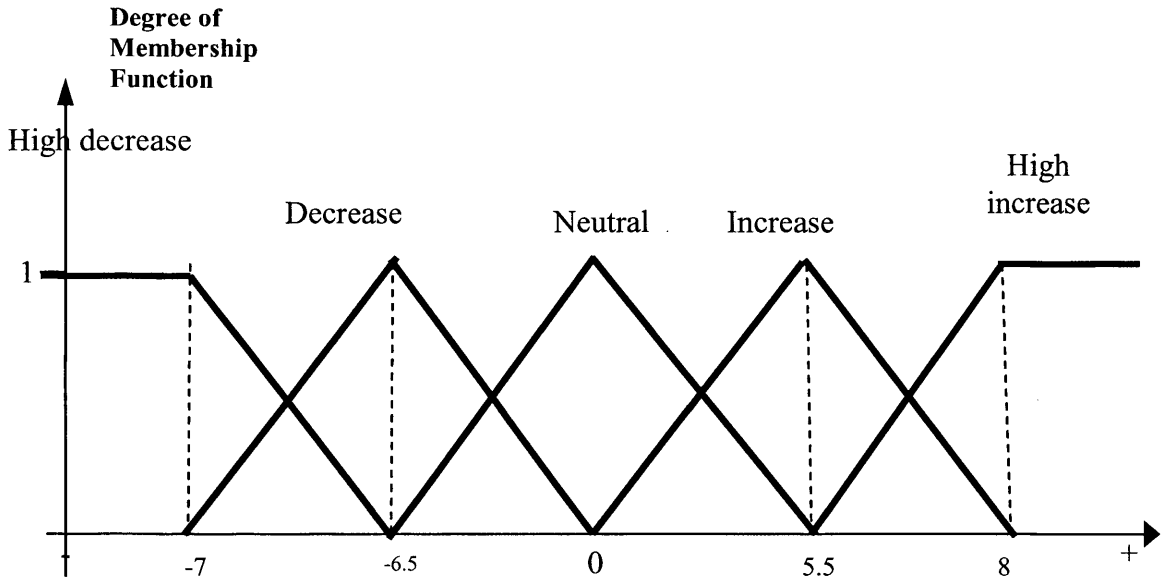


Figure 4.5 Membership function of ΔA_1

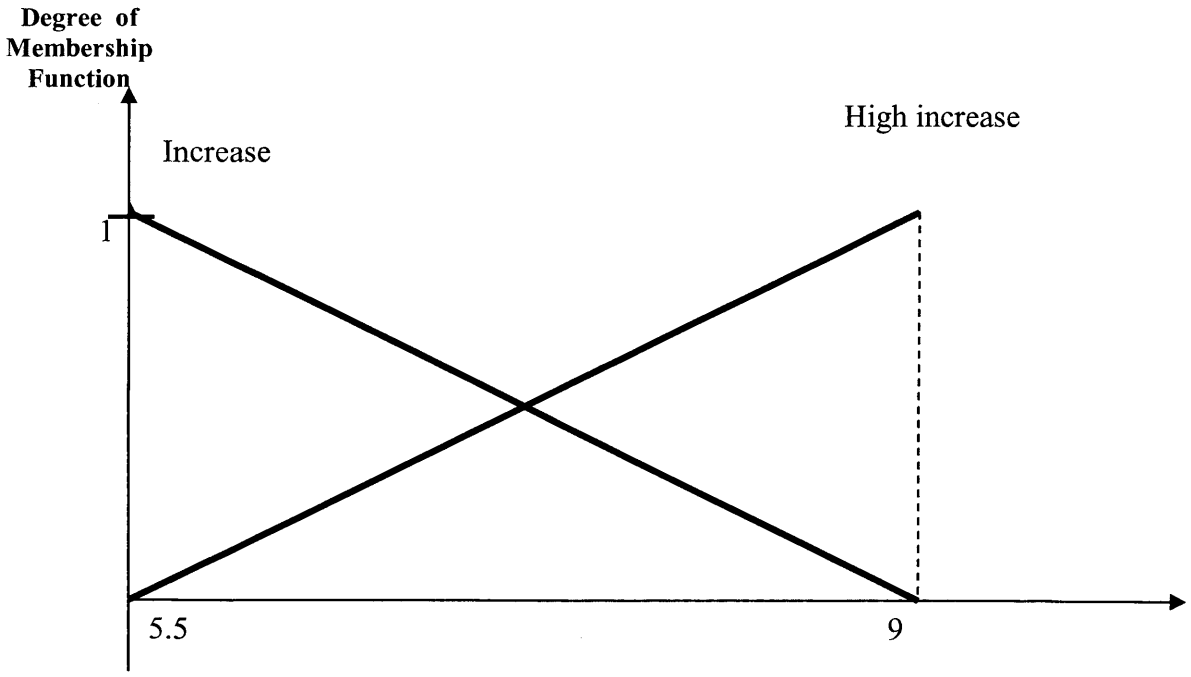


Figure 4.6 Membership function of ΔA_2

Δt	$F(\Delta A_1)$					$F(\Delta A_2)$		
	Hd	**D**	**N**	**I**	**Hi**	**N**	**I**	**Hi**
Δt_1	0	0	0	0	1	0.82	0.19	0
Δt_2	1	0	0	0	0	0	0	1
Δt_3	0	0.92	0.08	0	0	0.27	0.73	0
Δt_4	0	0	0.45	0.55	0	0	0.29	0.71

Table 4.10 Fuzzy database

Δt	$F(\Delta A_1)$	$F(\Delta A_2)$
Δt_1	Hi (= 1)	N (= 0.82)
Δt_2	Hd (= 1)	Hi (= 1)
Δt_3	D (= 0.92)	I (= 0.73)
Δt_4	I (= 0.55)	Hi (= 0.71)

Table 4.11 Pruned result

If Table 4.2 is presented, the membership function can be calculated for both ΔA_1 and ΔA_2. For ΔA_1, F_1 = -7, F_2 = 6.5, F_3 = 5.5 and F_4 = 8. For ΔA_2, F_3 = 5.5 and F_4 = 9.

Fully sympathetic support (ΔA_1, ΔA_2, ΔA_1) =

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.14a)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.14b)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.14c)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.14d)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.14e)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.14f)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.14g)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.14h)$$

Self-antipathetic support (ΔA_1, ΔA_2, ΔA_1) =

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.15a)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.15b)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.15c)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.15d)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.15e)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.15f)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.15g)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.15h)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.15i)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.15j)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.15k)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.15l)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.15m)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.15n)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.15o)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.15p)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.15q)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.15r)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.15s)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.15t)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.15u)$$

Self-sympathetic support $(\Delta A_1, \Delta A_2, \Delta A_1) =$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.16a)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.16b)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.16c)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.16d)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.16e)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hi))) \quad (4.16f)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = D), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.16g)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.16h)$$

Fully antipathetic support $(\Delta A_1, \Delta A_2, \Delta A_1) =$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.17a)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.17b)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.17c)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.17d)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.17e)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.17f)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.17g)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hd), \sum F((\Delta A_{1,\Delta t_{i+2}}) = Hd))) \quad (4.17h)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.17i)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hi), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))) \quad (4.17j)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.17k)$$

$$Avr(\sum(F((\Delta A_{1,\Delta t_i}) = Hd), \sum F((\Delta A_{2,\Delta t_{i+1}}) = Hi), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D))) \quad (4.17l)$$

Single attribute support $(\Delta A_m) =$

$$\Sigma\ (F\ ((\Delta A_{m,\Delta t_i}) = neutral\ )) \quad (4.18a)$$

$$\Sigma\ (F((\Delta A_{m,\Delta t_i}) = I)) \quad (4.18b)$$

$$\Sigma\ (F((\Delta A_{m,\Delta t_i}) = D)) \quad (4.18c)$$

where n is the total number of time stamps and m identifies the attribute of interest. I stands for increase, D for decrease, Hd for high decrease, and Hi for high increase. $\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = I), \sum F((\Delta A_{1,\Delta t_{i+2}}) = I))$ is a summation function giving the sum of the degree of membership functions for an increase in ΔA_1, which is followed

by an increase in ΔA_2 which induces another increase in ΔA_1 with respect to the time stamps Δt_i, Δt_{i+1}, and Δt_{i+2}.

Similarly, $\sum(F((\Delta A_{1,\Delta t_i}) = I), \sum F((\Delta A_{2,\Delta t_{i+1}}) = D), \sum F((\Delta A_{1,\Delta t_{i+2}}) = D)))$ is a function giving the number of times a decrease in ΔA_1 followed by a decrease in ΔA_2, which induces another decrease in ΔA_1 with respect to the time stamps Δt_i, Δt_{i+1} and Δt_{i+2}. The *neutral support* indicates the sum of degree membership functions for the linguistic terms "neutral" in a derived attribute. Avr() indicates the summation function over the total number of occurrences of that particular polarity combination.

4.7 Illustrative example

This example uses the same original database as presented in section 4.3. The membership functions given in Figures 4.7-4.10 are based on Table 4.7. Based on the membership functions, attribute linguistic term tables can be formed as shown in Table 4.12. By using fuzzy *support*, fuzzy dynamic rules are generated and these rules are shown in Table 4.13. The fuzzy *support* threshold used for generating dynamic rules is 0.1 and 0.3 respectively.

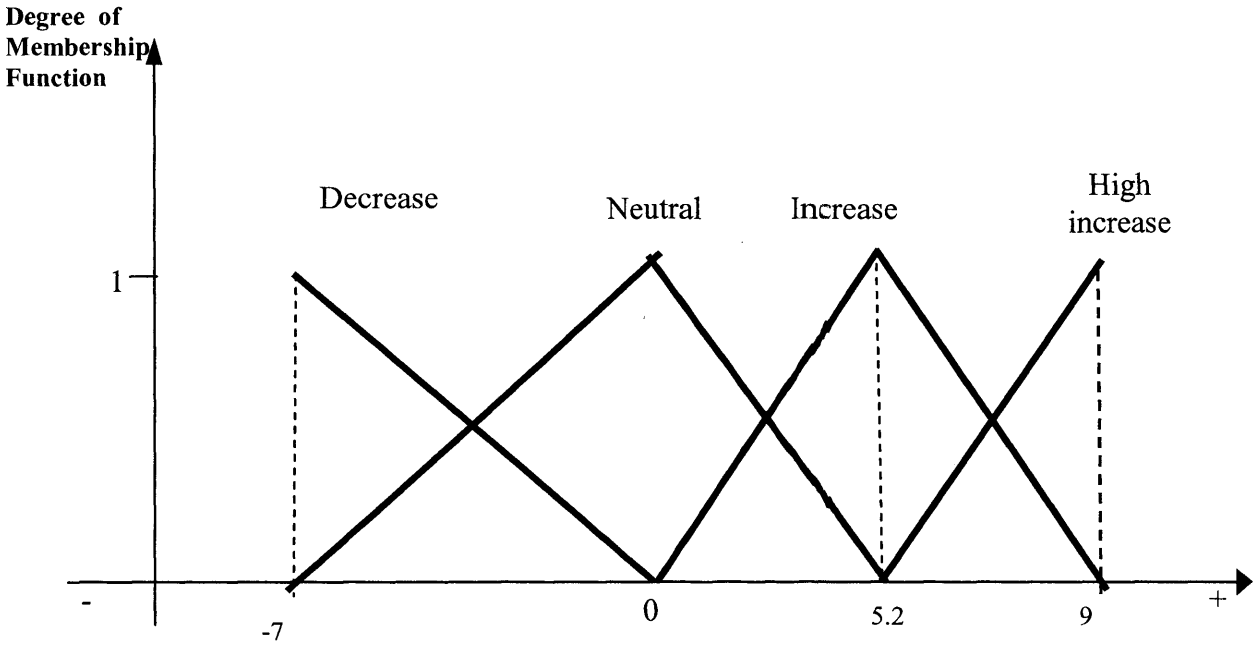


Figure 4.7 Membership functions for ΔA_1

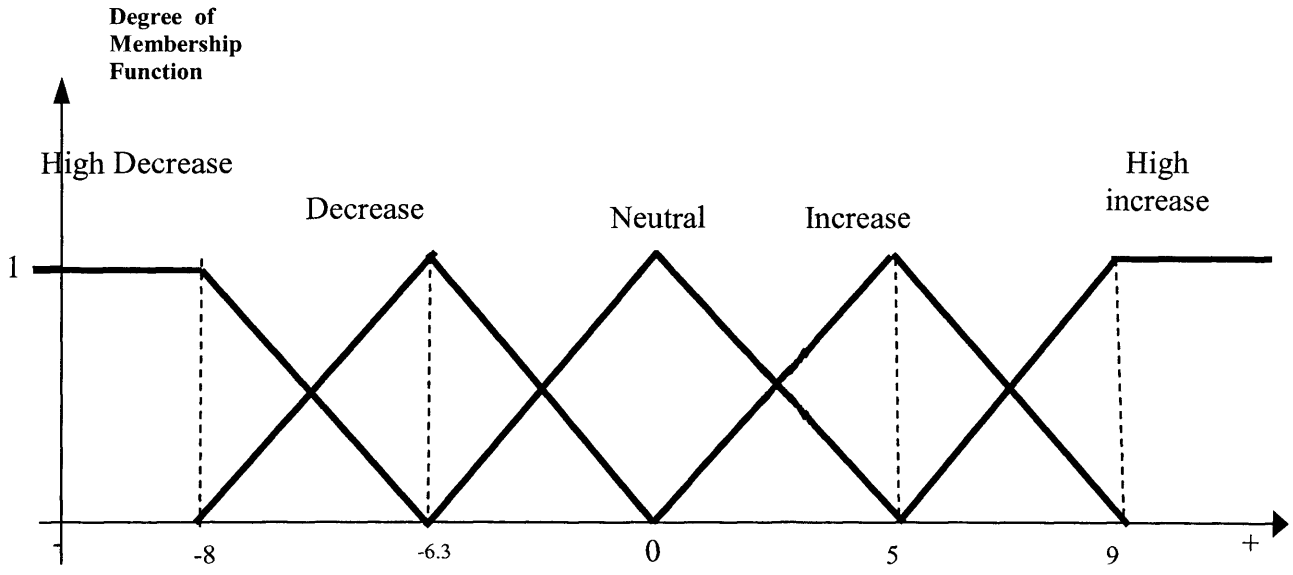


Figure 4.8 Membership functions for ΔA_2

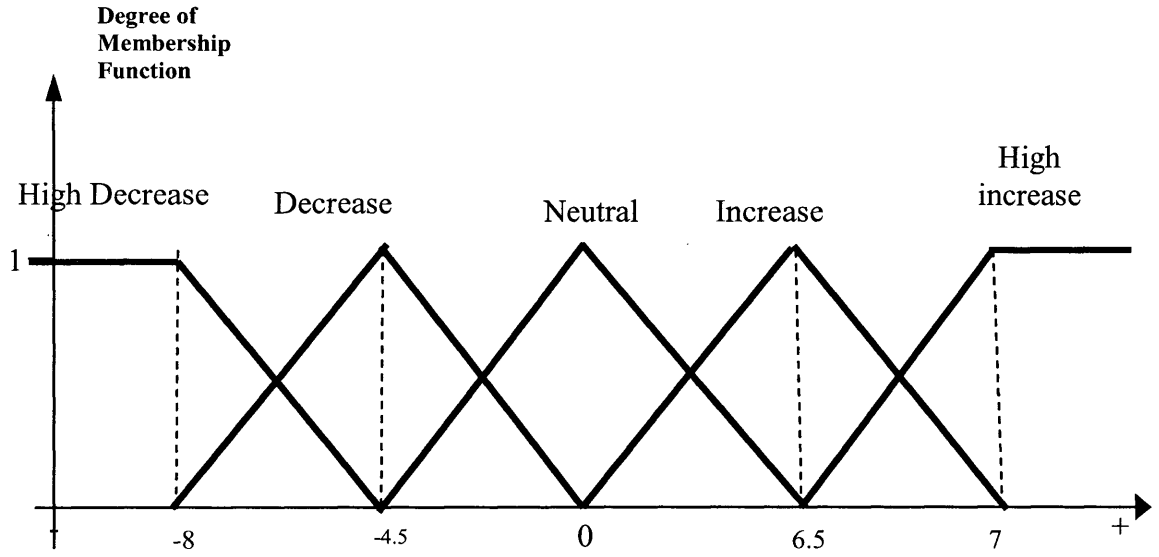


Figure 4.9 Membership functions for ΔA_4

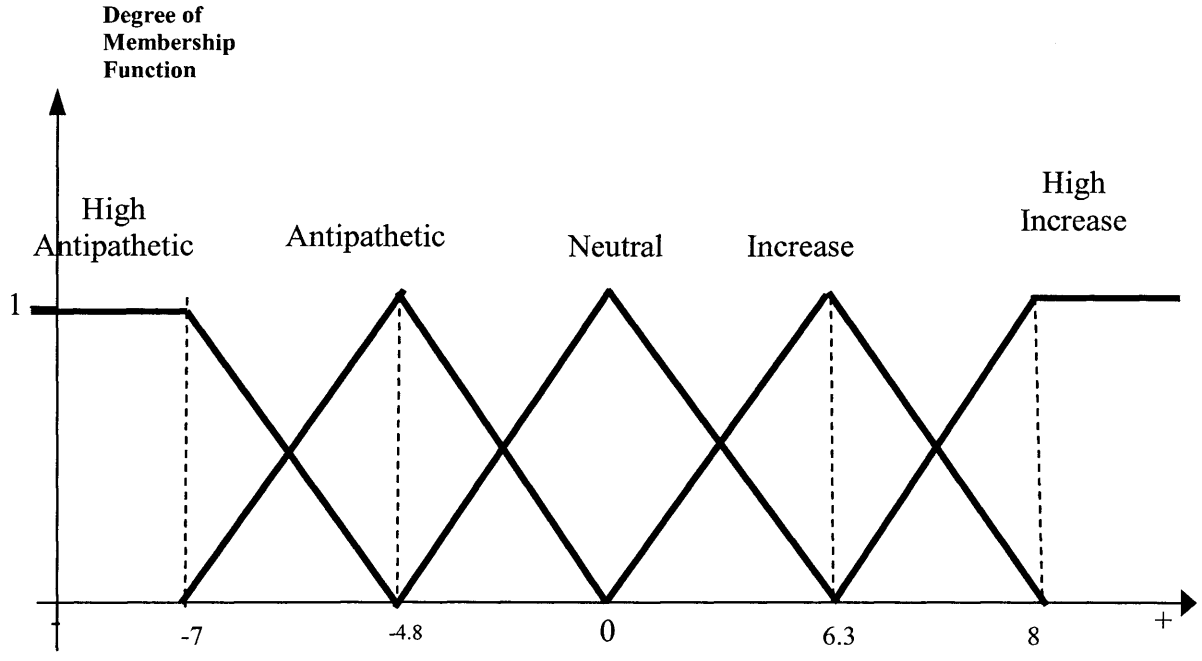


Figure 4.10 Membership functions for ΔA_7

	$F(\Delta A_1)$					$F(\Delta A_2)$				
Δt	Hd	D	N	I	Hi	Hd	D	N	I	Hi
Δt_1	0	0	0	0.27	0.73	0	0	0.80	0.20	0
Δt_2	0	1	0	0	0	0	0	0	0	1
Δt_3	0	0.88	0.12	0	0	0	0	0.20	0.80	0
Δt_4	0	0	0.42	0.58	0	0	0	0	0.11	0.89
Δt_5	0	0.14	0.86	0	0	0	0.95	0.05	0	0
Δt_6	0	0	0	0	0	0	0	0	0	0
Δt_7	0	0	0.04	0.96	0	0	0	0.40	0.60	0
Δt_8	0	0	0	0	1	1	0	0	0	0
Δt_9	0	0	0.81	0.19	0	0	0.79	0.11	0	0
Σ	0	2.02	2.25	2	1.73	1	1.74	1.56	1.71	1.89
	$F(\Delta A_4)$					$F(\Delta A_7)$				
Δt	Hd	D	N	I	Hi	Hd	D	N	I	Hi
Δt_1	0	0	0.07	0.93	0	0	0	0	0	0
Δt_2	0	0.22	0.78	0	0	1	0	0	0	0
Δt_3	0	0.89	0.11	0	0	0	0.83	0.17	0	0
Δt_4	1	0	0	0	0	0	0	0	0	1
Δt_5	0	0.67	0.33	0	0	0.09	0.91	0	0	0
Δt_6	0.25	0.75	0	0	0	0	0.20	0.80	0	0
Δt_7	0	0	0	0	0	0	0	0.21	0.79	0
Δt_8	0	0.88	0.12	0	1	1	0	0	0	0
Δt_9	0	0	0	0	1	0	0	0.05	0.95	0
Σ	1.25	3.63	1.41	0.93	2	2.09	1.94	1.23	1.74	1

Table 4.12 Fuzzified database

Δt	$MAX(F(\Delta A_2))$	$MAX(F(\Delta A_4))$	$MAX(F(\Delta A_7))$
Δt_1	D(= 0.80)	I(= 0.93)	0
Δt_2	Hi(= 1)	N(= 0.78)	Hd(= 1)
Δt_3	I(= 0.80)	D(= 0.89)	D(= 0.83)
Δt_4	Hi(= 0.89)	Hi(= 1)	Hi(= 1)
Δt_5	D(= 0.95)	D(= 0.67)	D(= 0.91)
Δt_6	0	D(= 0.75)	N(= 0.80)
Δt_7	I(= 0.60)	0	I(= 0.79)
Δt_8	Hd(= 1)	D(=0.88)	Hd(= 1)
Δt_9	D(= 0.79)	Hi(= 1)	I(= 0.95)

Table 4.13 Pruned results

$\Delta A_2, \Delta A_4, \Delta A_2$			$Support(\Delta A_2, \Delta A_4, \Delta A_2)$
Hi(= 1)	D(= 0.89)	Hi(= 0.89)	0.92
I(= 0.80)	Hd (= 1)	D(=0.95)	0.91
D(= 0.95)	D(= 075)	I(=0.60)	0.76
I(=0.60)	D(=0.88)	D(= 0.79)	0.75
$\Delta A_2 \Delta A_7 \Delta A_2$			$Support(\Delta A_2, \Delta A_7, \Delta A_2)$
Hi (= 1)	D(= 0.89)	Hi(= 0.89)	0.92
D(= 0.80)	Hd(= 1)	D(= 0.95)	0.91
D(=0.60)	Hd(= 1)	D(= 0.79)	0.79
$\Delta A_4 \Delta A_7 \Delta A_4$			$Support(\Delta A_4, \Delta A_7, \Delta A_4)$
I (= 0.93)	Hd(= 1)	D(= 0.89)	0.94
D(= 0.89)	Hi(= 1)	D(= 0.67)	0.85
Hd(= 1)	D(= 0.91)	D(= 0.75)	0.88
D(= 0.75)	I(= 0.79)	D(=0.88)	0.81

Table 4.14 Attribute combination

4.8 Fuzzy algorithm

The strategy consists of the following steps.

1. Fuzzify all dynamic attributes according to membership function using equations from (4.14) to (4.18).
2. Prune all dynamic attributes where the counted single attribute *support* is below the user-defined threshold.
3. Prune all dynamic attribute pairs where the counted pair wise *support* is below the user-defined threshold.

The mining process uses separate pruning method instead of brute force, which implies checking delay and feedback separately. First, the algorithm checks the delay relationships and prunes away the redundant attribute sets and then it checks the feedback relationships in the remaining sets. This reduces the numbers of scans in each pass. This algorithm works under the assumption that the *support* of a subset A is always at least the *support* of any other sets which contains A, similar to *DCM.* Tables 4.10 and 4.11 illustrate the pseudo code for the suggested algorithm.

The subroutine accepts the database, finds out and returns the complete dynamic set of the database and prunes away all attributes below the neutral support threshold. The algorithm generates a new transformed database from the derived database by specifying membership function which translates the dynamic attributes into fuzzy form. The *single fuzzy attribute* based on the membership function is first generated from the derived database.

Part 1: – Pre-processing: Removal of the "least" causal data from the database

Part 2: – Mining: Formation of a rule set that covers all training examples with minimum number of rules

Input: The original database (numerical database), the values of the pruning threshold for the neutral, sympathetic and antipathetic *supports*.

Step 1: Calculate F_1, F_2, F_3 and F_4.

Step 2: Initialize a new database with dynamic attributes based on the fuzzy memberships derived from F_1, F_2, F_3 and F_4.

Step 3: Sum the degree of membership for each of polarity with respect to each dynamic attribute.

Step 4: Prune away all the dynamic attributes with *supports* above the input thresholds.

Figure 4.10 The suggested algorithm part 1

Input: The preprocessed database, the values of the pruning threshold for the *supports* of the polarity combinations.

Step 1: Initialize a counter for attribute pair.

Step 2: Initialize an empty database.

Step 3: For each pair of attributes

Summarize the average of degree of polarity combination for the attribute pair, only $\Delta A_{m,\Delta t_i} \rightarrow \Delta A_{m,\Delta t_{i+1}}$

Store the non neutral linguistic terms contended pair with polarity combination above the input threshold into the empty database.

Step 4: For each pair of attributes in the new database

Summarize the average of degree of polarity combination for the attribute pair with feedback.

Prune the pair with polarity combination below the input threshold into the empty database.

Figure 4.11 The suggested algorithm part 2

If the sum of degree of membership is above the user specified *support* level, then the attribute will be kept, otherwise it will be pruned. In this subroutine, the fuzzy database is scanned in the same way as in section 4.4 and the fuzzy *support* based on the polarity combination is counted. The dynamic sets, with *support* larger than or equal to user defined support level, are generated following the separate pruning strategy.

4.9 Experiment

The algorithm was tested against 7 data sets, where 5 of the data sets were taken from the UCI Machine Learning Repository and 2 were taken from the real world (the ones with ® sign). The data is explained in appendix I. Table 4.15 shows the comparison between numbers of strong rules generated by *DCM* with *Support* level and *DCM* with fuzzy sets. Three *Support* thresholds were selected for the comparison. The result shows that the numbers of rules by the *DCM* algorithm are greater than those by fuzzified DCM (*FDCM*) in certain circumstances and are lesser in other circumstances.

This experiment was performed to compare various numbers of attribute and to identify the relationships between the numbers of rules mined and minimum *Support* values.

To assess the execution time of the algorithms, both fuzzified DCM (*FDCM*) and *DCM* with *Support* level were run on different datasets with different number of attributes. A comparison between the times spent running the datasets was then carried out. Table 4.16 shows the running time of the algorithms spend on different datasets in units of seconds. The result shows that the execution time of the fuzzy algorithm is longer than the *DCM* without and with *Support* level + normal value with a small margin.

It is clear that the number of rules mined increase along with an increase of the numbers of attributes for a certain minimum *Support* threshold. Execution times also increase along with an increase of numbers of attributes. The fuzzy method gives better experimental result than that with crisp partitions as it is shown here. However the run time is longer due to the fuzzification.

4.10. Real life example

This section continues from the real life example from last chapter. In this section FDCM is applied on the dataset to extract the dynamic causal rules. The fuzzifying process is done by Microsoft Excel with following code:

F_1 = MIN(data range)

F_2 = SUMIF(Data range, "<0")/COUNTIF(data range, "<0")

F_3 = SUMIF(Data range, ">0")/COUNTIF(data range, ">0")

F_4 = MIN(data range)

Support Data Set	DCM			FDCM		
	0.05	0.08	0.10	0.05	0.08	0.10
Adult	5	2	0	5	2	0
Bank	11	2	0	10	2	0
Cystine	5	3	0	5	3	0
Market basket	6	1	0	5	1	0
Mclosom ®	21	10	3	20	12	1
ASW®	17	8	0	18	6	0
Weka-base	6	5	0	6	5	0

Table 4.15 Comparison of algorithms based on number of rules

Data set	DCM	FDCM
Adult	< 1s	<1s
Bank	<1s	<1s
Cystine	<1s	1.5s
Market basket	2.5s	3s
Mclosom ®	230s	232s
ASW®	320s	328s
Weka-base	2s	3s

Table 4.16 Comparison of running time (in seconds)

Give dynamic data x, the membership function can be expressed as:

High increase fuzzy membership:

IF((x > F_3), (x- F_3/ F_4 -F_3),0)

Increase fuzzy membership:

IF(AND(x > 0, x < F_4),IF(AND(x > 0, x < F_3), x/ F_3, (F_4 – x)/ F_4 – F_3),0)

Neutral fuzzy membership:

IF(AND(x> F_2, x < F_3),IF(AND(x> 0, x < F_3), (F_3-x)/ F_3, (x –F_2)/ F_2),0)

Decrease fuzzy membership:

IF(AND(x > F_1, x < 0),IF(AND(x > 0, x > F_2), x/ F_2, (F_1 – x)/ F_1 – F_2),0)

High decrease fuzzy membership:

IF((x > F_2), (x- F_2/ F_1 –F_2),0)

Table 4.17 indicates a rule such as {A05005008, C15276179} has 0.90 percent occurrence on the whole database. This rule indicates that an increase in the production of metal type A05005008 will lead to either an increase/decrease in the production of C15276179 in next time stamp which will again lead to a decrease in the production of A05005008. This means A05005008 will be kept at a constant level even when there is an increase in its production due to its causal relation with C15276179. Therefore it is critical to monitor the changes occurring in C15276179 instead of reacting to the temporal changes occurring in the production of A05005008. The result is compared with the DCM result based on the following calculation:

$$Abs\left(\frac{FuzzySupport - DCMSupport}{DCMSupport}\right)*100\% \qquad (4.19)$$

where Abs() stands for the absolute value.

Strong rules	Support	Improvement
Sympathetic		
{C15276179, F030008}	0.095	2%
{J08008008, F060010}	0.090	1%
{A04004004, A05005005}	0.083	4%
{A05005006, C10251104}	0.080	5%
{A04004004, F100020}	0.082	0%
Antipathetic		
{A05005008, C15276179}	0.090	2%
{C10251104, F070010}	0.085	2%
{A05005008, F030008}	0.081	1%

Table 4.17 Result generated by the FDCM algorithm

4.11. Summary

In this chapter, the algorithm of *Fuzzy Dynamic Causal Mining* (*FDCM*) is proposed, which can handle transaction data sets with quantitative values and discover interesting rules/patterns among them. The rules thus mined exhibit quantitative regularity on multiple levels and can be used to provide suggestions to appropriate supervisors. Compared to the *DCM* mining methods for quantitative data, the *FDCM* approach gets smoother mining results due to the fuzzification of the data sets.

This chapter also shows that fuzzy logic has a number of advantages over classical crisp analysis in resolving *DCM problems*. Fuzzy logic theory makes it evident that the theoretical difficulty of realizing dynamic causality is related to the binary classification of concepts such as binary polarity. This makes it difficult to link the result to empirical behaviour observed in manufacturing and in other research areas.

The mined rules are expressed in linguistic terms with a more natural form. The representation of the knowledge discovered may be represented easily and understandably for human users. When compared to fuzzy mining methods, which take all fuzzy regions into consideration, the method achieves better time complexity since only the most important fuzzy term is used for each attribute. If all fuzzy terms are considered, the possible combinational searches are too large. Therefore, a trade-off exists between rule completeness and time complexity.

Although there are more complexities in the computation, the fuzzified *DCM* algorithm is able to use arrays as its data structure. The model is designed to assist

users in making decisions and to determine the impact of certain decisions and characteristics on the system. The model is used to evaluate the performance of the target system types subject to the domain knowledge.

Chapter 5 DCM applied in Game Theory

5.1 Preliminaries

This chapter applies *DCM* as a new method to analyse *the problem of a Game* and uses *Formal Concepts* as a representation for the solution. The work presented in this chapter is motivated by a rapid increase of interest in game theoretical analysis and by the need for a tool to extract and represent the knowledge and information from game-related databases (Tveit *et al.*, 2002).

This chapter presents a novel solution in the hope of outlining some fruitful directions for future research. Such a solution is called *Dynamic Causal Game Mining* (*DCGM*). *DCGM* uses a rule-based description of the dynamics in a game involving two or more players. The pattern of each player is presented in a simple and understandable structure. This presentation of the dynamics must be accompanied with specific linguistic terms for human understanding. The originality of the approach is in the mixing together of several branches of scientific analysis. This involves fuzzy logic, *Association Mining*, and *System Dynamics*, etc.

This chapter assumes that each individual selects a strategy based on a pattern. The pattern is extracted from historical databases and expresses the possible actions for each player. This pattern will assist in identifying future strategy selections for the player and for the opponent.

5.2 Related work

Data mining investigates the possibility of online learning for novel game characters as well as for more sophisticated experts. Data mining techniques can be used to determine which information is relevant to generate and switch between appropriate behaviours. Finally, besides its long-term commercial impact, the game domain also provides interesting impulses for behaviour learning in general.

There are two fields in the research of the game where *Data Mining* has been used heavily. The first is computer-based multiplayer games. These games are not just chess or poker. They can be described as Role Playing Games (RPG) or *Multiplayer Online Games*. Modern computer-based games create virtual worlds of enormous size in which the player slips into the role of a virtual character that has to fulfil a certain task. The characteristics of these games are that a large number of players take part in a persistent virtual world where they communicate, cooperate, fight and build virtual characters. Depending on the genre of the game, tasks involved in the game can vary from the building and administering of cities or civilizations, through solving adventurous quests, to simply staying alive on a virtual battlefield.

During the playing of these games, each player moves through a virtual world. The main task is to play against every other character in the game. These characters can be the computer or other players. The player will lose health, armour, cities and money, all of which can be compensated for by collecting corresponding items distributed in the game.

The state of the character therefore is almost completely determined by its current position in the game. Typical examples of such games are *Warcraft, DOOM, Neverwinter Nights,* and *Counter-Strike.* Screenshots taken from the game *Neverwinter Nights* are shown in Figure 5.1.

The purpose of *Data Mining* in these games is to discover players' patterns, e.g. rules or statistics that possibly can be used to improve the game. The actions of computer-controlled characters often appear artificial since they just cycle through fixed movements and action. With the extracted pattern, the game can simulate a computer generated virtual environment to behave more "intelligently" (Cass, 2002). If the computer generated virtual world acts more like the real world, then the human players become more interested and stay longer, which again increases the revenue of the game producer. The economic usage of game patterns and virtual players were analysed first by (Tveit *et al.*, 2002 and Tveit *et al.*, 2003). They suggested that the behaviour of non-personal or non-playable entities (monster, trees, animals) in a game should also be processed in the same way as the real players. The data related to non-playable entities must be combined with global information about the game (storyline, major events, user interface). Results of the mining process can either be used as input to a recommender service or as metrics to the game service operator.

The second type of game is board-based ones such as chess, backgammon, and poker. The earliest work on learning from a chess database is reported in (Michalski and Negri, 1977), where the inductive rule learning algorithm *AQ* (Michalski, 1969) is applied to the *KPK* database described in (Clarke, 1977).

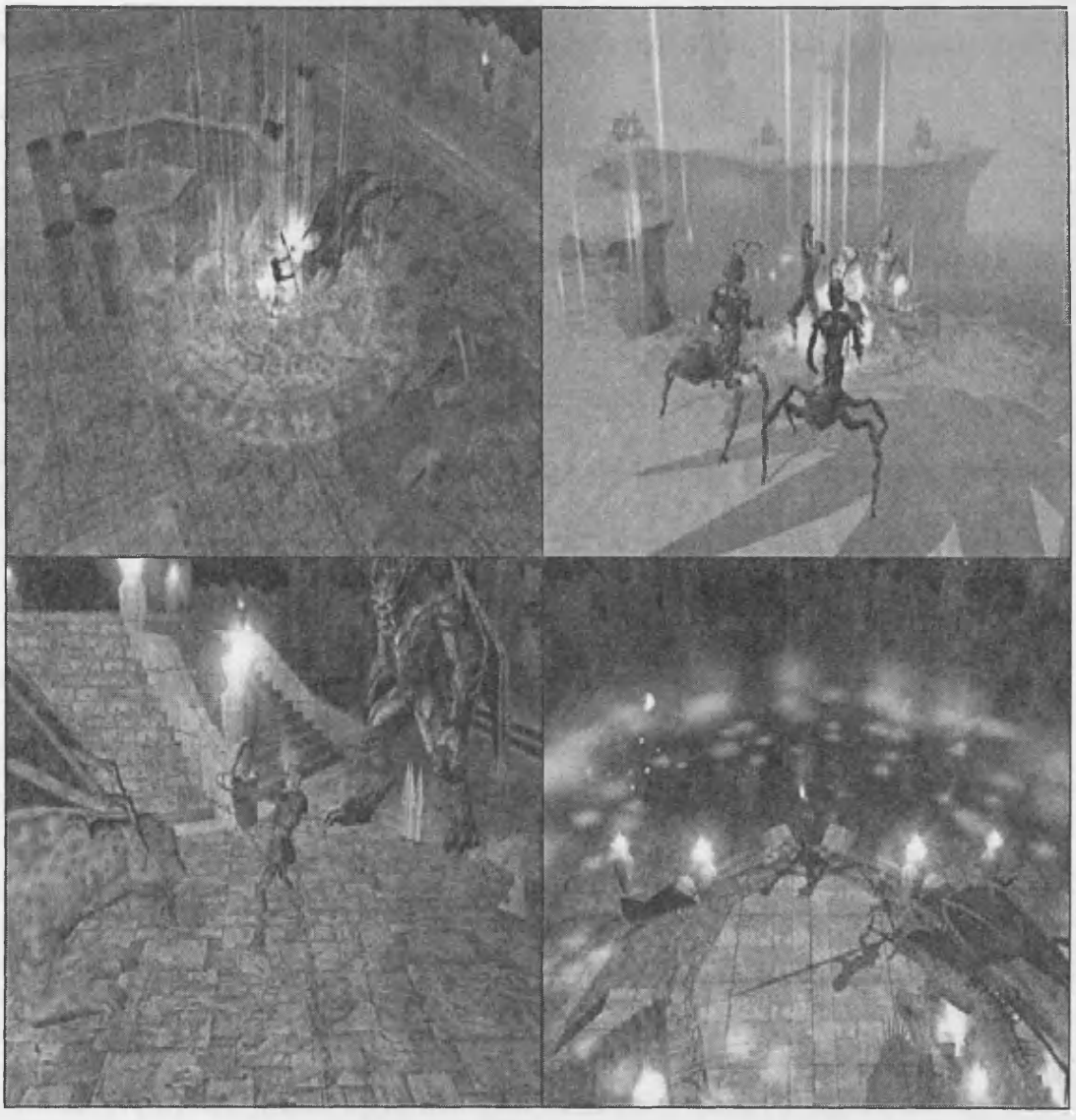


Figure 5.1 Screenshots of *Neverwinter Nights*

(http://www.nextway.ch/images/mac_nevershadows.jpg).

Quinlan described several experiments for learning rules for the chess endgame (Quinlan, 1983). More specifically, he used his decision tree learning algorithm *ID3* to discover recognition rules for positions of the endgame.

Shapiro decomposed the mining problem into a hierarchy of smaller sub-problems that could be tackled independently. A set of rules was induced for each of the sub-problems, which together yielded a more understandable result (Shapiro and Niblett, 1982). A similar semi-autonomous rule debugging process for refining the attribute set was used to generate decision trees for the game (Weill, 1994).

Hsu applied statistical techniques to yield optimal performance in a database of grandmaster moves *(Hsu, et al.*,1990). However, with the success of Tesauro's backgammon player, research in this area shifted gradually from tuning on databases to tuning by self-play via temporal difference learning (Tesauro, 1995).

Paterson used a clustering algorithm to automatically structure the chess game (Paterson, 1983). Muggleton applied *DUCE*, a machine learning algorithm that was able to suggest high-level concepts to the user autonomously (Muggleton, 1990). The results have been negative, similar to many other works (Bain and Muggleton, 1995, Nunn, 1994, Roycroft, 1988).

This chapter discusses a broader view of a game, which involves not only board games and computer-based games but also many other types of games such as economic games involving conflicts in bargaining, biochemical games involving

interactions between cells and bacteria, political games involving elections of a president, etc. All of these games can be analysed by using the principles of *Game Theory*. However *Game Theory* is not easy to implement and a proper representation is needed. The fundmentals of game theory are explained in Appendix B.

5.3 Drawbacks of Game Theory

5.3.1 Problems with rationality

Classical game theoretic analysis assumes *rationality of the player* (von Neumann 1965). However, this assumption does not guarantee a unique and well-understood solution. Often the solution is irrational and indicates that a player might pick one of the several strategy combinations but cannot determine which strategy is the winning one.

The assumption of rationality is also rather ambitious. It implies that decision-makers are capable of having complete knowledge of all relevant information, a perfect anticipation of future consequences and the full capacity and time availability. In reality, none of these assumptions is sufficiently met (Sterman, 2000).

Rational behaviour is difficult to address because people often just do not behave rationally (Meyer and Booker, 1991). Irrational behaviour may follow from the following factors.

- Failure to update action as new information becomes available.

- Acting on perceptions that are in fact false.
- Acting on personal agendas.
- Poor understanding of interdependencies between events.

The challenge is to capture the relevant causal behaviour between players that generate a solution of interest without making the model too complex. Selecting the important relationships from the less important ones can only be done by experience or learned from historical data.

5.3.2 Problem with designer's dilemma

Although *Game Theory* is fairly well developed, there are many aspects of it that are not perfect. Many of the analyzed games have provided a solution that is completely different from reality. For example, it is common knowledge that the solution to the prisoner's dilemma is that both prisoners confess. However, in real life many similar situation yields different results.

The *Designer's Dilemma* referred to in this section is adapted from the "*Prisoner's Dilemma*" (Axelrod, 1984). Imagine that two designers are available for a given design task for a manufacturing company that does not have sufficient resources for both designers. The two designers are isolated from each other and the manufacturer visits each of them and offers them a deal for designing a perfect product. If only one designer accepts the deal, this designer will carry out the whole project and will receive all of the payment. If neither of them accepts the offer, the manufacturer will keep them for future projects, paying them a royalty. However, if one of them refuses the job, that designer will take no further part in the process and will receive no

payment. If both accept they will be working together and each of them will receive less payment than if the other had refused the job. The dilemma resides in the fact that each designer has a choice between only two options but cannot make a good decision without knowing what the other one will do. This is shown in Table 5.1.

The example shows a one shot strategy profile setup according to *Game Theory*. A contractor is contracting project with 2 designers. As in the designer's dilemma, each of the players is tempted to *refuse*, because they can get 100 even without spending any effort on the project, because the contractor want to keep them for next project. However, due to the uncertainty about the opponent's action, a rational player would realise that the opponent might *accept*, thus leaving him with the worst payoff. To be on the safe side he will choose to *accept*. If both players are rational, the solution to the *designer's Dilemma* is that both should *accept*.

In real life, strategy selection is based on a player's knowledge of the opponent. If the player has perfect knowledge of the opponent, then the player has control of the game at all stages. This knowledge can be gathered from past records by applying *Data Mining* methods. As mentioned above, *DCM* is an alternative that can be applied in this work to extract knowledge in the form of causal rules from historical databases of strategies and to assist in strategy selection.

Designer 2 / Designer 1	Accept	Refuse
Accept	(500,500)	(1000,0)
Refuse	(0,1000)	(100,100)

Table 5.1 Designer's dilemma

5.4 Rule-based Game Theory

Game Theory originates from the work of Von Neumann and Morgenstern (see Appendix B). There are two ways to classify *Game Theory,* namely cooperative/non-cooperative (von Neumann, 1965, Nash, 1951) or rule/freewheeling-based (Kleindl, 1999). In rule-based *games,* players interact according to specified rules. These rules might come from contracts, loan covenants or trade agreements. In the second type, freewheeling *games*, players interact without any external constraints. For example, buyers and sellers may create value by transacting in an unstructured fashion. Business is a complex mix of both types of *games*. For rule-based *games*, *Game Theory* offers the principle that to every action there is a reaction. However, this reaction is not equal or opposite to the action. To analyse how one player will react to another player's move, one needs to play out all the reactions to their actions as far ahead as possible.

The patterns in the game are expressed also as rules and in the scope of this thesis *DCM* is used to bind and identify interconnections of the attributes to produce structural information. The analysis may be divided in macro level and micro level.

Micro level is closely associated with the details in short-term and real-time game play. Because the decisions are made for a short term, less accurate rules extracted in this level do not necessarily lead to total failure in the game. For instance, if the

chosen strategy is wrong, there is still time and resource to reroute the path to the new destination.

On the macro level, the decision is made over a long period of time. The amount of data can be large and therefore the main problem is to filter it down to a suitable form. Information can be lost and the problem is to ensure that all vital knowledge is kept. Decision-making at the macro level is speculative and the cost of a wrong decision is high. Because decisions are made more frequently at the micro level than at the macro level, the requirement of quality cannot be as high for the former..

Pattern recognition in the game means recognizing the underlying dependencies between the players. Previous work in game pattern recognition and game mining (Tveit et. al., 2002, Tveit et. al., 2003) focuses on the dependencies between the players in the sequence. They claim that it is sufficient to consider short term modelling. However, the previous chapters showed that long term modelling is equally important and that information feedback is also significant.

Definition 5.1 *A game consists of a set of dynamic causal rules governing a competitive situation. The rules represented by two or more variables reflect individuals' or groups' selectable strategies.*

Definition 5.2 *A rule in a rule-based game is common if it is known to all players and is hidden otherwise. The hidden rules can be revealed by techniques such data mining. Different data mining techniques will uncover different kinds of rules.*

Those players using data mining will have an information advantage or disadvantage (if the data mining is wrongly applied) in selecting their strategy and this leads to the asymmetric distribution of information (see appendix B for details about asymmetric information).

5.5 Dynamic causal game mining

Dynamic causal game mining describes the possible cause of conflicts between buyers and seller, disagreement between agents, etc. The process of Dynamic causal game mining (*DCGM*) defines and states the key variables. The time horizon for the problem is also defined, so that the cause and effects can be identified. *DCGM* provides the details about the players and decides which of their individual strategies should be included.

Definition 5.3 *Dynamic causal game mining is a data mining process of automatically revealing the dynamic interactions between different players involved in a game. This focuses on the dynamics in a game and treats the game as a system. The main idea is to decide what behaviour the players should select, based on the assumption that each action a player takes at a given time will influence the reaction of the opponent(s) at later time, since the actions of the player(s) influences the evolution of the state of a system over time.*

Definition 5.4 *An action is a move taken by a player at some point during the playing of a game. In the scope of this thesis, an action is a property inherent in a database entity or associated with that entity for database purposes. An action is obtained from*

real life in a defined time interval.

Definition 5.5 *A strategy is a complete plan of actions for a player and determines the player's behaviour. A strategy consists of the dynamic change of the actions over time. Let D denote a data set which contains a set of n records with actions* $\{a_1, a_2, a_3, \ldots, a_m\}$, *where each action is of a unique type (for example, sale price, production volume, inventory volume, etc). Each action is associated with a time stamp* t_i, *where* $i = \{1, 2, 3, \ldots, n\}$. *Let* D_{new} *be a new database constructed from D such that strategy S in* D_{new} *is given by:*

$$s_{m,\Delta t_i} = a_{m,t_{i+1}} - a_{m,t_i} \tag{5.1}$$

where m identifies the action of interest.

Definition 5.6 *A player is an intelligent participant in a game with a specific goal. The goal could be profit maximisation or risk minimisation. Normally, there is more than one player in a game.*

Theorem 5.1 A strategy is in a state of *Nash equilibriium* (Nash, 1951) if the neutral support is above the user-given threshold.

Proof: According to the definition of *Nash equilibrium,* the player should select this strategy no matter what other players do. Thus there exists no causal relationship between the players' strategy and other player's strategy. In such cases, the player believes that this strategy gives a better or equal payoff when compared to any other available strategy.

5.6 Dynamic causal game representation

An important step for *DCGM* is the development of a suitable structure of game concepts in which the discovered knowledge can be formulated. The previous work on the chess database has illustrated the need for an appropriate structure for learning (Furnkranz, 1995). However, such knowledge representation for game concepts could also contribute significantly to other research related to *Game Theory*, such as collaboration or conflict analysis.

The characteristics of such a representation should be expressive of the abstracted strategic concepts. The representation has to be extensible and easily understood by a user and be capable of efficient implementation.

Classical games use game tree and matrix as the presentation form and indeed these provide a very detailed description of the *game*. It is, however, not practical because the tree becomes absolutely huge very quickly, even for simple games. The normal matrix form is too simplified. An attempt to provide a complete description of a complex *game* like *Bridge* would lead to a huge amount of game tree nodes and thousands of game matrices.

This chapter uses the same concepts but a different representation. The *normal form* of the game is expressed with *Formal Concepts* and the extensive form of the game is represented with lattice.

Definition 5.7 *Given a set of strategies* $S_{playerA} = (s_{1,\ playerA}, s_{2,\ playerA} \cdots\cdots .. s_{m,\ playerA})$ *of player A, where some strategies are dynamic causally related to some strategies in* $S_{playerB} = (s_{1,\ playerB}, s_{2,\ playerB} \cdots\cdots .. s_{m,\ playerB})$ *of player B. If* $s_{i,pA}$ *is dynamic causally related to* $s_{i,\ playerB}$, *then such a pair* $(s_{i,\ playerA}, s_{i,\ playerB})$ *is called a formal concept.*

Theorem 5.2 *A dynamic causal game strategy is an ordered set based on time stamps.*

Proof: Given dynamic action Δa_1, Δa_2, Δa_3 and an hierarchy relation $\leq$ (smaller or equal to), thus

if $t_{\Delta a_1} \leq t_{\Delta a_2}$ and $t_{\Delta a_2} \leq t_{\Delta a_1}$ then $\Delta a_1 = \Delta a_2$ (antisymmetry)

if $t_{\Delta a_1} \leq t_{\Delta a_2}$ and $t_{\Delta a_2} \leq t_{\Delta a_3}$ then $\Delta a_1 \leq \Delta a_3$ (transitivity)

$t_{\Delta a_1} \leq t_{\Delta a_1}$ (reflexivity)

Theorem 5.3 Given the context $(S_{playerA}, S_{playerB}, R)$ describing a set of strategy $S_{player1}$, which belongs to player A and $S_{playerB}$ to player **B** and a binary relation R, there is a unique corresponding lattice structure, which is known as a concept lattice.

Proof: Each node in lattice L can be a couple, noted $(s_{t,\ playerA}\ ,\ s_{t,\ playerB})$, where $s_{t,\ playerA} \in S_{playerA}$ is called an extension of the concept, $s_{t,\ playerB} \in S_{playerB}$ is called an intension of the concept. Each couple must be complete with respect to R. A couple $(s_{t,\ playerA}\ ,\ s_{t,\ playerB}) \in S_{playerA} \times S_{playerB}$ is complete with respect to relation R if the following properties are satisfied.

(1) $s_{t,\ playerA} = \{ s_{t1,\ playerA} \in S_{playerA} . \forall s_{t2,\ playerB} \in s_{t,\ playerB}, (s_{t2,\ playerB}\ ,\ s_{t2,\ playerB}), t_1 \neq t_2 \}$

(2) $s_{t,\ playerB} = \{ s_{t2,\ playerB} \in S_{playerB} . \forall s_{t1,\ playerA} \in s_{t,\ playerA}, (s_{t2,\ playerB}\ ,\ s_{t2,\ playerB}), t_1 \neq t_2 \}$

5.6.1 The normal representation of DCGM

Definition 5.8 *A formal dynamic causal game context of the form* $K = (S_A; S_B; C)$ *where* S_A *and* S_B *are strategies belonging to player A and B; moreover,* $C(S_A, S_B)$ *is a dynamic causal relation between the sets of objects and attributes, respectively.*

Definition 5.9 A formal dynamic causal game context in its normal form can be considered as a table, which expresses causality between records in S_A and S_B.

Formal Concepts Analysis (*FCA*) (Willie, 1985) is a method of data analysis that identifies the associative relation between a group of defined *concepts* and a group of defined *objects*. Table 5.2 shows the matrix of the normal form representation of a game in which a player's causal relationships is already retrieved through *DCM. s* stands for sympathetic and *a* stands for antipathetic. In this particular example, player *A*'s strategy 2 has a strong sympathetic relation to player *B*'s strategies 2 and 3. If a player considers changing his strategy, he needs to be concerned about the causal effects of the change and if they are desirable.

5.6.2 The extensive representation of DCGM

The extension of concepts is retrieved by tracing all paths which lead down from the node to collect the formal objects. In this example, the formal object is strategy 3 of player *B*. To retrieve the intension of a *Formal Concept* one needs to trace all paths which lead up in order to collect all the formal attributes. In this example, there is the

empty node. Thus the concept with the extension "Strategy 3 player *B*" covers all the strategies used by player *A*.

"Strategy 2 player *B*" is a sub-concept of "Strategy 3 player *B*" because it covers only 2 strategies from player *A*. Thus the extension of "Strategy 2 player *B*" is a subset of the extension of "Strategy 3 player *B*" and the intension of "Strategy 2 player *B*" is a superset of the intension of "Strategy 3 player *B*". All edges in the extensive form of a concept lattice represent this sub-concept/super-concept relationship.

The top and bottom concepts in a concept lattice are special. The top concept has all formal objects in its extension. Often its intension is empty but it does not need to be empty. The bottom concept has all formal attributes in its intension. If any of the formal attributes mutually exclude each other then the extension of the bottom concept must be empty. The top concept can be thought of as representing the "universal" concept.

An extensive form as in Figure 5.2 is a graphical visualisation of the concept lattice. The extensive form consists of nodes and lines and each node represents one attribute of the given context. The relations of the context and the information about it can be read from the extensive form. It allows the investigation and interpretation of relationships between concepts, objects and attributes. This includes object hierarchies, if they exist in the given context. An extensive form contains the relationships between objects and attributes and thus is an equivalent representation of a context, i.e. it contains exactly the same information as the normal form.

PB \ PA	Strategy 1	Strategy 2	Strategy 3
Strategy 1	a		
Strategy 2		s	a
Strategy 3	a	s	a

Table 5.2 Normal representation of DCGM

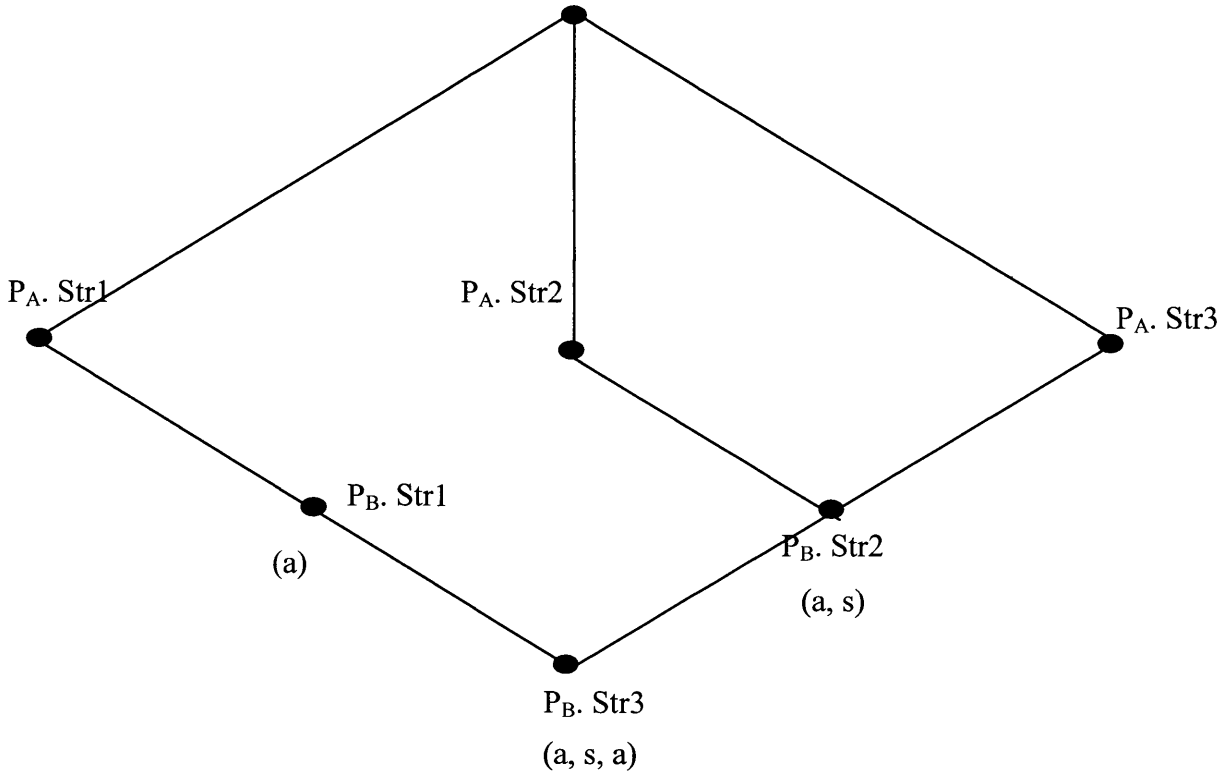


Figure 5.2 Extensive representation of DCGM

5.7 Fuzzy game concepts

The concept representation introduced in previous section is not feasible if the size of lattice becomes huge. This section suggests using fuzzy logic and linguistic terms for the polarity representation. The representation is built into the existing concepts which will capture the uncertainty, imprecision, or vagueness characteristics of such systems.

This section focuses on the development of a Formal Fuzzy Causal Game Concept (*FFCGC*) which defines a binary operator for the manipulation of fuzzy sets in a game environment. The *FFCGC* extends the work presented in previous sections to incorporate fuzzy logic in game concepts. *FDCGC* produces higher level observations, which are forwarded to the decision-making system. Each player involved in the game is allowed a set of possible actions and an appropriate response based on the information given by *FDCGC*.

Definition 5.10 A fuzzy dynamic action f is identified by a so-called membership function, which is a generalization of the characteristic function of a dynamic action $\Delta a_{m,\Delta t_i} \subseteq D_{new}$. For each record in $\Delta a_{m,\Delta t_i}$, a function specifies the degree of membership of μ that belongs to a linguistic term L. Usually, membership degrees are taken from the unit interval [0, 1], i.e. a membership function is a mapping $D_{new} \rightarrow$ [0, 1]. $F(\Delta a_{m,\Delta t_i})$ denotes the degree of membership of the dynamic attribute $\Delta A_{m,\Delta t_i}$ at time stamp i.

$$f = \mu L \qquad where\ \mu = m(\Delta a_{m,\Delta t_i}) \rightarrow [0, 1] \qquad (5.2)$$

Definition 5.11 *Give a player A with strategy S_A represented in a fuzzy form $F(S_{A,}\ t)$ for every time stamp t and a player B with strategy S_B represented in a fuzzy form $F(S_{B,}\ t)$ for every time stamp t and an indication of which $F(S_{A,t})$ is causally related to $F(S_{B,}\ t)$ for each time stamp t.*

A fuzzy dynamic game concept can then be defined as a pair ($F\ (S_{A,}\ t)_i$, $F\ (S_{B,}\ t)_i$) such that

$$F\ (S_{A,}\ t)_i \subseteq F(S_{A,}\ t)$$

$$F(S_{B,}\ t)_i \subseteq F\ (S_{B,}\ t)$$

If every strategy in $F\ (S_{A,}\ t)_i$ is causally related to every strategy in $F(S_{B,}\ t)_i$ then every strategy in $F(S_{A,}\ t)$ that is not in $F(S_{A,}\ t)_i$, is not causally related to at least one strategy in $F(S_{B,}\ t)_i$; and every strategy in $F(S_{B,}\ t)$ that is not in $F(S_{B,}\ t)_I$ is not causally related to at least one strategy in $F(S_{A,}\ t)_i$.

These concepts can be partially ordered by inclusion: if ($F\ (S_{A,}\ t)_i$, $F(S_{B,}\ t)_i$) and ($F(S_{A,}\ t)_j$, $F(S_{B,}\ t)_j$) are concepts, a partial order operator $\leq$ by saying that ($F(S_{A,}\ t)_i$, $F(S_{B,}\ t)_i$) $\leq$ ($F(S_{A,}\ t)_j$, $F(S_{B,}\ t)_j$) whenever $F(S_{A,}\ t)_i \subseteq F(S_{A,}\ t)_j$; and equivalently, ($F(S_{A,}\ t)_i$, $F(S_{B,}\ t)_i$) $\leq$ ($F(S_{A,}\ t)_j$, $F(S_{B,}\ t)_j$) whenever $F(S_{B,}\ t)_j \subseteq F(S_{B,}\ t)_i$. Every pair of concepts in this partial order has a unique greatest lower bound (meet) and a unique least upper bound (join), so this partial order satisfies the axioms defining a lattice. The greatest lower bound of ($F(S_{A,}\ t)_i$, $F(S_{B,}\ t)_i$) and ($F(S_{A,}\ t)_j$, $F(S_{B,}\ t)_j$) is the concept with objects $F(S_{A,}\ t)_i \cap F(S_{A,}\ t)_j$; it has as its attributes the union of $F(S_{B,}\ t)_i$, $F(S_{B,}\ t)_j$, and any additional attributes held by all objects in $F(S_{A,}\ t)_i \cap F(S_{A,}\ t)_j$. The least upper bound of ($F(S_{A,}\ t)_i$, $F(S_{B,}\ t)_i$) and

$(F(S_{A,t})_j, F(S_{B,t})_j)$ is the concept with attributes $F(S_{B,t})_i \cap F(S_{B,t})_j$; it has as its objects the union of $F(S_{A,t})_i$, $F(S_{A,t})_j$, and any additional objects that have all attributes in $F(S_{B,t})_i \cap F(S_{B,t})_j$.

Definition 5.12 *A fuzzy concept lattice is a group* $(F(S_{A,t}), F(S_{B,t}), R_f)$, *where* $F(S_{A,t})$ *is a set linguistic representation of strategy S belonging to player A for time stamp t,* $F(S_{b,t})$ *is a set linguistic representation of strategy S belonging to player B for time stamp t and* R_f *is the binary relationship between* $F(S_{A,t})$ *and* $F(S_{B,t})$.

The fuzzy approach reduces the complexity of concept lattices and simplifies the causal relations C to a binary relation instead of a triple. For some such groups, it is possible to predict the structure of the formal attributes without even considering the polarities. For example, the structure of data derived from a survey often represents rank orders ("high positive", "positive", "neutral", "negative", "high negative"). The lattice for such a rank order can be drawn without considering the actual polarity since it is already represented.

Table 5.3 describes the normal form of a fuzzy lattice game for some players. These are indicated by crosses. An empty cell indicates that the corresponding player does not have the corresponding relation with another player's attribute. It also means that it is not known if relationships exist between attributes.

Definition 5.13 *A table of crosses represents the normal form of fuzzy lattice. The structure used to describe formally these tables of crosses is called a formal game*

context. The context describes the relationship where strategies belong to player B which is causally related to the strategies of the player A.

The extensive form above produces information that PlayerB's *str2* is linked to playerA's *str2* and *str3*. It can easily be read from the extensive form the extent and the intent of each concept by collecting all objects below respectively all attributes above the circle of the given concept. Hence the object concept "$P_A.str2$" has the extent $P_A.str2$ and $P_A.str3$ and the intent $P_B.str2$ and $P_B.str3$. The extent of the top concept is (always) the set of all objects while the intent of it does not contain any attribute (in this context). In other contexts it may occur that the intent of the top concept is not empty, e.g. if a given context the attribute "All players" were added, then the top concept would be the attribute concept of "All players"

The concept lattice in Figure 5.3 has similar structure to that in Figure 5.2. The concept lattice has many more objects but they are all arranged in the pattern of Figure 5.3. Table 5.4 adds an extra dynamic attribute. This frequently happens in an environment where database updates constantly. Figure 5.4 shows how the new dynamic attribute can be represented in a lattice diagram.

This knowledge acquisition procedure has learned the knowledge from an expert. Usually the expert does not know the conceptual structure of his knowledge but is interested in understanding it. The procedure discovers the implications between the interested attributes. If an implication is not valid in the universe, then a counterexample is needed. In the actual context of the counterexamples, the program calculates valid implications and asks the expert if they are valid also in the universe.

PB \ PA	Str.1:HP	Str.2:HN	Str.3:P	Str.4:P
Str.1: HN	x			x
Str.2: N		x	x	
Str.3: HN	x	x	x	
Str.4:P				x
Str.5:P			x	

Table 5.3 Concept diagram

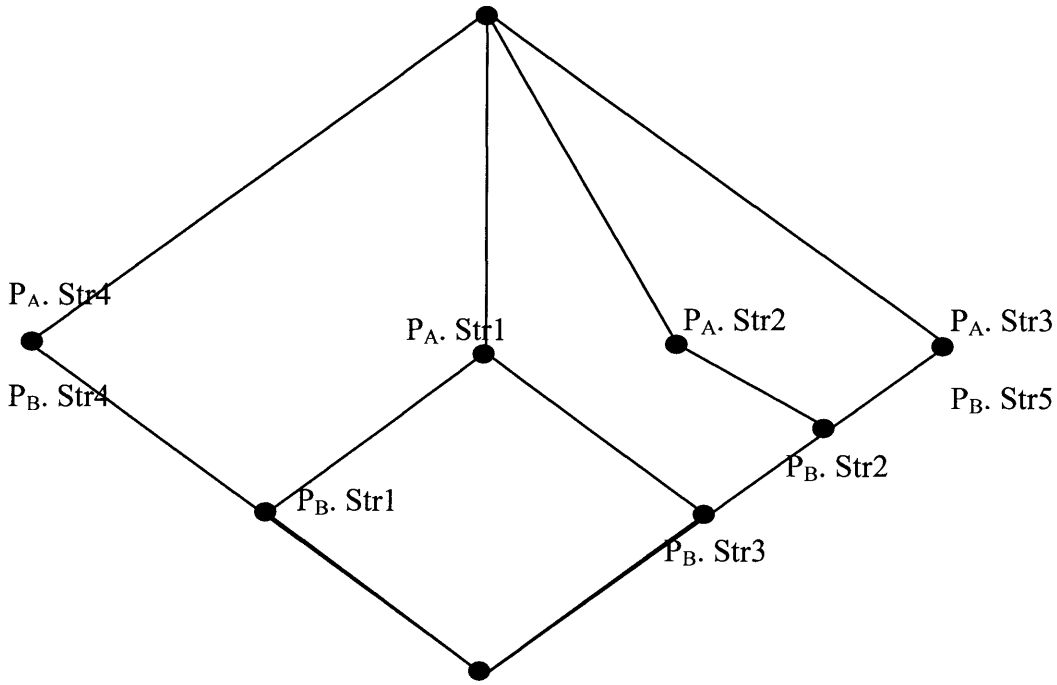


Figure 5.3 Lattice diagram

PB \ PA	Str.1:HP	Str.2:HN	Str.3:P	Str.4:P
Str.1: HN	x			x
Str.2: N		x	x	
Str.3: HN	x	x	x	
Str.4:P				x
Str.5:P			x	
Str.6: HN		x		x

Table 5.4 Altered concept table

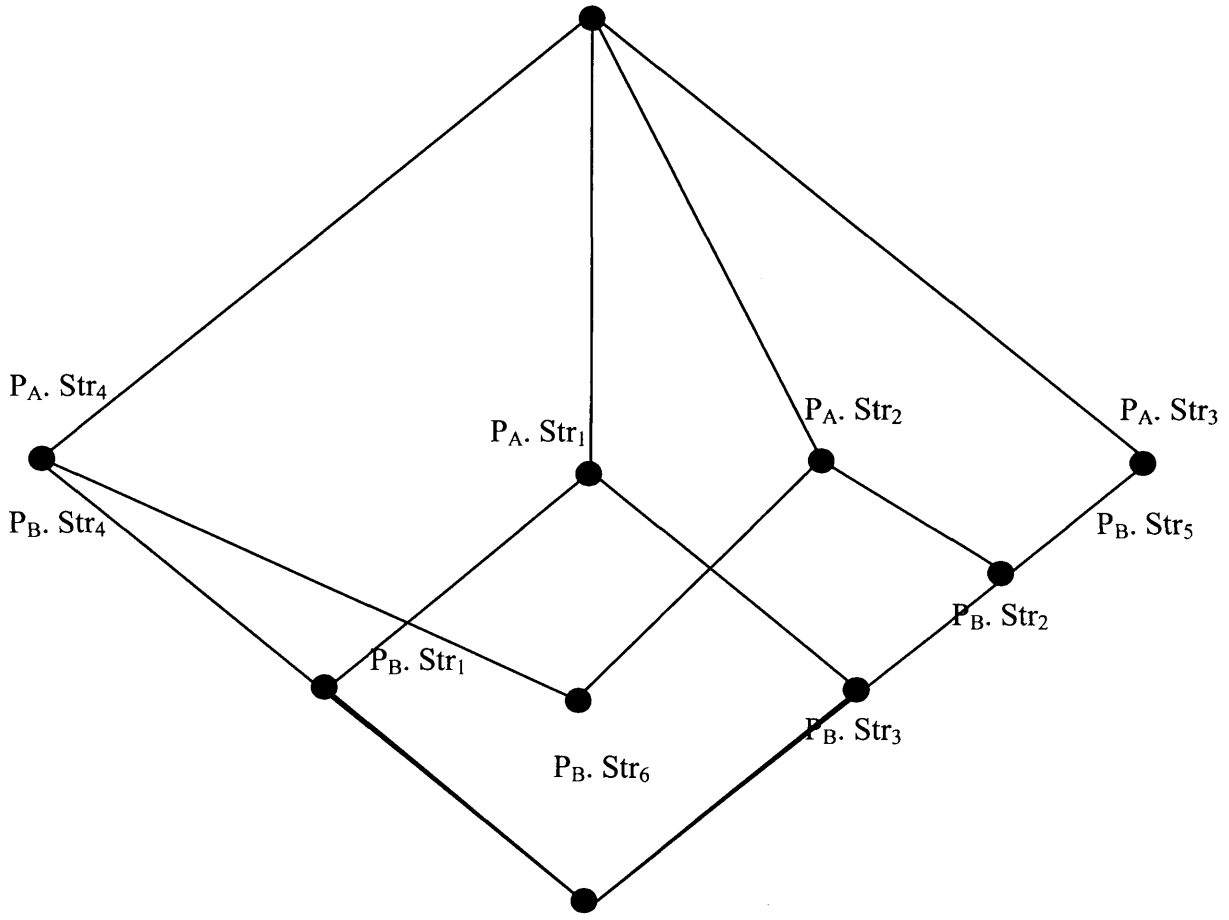


Figure 5.4 Altered lattice diagram

If all valid implications of the actual context are also valid in the universe, then the procedure stops, since in this case the concept lattices of the actual context and the universe are isomorphic. Hence it is not necessary to have full knowledge of all the concept lattices of the universe. This so-called attribute exploration can be dualized by interchanging objects and attributes to explore the conceptual structure between given objects with respect to a certain universe.

5.8 Fuzzy conceptual scaling

Fuzzy conceptual scaling reduces the complexity of concept lattices by dividing them into components based on groupings of related formal attributes. These groups are then separately visualized as lattices. For some such groups, it is possible to predict the structure of the formal attributes without even considering the formal objects.

The general process in conceptual scaling starts with the representation of knowledge in a data table with arbitrary values. Data in Table 5.5 are formally described by *fuzzy game concepts* (F_{p1}, F_{p2}, F, R), where F_{p1} and F_{p2} are sets of strategies, F is a set of 'multi strategies' and I is a ternary relation, $R \subseteq F_{p1} \times F_{p2} \times F$, such that for any $f \in F_{p1}$, $f \in F_{p2}$ there is at most one value w satisfying $(F_{p1}, F_{p2}, F) \in R$. Therefore, a many-valued attribute F can be understood as a partial function of F_{p2} *or* F_{p1}. The central granularity-choosing process in conceptual scaling theory is the construction of a formal context called *conceptual scales*. It represents a contextual language about several layers of strategy representation. This leads to a very useful visualization of multidimensional data in a so-called *nested extensive form.* An example of a scaled extensive form of a game is shown in Figure 5.5 and its normal form is shown in Table 5.5.

PA / PB	Str.1	Str.2	Str.3	Str.4	Str.5	Str.6	Str.7	Str.8	Str.9
Str.1	x	x					x		
Str.2	x	x					x	x	
Str.3	x	x	x				x	x	
Str.4	x		x				x	x	x
Str.5	x	x		x		x			
Str.6	x	x	x	x		x			
Str.7	x		x	x	x				
Str.8	x		x	x		x			

Table 5.5 Complex concepts table

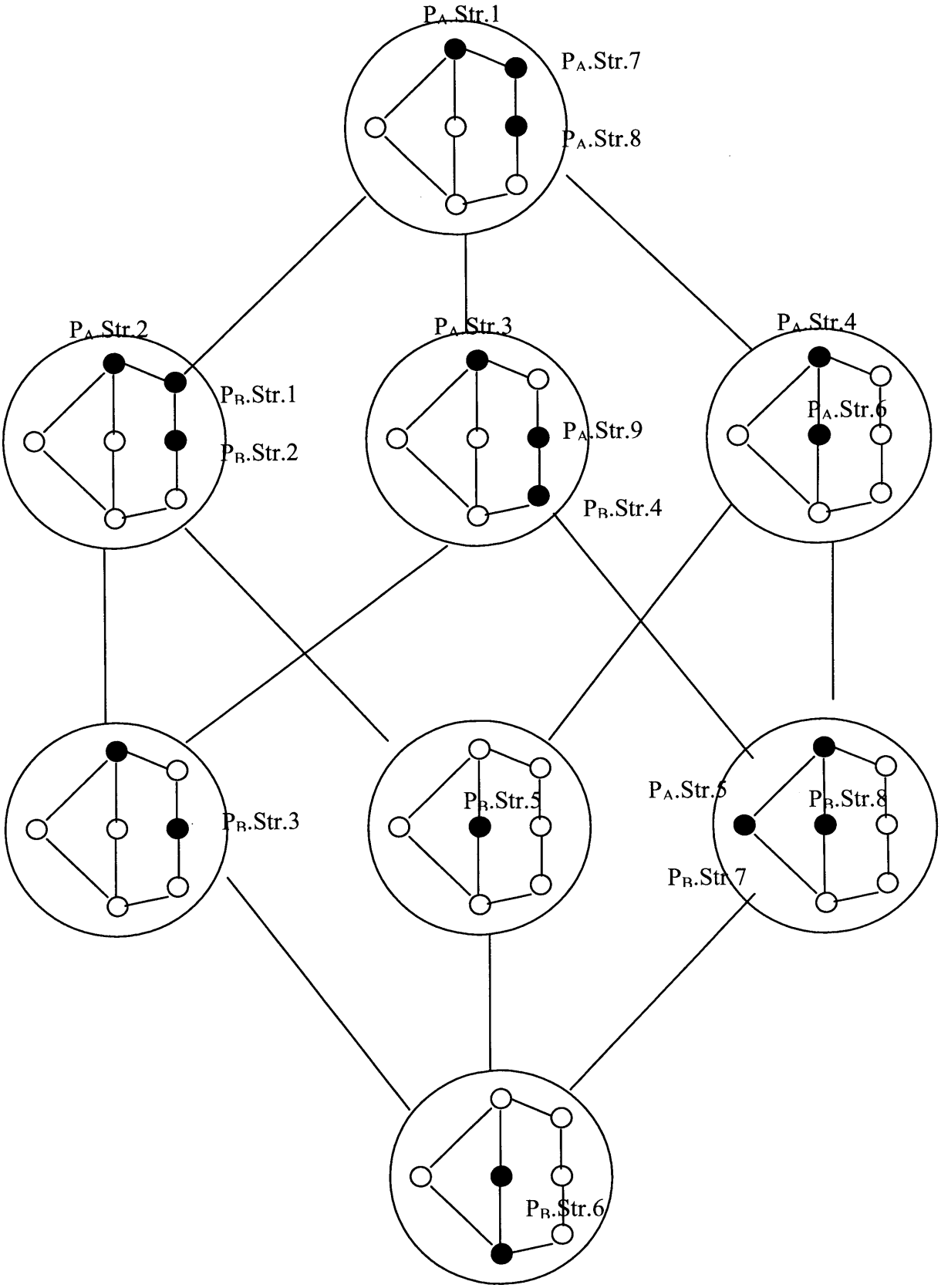


Figure 5.5 Scaled lattice diagram

5.9 Experiment

The experiment uses the database in format of *.pgn (Pgn stands for *Portable Game Notation)* and it can be downloaded from many sites on the internet. All commercial chess database software is built around some database format, which is proprietary to that software. Databases in that format can be exchanged only between owners of the software. A more general exchange of data between owners of incompatible software packages requires a neutral format which is understood by both packages.

The format of the database describes a chess game as such:

1.Nf3 Nf6 2.c4 g6 3.Nc3 Bg7 4.d4 O-O 5.Bf4 d5 6.Qb3 dxc4 7.Qxc4 c6 8.e4 Nbd7 9.Rd1 Nb6 10.Qc5 Bg4 11.Bg5 Na4 12.Qa3 Nxc3 13.bxc3 Nxe4 14.Bxe7 Qb6 15.Bc4 Nxc3 16.Bc5 Rfe8+ 17.Kf1 Be6 18.Bxb6 Bxc4 19.Kg1 Ne2+ 20.Kf1 Nxd4+ 21.Kg1 Ne2+ 22.Kf1 Nc3+ 23.Kg1 axb6 24.Qb4 Ra4 25.Qxb6 Nxd1 26.h3 Rxa2 27.Kh2 Nxf2 28.Re1 Rxe1 29.Qd8+ Bf8 30.Nxe1 Bd5 31.Nf3 Ne4 32.Qb8 b5 33.h4 h5 34.Ne5 Kg7 35.Kg1 Bc5+ 36.Kf1 Ng3+ 37.Ke1 Bb4+ 38.Kd1 Bb3+ 39.Kc1 Ne2+ 40.Kb1 Nc3+ 41.Kc1 Rc2+ 0-1

"1." in "1.Nf3 Nf6" indicating the players' moves in the first round. The first part "Nf3" indicates the movement of the first player and the second part "Nf6"indicates the movement of the second player. N stands for "Knight". "Nf3" indicates that the knight belonging to the first player is moved from its current position to the new position of f3. "Nf6" indicates that the knight belonging to the second player is moved from its current position to the new position of f6. It is shown in Figure 5.6. There are 41 moves in total and the result is that first player loses and second player wins (0-1 at the end).

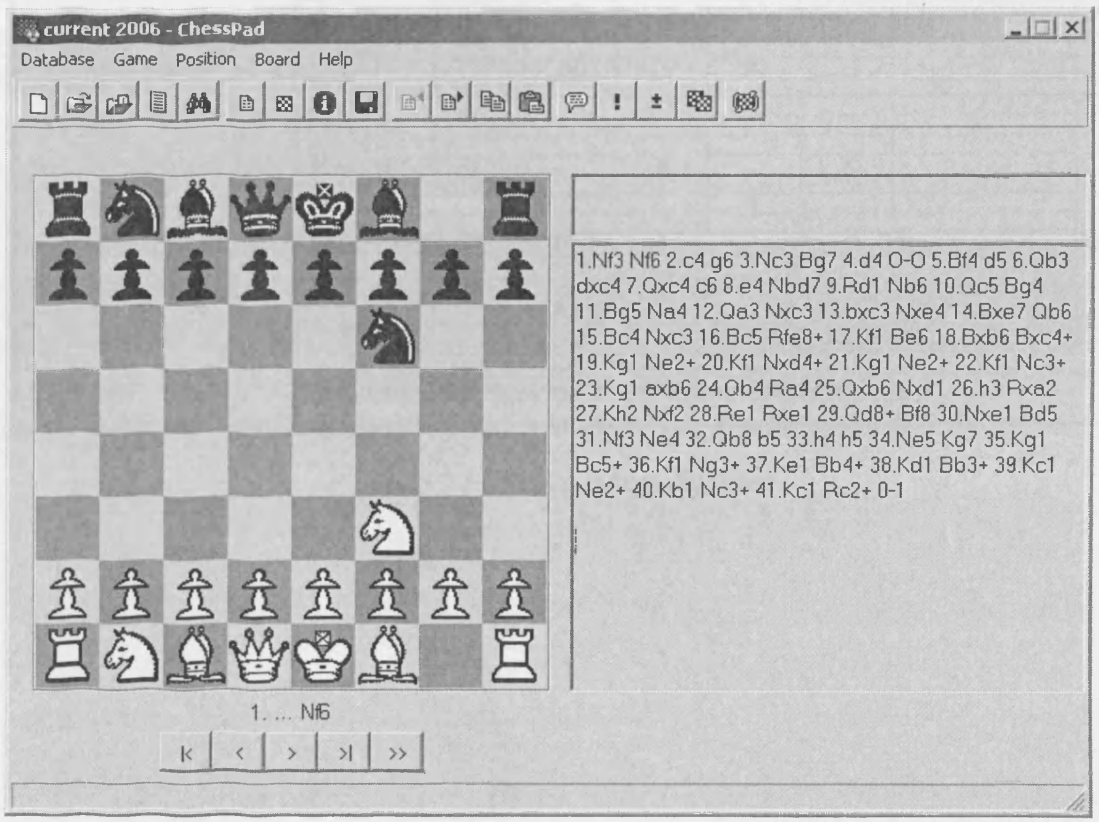


Figure 5.6 Chess game

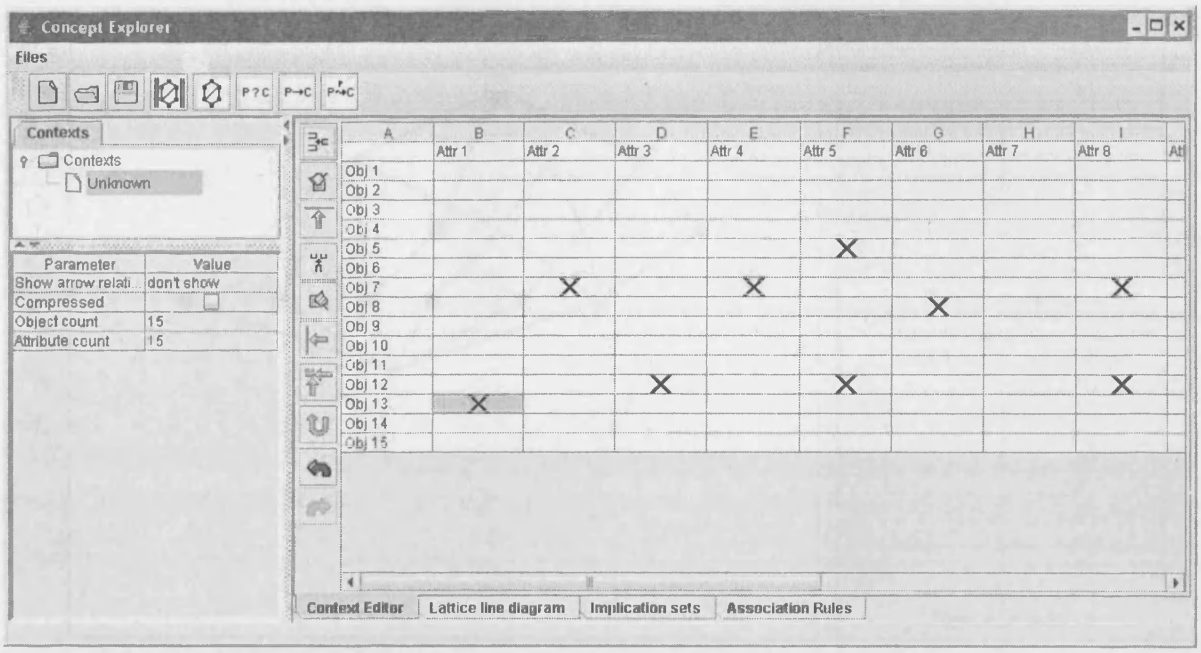


Figure 5.7 Formal concept database

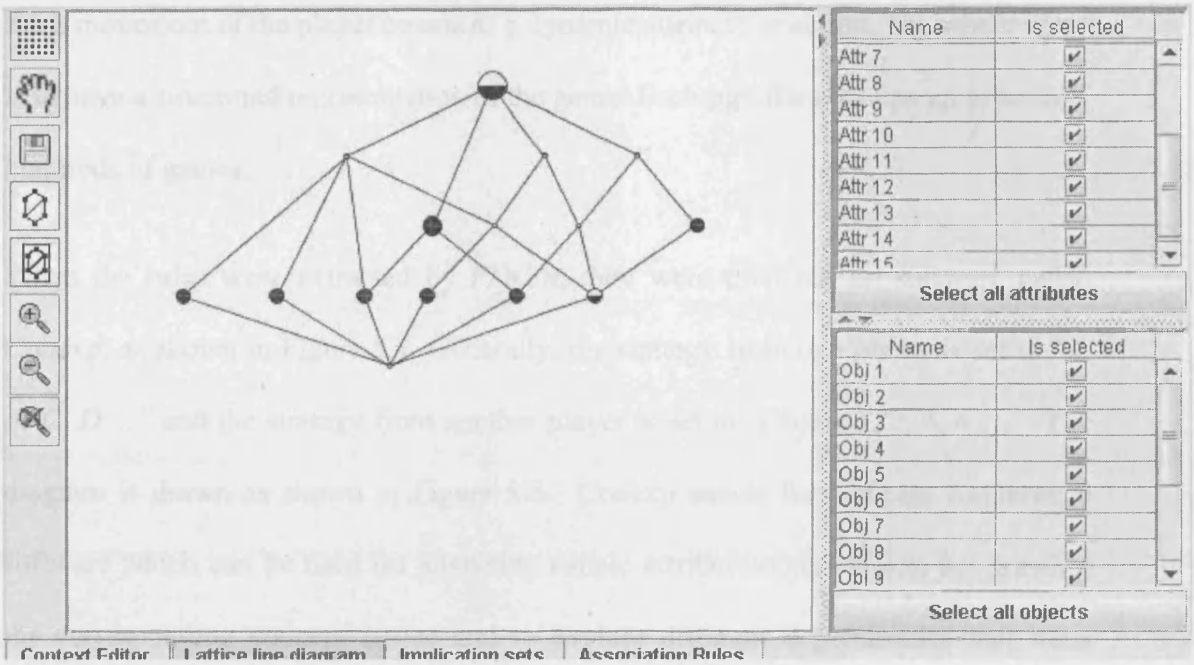


Figure 5.7 Formal lattice

Each movement of the player counts as a dynamic attribute or action. The whole idea is to have a structured representation of the game. Each pgn file contains up to several hundreds of games.

When the rules were extracted by *FDCM*, they were then put on software called Conexp, as shown in Figure 5.7. Generally, the strategy from one player is set to "*A, B, C, D ...*" and the strategy from another player is set to "Object 1, 2, 3, 4 ...". The diagram is drawn as shown in Figure 5.8. Conexp stands for Concept Explorer, a software which can be used for analysing simple attribute-object-tables, for drawing the corresponding concept lattice and to explore different dependencies that exist between attributes.

5.10 Summary

Game Theory itself is well developed. However, there is still some vagueness and lack of clarity about the fundamental assumption of rationality. This chapter casts a view on *Game Theory*, where all players have a pattern of behaviour and these patterns are the cause for the individual decisions.

Furthermore, these patterns can be extracted from historical data. By doing so, the players gain a better insight into the game in which they are involved and can enhance their performance. This chapter also suggests using a *Formal Concepts* lattice to structure the knowledge of a game in order to further improve the player's understanding.

In practice, finiteness of the data implies finiteness of the corresponding derived concept lattices. Clearly, each formal context is a special (realized) linguistic variable

where the characteristic functions of the extents of the attribute concepts are the membership functions. This demonstrates the main idea of how to prove that both theories are equivalent.

This chapter also uses the techniques developed from two previous chapters to identify the underlying causal relations. This assists in the development of new forms of game definition.

Chapter 6 Conclusions and Future Work

This chapter summarises the main contributions of the research and the conclusions of this thesis and provides suggestions for future work.

6.1 Contributions

This research addressed a way to identify dynamic causal relationships by combining *System Dynamics* and *Association Mining*. This combination led to the efficient discovery of hidden dynamic causal rules and policies, which might assist a manager in decision making. The contributions include:

An analysis of the issue of causal analysis. A critical study was carried out of the currently available causal discovery techniques, together with a discussion of their strengths and weaknesses. This led to the design of algorithms suitable for the discovery of the dynamic causal relationships.

A framework for the discovery of simultaneous causality. The new method focuses on the time stamps in existing databases and produced a general framework for dynamic causality discovery involving steps such as data transformation and policy generation.

An algorithm for the discovery of simultaneous dynamic causal relationships. The new algorithm focused on causal relationships happening in the same time frame and employed advanced search heuristics and search space pruning strategies that reduced search time significantly.

An algorithm for the discovery of dynamic causal relationships with time delay and feedback. The proposed algorithm focused on causal relationships happening in different time frames and also employed advanced search heuristics and search space pruning strategies to reduce search time and fuzzy logic to achieve for a more efficient and accurate search.

A simple way to represent the discovered relationships. The representation uses conceptual lattice structure to represent the knowledge discovered by the algorithms in an easily understood way.

6.2 Conclusions

This research has involved three different areas, *System Dynamics*, *Game Theory* and *Association Mining.*

The new *Association Mining* method proposed in this work, *DCM*, combines ideas from *System Dynamics* and *Data Mining. DCM* identifies the dynamic causal patterns

hidden in a database. *DCM* has the ability to process both numerical and categorical data. *DCM* enables efficient management of the dynamic relationships between attributes during the mining process.

The improved version of *DCM* which incorporates "delay" and "feedback concepts" by mining dynamic causal relationship between records with different time stamps, rather than records with the same stamp, has the advantage of being capable of discovering hidden dynamic structures and predicting future system behaviours. The fuzzy-logic version of *DCM* addresses the problem of boundaries between the attributes. This increases the accuracy of the rule sets generated. Rule pruning is carried out based on multiple and separate pruning methods. The strategies of multiple pruning and separate pruning enhance *DCM's* ability to decrease the data processing time of the algorithm.

The application of conceptual lattice is extended to the Game theoretical problem. The use of conceptual lattice in *Game Theory* has enabled the representation of the knowledge extracted via *DCM* in a clearer graphical format. This should facilitate better understanding of the extracted knowledge and provides a natural basis for complexity analysis

6.3 Future Research Directions

A number of aspects of the algorithms developed in this research could be improved. Parallel techniques, such as those based on lattices, can be introduced to *DCM* to reduce the time needed by the algorithm to find dynamic causal rules for a list of objects.

Another possible area of work might be to study different kinds of fuzzy relations that can be necessary in order to apply DCM to the analysis of fuzzy databases, in particular, when different ways to represent uncertainty and imprecision are employed in the database model.

Also, it might be useful to test Bell-shaped and trapezoid membership functions.

The thesis has presented one type of measure based on the support of the rules. Alternative measures may be useful for some applications. An example might be partial completeness measure based on the range of the attributes in the rules.

Fuzzy partitioning may not be the best approach for all types of data. This is because it tends to split adjacent values with high support into separate intervals even though

their behaviour would typically be similar. It may be worth exploring the use of clustering algorithms for partitioning and their relationship to partial completeness.

Finding interesting patterns in the output of association rules or sequential patterns is an area where more work needs to be done. Multiple supports incorporating additional domain knowledge will reduce the number of uninteresting rules even further. One of the challenges is to identify the types of domain knowledge (apart from taxonomies) that are applicable across several domains and that do not require a large amount of effort by the user to create them.

Visualising the output is an interesting research problem, since standard techniques do not scale beyond a few hundred attributes and very simple DCM rules. A problem is developing high-quality, easy-to-generate domain-specific or application-specific, visualisations.

Appendix A Data Sets

Many of the data sets used in this thesis are from the UCI repository of machine learning databases [Blake et al., 1998] and KDD databases [Hettich and Bay, 1999]. These databases were contributed by many researchers, mostly from the field of machine learning and data mining and collected by the machine learning group at the University of California, Irvine. Two of the datasets, MCslom and ASW, are taken from real life. These data sets are described briefly below.

Adult data: This data is donated by Barry Becker from the 1994 Census database, which is a record for an adult in US and their incomes. This data was extracted from the census bureau database with 48842 instances and can be found at http://www.census.gov/ftp/pub/DES/www/welcome.html.

ASW: This data consists of real life data. This dataset contains 65536 attributes of metal manufacturing, with 8 records in each attribute.

Bank data: Includes 600 instances of bank transactions and 12 attributes in each instance.

Cystine Database: This data arises from a large study to examine EEG correlates of genetic predisposition to alcoholism. It contains measurements from 64 electrodes placed on the scalp sampled at 256 Hz (3.9-msec epoch) for 1 second.

Market basket: Classical association data mining, used in WeKa analysis. It consists of 100 different transactions.

Mclosom: Manufacturing database for logistics. 72 time stamps and 50 attributes for 5 different classes.

Weka base: This dataset contains time series sensor readings of the Pioneer-1 mobile robot. The data is time series, multivariate. The few are binary coded 0.0 and 1.0. Two categorical variables are included to delineate the trials within the datasets. The data is broken into "experiences", in which the robot takes action for some period of time and experiences a controlled interaction with its environment.

Appendix B Galois Lattices and Formal Concept Analysis Theory

(*Partially taken from APPLYING FORMAL CONCEPT ANALYSIS TO DOMAIN MODELING IN an Intelligent help system* by Baltasar Fernandez-Manjon, Juan Cigarran,

And

Antonio Navarro and *Complexity of learning in concept lattices from positive and negative examples* by Sergei O. Kuznetsov)

Galois lattices and Formal Concept Analysis (FCA) are a relatively new approach to the mathematics normalization and representation of conceptual knowledge. FCA is a theory of concept formation derived from lattice and ordered set theory that provides a theoretical model for the analysis of conceptual hierarchies. FCA can be used as a tool for the data analysis that makes not only their conceptual structure visible and accessible but also that unfolds different views for the interpretation, making it possible to find patterns and regularities.

B.1. Formal Concepts and Formal Contexts

Let G and M be sets, called the set of objects and attributes, respectively. Let I be a relation $I \subseteq G \times M$ between objects and attributes: for $g \in G$, $m \in M$, gIm holds iff the object g has the attribute m. The triple $K=(G,M,I)$ is called a (*formal*) *context*. If $A \subseteq G$,

$B \subseteq M$ are arbitrary subsets, then the *Galois connection* is given by the following *derivation operators*:

$A' := \{m \in M | gIm \text{ for all } g \in A\}$,

$B' := \{g \in G | gIm \text{ for all } m \in B\}$

The pair (A,B), where $A \subseteq G$, $B \subseteq M$, $A'=B$, and $B'=A$ is called a *(formal) concept (of the context K)* with *extent A* and *intent B* (in this case we have also $A''=A$ and $B''=B$). The set of attributes *B is implied by the set of attributes D*, or *implication* $D \rightarrow B$ holds, if all objects from G that have all attributes from the set D also have all attributes from the set B, i.e., $D' \subseteq B'$. The operation $(\cdot)''$ is a closure operator ,i.e., it is idempotent ($X'''=X''$), extensive ($X \subseteq X''$), and monotone ($X \subseteq Y \Rightarrow X'' \subseteq Y''$). The set of all formal concepts of the context K, as any family of closed sets, forms a lattice, called a *concept lattice* and usually denoted by $\mathfrak{B}(K)$ in FCA literature.

B.2 Concept Ordering and Galois Lattices

The set of all concepts (X, S) of the context (O, A, R) is denoted by $\beta(O, A, R)$. This set is naturally ordered by a generalization-specialization relation (subconcept-superconcept) that produces a hierarchy for the context. The most general concepts are at the top of the hierarchy. These concepts have a smaller intension and a larger extension than any of the more specialized concepts below.

A concept (X1, S1) is a superconcept of the concept (X2, S2) if $X1 \supseteq X2$ which is equivalent to $S1 \subseteq S2$. (X2, S2) is then a subconcept of (X1, S1). Frequently, formal concepts of special interest are those concepts generated by a single attribute or by a

single object from the context. These concepts are called attribute concepts and object concepts respectively; they are useful because an object concept represents the smallest concept with this object in its extension while an attribute concept represent the largest concept with this attribute in its intension. In our example, the concept ({"lpr"}, {"printer", "job", "send"}) is an object concept and it is the most specific concept with the object "lpr" in its extension. On the other hand, the concept ({"lp" }, {"submit", "printer", "request"}) is the attribute concept corresponding to the attribute "submit". This attribute concept is the most general concept with the attribute "submit" in its intension.

The set of all the formal concepts with the generalization-specialization ordering (denoted by $\underline{\mathfrak{B}}(O, A, R) := (\mathfrak{B}(O, A, R), \leq)$) is a complete (Galois) lattice in which infimum and supremum can be described as follows.

$$\bigwedge_{t\in T}(X_t, S_t) = \left(\bigcap_{t\in T} X_t \;,\; \lambda\rho\left(\bigcup_{t\in T} S_t\right)\right)$$
$$\bigvee_{t\in T}(X_t, S_t) = \left(\rho\lambda\left(\bigcup_{t\in T} X_t\right) \;,\; \bigcap_{t\in T} S_t\right)$$

In general, a complete lattice L is isomorphic to $\underline{\mathfrak{B}}(O, A, R)$ if and only if there are mappings

$\gamma : O \to L$ and $\mu : A \to L$ such that γO is supremum-dense in L (i.e. $L = \{\vee X \mid X \subseteq \gamma O\}$), μA is infimum-dense in L (i.e. $L = \{\wedge X \mid X \subseteq \mu A\}$), and $(o,a) \in R \Leftrightarrow \gamma(o) \leq \mu(a)$ for all $o \in O$ and $a \in A$. In particular, $L \cong \underline{\mathfrak{B}}(L, L, \leq)$.

A concept lattice (a concept hierarchy) can be represented graphically using line (or Hasse) diagrams. These structures are composed of nodes and links. Each node

represents a concept with its associated intensional description. The links connecting nodes represent the subconcept-superconcep relation among them. This relation indicates that the parent's extension is a superset of each child's intension. A node covers all of the instances covered by the union of its descendents, making the concept hierarchy a subgraph of the partial ordering by generality. More abstract or general nodes occur higher in the hierarchy, whereas more specific ones occur at lower levels. Object concepts are named with the specific object that produces the concept while attribute concepts are labeled with the specific attribute that generates the concept.

B.3 Conceptual Scaling

In many situations the data about a domain are complex and it is not obvious how to apply FCA to analyze or structure domain information. For example, there are domains where objects are described by complex attributes that can have many values, and where even the values of some attributes are unknown. With the help of an expert and through an interpretation process called *conceptual scaling* a complex description can be transformed into a suitable formal context. In the interpretation process these complex attributes are considered as objects that are again described by new attributes, the so-called scale attributes. This transformation is not unique allowing different interpretations and views of the domain data. For example, this allows the visualization of the domain information with different levels of detail or granularity be means of nested line diagrams. Using those diagrams it is possible to begin doing a rough analysis of the data and then to focus on some interesting parts to produce a very fine presentation.

Appendix C Basic Elements of Game Theory

(partially taken from Wikipedia and Ross, Don 1997/ 2001 - in *Stanford Encyclopedia of Philosophy)*

C.1 Utility

An agent is, by definition, an entity with *goals and preferences*. Game theorists, like economists and philosophers studying rational decision-making, describe these by means of an abstract concept called *utility*. This refers to the amount of satisfaction an agent derives from an object or an event, so that an agent who, for example, adores the taste of pickles would be said to associate high utility with them, while an agent who can take or leave them derives a lower level of utility from them. Examples of this kind suggest that 'utility' denotes a measure of subjective *psychological* fulfillment and this is indeed how the concept was generally (though not always) interpreted prior to the 1930s.

C.2 Games and Information

All situations in which at least one agent can only act to maximize his utility through anticipating the responses to his actions by one or more other agents is called a *game*. Agents involved in games are referred to as *players*. If all agents have optimal actions regardless of what the others do, as in purely parametric situations or conditions of monopoly or perfect.

Each player in a game faces a choice among two or more possible *strategies*. A strategy is a predetermined 'programme of play' that tells her what actions to take in response to *every possible strategy other players might use*. The significance of the italicized phrase here will become clear when we take up some sample games below.

A crucial aspect of the specification of a game involves the information that players have when they choose strategies. The simplest games (from the perspective of logical structure) are those in which agents have *perfect information*, meaning that at every point where each agent's strategy tells her to take an action, she knows everything that has happened in the game up to that point. A board-game of sequential moves in which both players watch all the action (and know the rules in common), such as chess, is an instance of such a game. By contrast, the example of the bridge-crossing game from Section 1 above illustrates a game of *imperfect information*, since the fugitive must choose a bridge to cross without knowing the bridge at which the pursuer has chosen to wait and the pursuer similarly makes her decision in ignorance of the actions of her quarry.

The normal (or strategic form) game is usually represented by a matrix which shows the players, strategies and payoffs (see the example to the right). More generally it can be represented by any function that associates a payoff for each player with every possible combination of actions. In the accompanying example there are two players; one chooses the row and the other chooses the column. Each player has two strategies, which are specified by the number of rows and the number of columns. The payoffs are provided in the interior. The first number is the payoff received by the row player (Player 1 in our example); the second is the payoff for the column player (Player 2 in

	Player 2 chooses *Left*	Player 2 chooses *Right*
Player 1 chooses *Up*	**4, 3**	**–1, –1**
Player 1 chooses *Down*	**0, 0**	**3, 4**

Normal form or payoff matrix of a 2-player, 2-strategy game

Table C.1 Normal form

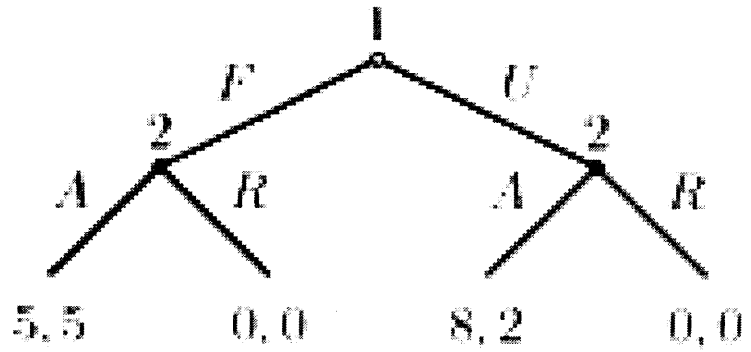


Figure C.1 An extensive form game

our example). Suppose that Player 1 plays *Up* and that Player 2 plays *Left*. Then Player 1 gets a payoff of 4 and Player 2 gets 3.

When a game is presented in normal form, it is presumed that each player acts simultaneously or, at least, without knowing the actions of the other. If players have some information about the choices of other players, the game is usually presented in extensive form.

The extensive form can be used to formalize games with some important order. Games here are often presented as trees (as pictured to the left). Here, each vertex (or node) represents a point of choice for a player. The player is specified by a number listed by the vertex. The lines out of the vertex represent a possible action for that player. The payoffs are specified at the bottom of the tree.

In the game pictured here, there are two players. *Player 1* moves first and chooses either *F* or *U*. *Player 2* sees *Player 1*'s move and then chooses *A* or *R*. Suppose that *Player 1* chooses *U* and then *Player 2* chooses *A*, then *Player 1* gets 8 and *Player 2* gets 2.

The extensive form can also capture simultaneous-move games and games with incomplete information. Either a dotted line or circle is drawn around two different vertices to represent them as being part of the same information set (i.e., the players do not know at which point they are).

Reference

Agrawal, R., Imielinski, T. &Swami, A. (1993) Mining Associations between Sets of Items in Massive Databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207-216, Washington D.C.

Agrawal, R. &Srikant, R. (1994) Fast Algorithms for Mining Association Rules. *Proceedings of the Twentieth International Conference on Very Large Databases*, pp 487-499, Santiago, Chile.

Agrawal, R. &Shafer, J. C. (1996) Parallel Mining of Association Rules. *IEEE Transactions on Knowledge &Data Engineering,* Vol. 8, No. 6. pp 962-969.

An, L., Jeng, J.& Gerede C. E. (2006) On Exploiting System Dynamics Modeling to Identify Service Requirements, *IEEE International Conference on Services Computing 06*,Vol 3, Issue 10, September, pp 277 – 280.

Axelrod, R. (1984) *The Evolution of Cooperation.* New York: Basic Books.

Bain, M. &Muggleton, S. (1994). Learning Optimal Chess Strategies. *Machine Intelligence 13* (eds. K. Furukawa &D. Michie), pp. 291-309. Oxford University Press, Oxford, UK. ISBN 0198538502.

Blanchard, B. S. &Fabrycky, W. J. (1998) Systems *engineering &analysis.* Upper Saddle River: Prentice Hall, New Jercey, USA.

Blake, C. L., Keogh, E. &Merz, C. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.htm

Boulicaut, J. F. &Bykowski, A. (2000) Frequent closures as a concise representation for binary data mining. In *PADKK '00: Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery &Data Mining, Current Issues &New Applications*, pp 62-73, Springer-Verlag, London, UK.

Brin, S., Motwani, R. &Silverstein, C. (1997) Beyond Market Baskets: Generalizing Association Rules to Correlations. *Proceedings of the ACM SIGMOD International Conference on Management of Data.* May, pp 265-276, Tuscon, Arizona.

Bykowski, A &Rigotti, R. (2001) A condensed representation to find frequent patterns. In *PODS '01: Proceedings of the twentieth ACM SIGMOD- SIGACT-SIGART symposium on Principles of database systems*, pp 267-273, ACM Press. New York, NY, USA.

Calders, T. &Goethals, B. (2002) Mining all non-derivable frequent itemsets. In *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining &Knowledge Discovery*, pp 74-85, Springer-Verlag. , London, UK.

Cass. S.(2002). Mind games. *IEEE Spectrum*, Vol.39. Issue 12. pp 40–44.

Cengiz, I.(1997) Mining Association Rules, *Bilkent University, Department of Computer Engineering &Information Sciences, Ankara, Turkey*, (URL: http://www.cs.bilkent.edu.tr/~icegiz/datamone/mining.html, last viewed April, 2008)

Chattratichat, J., Darlington, J., Ghanem, M. (1997) Large Scale Data Mining: Challenges &Responses, *Proceedings of the 3rd International Conference on Knowledge Discovery &Data Mining*, pp. 143-146, August.

Chen, M. S., J. W. Han &Yu, P. S. (1996) Data Mining: An Overview from a Database Perspective, *IEEE Transactions on Knowledge &Data Engineering,* Vol. 8, No. 6, pp. 866-883.

Chen, Y., Huang T. C. &Chang S.(2008) A novel approach for discovering retail knowledge with price information from transaction databases, Expert *Systems with Applications*, Vol 34, Issue 4, May, pp 2350-2359.

Cheung, D. W., Han, J. W., Ng, V. T., Fu, A. W. &Fu, Y. J., (1996) Efficient Mining of Association Rules in Distributed Databases, *IEEE Transactions on Knowledge &Data Engineering,* Vol. 8, No. 6, pp 911-922, December.

Churchman, C. W.(1968) *The Systems Approach*, Dell Publishing Co. Inc, New York, USA.

Cong , S., Han , J. Hoeflinger J. & Padua D. (2005) A sampling-based framework for

parallel data mining, *Proceedings of the tenth ACM SIGPLAN symposium on Principles &practice of parallel programming*, June 15-17, Chicago, USA

Coyle, R. G. (1977) *Management System Dynamics*, John Wiley & Sons, London.

Clarke, M. R. B. (1977) A Quantitative Study of King &Pawn against King. *Advances in Computer Chess*, No.1, Edinburgh University Press, pp 108-110. Edinburgh.

Dowe, P., (2000). *Physical Causation.* Cambridge University Press, Cambridge, Uk

Drew, D. R.(1995) System Dynamics Modeling &Application, *Course notes, ENGR 5104*, Virginia Tech.

Durlauf, S., &Young, H. P.(2001). *Social Dynamics*, MIT Press, Cambridge, MA.

Eagle, R. F. &Granger, C. W. J. (1988) Co-integration &Error Correction: Representation, Estimation, &Testing, *Econometrica*, vol. 55, issue 2, pp 251-76.

Esposito, F., Malerba, D., Ripaand, V. and Semeraro, G.,(1997) Discovering Causal Rules in Relational Databases. *Applied Artificial Intelligence* Vol.11, No.1, pp 71-84. (http://www.di.uniba.it/~malerba/publications/AAI97-1.pdf, last time checked, april, 2008)

Fay, G., (1973) *An algorithm for finite Galois connections*. Technical report, Institute for Industrial Economy, Budapest.

Fayyad, U. M., Shapiro, G. P.&Smyth, P. (1996) From Data Mining to knowledge Discovery: An Overview, *Advances in Knowledge Discovery &Data Mining,* Edited by Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padraic Smyth, &Ramasamy Uthurusamy, AAAI Press, , pp 1-34.

Ford, D. &Sterman, J. D.(1998) Expert Knowledge Elicitation for Improving Mental &Formal Models. *System Dynamics Review*, Wiley, Vol. 14, No. 4, p. 309-340.

Forrester, J. W.(1968) *Principles of Systems*, Wright-Allen Press Inc., Cambridge, MA.

Forrester, J. W., (1994) System dynamics, systems thinking, &soft OR. *System Dynamics Review,* Vol.10, pp 245–256.

Fu, A., Wong, M. H., Sze, S. C., Wong, W. C., Wong, W. L. &Yu, W. K., (1998) Finding Fuzzy Sets for the Mining of Fuzzy Association Rules for Numerical Attributes. *IDEAL 98, 1st International Symposium on Intelligent Data Engineering &Learning,* pp 263-268.

Furnkranz. J. (1995) A Brief Introduction to Knowledge Discovery in Databases. *OGAI-Journal*, Vol. 4, No. 4, pp14–17.

Ganter, B. (1984). *Two basic algorithms in concept analysis*. FB4 Preprint 831,TH Darmstadt.

Han. J. W. &Fu, Y., (1995) Discovery of Multiple-Level Association Rules from Large Databases, *Proceedings of the 21st International Conference on Very Large Databases*, Zurich, Swizerland, pp. 420-431,.

Han, E. H., Karypis, G. &Kumar, V., (1997) Scalable Parallel Data Mining For Association Rules, *Proceedings of the ACM SIGMOD Conference*, pp. 277-288.

Hettich, S. &Bay, S. D. The UCI KDD archive, 1999. [http://kdd.ics.uci.edu/, last time checked, november , 2007]

Hidber, C. (1999) Online Association Rule Mining, *Proceedings ACM SIGMOD International Conference on Management of Data*, pp.145-156, Philadephia, Pennsylvania. June.

Holt, J.D. & Chung, S. M. (2007) Parallel mining of association rules from text databases, *The Journal of Supercomputing*, v.39 n.3, March, pp 273-299,

Hong, T. P., Kuo, C. S., Chi, S. C. & Wang, S. L. (2000) Mining Fuzzy Rules from Quantitative Data Based on the AprioriTid Algorithm. *ACM SAC 2000*, pp.490-493, Como, Italy.

Houtsma, M. A. W. & Swami, A. (1995) Set-Oriented Mining in Relational Databases, *Data &Knowledge Engineering,* Vol.17 , Issue 3, Pp:245 - 262 .

Hsu, F., Anantharaman, T., Campbell, M., & Nowatzyk, A. (1990) A Grandmaster Chess Machine. *Scientific American*, 263, Vol.4, pp 44--50.

Hume, D. (1740). *A Treatise of Human Nature* (1967, edition). Oxford University Press, Oxford.

Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H. &A. Verkamo, I.(1994) Finding Interesting Rules From Large Sets of Discovered Association Rules", *Proceedings of the Third International Conference on Information &Knowledge Management (CIKM'94)*, pp. 401-407, November.

Kleindl B. (1999) Game theoretic perspective on market-oriented versus innovative strategic choice. *Journal of Strategic Marketing,* vol. 7, No. 4, pp 265–274.

Kondratenko, Y. (2003) *How Can Systems Thinking Be Used to Support Rural Development Decisions in Latvia*, Master thesis. University of bergen (http://www.lumes.lu.se/database/Alumni/02.03/theses/kondratenko_yuri.pdf, last checked April. 2008)

Kryszkiewicz, M., (2001) Concise representation of frequent patterns based on disjunction-free generators. *Proceedings of the 2001 IEEE International Conference on Data Mining*, pp 305-312, IEEE Computer Society. Washington, DC, USA.

Kuok, C. M., Fu, A. &Wong, M. H., (1998). Mining Fuzzy Association Rules in Databases. *SIGMOD* Vol. 27, No.1, pp. 41-46.

Lewis, D. (1973). Causation, *Journal of Philosophy*, Vol. 70, pp.556-567.

Li C., Qi J.&Shu, H. (2008) A System Dynamics Model for SMS Governance, Research &Practical Issues of Enterprise Information Systems II Vol 255/2008 pp1585-1595.

Liu, B., Hsu W. &Ma, Y.(1999) Mining Association Rules with Multiple Supports, *ACM SIGKDD International Conference on Knowledge Discovery &Data Mining (KDD-99)*, August 15-18, SanDiego, CA, USA. pp 245-25.5

Mohammad E.& Osmar R. Z.(2006) Parallel Bifold: Large-scale parallel pattern mining with constraints, *Distributed &Parallel Databases*, Volum 20, Issue.3, November. pp 225-243,

Meo, R., Psaila, G. & Ceri, S. (1996) A New Sql-Like Operator for Mining Association Rules, *Proceedings of the 22nd International Conference on Very Large Databases*, Mumbai, India, pp 122-133,.

Meyer, M. A. & Booker, J. M. (1991) *Eliciting and analyzing expert judgement: A practical guide, knowledge-based systems*, Vol. 5, Academic Press, London.

Michalski, R.S. (1969) On the Quasi-Minimal Solution of the Covering Problem, *Proceedings of the International Symposium on Information Processing*, Vol. 3 (Switching Circuits), Bled, Yugoslavia, pp. 125-128

Michalski, R. S. & Negri, P., (1977) An Experiment on Inductive Learning in Chess End Games, *Machine Representation of Knowledge, MACHINE INTELLIGENCE 8,* Ellis Horwood, pp. 175-192,

Mohapatra P. K. J., Bora M. & Mandal, P. (1994), *Introduction to System Dynamics Modelling*, University Press Ltd., India.

Moustakidesa G. V. & Verykios, V. S. (2008) A MaxMin approach for hiding frequent itemsets, *Data & Knowledge Engineering*, Vol 65, Issue 1, April, pp 75-89

Muggleton, S. (1990) *Inductive Acquisition of Expert Knowledge.* Glasgow, Scotland; Reading, Mass.: Turing Institute Press ;Wokingham, England; Addison-Wesley.

Mure, G. R. G (2007), *Posterior Analytics, (A translation of the works written by Aristotle),* The University of Adelaide. (http://etext.library.adelaide.edu.au/a/aristotle/a8poa/book2.html, last viewed April 2008)

Nanopoulos, A. Papadopoulosa, A. N. &Manolopoulosa, Y. (20070Mining association rules in very large clustered domains, Information Systems, Vol 32, Issue 5, July, pp 649-669.

Nash, J., (1951) Non-cooperative Games. *Annals of Mathematics Journal*, Vol. 54, pp. 286-295.

Nuhoglu, H. & Nuhoglu M.(2007) System Dynamics Approach In Science &Technology Education, *Journal of TURKISH SCIENCE EDUCATION*, Vol 4, Issue 2, September. Pp 234-237.

Nunn. J. (1994) Extracting information from endgame databases. In H. J. van den Herik, I. S. Herschberg, &J. W. H. M. Uiterwijk, editors, *Advances in Computer Chess,* Vol.7, pp 19-34. University of Limburg.

O'Connor, J. (1997). *The Art of Systems Thinking: Essential Skills for Creativity &Problem Solving*, An Imprint of HarperCollins Publishers, Thorsons, London.

Park, J. S., Chen, M. S. &Yu, P. S. (1995) An Effective Hash Based Algorithm for Mining Association Rules, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, California, May , pp. 175-186

Paterson, A. (1983) *An attempt to use CLUSTER to synthesise humanly intelligible subproblems for the KPK chess endgame.* Technical Report UIUCDCSR -83-1156, University of Illinois, Urbana, IL.

.

Pasquier, N., Bastide, Y., Taouil, R. &Lakhal. L. (2001) Closed set based discovery of small covers for association rules. *Networking &Information Systems*, Vol 3, No 2, pp 349-377, Hermès Science, Paris, June.

Pearl, J., (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Pearl, J. (1994) *Three statistical puzzles*. Technical Report R-217, U.C.L.A., February.

Pearl, J. (1997). From a Century of Statistics to the Age of Causation. *Computing Science &Statistics*, Vol. 29, No. 2, pp.415-423.

Pearl, J., (2000). *Causality: Models, Reasoning, &Inference*. Cambridge University Press.

Quinlan. J. R. (1983) Learning efficient classification procedures &their application to chess end games. *In R. S. Michalski, J. G. Carbonell, &T. M. Mitchell, editors, Machine Learning: An Artificial Intelligence Approach,* Tioga, Palo Alto, pp 463-482.

Raju, K. V. S. V. N. &Majumdar, A. K., (1988). Fuzzy Functional Dependencies &Lossless Join Decomposition of Fuzzy Relational Database Systems. *ACM Transactions on Database Systems*, Vol. 13, No. 2, pp.129-166.

Richmond, B.(1993) Systems Thinking: Critical Thinking Skills for the 1990's &Beyond, *System Dynamics Review* Vol.9, No.2, Summer, pp113-133.

Richmond, B. (2000). *The "thinking" in systems thinking: Seven essential skills.* Waltham, Massachusetts: Pegasus Communications.

Roycroft, A. J. (1988) Expert against oracle. *J. E. Hayes, D. Michie, &J. Richards (Eds.), Machine Intelligence,* No. 11, pp. 347–373. Oxford University Press, Oxford, UK.

Savasere, A., Omiecinski, E.& Navathe, S. B. (1995) An Efficient Algorithm for Mining Association Rules in Large Databases, *Proceedings of the 21nd International Conference on Very Large Databases*, Zurich, Swizerland, pp. 432-444

Senge, Peter M. (1990) *The Fifth Discipline.* Doubleday, New York

Simon H. (1957) *A behavioral model of rational choice.* R&Corporation, Santa Monica, California.

Shapiro A. D. & Niblett, T. (1982) Automatic induction of classification rules for a chess endgame. *In M. R. B. Clarke, editor, Advances in Computer Chess,* Vol. 3. Pergamon, Oxford, pp 73-92

Shintani, T. &Kitsuregawa, M. (1998) Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy, *Proceedings ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, USA, June 2-4, pp 25-36.

Spirites, P., Glymour, C., & Scheines, R. (1993) *Causation, prediction, and search* [Lecture Notes in Statistics 81]. Springer-Verlag. New York.

Spruill, N., Kenney, C. &Kaplan, L. (2001). Community development &systems thinking: theory &practice. National Civic Review, 90, pp105-117.

Srikant, R. &Agrawal, R. (1995) Mining Generalized Association Rules. *Proc. of the 21st Int'l Conference on Very Large Databases*, Zurich, Switzerland (http:zSzzSzwww.almaden.ibm.comzSzcszSzpeoplezSzsrikantzSzpaperszSzvldb95_rj .pdf/srikant95mining.pdf, last viewed April, 2008)

Srikant, R. & Agrawal, R. (1996) Mining Quantitative Association Rules in Large Relational Tables. *Proc. of the ACM-SIGMOD 1996 Conference on Management of Data*, Montreal, Canada.

Sterman, J. D. (1994) Learning in &about Complex Systems. *System Dynamics Review*, Vol. 10, No. 2-3, pp 291-330.

Sterman, J. D. (2000) *Business Dynamics: Systems Thinking &Modelling for a Complex World*, Irwin McGraw-Hill, Boston, MA.

Stumme, G. (1999) Conceptual knowledge discovery with frequent concept lattices. FB4- Preprint 2043, TU Darmstadt. (http://www.aifb.uni-karlsruhe.de/WBS/gst/papers/1999/P2043.ps Last checked 2007 Oct.)

Suppes, P., (1970). *A Probabilistic Theory of Causality*, North Holland.

Tesauro, G. (1995) Temporal difference learning &TD-Gammon. *Communications of the ACM,* vol .38, issue 3, March, pp 58-68.

Toivonen, H. (1996) Sampling Large Databases for Association Rules, *Proceedings of the 22nd International Conference on Very Large Databases*, Mumbai, India, pp. 134-145

Tveit A. & Tveit. G. B. (2002) Game Usage Mining: Information Gathering for Knowledge Discovery in Massive Multiplayer Games. *Proceedings of the 3rd International Conference on Internet Computing*, CSREA Press, Las Vegas, USA, June, pp 636-642

Tveit, A., Rein, Ø., Iversen, J. V. & Matskin. M. (2003) Scalable Agent-Based Simulation of Players in Massively Multiplayer Online Games. In *Proceedings of the 8th Scandinavian Conference on Artificial Intelligence (SCAI'03)*, Frontiers in Artificial Intelligence &Applications, IOS Press, Bergen, Norway, November.

Vivekananda, S. & Vashishta, Y.(1902) *Complete Works.* (http://www.ramakrishnavivekananda.info/vivekananda/complete_works.htm, last viewed april, 2008)

Von Neumann, J., &Morgenstern, O. (1965) *Theory of games &economic behaviour* (3rd ed.). Science Editions, John. Wiley, New York.

Wille, R. (1982) Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival, I. (ed.) Ordered Sets.445-470. Dordrecht-Boston, Reidel.

Weill. J. C. (1994) How hard is the correct coding of an easy endgame. *In H. J. van den Herik, I. S. Herschberg, &J. W. H. M. Uiterwijk, editors, Advances in Computer Chess,* Vol. 7, pp 163-176, University of Limburg.

Zadeh, L.A. (1975). The Concept of Linguistic Variable &its Application to Approximate Reasoning. *Information Sciences*, vol 8, pp. 199-249, 1975 (part I), vol 8, pp. 301-357, 1975 (part II), vol 9, pp. 43-80, 1976 (part III).

Zaki, M. J. & Hsiao, C. J. (1999) *Charm: An efficient algorithm for closed association rule mining,* Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180.

Zaki, M. J., Parthasarathy, S., Ogihara, M. &Li, W. (1997) New Parallel Algorithms for Fast Discovery of Association Rules, *Data Mining &Knowledge Discovery*, Vol. 1, No. 4, pp 343-373.

Zheng, L. (2007) A System Dynamics Based Study of Policies on Reducing Energy Use &Energy Expense for Chinese Steel Industry, master thesis, University of Bergen (https://bora.uib.no/bitstream/1956/2363/1/Masterthesis_Longbin.pdf, last checked April, 2008)