

# A Formalism for Visual Query Interface Design

by

Jiwen Huo

A thesis  
presented to the University of Waterloo  
in fulfillment of the  
thesis requirement for the degree of  
Doctor of Philosophy  
in  
Computer Science

Waterloo, Ontario, Canada, 2008

© Jiwen Huo 2008

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Abstract

The massive volumes and the huge variety of large knowledge bases make information exploration and analysis difficult. An important activity is data filtering and selection, in which both querying and visualization play important roles. Interfaces for data exploration environments normally include both, integrating them as tightly as possible.

But many features of information exploration environments, such as visual representation of queries, visualization of query results, interactive data selection from visualizations, have only been studied separately. The intrinsic connections between them have not been described formally. The lack of formal descriptions inhibits the development of techniques that produce new representations for queries, and natural integration of visual query specification with query result visualization.

This thesis describes a formalism that describes the basic components of information exploration and their relationships in information exploration environments. The key aspect of the formalism is that it unifies querying and visualization within a single framework, which provides a foundation for designing and analysing visual query interfaces.

Various innovative designs of visual query representations can be derived from the formalism. Simply comparing them with existing ones is not enough, it is more important to discover why one visual representation is better or worse than another. To do this it is necessary to understand users' cognitive activities, and to know how these cognitive activities are enhanced or inhibited by different presentations of a query so that novel interfaces can be created and improved based on user testing.

This thesis presents a new experimental methodology for evaluating query representations, which uses stimulus onset asynchrony to separate different aspects of query comprehension. This methodology was used to evaluate a new visual query representation based on Karnaugh maps, and showing that there are two qualitatively different approaches to comprehension: deductive and inductive. The Karnaugh map representation scales extremely well with query complexity, and the experiment shows that its good scaling properties occur because it strongly facilitates inductive comprehension.

## Acknowledgements

There are many people whose support, guidance, and encouragement have made this thesis possible.

I am especially grateful to my advisor, William Cowan, for his advice, support, and guidance throughout my study in Waterloo. He has taught me so much and been a great mentor.

In addition, I would like to thank my reading committee, Christopher Healey, Catherine Burns, Grant Weddell, and Ric Holt, for the time and effort they spent to read and comment on my work.

Also, I would like to thank all of people who participated in my experiments for their time and attention. Special acknowledgement goes to Yi Lin, Jie Xu, and Yinbin Liu, for giving me feedbacks and suggestions on my pilot study to improve the experiment.

Most of all, I would like to thank my parents and my husband Chong for always supporting and encouraging me. Without their love and support, the completion of this thesis would not be possible. Finally I'd like to thank my son Ray for the joy he brings to my life.

# Contents

<b>List of Tables</b>	<b>1</b>
<b>List of Figures</b>	<b>1</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Contributions . . . . .	2
1.2 User Interface Issues in Information Exploration . . . . .	3
1.3 A Formalism for Visual Query Interface . . . . .	4
1.4 Issues in Evaluating Query Languages . . . . .	5
1.5 The Experiment Methodology . . . . .	6
1.6 Thesis Organization . . . . .	7
<b>2 Related Work on Visual Information Exploration</b>	<b>9</b>
2.1 Information Exploration . . . . .	9
2.1.1 Information Exploration Process . . . . .	9
2.1.2 Visual Information Exploration . . . . .	11
2.2 Information Visualization . . . . .	13
2.2.1 The Process of Visualization . . . . .	13
2.2.2 Data Model . . . . .	14
2.2.3 Visual Mapping . . . . .	15
2.2.4 Data Selection and Visualization . . . . .	16
2.3 Boolean Query Specification . . . . .	17
2.3.1 Boolean Expressions and Truth-Tables . . . . .	18
2.3.2 Command Languages and Natural Language for Data Querying	19
2.3.3 Form-Based Query and Dynamic Query Technique . . . . .	21

2.3.4	Graphical Approaches to Query Specification Based on Boolean Expressions . . . . .	21
2.3.5	Graphical Approaches to Query Specification Based on Truth-Table . . . . .	23
2.4	Query Result Display . . . . .	25
2.4.1	Query Preview and Immediate Updates of Query Results . . . .	26
2.4.2	Context Information of Data Distribution along Each Attribute	27
2.4.3	Rank of Query Results . . . . .	27
2.4.4	The Relationship of Query Results with Query Terms . . . . .	27
2.5	Interactive Data Selection on Visual View . . . . .	28
2.6	Formal Models of Visual Information Exploration . . . . .	30
<b>3</b>	<b>A Formalism for Visual Query Interfaces</b>	<b>33</b>
3.1	The Table Model . . . . .	33
3.1.1	Definition of the Table Model . . . . .	34
3.1.2	Applying Tables to Tables . . . . .	35
3.1.3	Data Visualization by Tables . . . . .	36
3.1.4	Table Model for Queries . . . . .	37
3.1.5	Visualizing Queries . . . . .	38
3.2	The Transformation Model . . . . .	40
3.2.1	Query on Data Space . . . . .	41
3.2.2	Visual Query on Visual Representations . . . . .	42
3.3	Query Result Visualization . . . . .	44
3.3.1	Traditional Method of Query Result Visualization . . . . .	44
3.3.2	Example: Enhanced Visualization of Query Result . . . . .	45
3.3.3	The Transformation Model of Query Result Visualization . . . .	45
3.3.4	Examples of Query Result Visualization . . . . .	47
3.4	Visualizing the Mappings . . . . .	48
3.4.1	Visualizing the Visualization Mappings . . . . .	48
3.4.2	Visualizing the Query Mappings . . . . .	50
3.5	The Recursive Models . . . . .	52
3.5.1	Visualizing Visual Representations . . . . .	53
3.5.2	Querying on Queries . . . . .	54
3.5.3	The Iterative Model . . . . .	54

3.6	Summary . . . . .	56
<b>4</b>	<b>The Application System KMVQL</b>	<b>58</b>
4.1	Software Design based on the Formalism . . . . .	58
4.1.1	User Actions . . . . .	59
4.1.2	Software Domain Model . . . . .	60
4.1.3	Module Responsibilities . . . . .	61
4.1.4	Summary of the Software Model . . . . .	63
4.2	Overview of KMVQL . . . . .	64
4.2.1	Visualization of Query and Query Results . . . . .	64
4.2.2	Visualization of Data Space . . . . .	66
4.2.3	Query Device . . . . .	67
4.2.4	Information Seeking Nodes . . . . .	68
4.2.5	Formulate Boolean Queries in KMVQL . . . . .	68
4.2.6	Compound Query based on Intermediate Results . . . . .	69
4.3	Example of Exploring Data with KMVQL . . . . .	71
4.4	Summary . . . . .	75
<b>5</b>	<b>Related Work on Query Language Evaluation</b>	<b>76</b>
5.1	Introduction to Query Language Evaluation . . . . .	76
5.1.1	Framework of Variables for Query Language Evaluation . . . . .	77
5.1.2	Tasks for Query Language Evaluation . . . . .	79
5.1.3	Boolean Concept Complexity . . . . .	79
5.2	Review of Related Experiments . . . . .	82
5.2.1	Example Analysis: TEBI . . . . .	82
5.2.2	Example Analysis: Filter/Flow . . . . .	84
5.2.3	Example Analysis: Chui's User Study on Query Languages . . . . .	84
5.2.4	Summary of Previous Experiments . . . . .	86
<b>6</b>	<b>A Methodology for Evaluating Visual Query Languages</b>	<b>89</b>
6.1	Introduction . . . . .	89
6.2	Method . . . . .	90
6.2.1	Query Task . . . . .	90
6.2.2	Components of Query Task . . . . .	91

6.3	Isolating the Time Components . . . . .	91
6.3.1	Manipulating Stimulus Onset Asynchrony . . . . .	92
6.3.2	A Possible Experimental Hypothesis . . . . .	92
6.3.3	Data Fitting . . . . .	94
<b>7</b>	<b>The Experiment</b>	<b>96</b>
7.1	Introduction . . . . .	96
7.2	Method . . . . .	97
7.2.1	Query Task . . . . .	97
7.2.2	Experiment Trials . . . . .	98
7.2.3	Queries . . . . .	99
7.2.4	Graphical Icon . . . . .	100
7.2.5	Display Sequence and Delay Time . . . . .	101
7.3	Conduction of the Experiment . . . . .	102
7.3.1	Trial Setup . . . . .	102
7.3.2	Lab Facilities . . . . .	103
7.3.3	Subject Characteristics . . . . .	104
7.3.4	Process of the Experiment . . . . .	104
7.4	Experiment Results . . . . .	105
7.4.1	Error Rate . . . . .	105
7.4.2	Response Time at Zero SOA . . . . .	105
7.4.3	Plots of Response Time Sequence . . . . .	106
7.4.4	Analysis of the $\mu$ Values . . . . .	107
7.4.5	Analysis of the $\beta$ Values . . . . .	110
7.4.6	Analysis of the $\tau$ Values . . . . .	110
7.4.7	Consistency Check . . . . .	111
7.4.8	Subjective Evaluation . . . . .	113
7.5	Discussion . . . . .	114
7.6	Summary . . . . .	117
<b>8</b>	<b>Conclusions</b>	<b>119</b>
8.1	The Formalism for Visual Query Interfaces . . . . .	119
8.2	The Software Design Model and the Application . . . . .	121
8.3	The Experiment Methodology . . . . .	123



8.4	The Experiment . . . . .	123
<b>9</b>	<b>Future Work</b>	<b>125</b>
9.1	Extensions of the Formalism . . . . .	125
9.2	Applications of the Formalism . . . . .	126
9.3	The Experiment Methodology . . . . .	127
9.4	Empirical User Studies . . . . .	128
9.5	Future Work Summary . . . . .	129
	<b>Appendix</b>	<b>129</b>
<b>A</b>	<b>A story about information exploration</b>	<b>130</b>
A.1	Specify queries with Natural Language . . . . .	131
A.2	Specify queries with SQL commands . . . . .	131
A.3	Exploring Data With Form-Filling Interface . . . . .	133
A.4	Exploring Data With Dynamic Query Interface . . . . .	135
<b>B</b>	<b>Elbow Data Fitting Algorithm</b>	<b>138</b>
B.1	Introduction . . . . .	138
B.2	Minimization of the Objective Function . . . . .	138
B.3	Formalism . . . . .	139
B.4	The elbow fitting program . . . . .	140
<b>C</b>	<b>Subject Evaluation</b>	<b>142</b>
	<b>References</b>	<b>144</b>

# List of Tables

2.1	Information Exploration Process Models. . . . .	10
2.2	Data Table Model. . . . .	14
2.3	Data Type and Their Visual Encodings: Knowing the data type of the attributes (whether they are ordinal or quantitative) helps in determining which visual encodings are most effective [89]. This table gives some examples of how the visual encoding might be used given specific data types. . . . .	17
2.4	The SQL Syntax Error Rate. The errors are scored on a scale of 1 to 7. 1 means user never made the error, 7 means user made the error very often. . . . .	20
3.1	The Table Model. . . . .	34
4.1	User Actions and Their Processing Modules . . . . .	62
5.1	Tasks Used to Measure Ease of Use in Query Language Evaluation . . .	80
5.2	Review of the User Study for TEBI [39] . . . . .	83
5.3	Review of the User Study for Filter/Flow [106] . . . . .	85
5.4	Review of Chui's User Study [22] for Four Boolean Query Languages .	86
5.5	Commonly Used Query Tasks and the Factors Affecting the Performance	87
7.1	Queries Used in the Experiment . . . . .	99
7.2	ANOVA of Error Rate . . . . .	105
7.3	Summary of $\mu$ in the Four Presentation Conditions . . . . .	109
7.4	Slope of $\beta$ with Query Complexity . . . . .	110
7.5	ANOVA of $\tau$ in the Presentation Condition of Karnaugh Map Icon-first	111

# List of Figures

2.1	The Process of Visual Information Exploration. . . . .	12
2.2	The Process of Information Visualization. . . . .	13
2.3	Example of a Truth-Table . . . . .	19
2.4	Query Represented with Filter/Flow . . . . .	22
2.5	The TEBI Interface for Query Specification. . . . .	23
2.6	Venn Diagrams. . . . .	24
2.7	InfoCrystal Interface. . . . .	25
3.1	Applying a Table to a Table . . . . .	35
3.2	Example of the Data Visualization Mapping . . . . .	37
3.3	Example of a Truth-Table . . . . .	38
3.4	Iconic Representation for Queries . . . . .	39
3.5	Examples of Query Visualization . . . . .	40
3.6	Transformation Model of Visual Query. . . . .	41
3.7	Example of Data Query Mapping Table . . . . .	42
3.8	Example of Applying Query Mapping on a Data Set . . . . .	43
3.9	Example of Visual Query on Visual Features - Brushing . . . . .	44
3.10	Visualizing Query Results . . . . .	45
3.11	Visualizing Query Results . . . . .	46
3.12	The Model of Query Result Visualization. . . . .	47
3.13	Visualizing Query Results . . . . .	47
3.14	Visualizing Query Results . . . . .	48
3.15	Visualizing Query Results . . . . .	49
3.16	Example: Map and Map Legend . . . . .	50
3.17	Example: Legends for Visualization . . . . .	51
3.18	Example: Query Devices . . . . .	52

3.19	The Model of Visualizing Visualizations . . . . .	53
3.20	Example: Visual Encoding for Visual Attribute . . . . .	53
3.21	The Model of Query on Queries . . . . .	54
3.22	Example: Specifying Query on Query Space . . . . .	55
3.23	The Iterative Model . . . . .	55
4.1	User Actions Diagram . . . . .	60
4.2	Software Domain Model . . . . .	61
4.3	KMVQL User Interface in Karnaugh Map Mode . . . . .	65
4.4	KMVQL User Interface in Iconic Mode . . . . .	66
4.5	Modify Visual Encodings For X-Y-Plot . . . . .	67
4.6	Settings of Query Devices . . . . .	68
4.7	Karnaugh Map Query Devices . . . . .	70
4.8	Create Query Devices Based on Direct Visual Item Selections in KMVQL	71
4.9	Screen Shot of the Query “Meet at Least 3 Query Criteria” . . . . .	72
4.10	Direct Data Selection on the Visual Views . . . . .	73
4.11	Screen Shot of the Compound Query in KMVQL . . . . .	74
5.1	Framework of Variables Used in Prior Query Performance Research . .	78
5.2	Query Represented with TEBI Concrete and TEBI Abstract . . . . .	84
6.1	Response Time: (a) no delay time; (b) delay time less than query com- prehension time (c) delay time greater than query comprehension time . . . . .	93
6.2	The Plot of Response Time vs. Delay Time . . . . .	94
6.3	Elbow Data Fitting . . . . .	95
7.1	Stimuli Used in the Experiment. The upper panel shows the Karnaugh map, the lower panel shows the textual expression . . . . .	98
7.2	Query-first Example Trials: Display Query First . . . . .	102
7.3	Icon-first Example Trials: Display Icon and Query Terms First . . . . .	103
7.4	Average Response Time at Zero SOA: plotted against query, and grouped by representation. . . . .	106
7.5	Typical Data Set: the data set is plotted with the fitted curve drawn on.	107

7.6	The Fitting Parameters for all Twelve of the Presentation Conditions Shown Subject by Subject. $\beta$ , the amount of processing time, in milliseconds, when the early part of the stimulus has been completely processed, is shown in the left column. $\tau$ , the amount of time that the early part of the stimulus continues to be processed, in milliseconds, is in the centre column. $\mu$ , the effectiveness of early processing at reducing the amount of processing afterwards, is shown in the right column. . . . .	108
7.7	The Values of $T_{diff}$ of Each Query Condition: they are plotted against query, and grouped by representation. The values indicate the differences between expected response time with the practical response time at zero SOA . . . . .	114
A.1	AutoExplorer: Form-Filling Query Interface . . . . .	134
A.2	AutoExplorer Dynamic Query Interface . . . . .	135
B.1	The Elbow Data Fitting Program . . . . .	141

# Chapter 1

## Introduction

Extracting information from data is interactive: a user inspects the data, forms a hypothesis, and tests it, which leads to further inspection, and so on, until the desired information is obtained. Visualization plays an important role, supporting both inspection and testing. Querying is also important, allowing the user to isolate promising portions of the data. As a result, interfaces for data exploration environments normally include both, integrating them as tightly as possible [40, 68].

Most data queries are Boolean expressions, which are hard for users to manipulate successfully. Many query interfaces try to alleviate this difficulty by expressing queries using a visual query language. But there is at present no formal method for producing new representations of queries.

Furthermore, the intrinsic connections between querying and visualization are seldom described formally. The lack of formal description inhibits the development of techniques that produce new representations for queries. It also makes difficult integration of visual query specification with query result visualization.

This thesis describes and explores a formalism for visual query interfaces. It captures querying and visualizing in a single framework, which describes information exploration at a level below the interface presentation and above the implementation. The formalism can be used as a high-level framework for visual query interface analysis. It provides concepts for designing new visual query interfaces, and opportunities for extending existing ones.

To illustrate how the formalism is used in a practical visual query interface design, a new query/visualization interface is implemented, a Karnaugh map based visual

query language (KMVQL). K MVQL uses Karnaugh maps as its visual representation of Boolean queries. My implementation of K MVQL shows that the formalism is a practical mathematical abstraction of visual query interfaces.

Based on the formalism, a variety of innovative designs for query visualization can be created by defining multi-dimensional visualization techniques on a query space. But are these new representations for queries easy to understand? Are they easy to use? To answer such questions, it is necessary to evaluate the new query representations, and to compare them with existing ones. Experiments must be conducted.

Previous studies on visual query languages seldom examined why one visual representation performs better than another. To find the answer it is necessary to understand users' cognitive activities while they are using a query interface. Furthermore, it is more important to discover which cognitive activities are enhanced or inhibited by different presentations of a query. Knowing this is helpful for creating and analysing novel query interfaces.

This thesis presents a new method of experimental design and data analysis for query representation studies. The novelty of the method is that it isolates subjects' cognitive operations, and thus allows researchers understand subjects' behaviours in detail.

To illustrate its use, I designed and conducted an experiment comparing two query representations: Karnaugh maps and textual representations of Boolean expressions. A detailed description of the experiment, and the results I obtained are also reported in this thesis.

## 1.1 Contributions

The goal of this thesis is to apply visualization to designing and improving user interfaces for exploring and analysing data. The main contributions of this thesis are:

- a novel mathematical formalism which describes the basic components and their intrinsic connections in visual query interfaces for information exploration systems;
- a new visual query interface, K MVQL;
- a new experiment methodology for evaluating query languages; and

- an empirical user study for comparing the comprehensibility of Karnaugh map representations with textual representations of Boolean expressions.

## 1.2 User Interface Issues in Information Exploration

*“We do not always know precisely what we are looking for and, many times, we do not know even what its name is. In these conditions exploratory search is a strategy that allows us, with the help of visualization, to refine our search and reach the goal of our exploration by means of successive iterations.”*

———— by Juan C. Dursteler [30]

Searching for information, whether for purchasing a car, reserving a hotel room, or finding a book in the library, is an activity we do throughout our daily lives. But the massive volume and huge variety of available knowledge bases makes it difficult. Appendix A tells a typical story of information exploration. It describes the following difficulties a user meets when searching a database of second-hand cars:

- query specification, refinement and modification;
- query result analysis; and
- reuse of previous query results.

When searching for data in large data sets, it is not realistic to examine every available data item one by one, selecting interesting items. Data filtering, which breaks up the data set into meaningful subsets, is essential to decrease the amount of information users must perceive, remember, and recognize. Queries are used to filter data and return subsets of the data. Most commonly, they are built from simple terms, combined using Boolean operators, which have convenient formal properties. Boolean logic, which George Boole called “the Laws of Thought” [10], is the basis of most query models, yet is notoriously difficult to use with speed and accuracy [39, 55].

As a result, there are many query interfaces that attempt to put an easy to understand gloss on Boolean logic. Some are linguistic; others rely on visual presentations [41, 106], which might be described as **visualizations of queries**. Visual interfaces use visualization a second time, when the query results are provided to the



user. Visual display of query results is important because the user’s goals evolve during information exploration. Thus, the display of query results should help her to make sense of the data, providing information that facilitates query refinement and modification, so that she can decide what to do next.

Visualization is a powerful method for communicating the salient features of information, making them easy to perceive and understand. It is especially helpful when little is known about the data or when the user’s goals are poorly defined. Furthermore, many visualization implementations allow direct manipulation, so that users can filter data interactively.

But when I examined existing interfaces for information exploration, I found these complementary topics have usually been studied separately. Although a few systems, such as Tableau [40] and Spotfire [68], integrate visualization and query specification, their intrinsic connections are not described formally. The absence of a common formalization inhibits innovation.

In addition, there are few formal descriptions of the relationship between queries and query results, which emphasizes the existing conceptual gap between visual query specification and query result visualization.

In fact, having formal models of query interfaces for information exploration is beneficial. Because common formalizations are general. Thus they can be applied to a wide variety of design problems to help researchers understanding, designing, or analysing visual information exploration systems.

### 1.3 A Formalism for Visual Query Interface

This thesis presents a mathematical formalism that captures both visualization and query formulation within a single structure. It treats data presentation, query presentation, and query result presentation as instances of visual representations, which reveals the inherent connections between them.

The formalism uses the mapping model of visualization [15], in which a visualization is defined as a mapping of data attributes onto visual dimensions such as colour and position. A mapping is formalized as a two-column table, the first column listing every possible combination of attributes and the second column listing the corresponding visual presentation.

The formalism starts by defining a generic table model, in terms of which all entities in the query/visualization environment are defined: data, queries, query results, and visual representations of them. Mappings from table to table, including visualizing and querying, are also defined as tables.

Thus, the fundamental operation of the formalism is applying one table to another table to get a result table. As a result, the formalism supports querying a visual representation or visualizing a query, which enhances the versatility of visual query interfaces.

Based on the formalism, visualizing Boolean queries is identical to visualizing multi-dimensional data sets, which makes it possible to apply existing visualization techniques to query visualization. The formalism also describes an approach to query result visualization that tightly integrates query specification and the display of query results. It helps users make sense of the data and facilitates query refinement and modification. Furthermore, it also introduces recursive applications of both querying, querying a query, and visualizing, visualizing a visual representation. Recursion allows users to ask questions that help them understand complex queries or visualizations, which provides systematic reduction of complexity.

The formalism is an abstraction of visual query interfaces. There are many benefits of such an abstraction. For example, it helps to identify areas of the design space for future research and to clarify the similarities or differences between known techniques.

Based on the formalism I implemented an interactive system, KMVQL, which provides a visual interface based on Karnaugh maps. It allows users to specify arbitrary Boolean queries by direct manipulation, which is useful when the complexity of queries is high.

## 1.4 Issues in Evaluating Query Languages

Boolean logic is notoriously difficult for people to use with speed and accuracy [39, 55]. Many query interfaces alleviate this difficulty by expressing queries in a visual query language, It is important to evaluate the performance of query languages, comparing them to one another, in order to see how well they facilitate query understanding and use.

When analysing existing empirical studies of visual query interfaces, I found that

few of them showed why one query interface performs better than another. Users' cognitive activities were seldomly analysed. In fact, understanding users' cognitive activities and how they are affected by query presentations will help researchers create and improve novel query interfaces.

The time taken to finish a query task is often the response measure for query language evaluation. Tasks for query language evaluation are normally complex, requiring combinations of cognitive activities such as learning, understanding, remembering, and decision making. The required set of cognitive activities varies from task to task. Yet the raw time measured in empirical studies is a combination of many time components, so that it is hard to associate qualitative features of user performance with one or another aspect of the task. This complication is a block to analysing the utility of an interface based on user testing. Therefore, to analyse user behaviour and cognition based on the experimental data, it is important to isolate as many components as possible.

Besides, when analysing existing visual query representations, I found that most representations are suited to specific types of queries. It is quite possible that when comparing the time taken by two query representations using different queries, their relative performance may differ. To accomplish in limited time, empirical studies must limit the number of queries that are used, which makes it harder to measure performance on more complex queries. What happens when queries have twenty or more query terms? Is it possible for researchers to predict users' performance for more complex queries based on limited experimental data?

Isolating components of query tasks, and providing a data analysis method are two important issues in experiment design and analysis. But unfortunately, they have been neglected in previous empirical studies.

## 1.5 The Experiment Methodology

This thesis introduces a methodology for experiment design and analysis. It evaluates the quality of query languages, using an experimental task that requires users to perceive, understand, and respond to visual queries. It assumes a query task to be composed of three parts: query comprehension, stimulus comprehension, and response generation.

The central idea of the methodology is to present parts of the stimulus at different times, a technique known in perceptual psychology as stimulus onset asynchrony (SOA). When gradually changing the SOA value, a set of response times are measured, which form piece-wise linear curve. A data fitting algorithm computes the break points, from which the time for different query task components can be isolated. This method allows researchers to discover which cognitive activities are enhanced or inhibited by different presentations of a query.

Based on this methodology, I designed and conducted an experiment comparing the comprehensibility of two query representations: Karnaugh map and textual representations of Boolean expressions. From the study, I found that users understand queries in two qualitatively different ways: inductive and deductive. Karnaugh maps enhance inductive understanding, which is relatively independent of query complexity. This explains why Karnaugh maps increasingly outperform textual Boolean expressions as query complexity increases.

## 1.6 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 reviews previous research on visual query interfaces for information exploration, focusing on information visualization techniques, representations of Boolean queries, visualization of query results, and interaction techniques for data selection from visualizations.

Chapter 3 presents a mathematical formalism that relates the important components of visual query interfaces. The formalism includes a transformation model of queries on data, a model of query result visualization, and a set of recursive models. Formal definitions of the components and the transformation mappings between them are presented there.

Chapter 4 derives a software model of visual query interfaces from the formalism. The software model can be used as a reference model for designing practical visual query interfaces. Based on it, a new prototype system has been implemented.

Chapter 5 reviews previous research on query language evaluation. Chapter 6 presents the new experimental methodology on which the experiment design is based. Measurements of user performance for understanding Boolean expressions using textual

and Karnaugh map representations are then described in Chapter 7.

Chapter 8 presents conclusions, including a summary of the contributions of the thesis. Chapter 9 presents a discussion of directions for future work.

# Chapter 2

## Related Work on Visual Information Exploration

The research reported in this thesis lies within several sub-areas of graphical interaction: information exploration, information visualization, and visual query interfaces. Research relevant to my work in these areas is reviewed in this chapter.

### 2.1 Information Exploration

*“Information exploration is taken to mean the information seeking behaviours that can occur in primary databases and Internet resources in which end users interact with information, making relevance judgments and re-formulating the queries during a sequence of examinations and assessments of information”*

————— by Wilson [102]

By its nature, information exploration is an interactive process, with the user at its centre [50]. To better understand user interface issues in information exploration, it is necessary to have a model of the behaviour of users exploring information.

#### 2.1.1 Information Exploration Process

Models of the information exploration process have been described in similar ways by different researchers. Most of them assume an iteration cycle, like the ones outlined in Table 2.1.

Step	Norman (1988)	Marchionini (1995)	Hearst (1999)
1	Forming the goal	Recognize, accept problem Define, understand problem	Start with an information need
2		Choose a search system	Select a system and collections to search on
3	Specifying an action	Formulate a query	Formulate a query
4	Execute the action	Execute search	Send the query to the system
5	Perceiving the state of the world	Examine results	Receive results in the form of information items
6	Interpreting the state of the world	Extract information	Scan, evaluate, and interpret the results
7	Evaluating the outcome	Reflect/iterate/stop	Stop or reformulate the query and go to step 4

Table 2.1: Information Exploration Process Models.

These high-level, theoretical models of the information exploration process are generalizations; each of their authors points out that in practice they contain variations. The general model of information exploration they represent, however, remains the basis of most information exploration systems currently in use.

Many studies have examined user behaviour during information exploration, which led to several more detailed models of information seeking. These models take information seeking to be an evolving problem-solving activity in which users integrate results from the system with information they otherwise possess.

A representative model is the “berry-picking” model proposed by Bates [6]. In describing how users’ information needs are affected by the exploration process, the “berry-picking” model makes two important points.

1. As users internalise information encountered in the search process, their information needs, and consequently their queries, continually change. Information encountered at one point in a search often leads the search in a new, unanticipated direction.
2. Users’ information needs are not satisfied by a single final retrieved set of docu-

ments, but by a series of selections combined with fragments of information found along the way.

The berry-picking model is supported by several observational studies, including that of O’Day and Jeffries [75]. They found that information seeking consists of a series of diverse interconnected searches on a problem-based theme, and that satisfying one goal creates new goals, which push search in new directions.

### 2.1.2 Visual Information Exploration

Occasionally users know exactly what they are looking for and where it is, but often they have only a faint idea of what they are trying to find. How can they find information when they lack the specific knowledge needed to form a query?

Traditional search requires users to retrieve information by providing precise queries with meaningful keywords, which does not work. More elaborate systems are needed to provide exploratory search, which can function even when users do not know what they are looking for until they see and recognize it.

In exploratory search, users perceive and interact with the data. Researchers have found that visualizations communicates the salient features of the information, using methods that are natural for the human mind to perceive and understand [50]. Applying visualization techniques to help users exploring information is called **visual information exploration**.

Visual information exploration systems usually use visual query interfaces, which are essentially based on the use of visual representations to depict the domain of interest and express the related requests. They present data in a visual form, allowing users to see and interact directly with the data, which is a powerful way of perceiving patterns. The process of visual information exploration is illustrated in Figure 2.1.

Visual query interfaces include both a query language in a visual form and a variety of interaction mechanisms to facilitate data exploration. Visual query interfaces must provide two interaction modalities.

- **A visual query language.** Visual query languages represent queries as a set of visual features. Users can then reason visually to understand and specify queries. A successful visual query interface must facilitate query specification, refinement and modification, which are all important in information exploration.



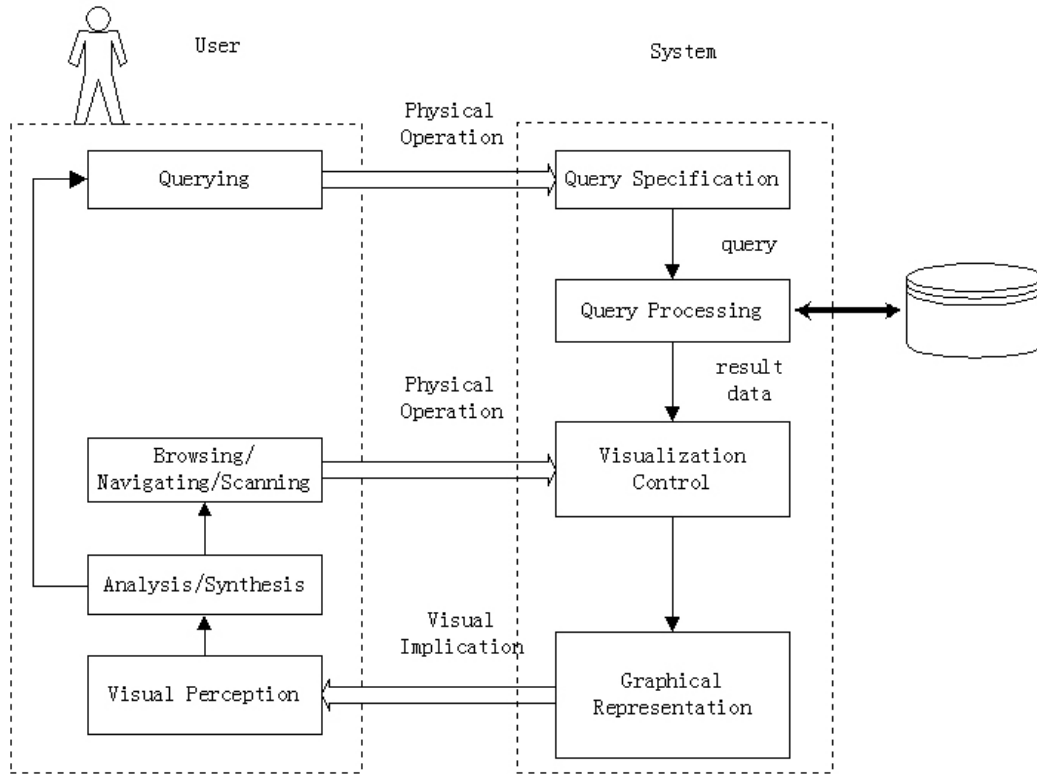


Figure 2.1: The Process of Visual Information Exploration.

- **Visualizations of the result data.** The result of a query must be easily perceived and understood by users so that they can analyse it and make decisions. When working with visualizations, selecting, browsing, navigating and scanning data are all important to users. A successful interface should allow users to do those operations easily and naturally.

Visual query interfaces are oriented to a wide spectrum of users who may have limited technical skills and who generally ignore the inner structure of the accessed database. Many visual query interfaces have been proposed in the past, adopting a range of different visual representations and interaction strategies. Some typical techniques are reviewed later in Section 2.3. In a visual query interface, visualization techniques play important roles, so that the next section reviews the basic concepts of information visualization.

## 2.2 Information Visualization

According to Card, Mackinlay, and Shneiderman [15], information visualization is defined as:

*The use of computer-supported, interactive, visual representations of abstract data to amplify cognition.*

—*Readings in Information Visualization: Using Vision to Think [15]*

Since the late 1980s, information visualization has developed as researchers seek ways to support understanding and analysis of abstract data (data that is not inherently geometric) by using interactive computer graphics and visualization.

### 2.2.1 The Process of Visualization

Visualization can be conceived as applying a mapping from data values to visual representations. A few attempts have been made to formalize the visualizing process. Card, Mackinlay, and Shneiderman developed a reference model [15] for visualization to describe the mapping process, which is shown in Figure 2.2.

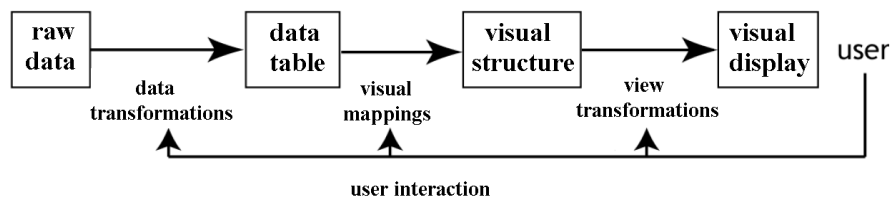


Figure 2.2: The Process of Information Visualization.

The reference model includes three steps of the visualization process:

- data transformations from raw data input to data tables;
- visual mappings from data tables to visual structures, and
- view transformations from visual structures to the final views.

The transformations and mappings are conditioned on the user's task and manipulated by the user through an interface.

The first step in the model is to transform the raw data into the data table, which sometimes can lead to a loss of information, usually from a reduction of variables or cases through statistical computations. Missing data values can also be constructed to fill out the structure of the table.

The second step in the model, the visual mapping from data tables to visual structures, is the most critical step in the visualization process.

The third step in the reference model is the view transformation from a visual structure to a view of it. Interactive view transformations allow the user to get more information from a visual view than would be possible from a static presentation.

Human interaction, which is part of the model, can alter all the transformations and mappings described above. Users can adjust their views of the data, change visual structures, or even affect data transformations.

### 2.2.2 Data Model

The raw data that is the input of a visualization comes in many forms, from spreadsheets to the text of novels. Usually this data is transformed into a format that is more structured and thus easier to map onto visual structures [15].

The usual data model is the relational data model introduced by Codd [24]. It is a formal mathematical model based on a single tabular data structure, the relation. The labels of the rows and columns in the data table are meta-data. The columns represent variables, sets that represent the range of the possible values in the tuples. Each row, called a tuple or data item, is an aggregation of related data values, one for each variable.

	<b>Variable 1</b>	<b>Variable 2</b>	<b>...</b>	<b>Variable n</b>
Item 1	$Value_{11}$	$Value_{12}$	...	$Value_{1n}$
Item 2	$Value_{21}$	$Value_{22}$	...	$Value_{2n}$
...	...	...	...	...
Item m	$Value_{m1}$	$Value_{m2}$	...	$Value_{mn}$

Table 2.2: Data Table Model.

The table model has several advantages.

- It provides an easy to understand conceptual model for users;

- it clearly depicts the variables associated with a collection of data, which is important when selecting visualizations;
- it provides access to many existing datasets, from computer systems logs to biological data to corporate data warehouses, many of which are already stored in relational databases, without needing any additional translation; and
- it provides a general abstraction layer so that all data sources look equivalent; Data transformations can be composed regardless of their actual implementation.

Conversely, some types of data, such as graphs or trees, are difficult to represent using the table model. Furthermore, even given such a representation, some data operations, such as a prefix tree traversal, are non-trivial. For these data structures, having a custom data model may be a solution that is both easier to design and likely to yield better performance than the more general table model.

Even given these disadvantages, the data table model is prevalent. In existing general-purpose visualization tools, the most common data model is the relational data model, which is used in Rivet [13], Snap-Together Visualizations [73], DEVise [63], VQE [28], DataSplash [103], Sage/Visage [80, 79], and VisDB [57].

### 2.2.3 Visual Mapping

The second step in the reference model (Figure 2.2) is the most critical step in the visualization process. It is the actual visualization step that maps data tables to visual structures constituted by a set of visual properties. The visual mapping encodes dimensions and measures of the data into visual properties such as colour, size, orientation, and shape.

A data table can be visualized in many different ways by choosing different visual properties for each data attribute and by specifying different visual items for each data item. Since the purpose of visualization is to amplify cognition [15], when defining visual mappings, it is important to follow principles derived from theoretical and empirical studies of human visual perception so that the visualizations can harness the perceptual capabilities of the human visual system to the task of data exploration.

There exists abundant research on the use of visual properties to effectively encode abstract data. For example, Ware [97] and Healey’s research [43, 44] connect psychological studies on human visual processing with information visualization techniques.

Such research helps visualization designers to understand the human visual system and helps improve both the quality and the quantity of information that can be displayed.

In order to determining which visual encodings are effective, or possible, it is important to understand the different measurement scales for data. Measurement scales for variables are commonly classified into four basic types [26, 97]:

- *Nominal* (only equal or not equal to other values),
- *Ordinal* (possess a total order),
- *Interval* (support addition and subtraction).
- *Ratio* (can do multiplication and division on them).

In some references [89, 90, 91], to simplify analysis, *Quantitative* data is used to denote both *Interval* data and *Ratio* data.

There are many visual properties that designers can work with in visually encoding data. Considerable research has taken place in finding guiding principles for encoding data visually. Examples include Bertins Semiology of Graphics [9], Wilkinsons grammar of graphics [99, 98], the Polaris formalism [89], and other similar work [44, 64, 71, 97]. Table 2.3 is a list summarized by Stolte [89] on the encoding relationship between visual properties and data types. These Research is the foundation of information visualization.

## 2.2.4 Data Selection and Visualization

During information exploration, which is an extended process in which users examine large scale collections of data with a view to understanding them, information selection and visualization go hand in hand. Data to be presented is determined by user-controlled selection; understanding resulting visualizations provides the criteria for further selection. Visualization systems always include data selection mechanisms; systems that primarily offer data selection, such as database management systems, often provide visualization capabilities. A tighter integration between visualization and selection will lead to more effective visualization systems for data exploration and analysis.

Data selection can occur at any stages of the visualization process. For example, raw data can be filtered before transforming into data table. Only selected data items

	Point	Line	Area	Nominal Encodings	Ordinal / Quantitative Encodings
Color					
Shape / Pattern					
Size / Granularity					
Rotation / Orientation					

Table 2.3: Data Type and Their Visual Encodings: Knowing the data type of the attributes (whether they are ordinal or quantitative) helps in determining which visual encodings are most effective [89]. This table gives some examples of how the visual encoding might be used given specific data types.

from the data table are encoded with visual mappings. Or, only selected items in the visual structure are displayed on the screen. Practically, how and where data selection is performed is determined by the implementation of a specific visualization system. No matter where it occurs, data selection is an integral part of information visualization and exploration.

There are many techniques for data selection, usually use Boolean query, sometimes explicitly, sometimes implicitly. Thus the following sections review existing Boolean query specification mechanisms and show how they are integrated with visualization techniques.

## 2.3 Boolean Query Specification

When exploring large data sets, it is necessary to partition the data set in order to focus on comprehensible subsets. User-specified queries select those subsets the user requires. Most such queries are based on Boolean algebra. Boolean queries have been

employed in information exploration systems for decades. With the recent proliferation of search engines, Boolean queries have become ubiquitous [86].

Many different interfaces have been introduced to support query specification. When analysing existing approaches to Boolean query specification, I found two underlying query models: the Boolean expression model and the truth-table model. The two query models lead to different representations of queries and interaction mechanisms for query creation.

Based on their human-computer interaction styles and their underlying query models, query approaches can be classified into the following categories:

- textual approaches based on command languages or natural languages;
- form-based and dynamic query interfaces;
- graphical approaches to query specification based on Boolean expressions, and
- graphical approaches to query specification based on truth-tables.

These techniques are discussed in the following sections.

### 2.3.1 Boolean Expressions and Truth-Tables

A Boolean query normally consists of two parts: (1) query terms that specify matching conditions of the data subsets, and (2) Boolean operators that join them into combinations.


A Boolean expression is composed of atomic query terms, which are either true or false, connected by Boolean operators into sub-expressions, which can be further connected into higher order sub expressions. Query terms, denoted  $T_1, T_2, \dots, T_N$ , are combined using Boolean operators: AND, denoted by  $T_1T_2$ , OR, denoted by  $T_1 + T_2$ , and NOT, denoted by  $\overline{T_1}$ .

A truth-table is a mathematical table used to compute the values of Boolean expressions. It is based on a canonical form of Boolean expressions: any Boolean expression using  $N$  terms can be written as the disjunction of  $2^N$  query fragments in which all terms are written either negated or not. These query fragments are known as minimal polynomials, and it is elementary to show that any Boolean expression can be written as a disjunction of them. Thus, for example:

$$T_1T_2 + T_1T_3 + T_2T_3 = T_1T_2T_3 + \overline{T_1}T_2T_3 + T_1\overline{T_2}T_3 + T_1T_2\overline{T_3}$$

Only a subset of the  $8 = 2^3$  fragments occur in this expression, and formally each expression can be uniquely defined by listing the minimal polynomials that appear in it. Figure 2.3 shows the truth-table representation for this Boolean expression.

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	Q
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
1	1	0	1
1	0	1	1
0	1	1	1
1	1	1	1



$$\begin{aligned}
 Q &= T_1 T_2 T_3 + \bar{T}_1 T_2 T_3 + T_1 \bar{T}_2 T_3 + T_1 T_2 \bar{T}_3 \\
 &= T_1 T_2 + T_1 T_3 + T_2 T_3
 \end{aligned}$$

Figure 2.3: Example of a Truth-Table

Each unique query term creates a column in the truth-table. All the attributes are Boolean values, each of which is “true” or “false” (here I use 1 to represent the “true” value, use 0 for “false”) depending on whether the corresponding term is negated or not.

Each possible minimal polynomial forms a row in it. There is an extra column used for indicating whether the Boolean expression is true for the minimal polynomial. “true” if the minimal polynomial is included in the expression, “false” otherwise. Any Boolean logical combination of the query terms can be represented using a truth-table.

### 2.3.2 Command Languages and Natural Language for Data Querying

Traditional query systems require users to express their queries in a textual command language based on Boolean expressions, which are powerful and precise. For example, SQL, is the most commonly used command language in database querying. When formulating a Boolean query in SQL, the query is created by conjoining query terms with Boolean operators *AND*, *OR* and *NOT*, and sometimes parenthesis. The following example is a typical SQL query.



```

SELECT Department, Name, Salary, Age
WHERE Department = 'Accounts'
      AND ( Salary > 15000 OR Age < 30 )

```

Unfortunately, many users find that Boolean logic is difficult, both slow to formulate and error-prone. For example, users have difficulty using parentheses to express the precedence of Boolean operators when building complex queries. The meaning of *AND* and *OR* in Boolean logic also conflicts with the meaning in English: *AND* tends to be inclusive in English but is exclusive in Boolean logic; *OR* tends to be exclusive in English but is inclusive in Boolean logic.

Table 2.4 shows the results of a survey [18] on the frequency of SQL syntax errors on a scale of 1 (never) to 7 (very often). In the table, “Misplaced brackets” ranks highest among all errors. “Improper nesting of AND/OR” and “use of AND for OR or vice versa” are also frequent.

SQL syntax errors	Score *
Misplaced brackets	3.3
Misspelling	3.1
Improper nesting of AND/OR	3.1
Synonyms	2.9
Omission of relation qualification	2.8
Use of AND for OR or vice versa	2.3
Omission of FROM clause	2.1
...	...

Table 2.4: The SQL Syntax Error Rate. The errors are scored on a scale of 1 to 7. 1 means user never made the error, 7 means user made the error very often.

Because of the difficulty that users have with Boolean operators, researchers have suggested that designers should avoid using logical operators and parenthesis [39]. Thus, various techniques have been developed as alternatives.

Natural language interfaces for query formulation have been recognized as an enhancement for end-users in some fields such as World Wide Web search engines. They allow users to choose from a selection of common simple ways of combining query terms, including ‘all the words’ (place all terms in a conjunction) and ‘any of the words’ (place

all terms in a disjunction). But sometimes *AND* and *OR* are still explicitly used in natural language interfaces, despite being ambiguous. It is also hard to express complex Boolean queries using such interfaces.

### 2.3.3 Form-Based Query and Dynamic Query Technique

Form-based query interfaces and dynamic query environments [1, 28, 84] allow users to construct queries by simply specifying query terms. Users do not need to worry about their logical combinations, because the structure is predetermined by the interface. Dynamic query systems also provide users with an immediate visual presentation of query results, giving the user guidance for completing partial queries.

But these interfaces accept only conjunctions among query terms, so that the query structure and the number of query terms is predetermined. The strong restriction on query structure makes them easy to use, but at the cost of power, because many queries are not supported.

### 2.3.4 Graphical Approaches to Query Specification Based on Boolean Expressions

Query-by-Example (QBE) [107] uses a tabular representation to create queries for relational databases. It uses visual tables where the user enters commands, example elements and conditions. Many of today's graphical front-ends for databases use ideas from QBE. But when represented in QBE, complex Boolean queries are not straightforward and thus they are difficult to form and understand.

Network(or graph) representations have emerged as popular graphical representations for database queries, with nodes and links representing components and their relationships. For example, in the Gql interface [76], entities are represented as circles and their associated attributes as ovals, with the attribute name appearing inside the oval, and with functions represented by labelled directed arcs. Such a representation eliminates the difficulty of dealing with the precedence of logical operators, and thus is more understandable than an textual expression that uses parenthesis, There are many examples of this type, such as LabView [38], GLOO [33], and GQL [5].

Magic Lens [36] uses windows to represent query terms, and uses windows that overlap other windows to represent logical combinations of terms. Each overlapping

window is associated with a Boolean operator that describes how it logically combines the windows that are underneath it. One advantage of Magic Lens is tight integration of the display of query results: data subsets are visualized in the window they satisfy, which helps users to analyse the results. To formulate Boolean queries, users must correctly arrange the windows, which is hard when queries are complex.

Both network representations and Magic Lens display the logic operators explicitly, and users must understand the meaning of operators.

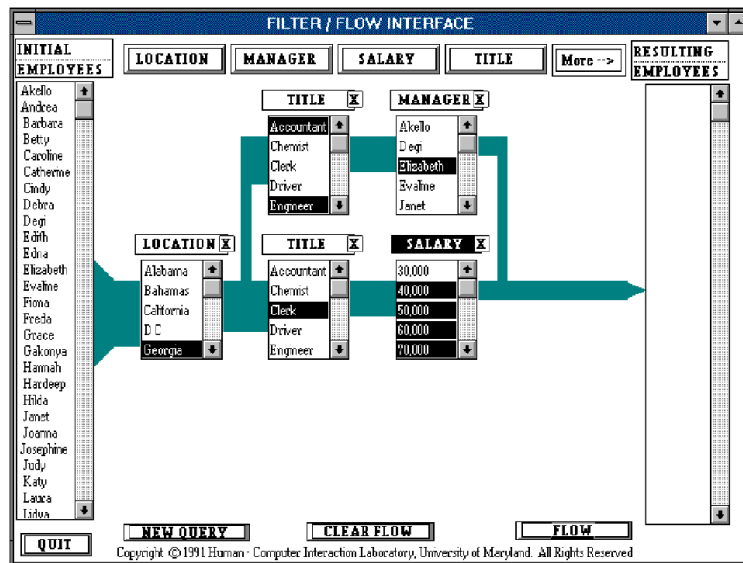


Figure 2.4: Query Represented with Filter/Flow

The Filter/flow representation [41, 72, 106] for Boolean queries solves the problem by visually representing the logical relationships. It uses the metaphor of water flowing through filters: sequential flow for conjunction, parallel flows for disjunction. A similar representation is the grid of tiles designed by Anick [4]. Each query term is represented within a block. The blocks are arranged into rows and columns. The positional relationship of blocks represents the Boolean relationship between query terms: Blocks in the same row denote conjunction; blocks in different rows denote disjunction. Thus Boolean relationships are visualized with visual features, which is intuitive. But visualizing complex queries requires much screen space, which limits its effectiveness.

When using Boolean expression based approaches for query specification, users need to have explicit logical structures in mind. They must be able to predict from the

query what records will be in the final listing, a difficult task that puts a heavy load on working memory when queries are complex.

### 2.3.5 Graphical Approaches to Query Specification Based on Truth-Table

TEBI [39], the Truth-table Exemplar-Based Interface, provides a user with a set of example data when she specifies a list of query terms. Example data items are organized in a truth-table, each row being a concrete example of a Boolean combination of the query terms. The user selects rows from the table to create a query, which is the disjunction of the selected rows. A similar system to TEBI is Query-by-Browsing (QbB) [29], which works by allowing the user to browse a database and select examples of records. The system then uses machine learning techniques to generate a query from the user’s examples.

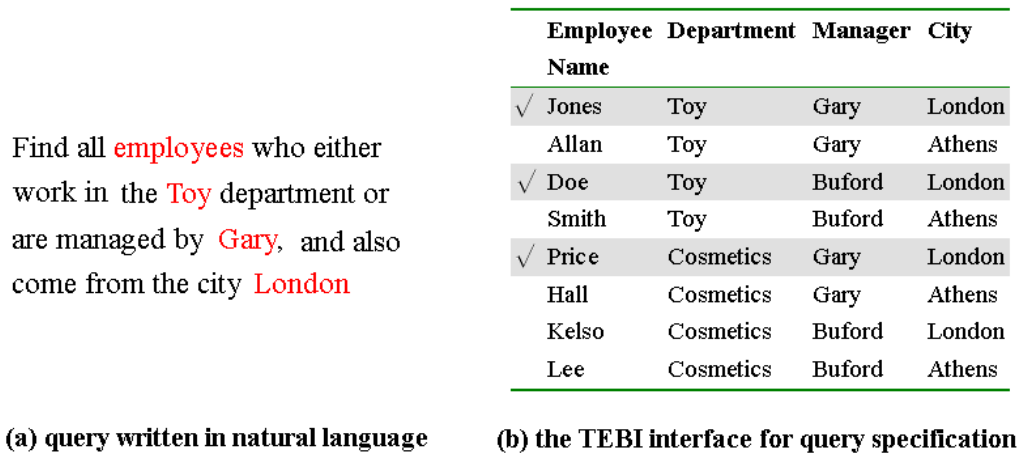


Figure 2.5: The TEBI Interface for Query Specification.

When creating queries using such interfaces, users only need to read, recognize, and select. They do not need to worry about Boolean combinations of query terms. Thus the task of specifying queries is easier. But the linear arrangement of the items in a truth-table is not efficient: users can only browse the items sequentially. When the number of query terms grows large, sequential browsing is time-consuming.

Truth-table based representations can be improved by visualizing elements with a set of visual features. Venn diagrams have been proposed as a more understandable way

of representing Boolean queries. Each query term is drawn as a circle, and intersections of circles denote conjunctions of terms. Each area in a Venn diagram represents a row in the truth-table. Venn diagram based query interfaces include GQL [70], VQuery [55], Cougar [45], and Link Crystal [81].

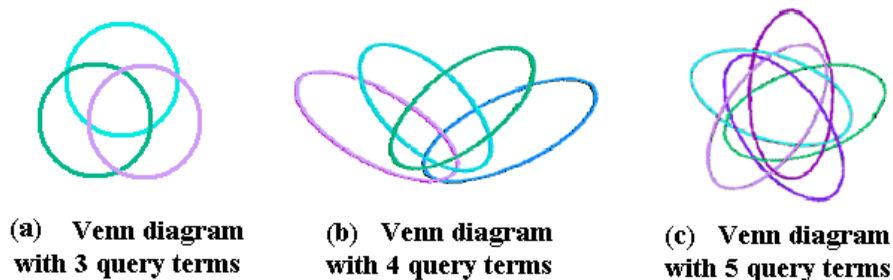


Figure 2.6: Venn Diagrams.

Experiments have proved that a simple Venn diagram can produce faster and more accurate queries than command language based syntax [49, 55]. However, a problem with Venn diagram is the limitation on the number of variables: it is not easy to expand the Venn diagram beyond 3 query terms.

InfoCrystal [88] was designed to improve the poor extensibility of Venn diagrams. It visualizes each row (query fragment) in a truth-table as a graphical icon. Multiple visual coding principles such as shape, size, colour and orientation, are used to enhance the representation, showing the connection between icons and query terms. Thus InfoCrystal can visualize queries with any number of query terms. When a user wants to specify a query, she needs to find the graphical icons that meet her information need and select them. The query is the union of the selected query fragments.

The problem with InfoCrystal is that even simple disjunctive queries can be complex. When the number of query terms grows, the InfoCrystal becomes difficult to use.

Compared to Boolean expression based approaches for query specification, truth-table based approaches eliminate the need to specify logical operators, and rely more on recognition than generation [39]. When creating queries, users only need to read, recognize, and select. They do not require users to have explicit logical structures in mind, which reduces the load on working memory and makes query specification much easier. This indicates that truth-table based query interfaces are promising.

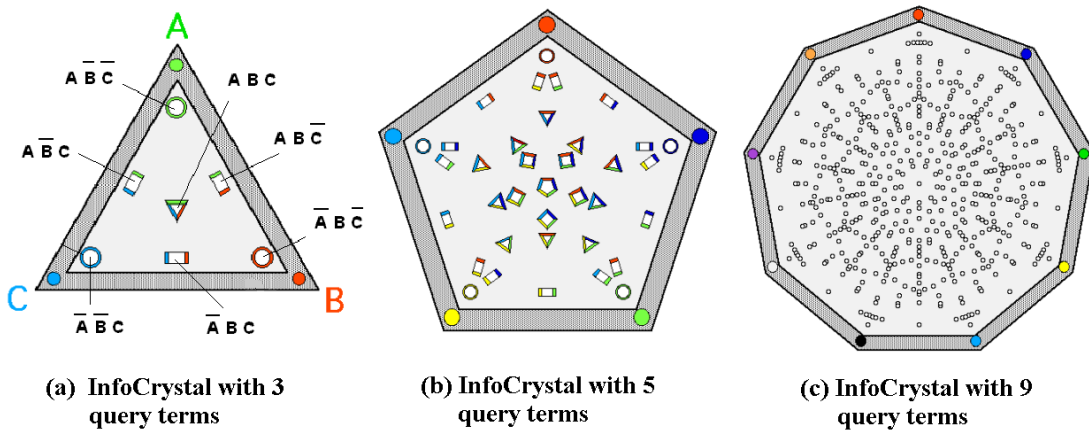


Figure 2.7: InfoCrystal Interface.

Unfortunately, existing query interfaces based on truth-tables have several limitations. How to design new visual representations for truth-tables that are more intuitive and extensible? The formalism reported in this thesis provides possible solutions. It allows designers to apply visualization techniques for visualizing truth-tables, so that more intuitive and extensible visual query interfaces can be designed, making query creation easier and less error-prone, and integrating the query with the visualization of its results.

In the literatures, the definitions of “graphical query language” and “visual query language” are not distinct. I would like to clarify the difference between them here. Graphical query languages include all the graphical approaches for query representation and creation, including those that use Boolean operators explicitly. Visual query languages visualize Boolean relationship with visual features, allowing users use their visual reasoning skills to understand and specify Boolean queries. Thus visual query languages are a subset of graphical query languages. This thesis focus on designing and analysing visual query languages.

## 2.4 Query Result Display

In many database systems and query interfaces, users can issue only one query at a time, and is not easy to change a query slightly or to express vague queries. Most importantly, only data items that exactly meet the query are returned and displayed,

which may contain either no data items, and thus no hint for continuing the search, or too many data items for efficient browsing.

When exploring large and unfamiliar data sets, it is often difficult to find desired data with such interfaces, especially when users have only a fuzzy understanding of their goals. Thus, better user interfaces would aid the understanding and expression of information needs by helping users to understand their query results and keep track of the progress of their exploration.

Researchers have developed various approaches to improve query interfaces so that better feedback is provided for users' queries. Information seekers often express a desire for a user interface that organizes search results into meaningful groups, in order to help them make sense of the data, and decide what to do next [46]. Some techniques have been developed for this purpose, such as providing immediate updates to query results, visualizing context information, ordering the display of results, and revealing their relationship with query terms. These techniques are reviewed in this section.

#### **2.4.1 Query Preview and Immediate Updates of Query Results**

Combining query preview with dynamic query interfaces provides immediate update of query results. When users manipulate selectors, they get immediate feedback from their actions. Such feedback gives them better understanding of data distributions.

The HomeFinder application [3], for example, provides an overview visualization of how houses are distributed geographically by representing each house as a dot on a map. The interface provides sliders that are mapped to individual data attributes. As the user adjusts the sliders, the results of the query are dynamically updated on the displayed map.

But dynamic query systems disclose only data that satisfies the current query limits, neglecting the possibility that contextual information provided by objects not matching the current limits might be useful.

## 2.4.2 Context Information of Data Distribution along Each Attribute

Dynamic query histograms [27], Data Visualization Sliders [31] and Attribute Explorer [94] are extensions of dynamic query techniques, which aim to provide context information about the data distribution. They display the data distribution along each attribute.

For example, in Attribute Explorer, the distribution of all data items on all attributes is represented by a set of histograms. Limits can be placed on an attribute to select a range and, consequently, the objects for which the attribute lies within the chosen range. When more than one attribute is explored, the histograms are linked so that a selection based on one attribute is “brushed” into the other histograms. It uses colour-encoded sensitivity to guide the user and reveal the data distributions along the attributes.

## 2.4.3 Rank of Query Results

Some systems choose to display the query data items in a ranked list. Ranked output has been used very commonly in information retrieval systems. In some digital library systems, for example, the ordering within the set of retrieved documents is based on well understood formal characteristics of the documents, such as date of publication, title, or author. In best-match systems [25], on the other hand, neither the matching rule nor the ranking rule is easily understandable.

One relevant database visualization system is VisDB [57], which focuses on displaying as many data items as possible to provide feedback as users refine their queries. This system even displays tuples that do not satisfy the query, indicating their “distance” from the query terms using spatial encoding and colour. This approach helps users avoid missing important data points that fall just outside of the selected query parameters.

## 2.4.4 The Relationship of Query Results with Query Terms

To help users understand query results and to support query reformulation, some visualization techniques make explicit the connection between the query terms and



result data items, with the goal of showing the contribution of each query term to the query result items. Several examples are described below.

The first method shows the relationship between each data item and the query terms. Examples include TileBars [45] and SeeSoft [32]. In TileBars, for example, a graphical bar beside the title of each retrieved document shows the degree of match for each term.

The second method shows how many data items satisfy each query term. Examples include InfoCrystal [88], VQuery [55], Cougar [45], Flow diagrams [41, 106], and Mosaic plot [53]. For example, InfoCrystal displays in each icon the number of data items that satisfy the corresponding query. Flow diagrams show quantity information of possible results flowing through the filters.

The third method visualizes data subsets in the context of the query terms they satisfy. Examples are Polaris [90], Magic Lens Filters [36] and the Envision system [37].

Some systems use spatial distance to show the relationship of query results to query terms. Examples are Bubbleworld [8], InfoCrystal [88], BEAD [17], VIBE [58], and LyberWorld [48]. For example, BEAD shows a landscape view of an entire document collection. Articles in a bibliography are represented by particles in 3D space. The spatial position of articles indicates their similarity to each other.

In summary, query result displays aim to provide immediate and explicit feed back for users' queries. The feed back should reveal the relationship of query results with query terms so that users can better understand the query and thus refine it more efficiently. Tight coupling of queries with results is especially important in exploratory search [66, 87].

As been reviewed in this section, there exist various approaches for query result display that help information seekers make sense of the data and the queries. But it is rare to find formal descriptions that are useful for designing new techniques of coupling queries with their results. Without formal models, visualization techniques cannot be fully utilized for query result representation.

## **2.5 Interactive Data Selection on Visual View**

In data visualization, users often focus on interesting subsets of the visual view for further analysis. This is accomplished by selection tools, which allow users to select

data intuitively.

Existing selection tools accomplish data selection by allowing the user to differentiate an area of a visual view, thereby selecting data items displayed within that area. Wills [101] comprehensively describes different possible combinations for the selectors in his selection calculus.

Though there are many different selection tools, they can be classified into two types: lassos and brushes. A lasso (often a “rubber-band” rectangle, or free hand drawing region) is defined by drawing it onto the plot. When the shape is complete, the selection operation is performed. A brush is a pre-defined region, typically a rectangle. With a brushing tool, user moves the brush to different positions on the display, which selects elements touched by the brush.

Brushing is a widely used mechanism for interactively selecting a subset of data in a visual view. The principles of brushing were first explored by Becker and Cleveland [7, 23] who used 2D rectangular brushes to highlight, label, and delete data points in matrices of linked scatter plots. Since then, a variety of brushing techniques and theories have been developed.

Various brushing techniques have been introduced, such as the N-dimensional, multiple, and composite brushes in XmdvTool [67, 96]; angular brushing for parallel coordinates [42], and similar techniques [51, 78, 95, 104]. Chen [20] also provides “compound brushing”, where multiple brushes can be joined via various logical operations and expressions (e.g. AND, OR, XOR, NOT etc) in a graphical data-flow language.

However, direct data selection on visual view is limited to spatial position. Lassos and brushes merely specify a region. Direct selection by other visual features such as colour, size and shape are not common. Some visualization applications [90, 93], Polaris [90], for example, use legends to explain the meaning of visual features used in a visual view. By brushing on the legends, the items displayed using the chosen visual properties are selected. To make data selection more general and efficient, Heer [47] uses a “query relaxation” technique, allowing users to expand their selections based on an initial selection. For example, a user may point to a blue square in the visual view and refer to “select all items sharing the same visual property like this one”. This approach enables generalized selections based on the properties of the item.

Most existing selection tools are direct manipulation mechanisms, because it is easier to indicate sets of objects by direct manipulation than by writing Boolean ex-

pressions. Moreover, in existing systems, specifying complex Boolean combinations of multiple selections is not easy. Although some work, such as Chen’s compound brushing [20], allows logical combinations of multiple selections, explicit logical operators are required. The visual displays of logical combinations are restricted to very simple expressions. With these systems, users face the same difficulty as with Boolean query specification. At the same time, only exact results of logical combinations are highlighted. Therefore, users face the problems they encounter when analysing query results.

When the user indicates interest in a subset, the degree of interest is a binary variable - either interested or uninterested. The user’s indication of interest is a mapping from the visual domain to  $\{0, 1\}$ . Based on the analysis, data selection in a visual view is no more than querying a visual view.

For instance, selecting a set of data by dragging out a bounding box on a scatter plot is a simple query on features mapped to spatial position. All existing lasso and brush tools are query mechanisms on position features. Therefore, two basic problems need to be solved for querying on visual views. First, how can users specify logical combinations of selections easily and efficiently? And second, what visual feedback should be provided for multiple selection? Answers to these questions will result in new interactive data selection techniques.

## **2.6 Formal Models of Visual Information Exploration**

The state-of-the-art of visual query interface design consists of many successful visualization techniques. But the many aspects of visual information exploration, visual representation of queries, visualization of query results, interactive data selection in visual views, have been studied separately. The intrinsic connections between them have not been described formally.

The lack of formal description inhibits the development of new representations for queries, and natural integration of visual query specification with query result visualization.

This section describes and analyses portions of the design space so as to understand the differences and similarities among designs and suggest new possibilities. After

reviewing the many existing solutions, I found that most visual query languages and interfaces have intrinsic similarities, which can be described within a single model.

This thesis presents a formalism that captures the entities in query/visualization environment in a single conceptual framework. The formalism fills the gap between the research work in different areas, describing most existing methods of visual querying, and allowing the designer to reason about them more rigorously.

Prior to my work several formal models of visual information exploration existed. The Polaris formalism [89] designed by Stolte describes table-based graphical representations of relational databases. It is capable of compiling visual specifications automatically into SQL queries and drawing commands that generate the display. This ability enables designers to design systems that closely integrate analysis and visualization. The Polaris formalism focuses on the generation of visualizations.

Lee’s visual database exploration (VDE) model [59] uses a directed graph structure to model the visual data exploration process. Vertices in the graph represent the state of the visualization while edges are relationships between states. The evolution of the states can be easily traced back to the database. The P-Set model of visualization exploration [54], designed by Jankun-Kelly, describes the process of visualization exploration and provides a framework to encapsulate, share, and analyse visual explorations. Both the VDE model and the P-Set model concentrate on describing the process of visual data exploration.

All these formal models emphasize on data visualization but overlook query visualization, which is equally important in visual information exploration environment. Critical issues in information exploration such as Boolean query specification and representation mechanisms are not addressed.

The formalism presented in this thesis describes all the essential components, together with their formal similarities and how they interact in a visual information exploration environment. All the entities in the formalism are defined as tables. Thus the querying and visualizing operations can be recursively applied on any component in the environment. It explains how visualization and interaction techniques can be used in designing user interface tools that help users to explore and analyse data. For example, a query can be visualized as a multi-dimensional data set using existing visualization techniques, which results in many interesting visual query representations.

Having formal models for visual information exploration is beneficial. Because

the formal models are general, they can be applied to a wide variety of visualization problems. They can be used to help researchers understanding, designing, or analysing visual information exploration systems.

# Chapter 3

## A Formalism for Visual Query Interfaces

This chapter presents a formalism of the basic components and behaviour of visual query interfaces. The components include data, queries, query results, and visual representations of them. Each component is defined as a table. The formalism is augmented by mappings between components, visualizations and queries, which are also defined as tables. The formalism is novel in that all entities may be either visualized or queried. In these terms a visual query interface is the application of visualization to queries, and selecting data from a visual view is the application of a visual query on a visual representation of data. The formalism supports recursive visualization, visualizing a visualization, and recursive querying, querying a query. It also provides a natural visual representation of the information exploration process. Most important, it unifies querying and visualization within a single framework that supports the analysis and design of visual query interfaces.

### 3.1 The Table Model

As illustrated in Figure 2.2, high-level descriptions of data visualization define it as a pipeline, through which a set of mappings transform raw data onto one or more visual displays. The raw data, which is to be explored, comes in many forms. It is usual to structure the data prior to visualization, normally as a table [15].

### 3.1.1 Definition of the Table Model

A data set is tabularly defined by listing the attribute values of each data item in the set. That is, each data item is a tuple of attribute values, as shown in Table 3.1, the collection of tuples being a relation.

	<b>Variable 1</b>	<b>Variable 2</b>	<b>...</b>	<b>Variable n</b>
Item 1	$Value_{11}$	$Value_{12}$	...	$Value_{1n}$
Item 2	$Value_{21}$	$Value_{22}$	...	$Value_{2n}$
...	...	...	...	...
Item m	$Value_{m1}$	$Value_{m2}$	...	$Value_{mn}$

Table 3.1: The Table Model.

The formal model of a table has the following features.

- An item set, or table, is a set of Items,  $\{item_1, item_2, \dots, item_m\}$ , with each item  $item_i$  a tuple of attribute values. Each item in a table has the same set of attributes, which differ in value from item to item.
- Attribute values are elements of value sets  $A_j$ . The value of attribute  $j$  of item  $i$  is written  $value_{ij} = item_i.Attribute_j$ .  $value_{ij} \in A_j$ .
- The item space  $IS$  is the set of all possible items that can appear in an item set. The item space is the Cartesian product of the value sets of all attributes in an item set:  $IS = A_1 \times A_2 \times \dots \times A_n$ .
- The set space is the power set of the item space, set of all item sets.
- The relationships between an item set, its item space, and its set space are: the item set is a subset of the item space; the item set is an element of the set space; and the item space is an element of the set space.

One representation of an item set, a very useful one, is a join of the item space with a Boolean attribute that indicates whether or not a particular item is in the item set.

In a pure relational model, duplicate tuples are not allowed. But in information exploration, there might be multiple tuples with identical values of the attributes. In order to accommodate duplicate tuples, the formalism defines them as instances of the same item. When it is important to know the number of instances of an item in

the item set, the Boolean-valued attribute can be replaced with a cardinal-valued one indicating how many copies of that item exist.

Many visualization systems utilize similar relational data models. The key aspect of the table model presented here, distinguishing it from existing ones, is that it explicitly introduces the concepts of item space and set space, which are central concepts of the formalism. The item space has been introduced as a property of a single item set. But data exploration environments are designed for a universe of data sets, any one of which may be visualized and queried. It makes sense to define the set space as a property of a set of item sets, all of which have the same attribute structure. Therefore, the set space is a natural domain for query and visualization mappings.

Tables also provide a basis for defining queries and visualizations. I also show how tables can be used to calculate the basic operations of information exploration, visualizing and querying.

### 3.1.2 Applying Tables to Tables

To describe data visualization using tables it is necessary to define visual structures as tables, and to describe how tables are applied to tables.

Figure 3.1 depicts the basic process of applying tables to tables. A discrete mapping is a table composed of two parts: keys and values. When applying the mapping table on an item set, the items are used as keys in the mapping table, the resulting item set is retrieved by looking up the associated values of the input items.

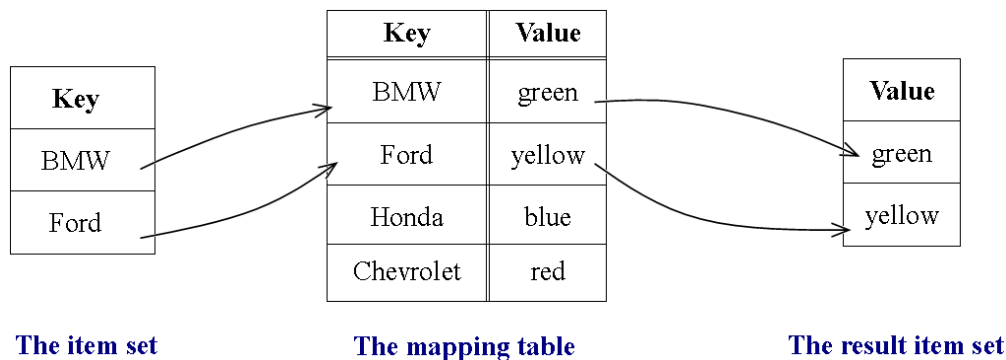


Figure 3.1: Applying a Table to a Table

Assume a mapping  $m$  is defined as a table. The keys in the table define an item set



$I$ . The table can be applied to any subset of  $I$ , therefore the domain of the mapping  $m$  is  $P(I)$ , the power set of  $I$ . Similarly, extracting the values from the mapping table will get an item set  $O$ . When applying the mapping on an input item set, the result item set is always a subset of  $O$ . therefore the codomain of the mapping  $m$  is  $P(O)$ , the power set of  $O$ . Then the mapping is denoted  $m : P(I) \rightarrow P(O)$ .

### 3.1.3 Data Visualization by Tables

In the formalism, a visualization is defined by a mapping table, and this definition subsumes most formal definitions that have been used in previous visualization models. That is, a visualization is a mapping from data items to visual presentations, which associates each item in the data set with a tuple of visual qualities. These visual features, combined in a single entity, are called an “icon”. The visualization mapping is formally defined as  $m_{vd} : DS \rightarrow VS$ , with domain  $DS$ , the data set space, all possible data sets; and codomain  $VS$ , the visual representations space, which includes all possible combinations of icons.

Similar data visualization mappings have been explored formally in many visualization models and systems, such as Lee’s GDE model [60], the APT system [65], Rivet [13] and DEVise [63]. Compared with existing ones, the visualization mapping in this formalism has a more rigorous mathematical foundation because it is a mapping from set space to set space, instead of using simple item sets as domain and codomain. It allows researchers to analyse the data visualization environments on a higher level.

Furthermore, in the formalism, a visualization mapping itself is defined as a table too, with elements of the item space as keys and the associated visual presentations (the icons) as data. (As described, the table is normally a poor implementation strategy, but any implementation is formally equivalent to the table.)

Each icon is a combination of visual features. The concept “visualization term” is used to denote a mapping that maps a data attribute to a single visual feature. A visualization mapping that uses multiple visual features to encode information is defined as the Cartesian product of its visualization terms. A visualization table is applied to a data set with the data items as keys for retrieving icons. Figure 3.2 shows an example of a visualization mapping.

The above analysis shows that all entities in a data visualization: the data, visual representation of the data, and the visualization mapping, can be described as

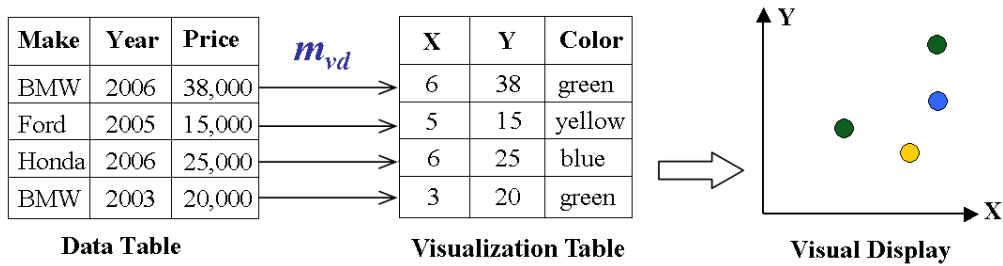


Figure 3.2: Example of the Data Visualization Mapping

tables. Treating data visualizations as applying visualization tables to data tables is the essential principle of the formalism.

### 3.1.4 Table Model for Queries

Many techniques exist for visualizing elements of data spaces. These techniques can be used to visualize queries so that query visualizations are tightly integrated with data visualizations. Section 2.2 showed how to visualize tabularly defined data. It is now necessary to define queries tabularly.


Tabular definition of Boolean queries is based on truth tables, which, as described in Section 2.4.3, are a canonical form of Boolean expressions: any Boolean expression using  $N$  unique terms can be written as the disjunction of no more than  $2^N$  query fragments. For example,

$$T_1T_2 + T_1T_3 + T_2T_3 = T_1T_2T_3 + \overline{T_1}T_2T_3 + T_1\overline{T_2}T_3 + T_1T_2\overline{T_3}$$

Any Boolean expression can be defined using a truth-table. Each row in a truth-table is related with a query fragment. In a truth-table, the last column indicates the value of the Boolean expression. “true” if the query fragment is included in the expression, “false” otherwise. Here I use “isTrue” to be the title of the last column, use 1 to represent the “true” value, and use 0 for “false”. The truth-table shown in Figure 3.3 represents the Boolean expression  $T_1T_2 + T_1T_3 + T_2T_3$ .

This construction gives queries the same formal structure as data sets and visual presentations. Based on the table model,  $Q$  is defined as the set of all the query fragments included in a truth table. A particular truth-table defines a specific Boolean query function. The query space  $QS$  is defined as the set of all the possible queries

	$T_1$	$T_2$	$T_3$	isTrue
$\bar{T}_1\bar{T}_2\bar{T}_3$	0	0	0	0
$T_1\bar{T}_2\bar{T}_3$	1	0	0	0
$\bar{T}_1T_2\bar{T}_3$	0	1	0	0
$\bar{T}_1\bar{T}_2T_3$	0	0	1	0
$T_1T_2\bar{T}_3$	1	1	0	1
$T_1\bar{T}_2T_3$	1	0	1	1
$\bar{T}_1T_2T_3$	0	1	1	1
$T_1T_2T_3$	1	1	1	1



$$T_1T_2 + T_1T_3 + T_2T_3 =$$

$$T_1T_2\bar{T}_3 + \bar{T}_1T_2T_3 + T_1\bar{T}_2T_3 + T_1T_2\bar{T}_3$$

Figure 3.3: Example of a Truth-Table

that can be formed.

### 3.1.5 Visualizing Queries

A query visualization is a mapping from queries to visual presentations where each query fragment associated with a set of visual qualities. This is described as the mapping  $m_{vq} : QS \rightarrow VQS$ , with domain  $QS$ , the query space, and codomain  $VQS$ , the query visualization space.

Figure 3.4 presents a novel visualization of a 3-dimensional query table. Each query fragment is represented as an icon with coloured petals. A unique colour, orientation and position is assigned to each petal. The presence of a petal is determined by the value of the related attribute. If the *isTrue* value of a query fragment equals to 1, then the icon is surrounded by a blue circle. For example, if a query fragment is  $\langle 1, 1, 0, 1 \rangle$ , then its associated icon has a red petal, a green petal, and is surrounded by a blue circle. In this example, the query is  $T_1T_2 + T_1T_3 + T_2T_3$ . It is a disjunction of four query fragments:  $T_1T_2\bar{T}_3$ ,  $T_1\bar{T}_2T_3$ ,  $\bar{T}_1T_2T_3$ , and  $T_1T_2T_3$ . Therefore, only the icons associated with the four query fragments are surrounded by blue circles.

Such visual representations can be used for specifying Boolean queries: a set of query fragments is presented and a user needs only to find the fragments that meet her information requirement and select them. Thus, specifying Boolean queries is transformed into visual search, which eliminates the explicit specification of logical operators. TEBI [39], InfoCrystal [88] are examples of such interfaces.

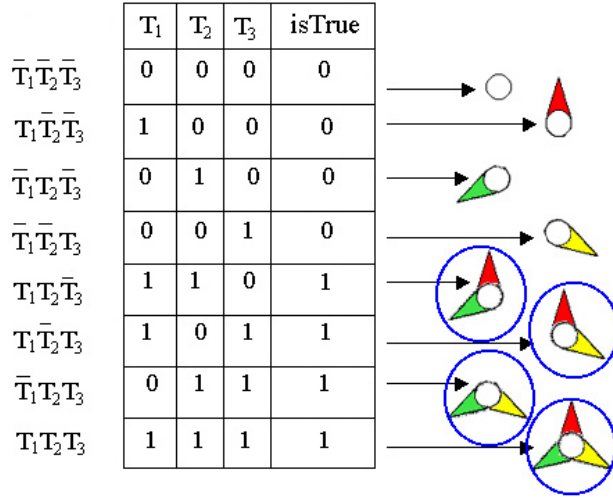


Figure 3.4: Iconic Representation for Queries

Since queries are defined tabularly, techniques for visualizing multi-dimensional data can be applied to visualize them. Figure 3.5(a) shows an example of nested coordinate representation for a 4-dimensional query fragment space. A  $t$ -dimensional query fragment space is represented as a binary tree, the height of which is  $t$ . Each query fragment is a leaf node. Therefore, techniques for visualizing trees suffice to visualize queries. For example, Figure 3.5(b) uses a treeMap to visualize a 3-dimensional query fragment space.

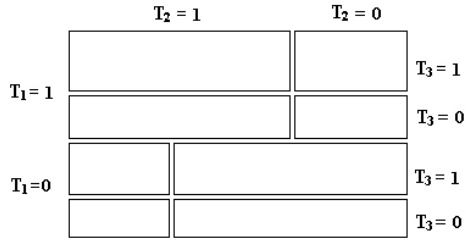
Furthermore, two query fragments that differ in only one attribute can be combined to form a simplified expression. For example,  $T_1T_2\bar{T}_3 + T_1T_2T_3 = T_1T_2$ . Gray code visualization techniques use this property: fragments differing in a single attribute are always adjacent, which aids query simplification.

Figure 3.5(c) shows a Karnaugh map [56], a 2-dimensional tabular representation of a truth-table that facilitates management of Boolean algebraic expressions. Each cell stands for a single query fragment. The rows and columns of the map are ordered according to the principles of Gray codes: a single variable changes value between neighbouring cells. This tabular representation is familiar and intuitive to use. The regular tabular layout of the cells makes the Karnaugh map easy to view, navigate, and interact with. Figure 3.5(d) shows a circular representation based on Gray codes.

In Figure 3.5(b) and 3.5(d), the cells differ in size, which encodes extra information, such as the number of data items associated with each query fragment.

$T_1 = 1$						$T_1 = 0$					
$T_2 = 1$			$T_2 = 0$			$T_2 = 1$			$T_2 = 0$		
$T_3=1$		$T_3=0$									
$T_4$											

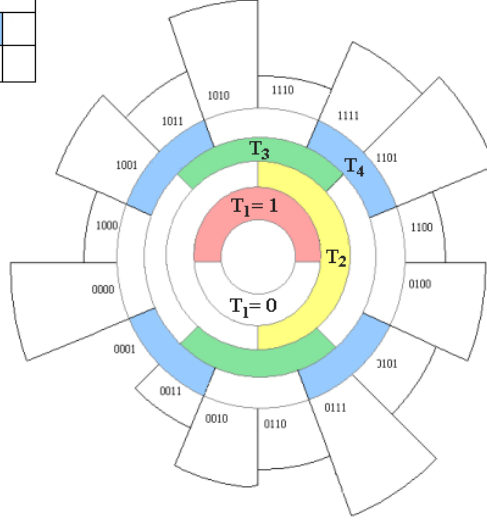
(a) Nested Coordinate Representation



(b) TreeMap Representation

		$T_2 = 0$		$T_2 = 1$	
		$T_3 = 0$		$T_3 = 1$	
$T_1=0$	$T_1=0$	$T_3 = 0$	$T_3 = 0$	$T_3 = 1$	$T_3 = 1$
$T_1=1$	$T_1=1$	$T_3 = 0$	$T_3 = 0$	$T_3 = 1$	$T_3 = 1$

(c) Karnaugh Map



(d) Circular Gray Code Arrangement

Figure 3.5: Examples of Query Visualization

## 3.2 The Transformation Model

So far, four objects have been introduced:

- $DS$ : the space of data sets, on which visualizations, or queries can be created,
- $VS$ : the space of visual representations for data sets,
- $QS$ : the space of Boolean queries on the data sets, and
- $VQS$ : the space of visual representations for queries.

Each element of these spaces can be represented as a table.

Two visualization mappings have also been introduced:  $m_{vd}$  defines visualizations on the data space, and  $m_{vq}$  defines visualizations on the query space, both of which can be represented as tables.

To model formally the connections between the concepts, I use the transformation model shown in Figure 3.6, which uses diagrams like those that appear in category theory. It depicts the mappings between data, queries and visual representations. Two new mappings are introduced into the model:  $m_{qd}$ , which defines queries on the data space, and  $m_{qv}$ , which defines queries on the visual representation space.

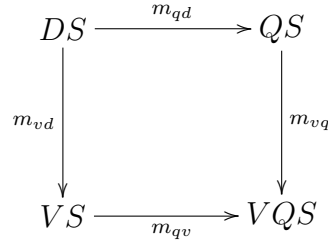


Figure 3.6: Transformation Model of Visual Query.

### 3.2.1 Query on Data Space

Query specification consists of two steps: (1) specifying the query terms, and (2) specifying the query structure. In the formalism, a query term is a table indexed by item attribute values, with a single Boolean-valued attribute. The query structure is a truth-table. If a query is composed of several independent query terms, then the query mapping table is the Cartesian product of the query term tables joined with a truth-table.

Therefore, the query mapping table for  $m_{qd}$  is a table indexed by data items, each item associated with a query fragment. The *isTrue* value of a query fragment indicates whether or not the items associated with it are selected by the query.

Figure 3.8 presents an example of a query which is composed from 3 query terms:

$$T_1 = (\text{Make} = \text{BMW}), T_2 = (\text{Size} = \text{small}), T_3 = (\text{colour} = \text{black}).$$

The logical combination of the terms is “meet at least two of the three query terms”. If a data item is  $\langle \text{BMW}, \text{full}, \text{black} \rangle$ , then its associated query fragment is  $\langle 1, 0, 1, 1 \rangle$ . The *isTrue* value is 1, indicating the item meets the query.

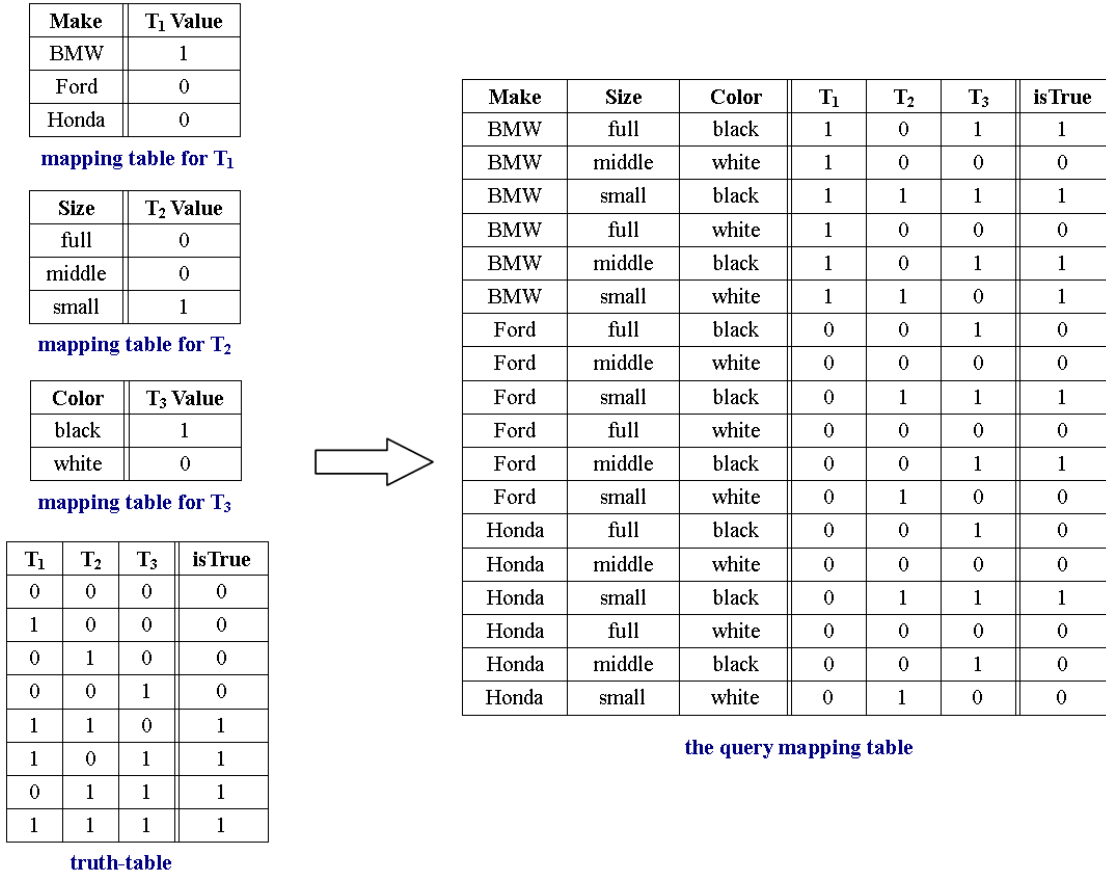


Figure 3.7: Example of Data Query Mapping Table

Applying the query mapping table to a data item set results in a query fragment set. Sometimes it is important to know how many data items are associated with a query fragment. Then the query fragment set is represented as a join of the truth-table with a cardinal-valued attribute indicating how many copies of that fragment exist. Figure 3.5(b) and 3.5(d) are examples of visualizing the query fragment set in this form.

### 3.2.2 Visual Query on Visual Representations

The mapping  $m_{qv}$  defines visual queries on visual representation spaces: it specifies both queries on visual features and visualizations for the queries.

In the transformation model, from the data space  $DS$ , there are two ways of defining  $VQS$  by composition.

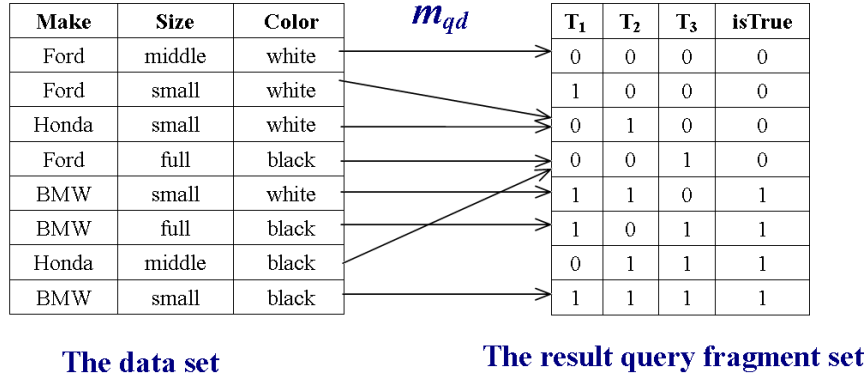


Figure 3.8: Example of Applying Query Mapping on a Data Set

$$(1) \quad DS \xrightarrow{m_{qd}} QS \xrightarrow{m_{vq}} VQS$$

$$(2) \quad DS \xrightarrow{m_{vd}} VS \xrightarrow{m_{qv}} VQS$$

It is reasonable to restrict the table so that each element of  $DS$  maps to only one element of  $VQS$ , which means the four mappings in the model obey the following identity.

$$m_{qv} \cdot m_{vd} = m_{vq} \cdot m_{qd} \implies m_{qv} = m_{vq} \cdot m_{qd} \cdot m_{vd}^{-1}$$

In the above expression, the composite function  $m_{qd} \cdot m_{vd}^{-1}$  is a Boolean query that maps visual presentations to Boolean values, while  $m_{vq}$  maps Boolean queries to visual presentations. Therefore  $m_{qv}$  defines visual queries on the visual representation space. It specifies not only queries on visual features but also visualizations of the queries. Of course, the visualization function  $m_{vd}$  may not be one-to-one, in which case it induces equivalence classes on  $DS$ , which must be respected if the visual query is to be well-defined.

The above analysis describes an approach for interacting with visual representations: first specify constraints on visual attributes, then map the visual items to another set of icons based on the query visualization function.

In Figure 3.9, data items are visualized as dots of differing size. The user has dragged out a rectangular box in the dot size list, which specifies a query term on the size attribute. Then the selected dots are assigned a colour to differentiate them from the others. The colour-coding is the visual mapping of the query. This is an



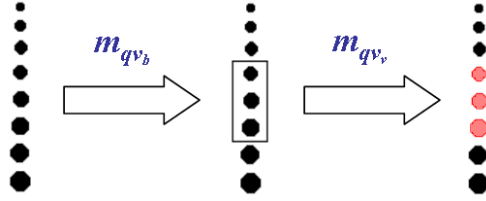


Figure 3.9: Example of Visual Query on Visual Features - Brushing

example of brushing, a widely used technique for interactively selecting a subset of data in visualization applications [67, 78].

This formal definition of  $m_{qv}$  shows that the essence of making a data selection within a visual view is making a query on visual attributes. Each individual selection (or brushing operation) is a visual query term. Logical combinations of multiple selections are specified using the query specification method derived from the formalism. An example of compound selection is shown in Figure 3.15.

### 3.3 Query Result Visualization

Tight coupling of queries with results is important in exploratory search [66, 87]. But how should the coupling be implemented? Are there formal descriptions that are useful for designing new techniques of query result visualization? The formalism provides an answer.

#### 3.3.1 Traditional Method of Query Result Visualization

In the formalism, the term “exact query result”  $R_d$  is used to denote the set of data items that exactly meet a query.  $R_d$  is composed of only the data items associated with the query fragments having *isTrue* equal to 1.

In most existing query interfaces, when a user specifies a query, only exact query results,  $R_d$ , are returned and presented. With large and unfamiliar data sets, users find it difficult to find the desired data with such interfaces.

In the example presented in Figure 3.10, the Boolean query is **(Make = BMW) OR (Mileage < 50000)**, which is composed of two query terms. The data items that meet either query term are displayed. But the display provides no insight into which

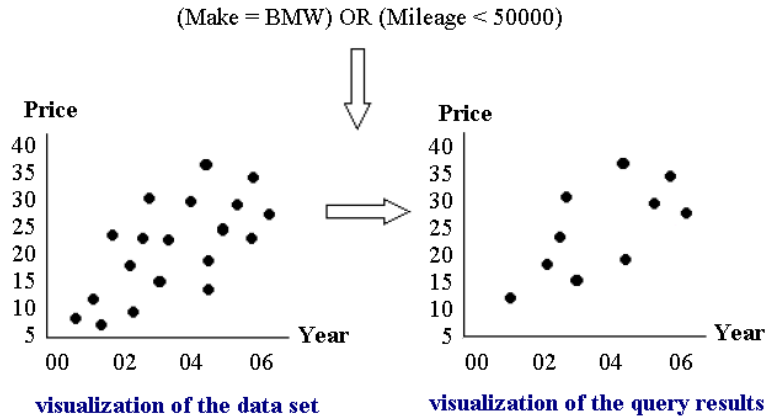


Figure 3.10: Visualizing Query Results

data item meets which query term or how many data items meet both the two query terms.

Sometimes the user wants the relationship between data items and query terms. In such cases the above display provides little support for further data analysis.

### 3.3.2 Example: Enhanced Visualization of Query Result

However, the visualization in Figure 3.10 can be enhanced, as shown in Figure 3.11. Icons are coloured by query fragment, an addition to the original data visualization. In this example, the icons have two sets of attributes: the original visual attributes (x-axis and y-axis positions), and the visual coding by query fragment. Thus, the visual representation shows how the data is distributed by query term so that users can better understand the result.

### 3.3.3 The Transformation Model of Query Result Visualization

The formalism provides a foundation that is useful for designing new techniques of query result visualizations. To define query result visualization formally, the **query result item set**  $R$  is defined as the join of a data item set  $D$  and the query  $Q$  associated with it. Suppose  $r_i$  is an item in  $R$ , then  $r_i = \langle d_i, q_j \rangle, d_i \in D, q_i \in Q, q_j$  is the associated query fragment of  $d_i$  based on the query mapping  $m_{qd}$ .

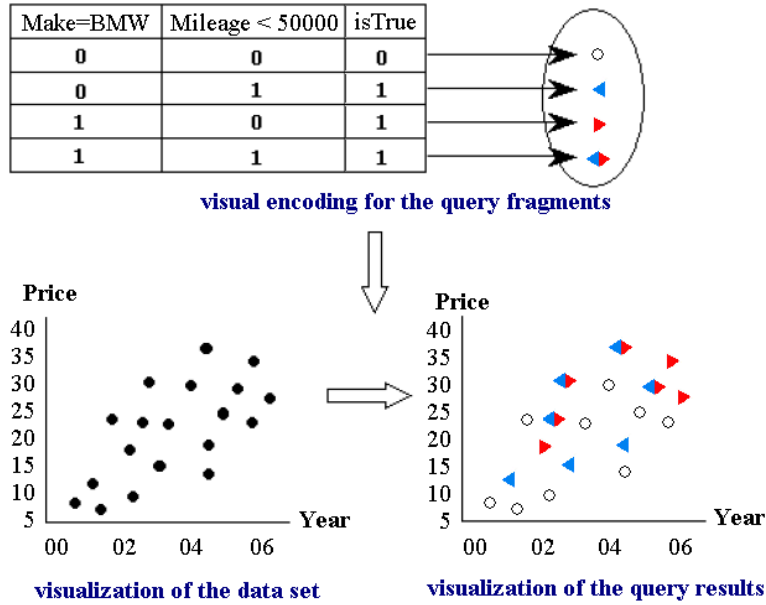


Figure 3.11: Visualizing Query Results

Compared with the “exact query results”  $R_d$ ,  $R$  shows explicitly the relationship between the data and the query. Therefore, visualizing  $R$  provides good feedback to the user.

The visual representation of  $R$  is denoted as  $VR$ , which is a set of icons, one for each query result item.  $R$  consists of two parts: a data item set, and a query. When visualizing  $R$ , the visualization mappings  $m_{vd}$  and  $m_{vq}$  are applied. Therefore,  $VR$  is the join of two visual sets:  $m_{vd}(D)$  and  $m_{vq}(m_{qd}(D))$ . That is,  $VR$  is the join of the visual representation of data and the visual representation of the query specified on the data. Since  $m_{vd}(D) = V$ , and  $m_{vq}(m_{qd}(D)) = m_{qv}(V)$ ,  $VR$  can also be obtained by joining the visual representation of data and the visual query on the visual representation.

Here  $DS \xrightarrow{m_{qd}} RS$  is used to denote the mapping from  $DS$  to  $RS$ .  $RS$  is the query result space, the set of all possible query result item sets. The model in Figure 3.12 describes the mapping relationships between the data set space  $DS$ , the data visual representation space  $VS$ , the query result space  $RS$ , and  $VRS$ , the visual representation space of query results.

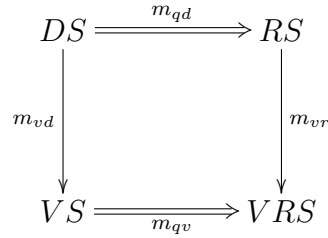


Figure 3.12: The Model of Query Result Visualization.

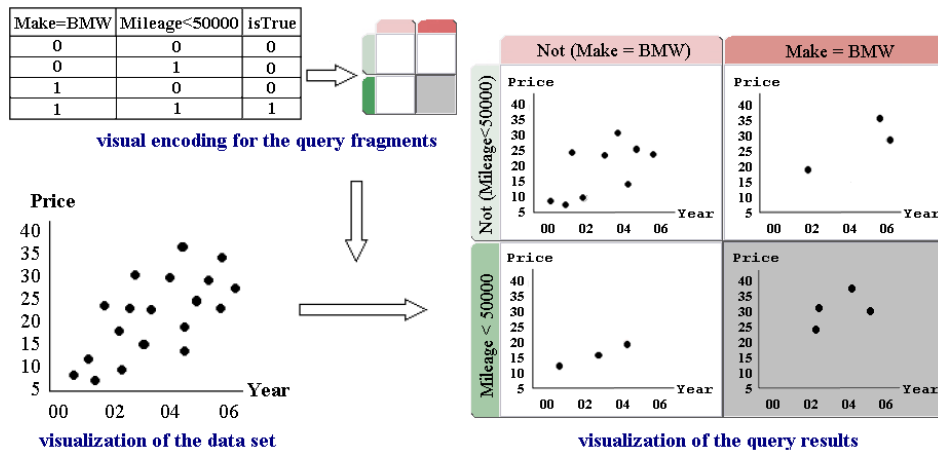


Figure 3.13: Visualizing Query Results

### 3.3.4 Examples of Query Result Visualization

Based on the model, query results can be visualized, showing simultaneously both data set and query structure. In Figure 3.11, icon codings for query fragments are appended to the data visualization. Figure 3.13 uses a Karnaugh map to visualize the query space. Items associated with a specific query fragment are displayed in the corresponding cell. Each cell is a small visual view showing the icons of the data items.

In Figure 3.14, data items are visualized as dots differing in size. The user is interested in a specific range of size, and selects the range by brushing on the dot list. Then the selected dots are assigned a colour different from the others. Such a colour-coding mechanism defines a visual mapping for the query based on the visual attribute (dot size). Then all the items within the size range are displayed in that colour in the

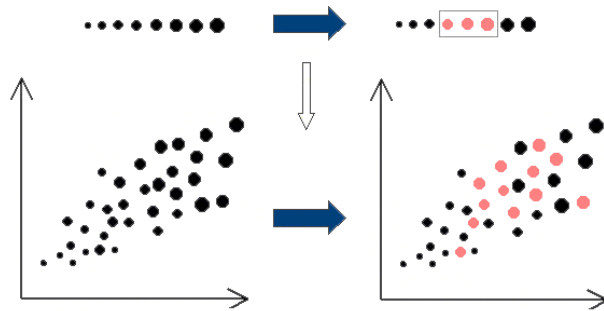


Figure 3.14: Visualizing Query Results

visual representation of query results.

In multiple view visualizations with linking, brushing in one view colours objects in other views. Brushing in a view is a visual query, while the linked views are visual representations of the query results. Integrated with query structure specification mechanisms, Boolean combinations of multiple brushing operations, compound brushing [20], are easily supported. For example, in Figure 3.15, selections on the small x-y-plots are query terms. Logical combinations of selections are specified by operating on the Karnaugh map at the bottom right, which is also a legend for the visual coding of the query fragments. Visualizing compound selection is an instance of query result visualization.

## 3.4 Visualizing the Mappings

In the formalism, all the mappings in the query/visualization environment are defined as tables. Therefore, the mappings themselves can be visualized.

### 3.4.1 Visualizing the Visualization Mappings

From maps to modern visualization systems, there are many examples of visualizing the visualization mappings. They are commonly called “legends”.

Map legends are traditionally provided to explain the content of a map. According to the Kansas Association of Mappers (2005), the legend is defined as “An explanation of the symbols, codes, names given to variables and other information appearing on a map drawing or chart. It includes a sample of each symbol, line pattern, shading,

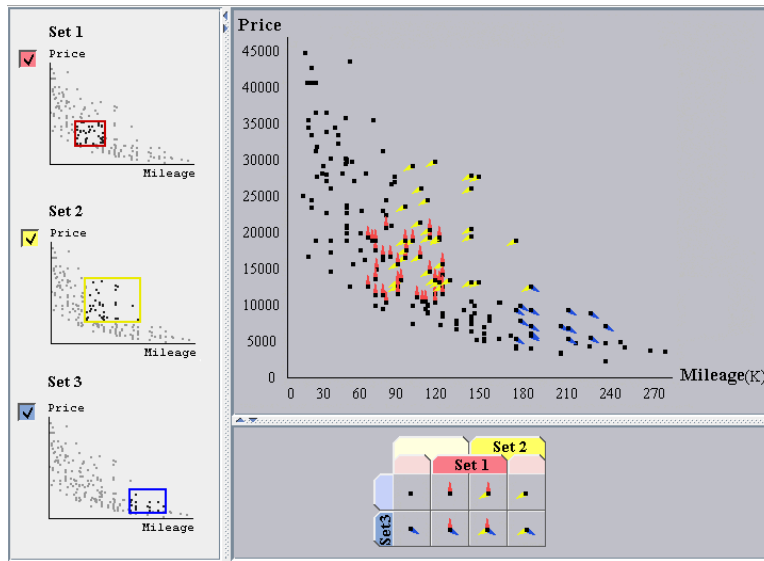


Figure 3.15: Visualizing Query Results

or hatching appearing on the map along with annotations describing the meaning of each.” In other words, it is an important map feature - the symbol used to interpret a map that lists and explains the map elements.

Figure 3.16 presents a temperature map where colours are used for visualizing temperatures. In the map legend, all the possible temperature values from the data space and their associated colour values are ranked and presented, even though only a few colours appear in the map.

In Figure 3.17, each icon comprises two visual features: colour and shape. Two legends are provided explaining the meaning the visual features.

In the formalism, a legend is itself a graphical representation of the visualization mapping. It depicts the mapping relationship between the data attributes and the the visual features that constitute the visual representation space.

Legends can be interactive [85, 93]. Generally, an interactive legend has two roles: presentation and interaction. Presentation depicts the current mapping from data attributes to visual features, thereby supporting user interpretation of the visualization. Interaction allows users to alter the visualization mappings dynamically, improving the presentation of information features that interest the user.

Interactive legends exist in some modern visualization systems [85, 93], but they

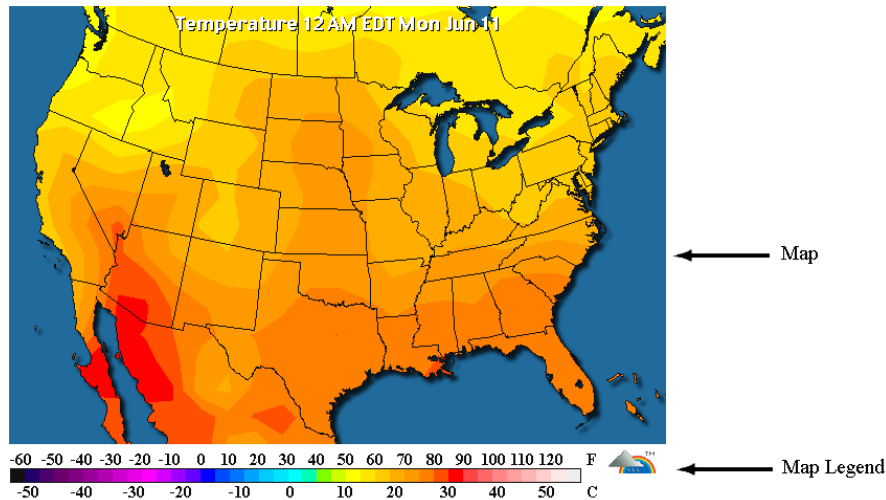


Figure 3.16: Example: Map and Map Legend

are not treated as the results of visualizing the visualization mappings. The formalism makes explicit the formal identity, which helps users and designers to understand and enhance them more effectively, and with less implementation effort.

### 3.4.2 Visualizing the Query Mappings

Similarly, query mappings can be visualized. Such visualization includes two parts: truth-table visualization and query term visualization. The first part was described in section 2.4; this section focuses on the second.

Query term visualization examples include UI widgets used to control visual representations of Boolean queries, including sliders, brushing histograms and toggle buttons. They are sometimes called “query devices” [2], especially when they are combined with graphical data representations such as dynamic query interfaces. Like interactive legends, query devices have two roles: presentation and interaction. For example, a rangeslider visualizes 1-dimensional attribute values and uses draggable buttons to specify range limits. Selected values are visualized in place, using, for example, different colours than unselected ones.

Rangeslider-like widgets can be created to answer questions such as: what information is to be visualized? how should the attribute values be arranged? or what interaction mechanisms are provided? Figure 3.18 shows a few examples. The attribute values

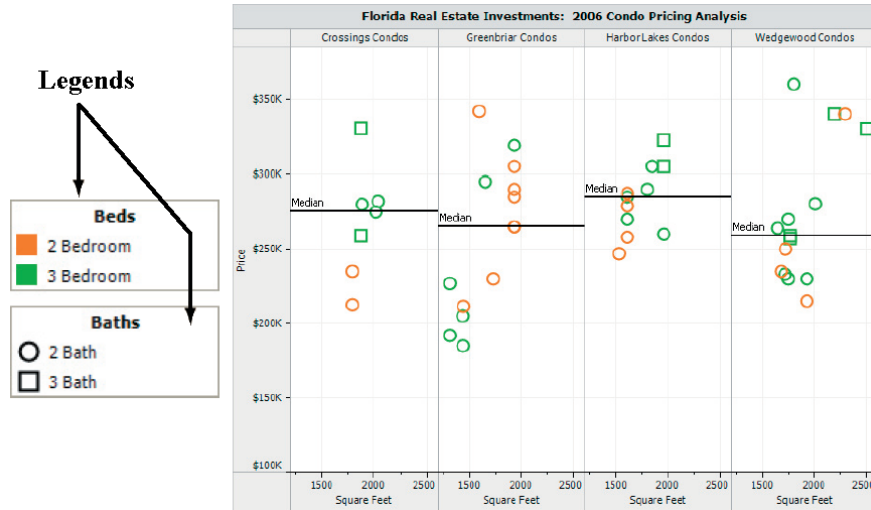


Figure 3.17: Example: Legends for Visualization

can be mapped on a semicircle, on a spiral curve, or on a straight line. Data visualization sliders visualize data distribution along the attribute values. Scented widgets [100] add visual cues to common user interface widgets such as radio buttons, sliders, and combo boxes. They enhanced the widgets with embedded visualizations that facilitate navigation in information spaces. For example, in Influence Explorer [81], the relationship between different query devices is encoded with colours. Brushing Histograms also support brushing operations for multiple value range selection.

Because query devices are graphical representations, using them for query specification is no more than making a visual query on a visual representation. In Influence Explorer [81], each slider is a visual view showing one aspect of the data set. Operating on one slider updates the others, an example of brushing and linking. Thus, the commonalities identified by the formalism make any visualization and interaction technique an idea for a new query device or for extending an existing query device.

Furthermore, integrating legends with query devices may provide more informative and powerful user interface widgets for visual queries. For example, if data items are visualized as dots differing in size, the list of dots associated with attribute values is a legend. On the other hand, the dot list also functions as a query device: brushing it accounts for specifying a query term. This mechanism allows users select items by non-spatial visual features such as colour, size and shape. Another example is using



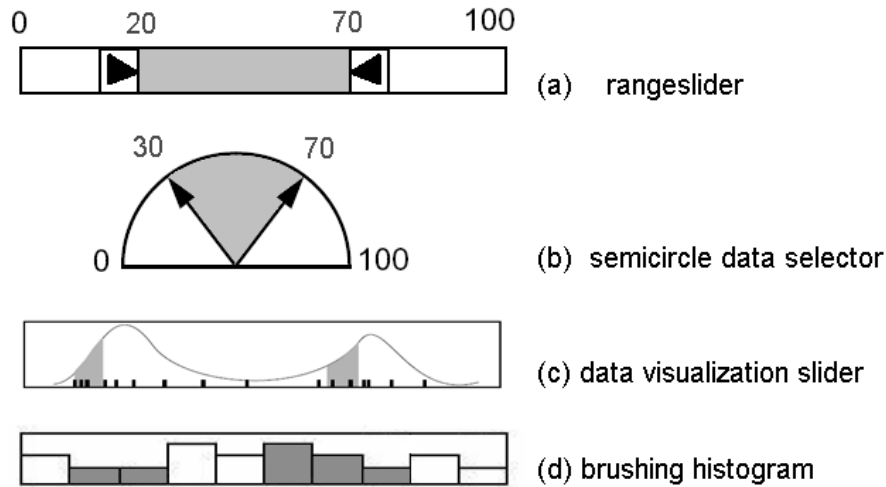


Figure 3.18: Example: Query Devices

coloured pie-chart as a legend which also provides information of the proportion of attribute values. Clicking a pie slice to select the related data items amounts to a query.

### 3.5 The Recursive Models

According to the formalism, all entities in the query/visualization environment are tables:

- Data, queries, and visual representations, and
- mappings, including visualizing and querying.

Data sets are cardinal-valued; queries are Boolean-valued; and visual representations are icon-valued. Applying one table to another, as occurs when a data set is queried or visualized, amounts to a join between the two tables.

The formalism opens up the new world of recursive mappings: a query can be queried; a visual representation can be visualized. The mappings themselves can be queried and visualized. The application of queries and visualizations to themselves and each other occurs frequently and informally within the brain of the user: the formalism makes it explicit, so it can be put into the computational environment.

### 3.5.1 Visualizing Visual Representations

Figure 3.19 illustrates the application of the transformation model to the space of visual representations. It shows how visualization and query mappings can be defined as easily on visual representations as on data.

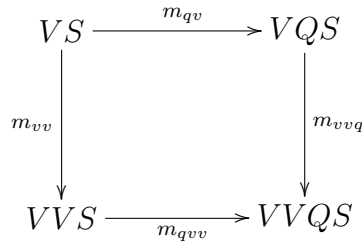


Figure 3.19: The Model of Visualizing Visualizations

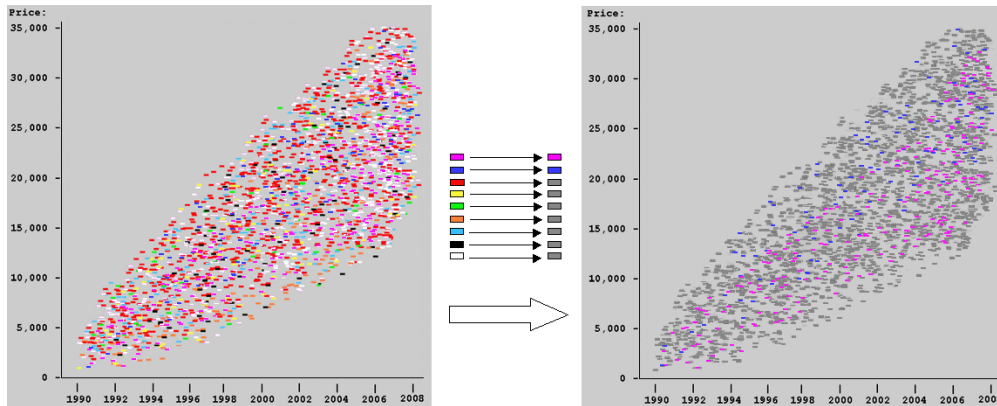


Figure 3.20: Example: Visual Encoding for Visual Attribute

Figure 3.20 shows a concrete example. Suppose the visual representation includes thousands of icons. A user is interested in some of them, such as those that are purple or blue. She wants to detect and compare their distributions as well as comparing them with the other items, which is hard in the given colour rich representation.

However, she can define a new visual mapping  $m_{vv}$ : if the original colour is purple then the new colour is still purple, if the original colour is blue then the new colour is still blue, otherwise the new colour is grey. As desired, the purple dots and the blue dots

stand out in the new visualization. At the same time, the other icons are still displayed to provide context. This method allows reduction of the visual complexity. Figure 3.20 shows the result of visualizing a visual representation, and gives a visualization of the visual mapping itself.

### 3.5.2 Querying on Queries

Similarly, the query space is also defined based on the table model, so that querying the query space is identical to querying the data space.

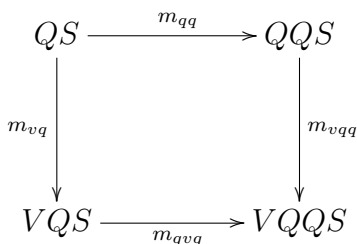


Figure 3.21: The Model of Query on Queries

Figure 3.22 presents an example of querying a query by specifying query terms on the query space. Suppose there are 6 query terms:  $T_1, T_2, \dots, T_6$ . The query specified based on these 6 query terms looks a little complex. Suppose two new query terms are defined on the query space as:  $T'_1 = T_1(T_2 + T_3)$ ,  $T'_2 = T_4T_5T_6$ , then the new query representation is based on only two query terms, which is easier to understand.

Querying a query appears little in information exploration systems: the importance of supporting it explicitly is a contribution of my research. In the information exploration cycle queries evolve as users refine them, often to the point of unintelligibility. Then querying the query is a means of asking questions about it and understanding it.

### 3.5.3 The Iterative Model

Once recursion is established it can be repeated as often as desired, visualizing a query about a visualization mapping, for example. With today's information exploration tools, which support only simple query and visualization mappings, these possibilities

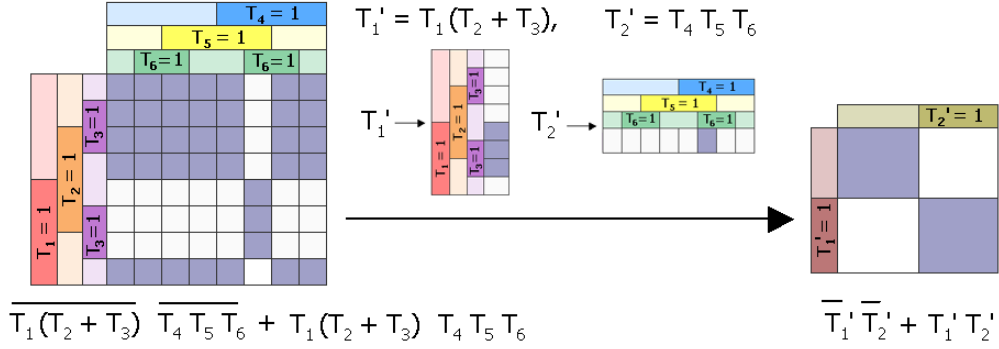


Figure 3.22: Example: Specifying Query on Query Space

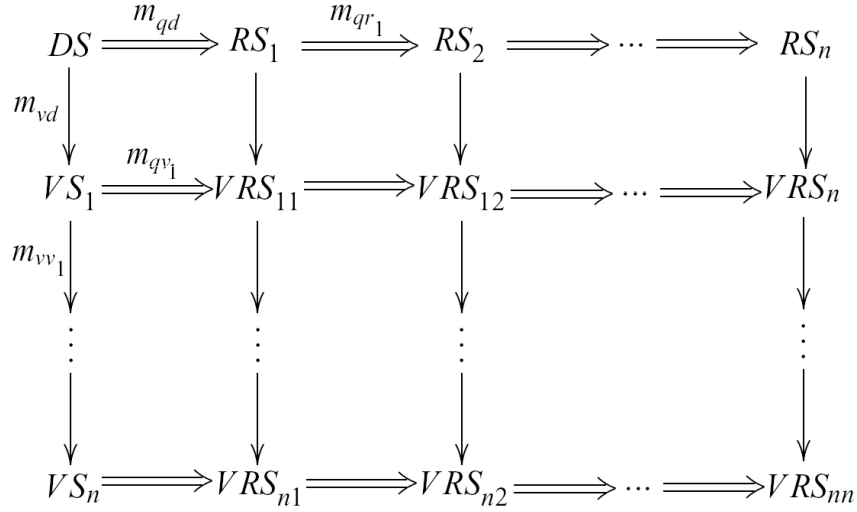


Figure 3.23: The Iterative Model

are surely overkill. But, more complex interactions between users and data are the future of information exploration, and complexity requires tools that support it. Thus, the extensibility of the formalism is sure to be important in the future.

Recursion is also possible in another dimension. Query results are also tables, and can be visualized or queried. By querying previous query results, complex data searching tasks can be decomposed into stages. It is a natural approach that most information exploration systems use. But many of them only support querying within the exact query results  $R_d$ . Thus only pure conjunctions of the queries are possible. However, the query result item set  $R$  defined in the formalism contains both data and

query. As described by the recursive model,  $R$  carries over the information of all queries that have been specified. Therefore, specifying queries on  $R$  enables arbitrary Boolean combinations of them.

In fact, the information exploration process is no more than a sequence of queries and visualizations on visualized data. A particular sequence is itself easily visualized using the mapping diagrams of the formalism. This capability is important: users often wonder how they got to a particular result. Furthermore, this is an aesthetically satisfying result of the formalism, which was created by a months-long sequence of queries and visualizations, albeit informal ones. Closure, in which the process of discovery is described using its result, is typical of computer science.

### 3.6 Summary

Prior to the formalism, complementary topics in information exploration such as data visualizations, query result visualizations, query structure visualizations, query devices, and interactive legends were studied separately.

The formalism, which is based on the table model, brings them together. All entities in the query/visualization environment are tables: data sets are cardinal-valued; queries are Boolean-valued; and visual representations are icon-valued. Applying one table to another, as occurs when a data set is queried or visualized, amounts to a join between the two tables. This is the source of the formalism's power. It makes it possible that visualization and interaction techniques in one field can be used in the others. More specifically,

- a query is visualized as a multi-dimensional data set, which results in many interesting visual query representations. Visual representations can be queried. Complex logical combinations of multiple visual selections can be implemented in the same way as querying a data set.
- query specification can be integrated with the display of query results to help users understand the distribution of data.
- the formalism defines all the mappings as tables, which makes it possible to apply existing visualization techniques to designing more informative and powerful user

interface control widgets for visual querying, such as interactive legends, and new query devices.

- the formalism supports querying a query or visualizing a visual representation, both of which exist sparingly at present, but will increase in importance as queries and visualization mappings become more complex.

Many examples are presented in this chapter. They demonstrate that the formalism is capable of providing a common description of different aspects of visual query interfaces.

# Chapter 4

## The Application System KMVQL

The formalism presented in Chapter 3 provides a general framework for visual query interface design. It clearly states the most important elements, phases and issues in the design process, which, I believe, makes the task of designing visual query interfaces much easier.

The formalism is a theoretical framework at a level more abstract than the implementation. To fill the gap between the formalism and implementing practical systems, I designed a software domain model. The software modules defined in it are correspondences to the elements in the formalism. Thus, their responsibilities and interactions are easily defined.

The software model can be used as a reference for designing visual query interfaces. Based on it, a prototype software program KMVQL (Karnaugh Map based Visual Query Language) was implemented as an application.

### 4.1 Software Design based on the Formalism

The software model mainly answers the following questions:

- What are the user actions that should be handled by a visual query interface for information exploration?
- What software modules should be included in the visual query interface architecture?
- What are the responsibilities of the software modules?

### 4.1.1 User Actions

In the process of visual information exploration, users normally perform the following basic actions:

- select the data source, specifying which collections of data to search on,
- query data, specifying queries that express their information needs, which normally requires specification of query terms and specification of the query structure,
- interact with the visualizations, browsing, navigating, scanning and zooming the view to understand and analyse the data ,and
- select data from the view.

In the work reported in this thesis, selecting data from the visual view is considered to be querying it. Therefore, selecting is studied separately from other operations such as browsing, navigating and scanning. A successful visual query interface should provide interaction control mechanisms allowing users to conduct those operations easily and naturally. It allows the user to specify queries visually, eliminating the need to understand Boolean logic. Furthermore, it should also provide control mechanisms allowing users to customize their visualizations. For example, it should

- provide a variety of visualizations, allowing the user to select which one to use,
- provide a flexible visualization mapping mechanism, allowing the user to choose how to visualize the data, and
- provide a flexible query visualization mapping mechanism, allowing the user to specify how to visualize the queries.

Therefore, the software model accommodates the following user actions: selecting data source, specifying query terms, specifying query structure, interacting with the visualization, selecting data from the visual view, and specifying visualization methods for the data set and query.



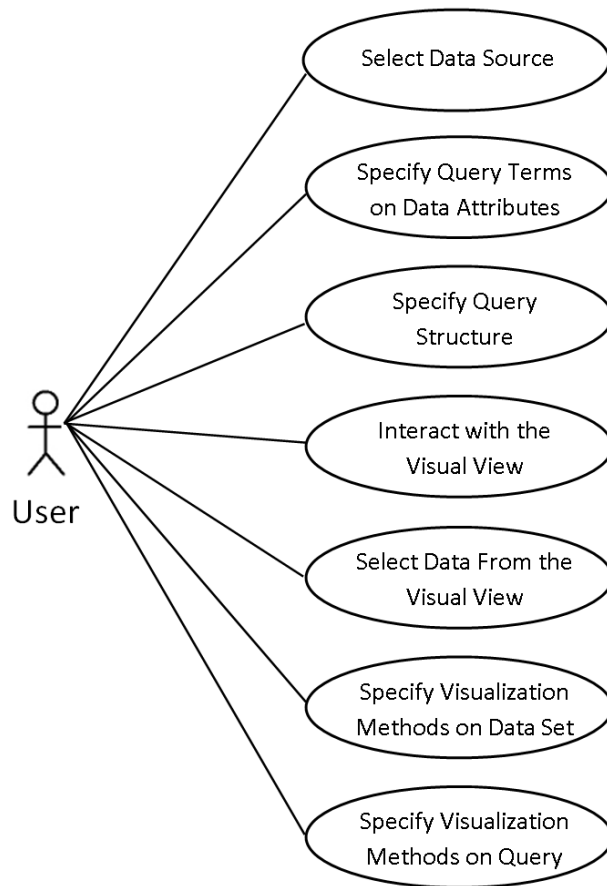


Figure 4.1: User Actions Diagram

### 4.1.2 Software Domain Model

The software domain model describes the basic software modules in a visual query interface and their interrelationship. The modules manage data, queries, query results, visualizations of them, and the mechanisms for defining the visualization and query mappings. To be general, the granularity of the modules is coarse, as shown in Figure 4.2.

Some of the modules respond to user actions. The user actions and their processing modules are listed in Table 4.1.

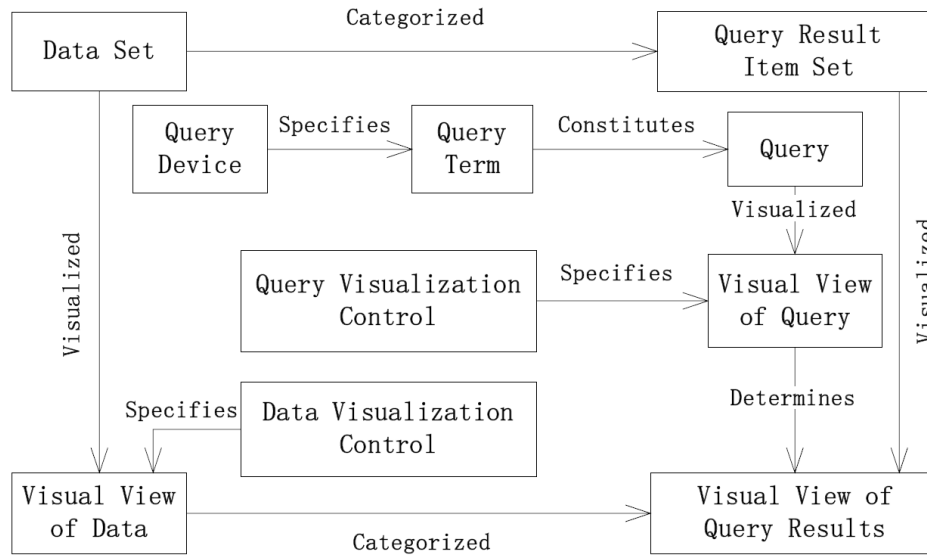


Figure 4.2: Software Domain Model

### 4.1.3 Module Responsibilities

Defining the responsibilities of the modules is a critical step in the design process. In this section, the functionality and responsibilities of the modules are described in detail.

**The Data Set Module** provides functions for data access, data transformation, data manipulation, and meta-data management.

- (1) Data access is the mechanism for importing data from data sources. The system should accommodate different types of data and different types of data sources.
- (2) Data transformation transforms raw data from the data source into a data table. It is important for the module to provide data manipulations that are commonly used in the analysis process, such as count and aggregate, sort, join, project, extract, insert, delete, and update.
- (3) Meta-data management is needed for two main reasons: to expose semantically meaningful data transformations to the user and to drive design decisions when generating visual representations. Meta-data typically consists of the at-

User Actions	Modules
Select data source	Data set module
Specify query terms on data attributes	Query device
Specify query structure	Visual view of query module
Interact with the visual view	Visual view of data module and Visual view of query results module
Specify visualization methods on data set	Data visualization control module
Specify visualization methods on query	Query visualization control module

Table 4.1: User Actions and Their Processing Modules

tribute name, the attribute type, the storage type, the attribute domain and possible hierarchical relationships.

**The Visual View of the Data Module** carries out data visualization, which transforms data tables into visual structures. The module then displays the visual structure on the screen. It provides interaction mechanisms, responding to user input (keyboard, mouse, and other input devices), to provide view transformations, such as zooming and panning, and updates the display. It also allows users do direct data selection within the view.

**The Query Module** manages query terms and query structure. It keeps a record of query terms that have been specified. It supports manipulations of the query terms list such as add, delete, update, select and unselect.

**The Visual View of the Query Module** carries out query visualization. Users interact with it to specify query structures. It transforms truth tables into visual structures and displays them on the screen. It also provides interaction mechanisms, responding to user input. When operations are applied to visual query fragments, the query structure is updated.

**The Query Result Item Set Module** computes a Boolean representation of query fragments for data items in the data set. Similar to the data set module, it provides manipulation of the query result item set, such as count and aggregate, sort, project, extract, insert, delete, and update.

**The Visual View of Query Results Module** provides visualizations of the query

result item set. It transforms the item table into a visual structure and displays it on the screen. It provides interaction mechanisms, responds to user input, and updates the display. It also allows direct data selection within the visual view.

**The Data Visualization Control Module** provides definitions for data visualization mappings, which map data attributes and values to visual features. It may also provide interfaces allowing users to specify visual mappings. There is a spectrum of choices, with a tradeoff between simplicity and expressibility.

(1) Use scripting (a procedural approach) to create the visualization. Scripting requires the most expertise and takes the most time, but has the most expressiveness and flexibility.

(2) Based on visualization structures, write a specification (a declarative approach), which is like filling out a form with the specific fields the user wants to visualize.

(3) Based on visualization structures, fill in GUI forms specifying which data attribute maps to which visual feature.

(4) Use a direct manipulation interface to drag-and-drop field names to create a visualization.

**The Query Visualization Control Module** provides definitions for query visualization mapping. Like the Data Visualization Control module, it defines visual mappings that map query fragments to visual features. It may also provide interfaces allowing the user to specify visual mappings.

**The Query Device** is a user interface widget that allows users to specify a query term. It provides interaction mechanisms for responding to user input (keyboard, mouse, or other input devices), updating its display, and creating a query term.

**The Query Term** is a Boolean query expression that specifies constraints on item attributes, which may be either data attributes or visual attributes.

#### 4.1.4 Summary of the Software Model

The software model defines software modules corresponding to the elements of the formalism. The model uses the object design technique which can improve the exten-

sibility and ease of maintenance of the software. Such a modularised architecture has the following benefits.

- Clarity and ease of comprehension of the design is increased.
- Maintenance and enhancements are simplified.
- Reuse of modules is increased.

In order to further validate the formalism and the software model, and to provide concrete examples of how the techniques are used in a practical system, I implemented a new visual query interface - KMVQL.

## 4.2 Overview of KMVQL

KMVQL has been designed to support interactive exploration of multidimensional datasets. It uses the Karnaugh map [56] as the visual representation of query space. KMVQL incorporates dynamic query techniques and provides a visual query interface to help users formulate arbitrarily complex Boolean queries by direct manipulation.

Its interface has three parts: a view that provides a graphical presentation of the data, a Karnaugh map, and a set of query device widgets for data value selection. This section introduces the components and how they work together when users formulate Boolean queries and explore the data.

### 4.2.1 Visualization of Query and Query Results

Two visual query representations are supported in KMVQL: a Karnaugh map and an iconic representation. Chapter 3 described how to use them for Boolean query specification and query result visualization.

A Karnaugh map arranges the cells in a regular grid, making them easy to understand and navigate. As a table-based representation it is familiar and intuitive. In data analysis, table-based visualizations predominate [91]. Using a Karnaugh map to visualize query results makes it possible to incorporate a variety of data visualization mechanisms. Therefore, it works as a visualization spreadsheet [21], which allows

users make comparisons between cells. Users can specify query structures by directly selecting cells of interest, which is easy and intuitive.

On the other hand, a Karnaugh map splits the visualization into discrete cells, with the context of each cell not explicit. Therefore, iconic representations are used so that query results can be visualized in a single visual view. Corresponding to the two visual representation types, the visual view of query results can be in two modes: a Karnaugh map mode, and an iconic mode, as shown in Figures 4.3 and 4.4.

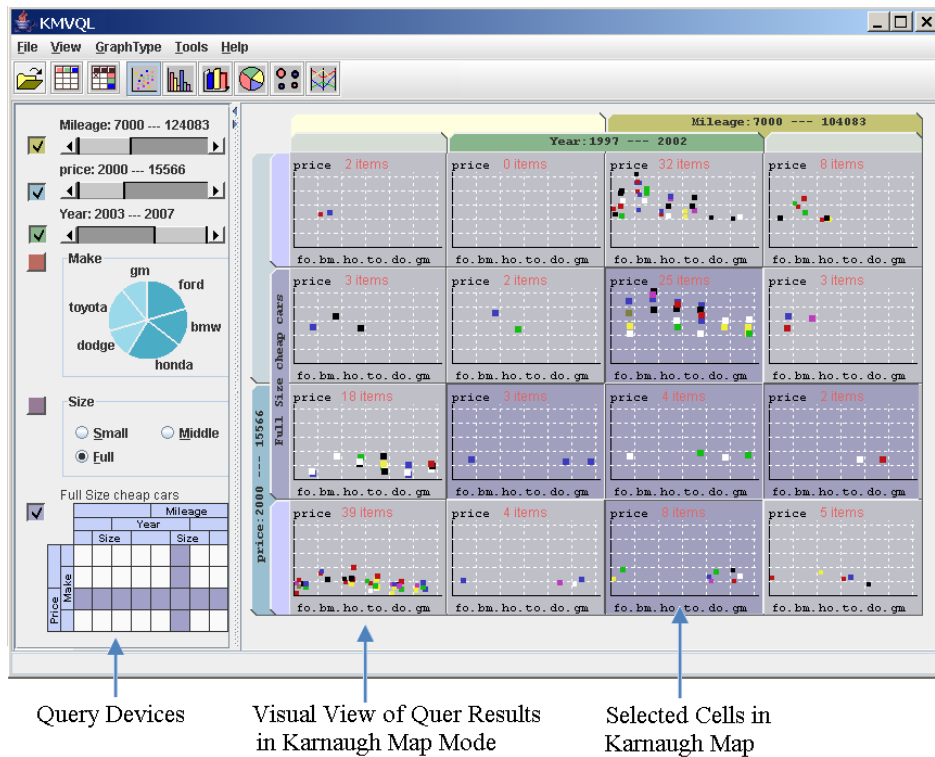


Figure 4.3: KMVQL User Interface in Karnaugh Map Mode

In iconic mode, the representation is not convenient for query specification. Therefore, a separate Karnaugh map is provided for specifying the Boolean query structure. The Karnaugh map also provides a legend for the visual codings of the query fragments, which can be modified by users.

The icons with coloured petals introduced in Figure 3.4 is a typical example of iconic coding for query fragments. Such visual codings are complex, which can result in clutter when the number of query terms or data items is large. Simpler coding

methods are also supported in KMVQL. For example, visual query fragments can simply differ in colour or size. Such visual codings have been adopted by systems such as Attribute explorer [81]. On the other hand, it is hard to compare different query fragments with the simple encodings. KMVQL provides the flexibility to allow users to define visual encodings based on their needs.

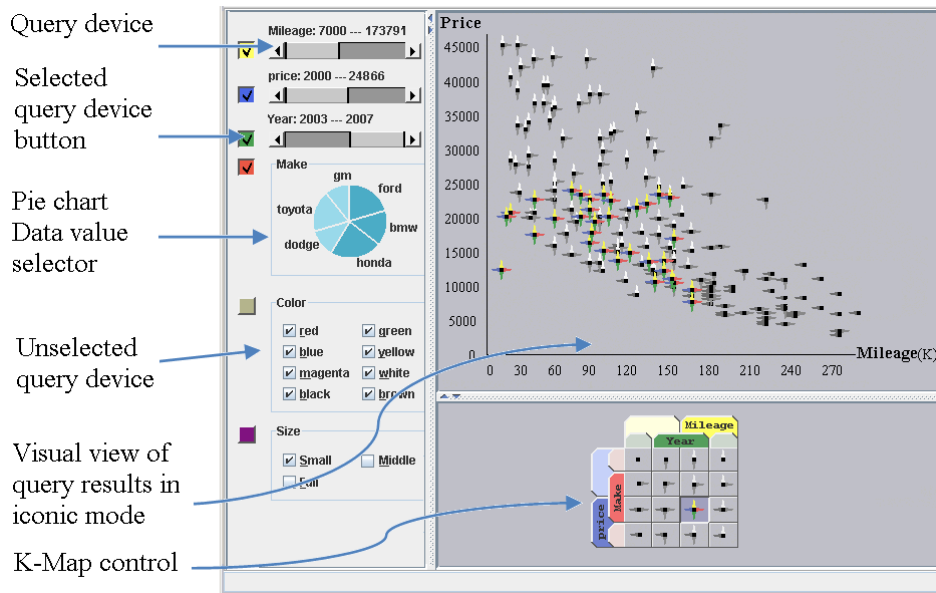


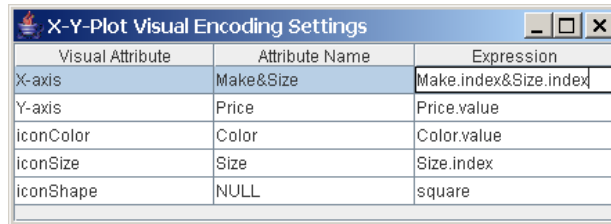
Figure 4.4: KMVQL User Interface in Iconic Mode

## 4.2.2 Visualization of Data Space

KMVQL provides several pre-defined visualization structures: x-y-plots, bar-charts, pie-charts, and parallel-coordinates plots. Each visualization structure corresponds to a visualization space constituted by a set of visual features.

For example, the x-y-plot space is constituted by X-axis(horizontal position feature), Y-axis(vertical position feature), icon colour, icon size, and icon shape. In the bar-chart space, the icons can only be vertical bars. X-axis(horizontal position feature), Y-axis(vertical position feature), and icon colour are constitutive features, visual features like icon size, and icon shape are not needed. A parallel-coordinates plot comprises a set of axis and colour of icons.

For each visualization structure, the user can describe the mapping relationship between data attributes and visual features with a tool provided in KMVQL. Figure 4.5 shows an example of the tool for editing visual encodings for X-Y-Plot.



Visual Attribute	Attribute Name	Expression
X-axis	Make&Size	Make.index&Size.index
Y-axis	Price	Price.value
iconColor	Color	Color.value
iconSize	Size	Size.index
iconShape	NULL	square

Figure 4.5: Modify Visual Encodings For X-Y-Plot

### 4.2.3 Query Device

In KMVQL, each query device is composed of two parts: a data value selector and an associated button displayed in front of it (see Figure 4.3 and 4.4). Simple data value selectors are dynamic query widgets, such as rangesliders, checkboxes, radiobuttons, comboboxes, and pie-charts.

Normally, rangesliders are used for ordinal and quantitative values. Checkboxes, radiobuttons, comboboxes and pie-charts are used for discrete values. Query devices in KMVQL are adjustable. Users can specify what type of widget is to be used for a data attribute. KMVQL also allows users to dynamically remove unnecessary query devices from the interface, or add new ones when needed. This feature saves the display from being occupied by a lot of useless widgets, which is a common problem in many query interfaces. The tool provided in KMVQL for query device settings is shown in Figure 4.6.

Interacting with a data value selector generates a Boolean function which is used as a query term. If the associated button of a query device is selected, the query term generated by the query device is added into the Karnaugh map and used for query definition. Releasing an already selected button removes the associated query term from the Karnaugh map.



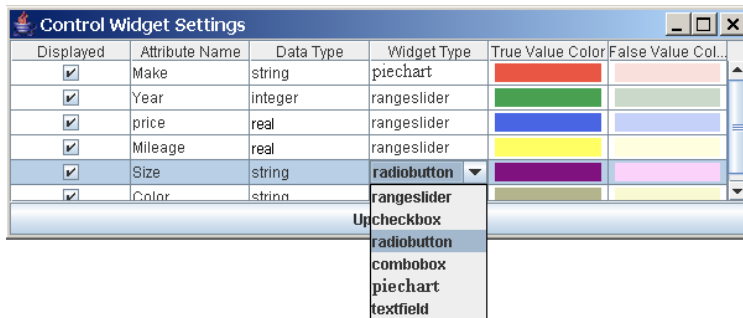


Figure 4.6: Settings of Query Devices

#### 4.2.4 Information Seeking Nodes

The information seeking process consists of a series of interconnected but diverse searches. The search process might involve sophisticated queries. Therefore, a successful interface for information exploration should allow users to save important intermediate results and reuse them later.

In KMVQL the concept of an “information seeking node” is introduced to describe a checkpoint in the information exploration process. As explained in the formalism, the information exploration process is a sequence of queries and visualizations of data. Therefore, each information seeking node  $N$  is defined as a tuple  $N = \langle D, m_{qd}, m_{vd}, m_{vq} \rangle$ , which contains the following components: the data set being analysed  $D$ , the query being specified  $m_{qd}$ , and the visualizations of the data and query.

At any point in the information seeking process, the user can save the current state as an information seeking node for later review, modification, or for new query creation.

When a user loads an information seeking node, the working data set and the interface components (including the query devices displayed on the screen, the Karnaugh map, and the visual view of query results) are deployed based on information stored in the node. This feature allows users to reuse previous query results.

#### 4.2.5 Formulate Boolean Queries in KMVQL

A typical scenario of formulating a query in KMVQL is described below:

1. Load data from a data source. Visualize and display the data. KMVQL creates

query devices based on the properties of data attributes.

2. Specify the query terms by interacting with the query devices. The associated button of the active data value selector is automatically selected. Selecting a button adds a new query term to the Karnaugh map; releasing a button removes the related query term.
3. Specify the query structure by selecting or unselecting cells in the Karnaugh map. The default setting of the Karnaugh map is to automatically select the cell that represents the conjunction of the query terms. If a user wants to specify pure conjunction queries, she does not need to operate on the Karnaugh map.
4. Double click the view of the query results. The set of data items that exactly fulfil the query is the new working data set, on which users can make further queries by repeating steps 2 and 3. Users can also save the query results for later use, or open a previous information seeking node for analysis.

Upon any of the above operations, the view of query results updates its display immediately to give feedback. The number of query terms in the Karnaugh map equals the number of selected query devices. In Figure 4.4, there are four selected query devices; thus the query is composed of four query terms.

Each query device is assigned a colour, in which its associated button is displayed. The same colour coding is used for the Karnaugh map tabs, to show which tab is associated with which query device. In this way, the connection between the query devices and the query visualization is explicitly revealed to users.

Of necessity, the query devices, the Karnaugh map, and the visual view of query results are tightly coupled. The Karnaugh map acts as middle-ware joining the other components. In traditional dynamic query systems, no such middle-ware exists, the resulting query is limited to the conjunction of predetermined query devices. But using Karnaugh maps, arbitrary Boolean queries can be easily formulated.

#### **4.2.6 Compound Query based on Intermediate Results**

With KMVQL, users are allowed to formulate compound queries by using previous query results to specify data constraints for a new query. KMVQL supports three types of query devices:

1. widgets for simple value selection, such as rangesliders, radio buttons, checkboxes, and so on.
2. small views of a data set, and
3. small Karnaugh maps.

Query devices can be created based on previous information seeking nodes. With the query device settings tool, users can specify the type of the query device to be either a small Karnaugh map or a small view, which is created based on the information stored in the node. Figure 4.7 shows an example of using small Karnaugh maps as query devices.

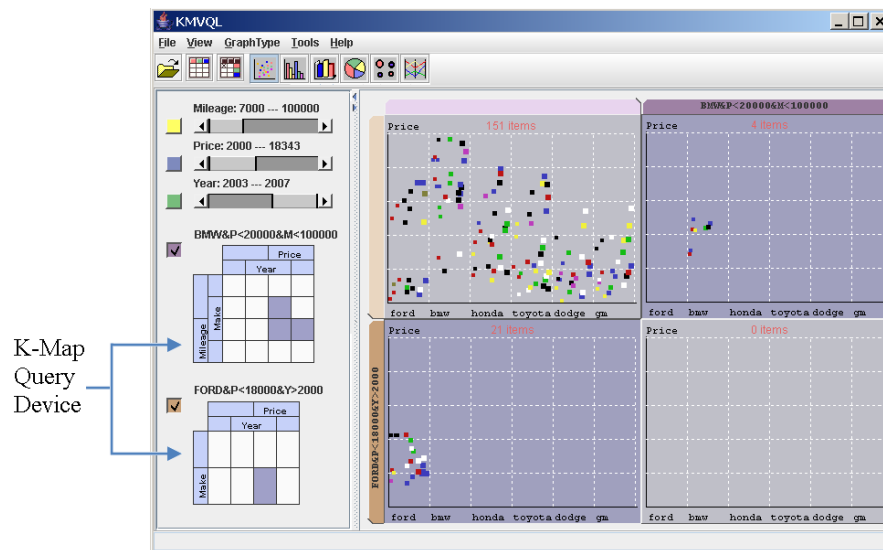


Figure 4.7: Karnaugh Map Query Devices

KMVQL also allows a user to directly select data from the visualization. For example, dragging out a rubber-band in an x-y-plot, or brushing an axis in a parallel-coordinates plot are common direct data selection mechanisms.

In response to a selection operation, the selected items are highlighted. If the user drag-and-drops the selection into the frame of query devices, a new query device is created and added to the interface. The new query device is a small view of the data set, with the selection highlighted, as shown in Figure 4.8.

Query devices are used in the same way for query structure specification. Therefore, users can reuse intermediate query results and specify Boolean combinations of them.

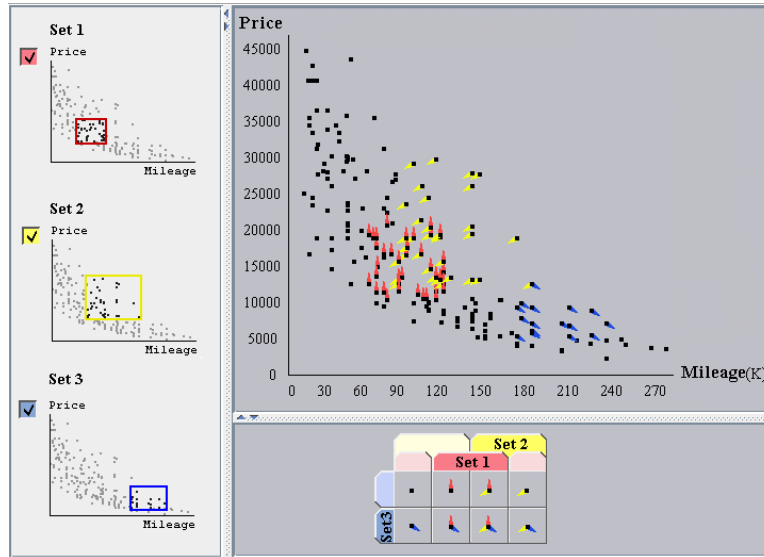


Figure 4.8: Create Query Devices Based on Direct Visual Item Selections in KMVQL

### 4.3 Example of Exploring Data with KMVQL

This section uses an example to describe the capabilities of KMVQL as an information exploration interface. Suppose the user, Iris, is interested in buying a second-hand car. But she is not familiar with the car market, and she has no particular preference of the car. This example shows how KMVQL helps her in seeking what she wants.

At the beginning, Iris loads into KMVQL a second-hand car database. When the data is loaded, KMVQL creates a set of query devices based on the data types of the attributes. For example, the Makes of the cars are strings, which are nominal. KMVQL uses a combobox to list all the string values, allowing the user to select the preferred Makes. But Iris is not satisfied with the automatically generated query device for this attribute. She notices that the list is short, so she chooses a pie-chart as the data value selector, which provides easier data selection, and reveals the proportions of each Make.

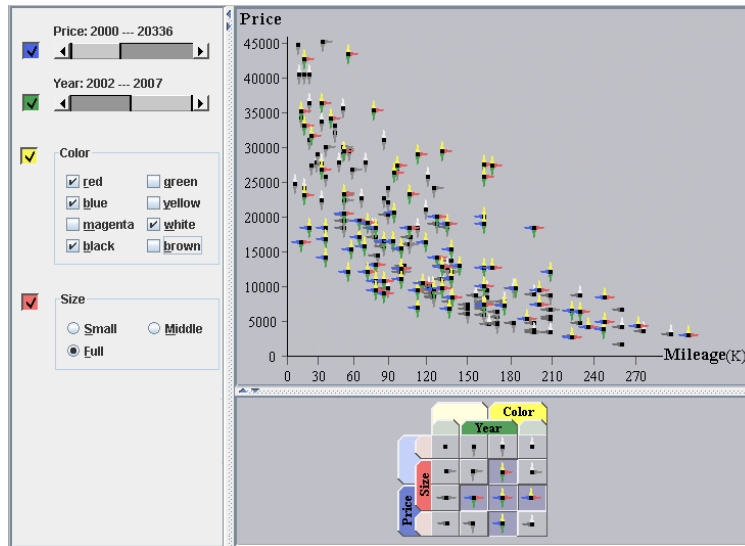


Figure 4.9: Screen Shot of the Query “Meet at Least 3 Query Criteria”

Then Iris chooses to visualize the data with a x-y-plot, mapping the Mileage attribute to the x-axis, and the Price attribute to the y-axis. The data visualization is displayed on the screen. Operating on the query devices, Iris specifies query terms. In the Karnaugh map the cell representing the conjunction of the query terms is automatically selected. The data items associated with the selected cell are displayed in coloured icons, while other items are displayed in grey scale. This makes the selected items stand out as well as providing context information.

When Iris analyses the visual view she finds that few items meet the pure conjunction. So she clicks other cells in the Karnaugh map to broaden the query, adding more data items to the results, helping her to see other promising candidates. Iris continues interacting with the query devices to modify her query. After each operation, the view is updated immediately to give her visual feedback to help her understand the data more efficiently.

When she feels satisfied with the query and the results, she saves the state as an information seeking node  $N_1$  so that she can analyse or use it later. The specified query is  $Q_1$ : “satisfy at least 3 query criteria among the four”. Thus she names the node “Meet at least 3 criteria”.

Then Iris drags out a rectangular box on the x-y-plot, forming the second query,  $Q_2$ . She drag-and-drops the rectangular box into the frame of query devices. A new

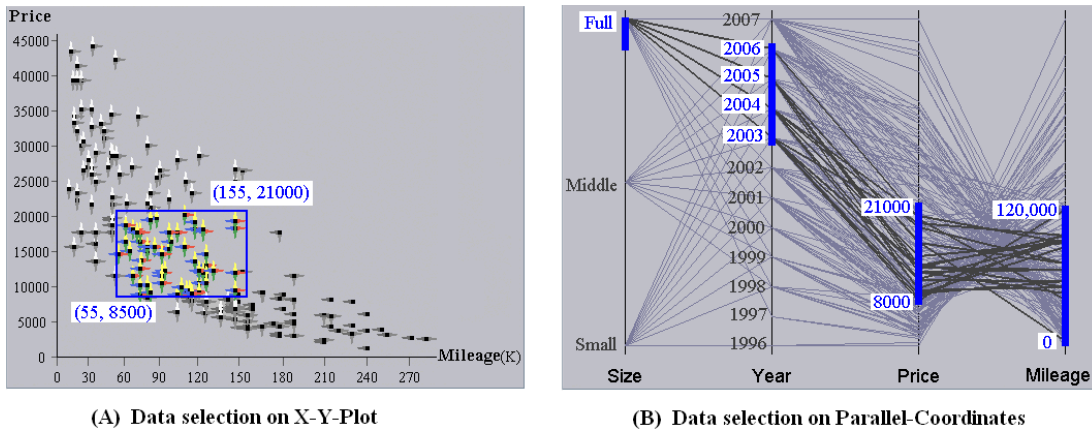


Figure 4.10: Direct Data Selection on the Visual Views

query device is created and added to the window, showing a small view of the selection. Iris names the new query device as “Price-Mileage Selection”.

Iris thinks that the last two queries might filter out some important candidates. She wants to get more information about the data set. So she resets the interface and views the data set with a parallel-coordinates plot. She brushes the axis to indicate the interested value ranges, forming the third query  $Q_3$ . She saves the current state as a second information seeking node  $N_2$ , named “FullSize-After 2003”. As we can see, although the user-defined names are not precise, they allow users to remember and recognize intermediate results.

Then Iris resets the interface again using an x-y-plot to visualize the data set. This time the Year attribute is mapped to the x-axis so that she can analyse the data from a different perspective. She adds two new query devices into the interface. One shows a small parallel-coordinates plot from information seeking node  $N_2$ , another is a small Karnaugh map from node  $N_1$ . To make the interface look clean and simple, she removes unused widgets from it. She chooses the Karnaugh map mode for the interface and selects four cells in the Karnaugh map to form a query. As shown in Figure 4.11, the query means “satisfy at least two queries among  $Q_1$ ,  $Q_2$ , and  $Q_3$ ”.

Then Iris double clicks the view of query results to analyse the data items that fulfil the query. The other items are filtered out so that she can focus her analysis on a relatively small data set. Iris saves the current interface status to a file so that when she explores the same data in the future, the interface will be automatically configured

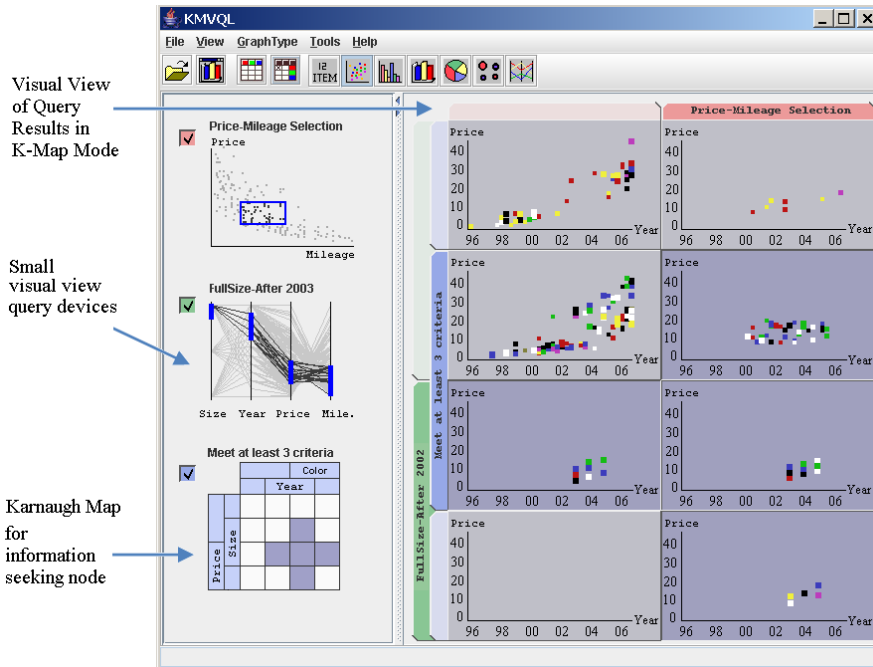


Figure 4.11: Screen Shot of the Compound Query in KMVQL

based on it and the intermediate results will be restored. She can start from this point to continue her search until she found the desired car to purchase.

The above example shows that with KMVQL, complex queries can be formulated easily by direct manipulations. KMVQL is promising in several aspects:

- With Karnaugh map, arbitrary Boolean queries can be specified by direct manipulation. Users do not need to worry about Boolean logic. User queries are no longer restricted to conjunctions of predetermined query devices.
- The tight coupling of query and query result visualization makes it easier for users to analyse the results and refine their queries.
- Important search status can be saved and reloaded for review, for modification, or for new query creation. Intermediate query results can be reused.
- Users can directly select data by interacting with the views. Boolean combinations of multiple selections can be specified easily by operating on the Karnaugh map.

- Query devices are adjustable. Users can remove unnecessary query devices from the interface, or add new ones. This feature saves the screen from being occupied by useless widgets, which is a common problem in many query interfaces.

## 4.4 Summary

This chapter presents a software model and a new visual query interface based on it.

The software model defines the software domain model of visual query interfaces. In the domain model, software modules and their functions are defined, in addition to the common user actions in information exploration. All these actions are handled by software modules defined in the domain model. The software model functions as a reference which makes the task of designing a practical visual query interface easier.

Based on the software model, a new visual query interface, KMVQL, was designed and implemented. It uses a Karnaugh map as the visual representation for Boolean queries. It incorporates dynamic query techniques that allow users to formulate Boolean queries of arbitrary complexity by direct manipulation. With KMVQL, users do not have to concern themselves with explicit logic operators, which makes Boolean query specification much easier. It also provides seamless integration of queries with their results, which helps users to understand the data set and refine their queries efficiently.

The implementation of KMVQL shows that the formalism is a practical abstraction of visual query interfaces for information exploration. There are many advantages in developing interactive visual query interfaces using the formalism: the task of designing new interfaces is easier. The innovative features implemented in KMVQL are inspired by the formalism. With them, users are able to complete complex tasks in short periods of time.



# Chapter 5

## Related Work on Query Language Evaluation

A Visual query interface includes a query language in a visual form and a variety of interaction mechanisms to facilitate data exploration. Empirical study of query languages is important for improving the quality of visual query interface design. For example, when designing a visual query interface using a new query language, it is necessary to evaluate the new query language by comparing it with existing ones. There exist many user studies that evaluate query languages. This chapter reviews previous research on query language evaluation. Several typical examples of evaluating visual query languages are also discussed.

### 5.1 Introduction to Query Language Evaluation

The purpose of empirical studies on query languages that have been undertaken can be classified into the following five categories:

- measuring ease-of-use,
- comparing two or more languages for ease-of-use,
- studying controversial issues in query language design,
- providing feedback to aid design of a query language, and
- understanding and modeling human behaviour when writing and understanding queries.

The general approach of an experiment is to

1. define precisely what is to be measured,
2. develop a task that human subjects perform, and
3. measure the relevant parameters of subject performance.

At the core of the query language studies is measuring ease-of-use. It is hard to develop a definition for the ease-of-use that corresponds to intuitive notions of ease-of-use and permits measurement with scientific rigor. Several categories of variables are predicted to have an effect on query performance. They are included in a framework of variables for query evaluation.

### **5.1.1 Framework of Variables for Query Language Evaluation**

In an experiment, many factors affect the experimental data. Without controlling or considering these factors, the validity of experimental data is questionable. Prior research on query language evaluation [19, 52, 92] provides a set of variables related to performance, which is shown in Figure 5.1. This set is a good basis for choosing the variables for an experiment.

In the experiments, user performance, which is a dependent variable, is normally measured by the time to complete a task, the accuracy of results, and/or a subjective response. The time taken to complete a task has been used in many studies as an efficiency measure of user performance. Many empirical studies have also used accuracy of results. Time and accuracy are quantitative proxy measurements for the vague notion of “ease-of-use”. They are measured under controlled conditions.

Sometimes, subjective measurements, such as subjects’ confidence in their results, the perceived ease-of-use of the language, their satisfaction with the language, or their performance expectations, are also used for query language evaluation. These subjective data are normally collected through a questionnaire.

Based on the results of previous experiments, the four groups of factors that directly affect user performance are characteristics of the subject, characteristics of the query language, nature of the task, and set-up of the laboratory. They are normally controlled either by keeping them constant, by random assignment, or by treating them as covariants in the statistical analysis.

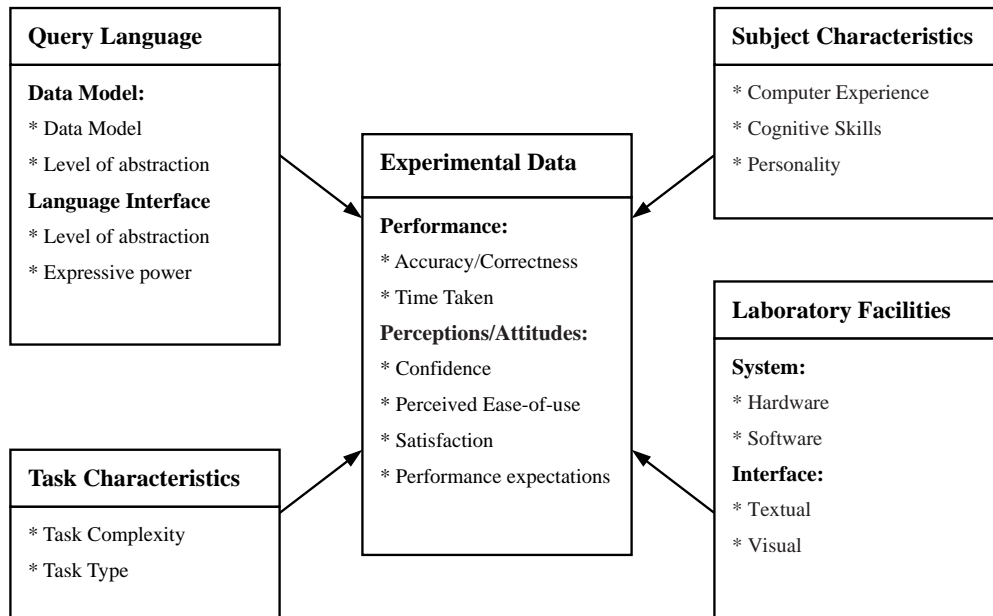


Figure 5.1: Framework of Variables Used in Prior Query Performance Research

- **Subject characteristics.** Subject characteristics refer to dimensions on which subjects differ, such as intelligence, computer programming experience, or the ability to concentrate during the experiment. Subject characteristics are normally indirectly controlled by randomising the subject placement in the experimental groups. In most experiments, training programs are included to lessen the relevant individual differences.
- **Query language characteristics.** Query language characteristics include the data model and the language interface. When investigating a query language, such variables are constant. When comparing different query languages, these characteristics are effectively covarying independent variables.
- **Task nature.** The task nature includes task complexity and task type, which are very important factors in experiment design. Task complexity is used as independent variable for data analysis in almost all experiments.

- **Laboratory facilities.** Laboratory facilities refer to details such as whether the experiment is manual or computer based. If the experiment is computer based, relevant system characteristics include the physical aspects of the system, such as its response lag and its variability, the physical input/output devices being used, and the dialogue style. Which variables are held constant, and which are used as independent variables is determined by the purpose of the experiment.

There is a tradeoff between holding variables constant and treating them as independent variables. If more variables are held constant, the experiment is simpler and cheaper, but less can be learned about the direct effects of and the interactions among the variables.

### 5.1.2 Tasks for Query Language Evaluation

Among task variables, query task complexity and task type are core factors in experiment design. Query languages are complex and require rich combinations of cognitive activities, such as learning, understanding, and remembering. To capture ease-of-use, experimenters have developed a number of different tasks, which are used in experiments to capture different aspects of ease-of-use.

Reisner summarized the commonly used tasks for measuring ease of use for query languages in “Human Factors Studies of Database Query Languages: A Survey and Assessment” [77]. These tasks are listed in Table 5.1.

### 5.1.3 Boolean Concept Complexity

Despite much attention to task complexity in data querying, no widely accepted definition of task complexity exists [92]. Psychology research on human concept learning recognizes Boolean logic as an essential component of concepts [74]. Thus, the near-universal presence of Boolean logic in query interfaces is not surprising. But experimental studies show that logical concepts vary widely in ease of comprehension and production in humans [83] and much research in visualization has focussed on building representations of Boolean concepts that are more easily understandable. Conceptual complexity is currently regarded as the key variable affecting the difficulty of comprehension, so this section describes the state of the art in psychological research on Boolean concept complexity.

<b>Task</b>	<b>Description</b>
Query writing	Users are given a question stated in natural language and required to write a query in the given query language.
Query reading	Users are given a query written in the query language and asked to write a translation into natural language.
Query interpretation	Users are given a query in the query language and a printed database with the data filled in. They are asked to find the data asked for by the query.
Question comprehension	Users are given a natural language question and a printed database and are asked to find the data asked for.
Memorization	Users are asked to memorize and reproduce a database.
Problem solving	Users are given a problem and a database and are asked to generate questions in natural language that would solve the problem. The questions should be answerable from the database.

Table 5.1: Tasks Used to Measure Ease of Use in Query Language Evaluation

### Rough Classification of Boolean Concepts

In most psychological studies, Boolean concepts are classified into the following 5 types, according to the Boolean logic present in them:

- Type 1: Pure conjunction, contains AND operators only.
- Type 2: Pure disjunction, contains OR operators only.
- Type 3: Predominately NEGATION queries.
- Type 4: Contains AND as well as OR operators.
- Type 5: Contains all the operators: AND, OR, NOT.

Based on these studies, Boolean complexity is determined by the number of terms and the Boolean concept type. The smaller the number of terms, the lower the Boolean concept complexity, and vice versa. When the number of terms is equal, a Boolean concept with pure conjunction operators is normally considered less complex than other types. The NEGATION concepts, type 3, are generally more complex than the “AND alone” or “OR alone” concepts. Finally, concepts in type 4 and type 5 are more complex than the others [34, 39].

This classification is very rough. It does not provide a quantitative basis for data analysis: for example, how might one compare the complexity of a pure conjunction concept with 3 terms and a pure disjunction concept with 2 terms? Furthermore, concepts in the same category might have different complexity. For example, here are two concepts,

- (1) (NOT A) AND (B OR C)
- (2) (NOT A) AND B AND C OR A AND (NOT B) AND (NOT C)

They both belong to type 5. Do they have the same complexity? Common sense tells us that the second expression is more complex than the first one, but the rough classification does not support this assert.

### **Feldman's Catalog of Boolean Concepts**

There is a long but sporadic history of measuring Boolean complexity, of which Shepard's [83] contribution is seminal. Recently, Feldman [34, 35] introduced and substantiated a quantitative measure of Boolean complexity. The measure identifies the subjective difficulty of a concept with its Boolean complexity, that is, to its logical incompressibility. He defines Boolean complexity as the length of the shortest logically equivalent Boolean formula, calculated as the number of literals (positive or negative variables). For example, the concept  $AB + A\bar{B}$  is equivalent to  $A(B + \bar{B})$  and thus to  $A$ , and hence has Boolean complexity 1; whereas  $\bar{A}B + A\bar{B}$  has no shorter equivalent, and hence has Boolean complexity 4.

Based on Feldman's algorithm, the complexities for the two Boolean expressions (1) and (2) can be computed and compared: the complexity for (NOT A) AND (B OR C) is 3, while the complexity for (NOT A) AND B AND C OR A AND (NOT B) AND (NOT C) is 6. Therefore, by Feldman's measure, expression (2) is more complex than expression (1).

Analysing experimental data based on a quantitative definition of Boolean complexity makes it possible for researchers to reason about subjects' performance more rigorously.

## 5.2 Review of Related Experiments

Many different query languages and interfaces have been built. There have also been many user studies conducted for evaluating them.

The many evaluations of query languages are broadly similar. The evaluation of the Truth-table Exemplar-Based Interface (TEBI) [39] is a typical example. Young and Shneiderman’s evaluation study for Filter/Flow [106], and Chui’s evaluation of comparing four query languages: Boolean expressions, an if-then-else procedural language, Filter/Flow, and Venn diagrams [22], are also worth mentioning. This section reviews the following aspects of these experiments,

- query languages studied in the experiment,
- laboratory facilities and subject characteristics,
- query tasks being used for query language evaluation,
- training,
- dependent variables, and
- how the experiment was conducted.

### 5.2.1 Example Analysis: TEBI

TEBI [39] is the Truth-table Exemplar-Based Interface. There are two types of TEBI interface, TEBI concrete, and TEBI abstract. The only difference them is that in TEBI concrete, all values that appear in a truth-table are concrete, while in TEBI abstract, the irrelevant values are replaced with “Don’t Care”. Figure 5.2 shows a query represented by TEBI concrete and TEBI abstract. In their experiment, Greene studied the difference between queries produced using SQL and queries produced using TEBI.

In their study, the performance of subjects and the types of errors were analysed. The authors found a significant improvement in the speed and accuracy of query production using TEBI, especially with TEBI concrete. They also found that the fewer the formal logic language features the better the performance. The authors conclude that for query specification, it is easier to recognize than to generate.

<b>Query Languages</b>	TEBI and SQL
<b>Lab Facilities</b>	Paper and pen based test.
<b>Subject Characteristics</b>	<ol style="list-style-type: none"> <li>1) 80 female subjects, age from 20 to 60 (mean age of 45)</li> <li>2) with at least a high-school education</li> <li>3) no previous experience in information retrieval</li> <li>4) subjects are assigned randomly into four groups</li> </ol>
<b>Description of the Tasks</b>	<p>Four subject groups performing different tasks:</p> <ol style="list-style-type: none"> <li>1) <b>TEBI Concrete</b>: Given an exemplar table printed on a paper, user need to select the rows that represent a query.</li> <li>2) <b>TEBI Abstract</b>: same as in <b>TEBI Concrete</b> group.</li> <li>3) <b>SQL Choice</b>: One correct and three incorrect sets of SQL code (ordered randomly) printed on a paper, select the correct one.</li> <li>4) <b>SQL Generation</b>: Given the first part in a query: “ <b>SELECT CodeNumber FROM Shape</b> ” , subjects are asked to finish the query expression on the paper.</li> </ol>
<b>Training</b>	<p>The training was finished within 3 hours:</p> <ol style="list-style-type: none"> <li>1) Teach the basic concepts of databases and Boolean operators</li> <li>2) Provide manuals, introducing the query languages they will use.</li> <li>3) Provide examples, ask them take exercises.</li> </ol>
<b>Experimental Data</b>	<ol style="list-style-type: none"> <li>1) The answer for the query</li> <li>2) The time taken to finish the task related with the query</li> <li>3) The subjects' confidence in their answer</li> </ol>
<b>Experiment Process</b>	<ol style="list-style-type: none"> <li>1) The test is composed of 24 query tasks, contains all the 6 query types in rough classification</li> <li>2) Each subject is given a booklet of queries written in English</li> <li>3) Based on the English description of the queries, subjects need to finish the required tasks: <ol style="list-style-type: none"> <li>a) subjects in <b>TEBI</b> groups need to specify the query by selecting the related rows in the given truth-table;</li> <li>b) subjects in <b>SQL choice</b> group need to select the correct query from a list of choices;</li> <li>c) subjects in <b>SQL generation</b> group need to write down the query expression on the paper.</li> </ol> </li> </ol>

Table 5.2: Review of the User Study for **TEBI** [39]



Query: Find all **employees** who either work in the **Toy** department or are managed by **Gary**, and also come from the city **London**

Employee Name	Department	Manager	City
✓ Jones	Toy	Gary	London
Allan	Toy	Gary	Athens
✓ Doe	Toy	Buford	London
Smith	Toy	Buford	Athens
✓ Price	Cosmetics	Gary	London
Hall	Cosmetics	Gary	Athens
Kelso	Cosmetics	Buford	London
Lee	Cosmetics	Buford	Athens

**TEBI Concrete**

Employee Name	Department	Manager	City
✓ Jones	Toy	Gary	London
Allan	Toy	Gary	Don't Care
✓ Doe	Toy	Don't Care	London
Smith	Toy	Don't Care	Don't Care
✓ Price	Don't Care	Gary	London
Hall	Don't Care	Gary	Don't Care
Kelso	Don't Care	Don't Care	London
Lee	Don't Care	Don't Care	Don't Care

**TEBI Abstract**

Figure 5.2: Query Represented with TEBI Concrete and TEBI Abstract

### 5.2.2 Example Analysis: Filter/Flow

Filter/Flow [106] is a procedural, graphical representation for Boolean queries designed by Young and Shneiderman. It gives Boolean queries a visual form that uses the metaphor of water flowing through filters. Sequential flow represents the AND operator; parallel flows represent the OR operator. Young and Shneiderman's evaluation study is briefly described in Table 5.3.

The statistical analysis of their results indicate that for query comprehension tasks, fewer Boolean errors were made in Filter/Flow than were made in SQL; for query composition tasks, there was a significant difference between the number of correct queries in SQL and Filter/Flow, favouring Filter/Flow.

### 5.2.3 Example Analysis: Chui's User Study on Query Languages

Chui [22] conducted an empirical study examining the query formulation performance of naive users on four query languages: Boolean expressions, If-then-else procedural languages, Filter/Flow diagrams, and Venn diagrams. Among them, If-then-else procedural languages and Boolean expressions are textual query languages; Filter/Flow diagrams and Venn diagrams are graphical query languages. The experiment is briefly

<b>Query Languages</b>	Filter/Flow and SQL
<b>Lab Facilities</b>	<ol style="list-style-type: none"> <li>1) Paper and pen, and</li> <li>2) Computer based test: An <i>AT&amp;T</i> computer with a 80386 processor running at 25 Megahertz with a 17 inch VGA monitor Both the SQL and Filter/Flow interfaces were created using Tool-Book on the same computer</li> </ol>
<b>Subject Characteristics</b>	<ol style="list-style-type: none"> <li>1) 20 subjects (10 male and 10 female)</li> <li>2) some experience using a computer and mouse</li> <li>3) little or no programming language experience</li> <li>4) no Boolean logic experience</li> </ol>
<b>Description of the Tasks</b>	<p>Two types of tasks: query comprehension (query reading) and query composition (query writing)</p> <ol style="list-style-type: none"> <li>1) query comprehension tasks: given a database printed on a page of paper, and a query displayed on the computer screen, subjects were required to indicate on a sheet of paper the results of the query.</li> <li>2) query composition tasks: given a query description in English sentence, subjects were required to form a query using either the SQL or Filter/Flow interface on the computer: <ol style="list-style-type: none"> <li>a) in SQL mode, subjects were asked to type in statements</li> <li>b) in Filter/Flow mode, subjects were asked to use the mouse to select and drag boxes on the screen to formulate a query</li> </ol> </li> </ol>
<b>Training</b>	<p>A fifteen to twenty minute training period for each interface:</p> <ol style="list-style-type: none"> <li>1) Exercising with the given interface (SQL or Filter/Flow)</li> <li>2) Learning and exercising with Boolean logic</li> </ol>
<b>Experimental Data</b>	<ol style="list-style-type: none"> <li>1) The number of correct queries in each task and</li> <li>2) The number of errors made in each task</li> </ol>
<b>Experiment Process</b>	<ol style="list-style-type: none"> <li>1) subjects performed the comprehension task (consisting of five queries), followed by the composition task using one of the designated interfaces (SQL or Filter/Flow)</li> <li>2) a five minute break</li> <li>3) subjects repeated the two tasks using the alternate interface and another set of queries</li> <li>4) subjects were asked for their opinion of the two interfaces and the reasons for their preferences</li> </ol>

Table 5.3: Review of the User Study for Filter/Flow [106]

described in Table 5.4.

<b>Query Languages</b>	Boolean expressions, If-then-else procedural languages, Filter/Flows, Venn Diagrams
<b>Lab Facilities</b>	Paper and pen
<b>Subject Characteristics</b>	1) 24 undergraduate students (17 female and 7 male) 2) no experience for computer programming, Boolean logic, set theory, or Venn diagram
<b>Description of the Tasks</b>	query writing: given a query description in English sentence, subjects were required to write/draw the query using one of the query language on a piece of paper.
<b>Training</b>	1) teaching subjects Boolean logic, and how to form queries with a particular query language. 2) a pre-test was conducted.
<b>Experimental Data</b>	1) The answer for each query 2) The time take for completing each query
<b>Experiment Process</b>	1) subjects are assigned randomly into 2 groups: text and graphics. In text group, subjects formulated queries using Boolean expression and If-then-else procedural language. In graphics group, subjects used Filter-Flow diagram and Venn diagram. 2) each subject were required to formulate 16 queries using each query language.

Table 5.4: Review of Chui’s User Study [22] for Four Boolean Query Languages

Statistical analysis of the results indicates that user performance in accuracy and speed was highest using Boolean expressions, followed by Venn diagrams, Filter-Flow diagrams, and then If-Then-Else procedural languages. In summary, Chui claimed that textual Boolean expressions outperform the graphical query languages in both speed and accuracy. It is not hard to notice that the conclusion from this user study conflicts with Young and Shneiderman’s study.

### 5.2.4 Summary of Previous Experiments

After analysing previous studies on query languages, I found that there are three most commonly used query tasks on Reisner’s list: query writing, query reading and translation, and query result data selection. Detailed descriptions for them are listed in Table 5.5, it also summarizes the main factors that affect performance.

Query task	Description	Main factors affecting performance
Query writing	Given a natural language description of the query, subjects write the query in the tested query language	<ul style="list-style-type: none"> <li>• <b>query complexity and query language type</b></li> <li>• subjects' reading and writing skills</li> <li>• natural language is ambiguous</li> </ul>
Query reading and translation	Given a query written in a query language, translate it into natural language sentences	
Query result data selection	Given a query and a list of data, find all the data items that meet the query	<ul style="list-style-type: none"> <li>• <b>query complexity and query language type</b></li> <li>• the length of the list</li> <li>• the number of qualified data items</li> <li>• subjects' reading skill</li> </ul>

Table 5.5: Commonly Used Query Tasks and the Factors Affecting the Performance

The essence of query writing tasks and query reading and translation tasks is translating a query from one form to another. Performance on such tasks is strongly affected by individual differences between subjects, such as familiarity with the natural language, writing speed, and so on. For the task of query result data selection, the performance of finding query answers from the data list is closely related to such factors as the length of the list, the number of qualified data items, etc. These nuisance variables are confounded with query task complexity and query language characteristics which also affect performance. Without careful analysis, the nuisance variables can easily cause inferential errors.

Furthermore, many query languages are better suited to one type of query than another. It is quite possible that when comparing two query languages that the result would be quite different for a different set of queries.

In addition, most studies used only rough classifications of query complexity for data analysis. Quantitative analyses were not performed. Since in any empirical study, only limited number of queries are used, this restriction makes it harder to measure or predict performance on more complex queries.

In summary, most previous experiments on visual query languages suffer from one or more of the following deficiencies.

- The response time measured in the study was used directly for evaluation. But sometimes, the overall response time is affected by more than one factor;
- Only simple queries were studied in the experiments, or an insufficiently precise classification of query complexity was used for data;
- Users' cognitive activities when using the query interface were not separated from one another.
- The reasons why one query language performs better than another were not investigated.

Thus it is no surprise that these experiments sometimes lead to superficially conflicting results. Other evaluations, not discussed here have similar problems. The ones described are not chosen to be typical, but to be representative of the better evaluations I found, which indicates that further research into possible methodologies for experiment design and analysis is needed.

# Chapter 6

## A Methodology for Evaluating Visual Query Languages

This chapter introduces a new methodology of experimental design and data analysis for query language studies.

### 6.1 Introduction

When analysing a novel query language, it is both necessary to know how it compares to other query languages, and to know why it performs better or worse. The latter requires us to know what users are thinking when using a query language, and specially, which cognitive activities are enhanced or inhibited by the language. Doing so we can better predict user performance with a proposed language early in its design.

Query tasks are normally complex, requiring a combination of cognitive activities, such as learning, understanding, remembering, and decision making. These activities are differently affected by query languages. The response times measured in an experimental study are a mixture of many time components. Each related to a different combination of activities. To understand users' cognitive activities and how they are affected by query languages, it is important to isolate as many of these time components as possible.

This chapter presents a methodology of experimental design and data analysis for query language studies. It isolates several time components by making a rough model of query processing, so that cognitive activities of users can be analysed in more detail.

## 6.2 Method

To evaluate a new visual query language, a comparison standard is needed. Normally the standard is the most commonly used query language. The new visual query language is then compared with the standard using several criteria. (In most user studies, a textual representation of Boolean expressions is used as the standard.)

When languages are compared, subjects are asked to accomplish a set of tasks with either query language. For each subject, task, and query language combination, the correctness of the task result and the time taken to complete the task are measured.

### 6.2.1 Query Task

Most evaluations of query languages try to mimic real conditions of interface use as closely as possible, and therefore measure tasks with many intermixed perceptual and cognitive components. Such tasks may require users to perceive, understand, respond to, or create visual queries. Thus, users' performance is affected by many factors, which are hard to disentangle. In order to identify individual cognitive components, and to reduce the effects of nuisance variables, it is important to abide by the following principles for experiment design:

- keep the query task as simple as possible;
- keep the user actions as simple as possible; and
- keep the experimental environment as simple as possible.

The simplest query task is matching: users are given a query and a single data item, and are asked only whether or not the data item is selected by the query. Simple as this task is, it is important to understand, because the components of query matching are certainly components of query creation and other tasks required for data selection.

In the experiment described in this thesis the data item is a simple coloured shape, which is called an icon. In the matching task the stimulus consists of an icon plus a query that defines a set of properties the icon might have, such as colour, shape and size. The user determines whether or not the query selects the icon.

## 6.2.2 Components of Query Task

In the experiment, a matching task is assumed to be comprised of three parts.

- Query comprehension. The subject reads the query represented in a query language and forms a mental model of the query. The time taken for query comprehension is denoted as  $T_q$ .
- Stimulus comprehension. The subject identifies the relevant qualities of the icon. The time taken for stimulus comprehension is denoted as  $T_s$ .
- Response generation. Based on the mental model of the query and the qualities of the stimulus, the subject decides which response is correct and executes it. The time taken for response generation is denoted as  $T_g$ .

Different query languages can affect the components differently. Thus, it is good to measure the time components individually, so that we can discover how each cognitive activity is affected.

The response time measured in most experiments is a combination of the three time components mentioned above, and of others. It is hard to measure directly the time taken for any individual subtask. The methodology described in this chapter partially isolates different components.

## 6.3 Isolating the Time Components

The essence of the experimental methodology is to measure the response time when different parts of the stimulus are presented at different times, thus encouraging the subject to decompose the task into component stages. The time differences are called stimulus onset asynchronies (SOAs).

This type of stimulus manipulation is common in psychology, in studies of priming and masking [12, 69], for example, and recently in studies of the psychological refractory period [62]. There is an abundant literature on SOA manipulation in experimental psychology, but in user interface studies it is rare and introducing it into perceptual studies of visualization is an important contribution of this thesis.



### 6.3.1 Manipulating Stimulus Onset Asynchrony

SOA is used two ways in the experimental task. Presenting the query before the icon gives the user has the opportunity to start forming a mental model of the query in the absence of the icon. Contrarily, when the icon is presented early the user can begin analysing it. In either case, response time,  $T$ , is calculated from the time when the stimulus is complete. The length of time when only part of the stimulus is presented is called the delay time,  $t$ .

### 6.3.2 A Possible Experimental Hypothesis

A hypothetical model of the matching task shows how SOA can disentangle different parts of the matching task. Assume that the components are deployed sequentially. Then the relationship between response time  $T$  and delay time is illustrated in Figure 6.1.

Consider the matching task when SOA is zero, and assume that the user first comprehends the query,  $T_q$ , then analyses the stimulus in terms of the query,  $T'_s$ , then generates the response,  $T_g$ . The response time is

$$T = T_q + T'_s + T_g$$

Alternatively, assume that the user starts with the stimulus,  $T_s$ , then analyses the query in terms of the stimulus,  $T'_q$ , producing the response time

$$T = T'_q + T_s + T_g$$

Whichever strategy is faster will be chosen within a few trials, even by an inexperienced subject.

Now, if the query is presented  $t$  seconds before the remainder of the stimulus, the user can start comprehending the query before the trial starts. This condition is called query-first. The response time  $T$  in this condition is then

$$T = T_q - \min(T_q, t) + T'_s + T_g$$

The response time, graphed as a function of delay time, would be the curve shown in Figure 6.2.

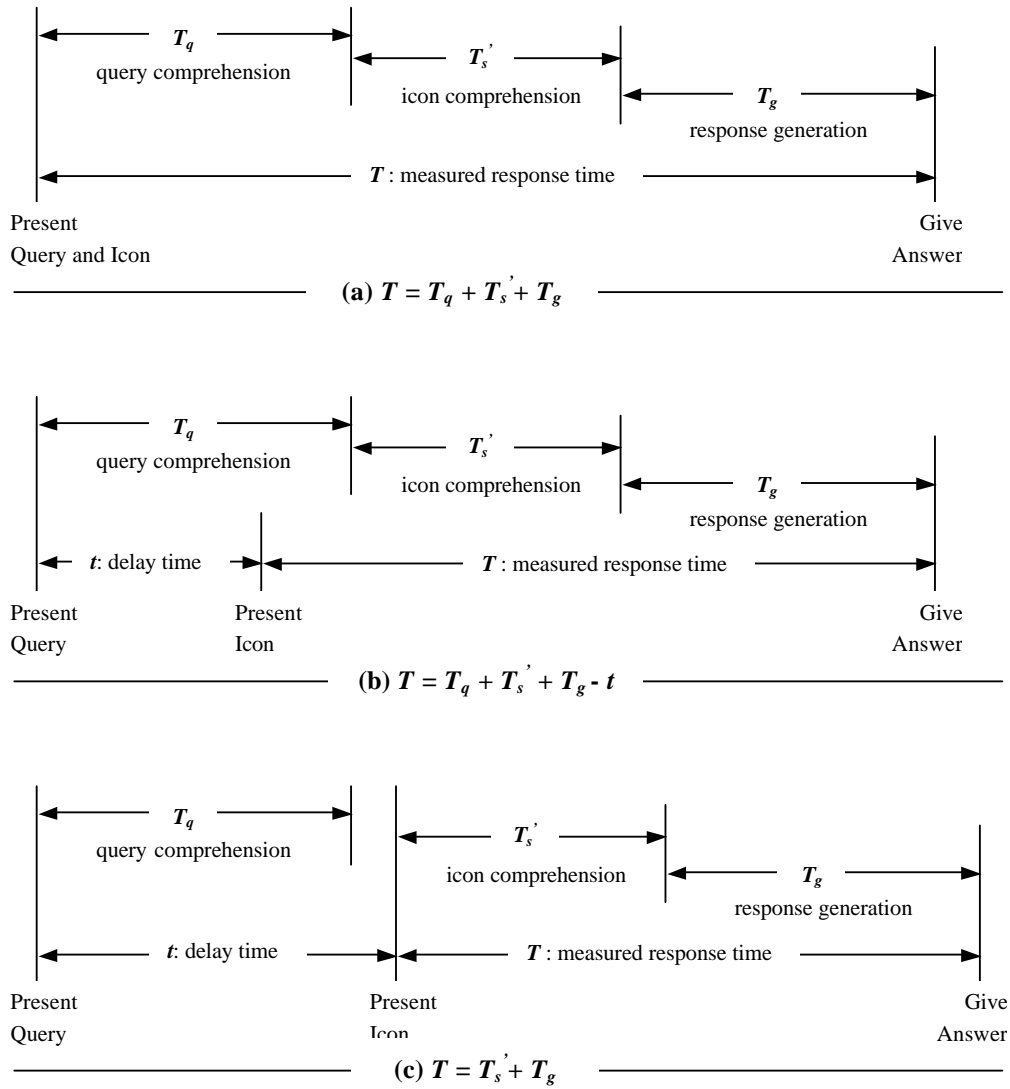


Figure 6.1: Response Time: (a) no delay time; (b) delay time less than query comprehension time (c) delay time greater than query comprehension time

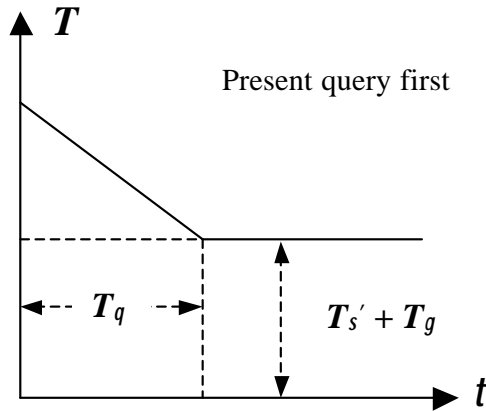


Figure 6.2: The Plot of Response Time vs. Delay Time

And, if the icon is presented before the remainder of the stimulus, which is called icon-first, then the response time is

$$T = T_s - \min(T_s, t) + T'_q + T_g$$

In both cases the response time is linear with a slope of -1 until the early part of the stimulus has been completely processed,  $T_q$  or  $T_s$ , then remains constant for greater delay times. This model of the matching task is intuitively appealing, but unquestionably naive. Among other things, it ignores the possibility of activities proceeding in parallel, and of interactions between activities, which undoubtedly exist. However, it fits the data qualitatively, though not quantitatively, and provides a very useful way of conceptualising the results.

### 6.3.3 Data Fitting

When the delay time is varied, the measured set of response times, plotted against the delay time, approximate an elbow curve. Elbow curves are fit to sets of data points that differ only in delay time, estimating the height of the constant part of the curve,  $\beta$ , the slope of the descending part of the curve,  $\mu$ , and the delay time at which the two curves intersect,  $\tau$ .

The fitting algorithm minimizes the least squares error with respect to the three variables. A detailed description of the elbow data fitting algorithm is given in Ap-

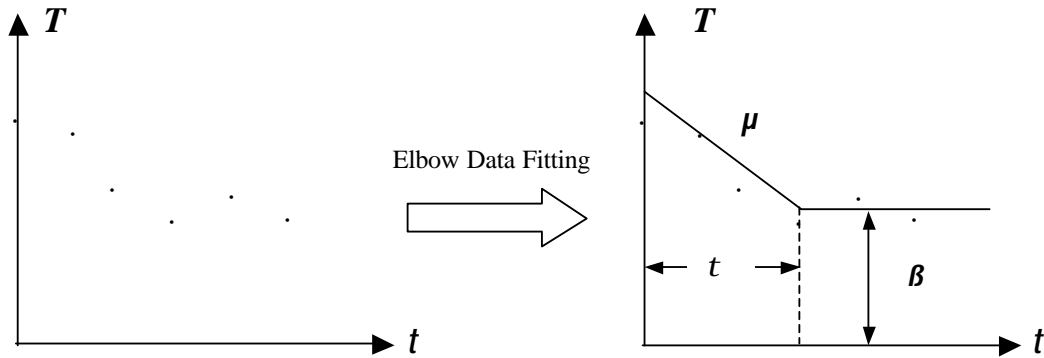


Figure 6.3: Elbow Data Fitting

pendix B. Based on the algorithm, a program has been developed to estimate the parameters of an elbow curve from a set of experimental data.

Analysis of the fitting procedure shows that the estimates for  $\beta$  and  $\mu$  are normally distributed random variables, so ordinary statistical analysis of them can be performed. However, the estimate for  $\tau$  is distributed by a Cauchy distribution, so the results must be handled carefully. The analysis of them is confined to graphing.

In terms of the above hypothesis the fitting parameters have simple interpretations. In the query-first condition  $\beta = T'_s + T_g$ , and in the icon-first condition  $\beta = T'_q + T_g$ . In the query-first condition  $\tau = T_q$ , and in the icon-first condition  $\tau = T_s$ .

Using the methodology described above, I conducted an experiment comparing the relative performance of Karnaugh maps and textual Boolean expressions. It is described in the following chapter.

# Chapter 7

## The Experiment

This chapter describes an experiment that studies the difference between two different representations of Boolean queries: Karnaugh map and textual. It investigates a user's ability to comprehend Boolean queries, which is important in its own right and necessary, though not sufficient, for creating queries successfully. It makes three main contributions.

- The first is the use of stimulus onset asynchrony as an experimental technique for separating different aspects of query comprehension.
- The second is the discovery that there are two ways in which users comprehend Boolean expressions, inductive and deductive, which are operationally very different.
- The third contribution is the demonstration that the Karnaugh map representation scales extremely well with query complexity, and the experiments show that its good scaling properties occur because it strongly facilitates inductive comprehension.

### 7.1 Introduction

Boolean queries are usually expressed textually, as query terms connected by the Boolean operators. Textual representations have convenient formal properties, but users have difficulty understanding them [39, 55].

A Karnaugh map represents logical combinations of query terms in a form totally different from textual expressions. It is a two-dimensional table, which uses a Gray code pattern to represent truth tables in easy to understand diagrams. Chapter 4 describes the Karnaugh map based visual query language (KMVQL), in which Karnaugh maps play important roles in query specification and query result visualization. Thus it is natural to ask “Is the Karnaugh map truly easy to understand and to use?”

Furthermore, as reviewed in Section 2.3, there are two underlying query models: the Boolean logical expression model and the truth-table model. Many existing query languages are based on the logical expression model. Karnaugh maps and many novel visual query representations presented in Chapter 3 use truth-tables as their underlying model for representing Boolean queries.

Therefore, it is important to find out how the different query models affect users’ cognitive activities, and how different query representations behave. To answer the questions, I conducted an experiment comparing the relative performance of Karnaugh maps and textual Boolean expressions.

To understand users’ cognitive activities and how they are affected by query representations, the experiment is designed based on the experimental methodology described in Chapter 6: by manipulating SOAs, different cognitive components can be partially isolated and analysed.

## 7.2 Method

In order to identify individual cognitive components, and to reduce the effects of nuisance variables, it is important to keep the query task, the user actions, and the experimental environment as simple as possible. Thus the experiment uses a matching task.

### 7.2.1 Query Task

In the matching task, the match items are icons. In each experimental trial, a graphical icon with a set of visual features (colour, shape, and size) is shown on a display beside a query represented either textually or using a Karnaugh map, as illustrated in Figure 7.1. Subjects respond according to whether or not the icon is selected by the query.

To respond subjects press one of two keys, ENTER for a positive answer, SPACE for a negative one. In this way, the query task is simple; perceiving the graphic icon is easy; the method for giving answers is straight forward. There is little opportunity for variations that interact with the experimental variables.

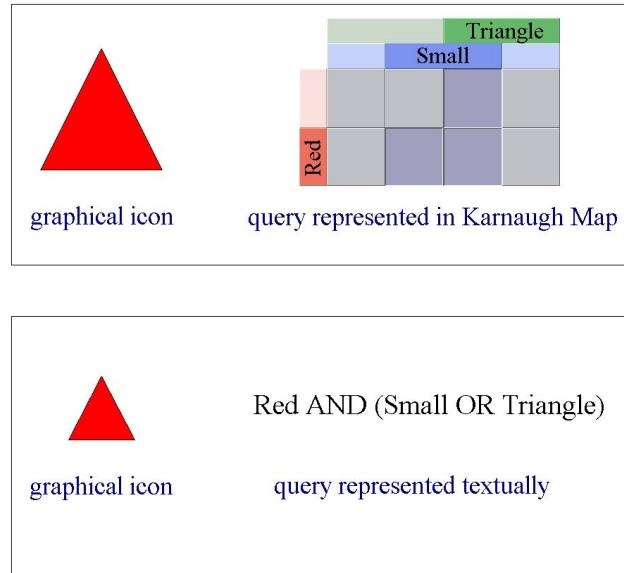


Figure 7.1: Stimuli Used in the Experiment. The upper panel shows the Karnaugh map, the lower panel shows the textual expression

## 7.2.2 Experiment Trials

The experiment comprises a series of trials. In each trial, the stimulus has two parts: a graphical icon, and a query. The query is represented in either textual form or using a Karnaugh map. Since this experiment was designed based on the methodology of manipulating SOA, different parts of the stimulus are presented at different times.

Each trial is an instance of the tuple  $\langle \text{Query}, \text{Icon}, \text{Representation}, \text{Display Sequence}, \text{Delay Time} \rangle$ . Among the variables, *Representation* has two fixed values: textual, or Karnaugh map. Potentially, other variables can take on many values. To limit the number of trials values for the variables must be carefully chosen. The remainder of this section describes what values were chosen for the variables and why they were chosen.

### 7.2.3 Queries

This experiment included 1-dimensional, 2-dimensional, and 3- dimensional queries with different query structures and Boolean complexities, including several queries with the same Boolean complexity but different logical operators. The queries being used in the study are listed in Table 7.1.

Query ID	Number of Terms	Boolean Complexity	Boolean Expressions
1	1	1	A
2	1	1	not A
3	2	2	A AND B
4	2	2	(not A) AND (not B)
5	2	2	(not A) OR (not B)
6	2	4	A AND (not B) OR (not A) AND B
7	3	3	(not A) AND (not B) AND (not C)
8	3	5	C AND (A AND B OR (not A) AND (not B))
9	3	5	A AND (B OR C) OR (not A) AND (not B)
10	3	6	A AND ((not B) OR (not C)) OR (not A) AND (B OR C)
11	3	8	A AND ((not B) AND C OR B AND (not C)) OR (not A) AND B AND C
12	3	10	A AND ((not B) AND (not C) OR B AND C) OR (not A) AND ((not B) AND C OR B AND (not C))

Table 7.1: Queries Used in the Experiment

The table provides a rank of the queries, which is primarily based on the number of query terms. For the queries with the same number of query terms, they are ranked based on Boolean concept complexity. There are several queries having the same Boolean complexity but using different logical operators. For example, based on Feldman’s algorithm, the Boolean concept complexity of both query 1 (A) and query 2 (not A) is 1. But query 2 uses the negation operator which, based on Shephard’s classification of queries, is harder than query 1. Therefore, query 2 is ranked after query 1. Another example of queries with the same complexity but different logical operators is query 4 ((not A) AND (not B)) and query 5 ((not A) OR (not B)). Since



query 5 uses the OR operator, which is supposed to be harder than the AND operator, query 5 ranked after query 4.

This ranking method, using the rough classification rule to break ties in Feldman's definition of Boolean concept complexity provides the ranking in query complexity that is used for data analysis. Note there is an exception to the ranking by Boolean complexity: query 6 (A AND (not B) OR (not A) AND B) is more complex than query 7 ((not A) AND (not B) AND (not C)). Query 7 ranks after query 6 is because it involves more query terms than query 6.

As shown in the table, several queries (query 9, 10, 11, and 12) are very complex. Such complex queries are rare in previous query language studies, where a small number of low complexity queries is normally used. In my experiment, such high complexity is necessary, since an important aspect of the experiment is to examine how the difficulty of understanding a query representation scales with query complexity. Thus queries with a wide range of complexities are needed.

## 7.2.4 Graphical Icon

Since the experiment uses a matching task, query complexity is not the only factor that affects the complexity of the task. The same subject, seeing the same representation of the same query but a different graphical icon, sees a different level of complexity. When the set of visual features relevant to a match changes, the resulting response time may vary, because, to correctly categorize an icon, the set of relevant features must be tested. In different conditions, the number of features to be tested is different.

For example, if the query is Red AND (Small OR Triangle), and a green small triangle is presented, subjects can answer "No" perceiving only the icon colour. They need not check other features. But if a red big circle is presented, subjects must test all of colour, size and shape, slowing the response.

Therefore the complexity of a matching task is determined by the Boolean query and the graphical icon taken together. But the icon factor is easy to overlook. It is explicitly described here because considering the number of features to be tested is an important principle that I used for choosing graphical icons in the experiment.

For each query, I used multiple graphical icons which differ in the number of features that require testing. Six icons were chosen per query, three positive and three negative. The mean of their response times reflects the overall complexity of a query.

In the experiment, graphical icons were compound stimuli composed of shape, colour and size attributes. Five colours were used: red, green, blue, yellow, and purple; two sizes were used: big and small; and three shapes were used: triangle, circle, and square.

To reduce boredom and carry-over effects, the visual stimuli used for each query were varied. For example, if the Boolean concept is A AND B, queries having different terms can be used for the same concept: Red AND Small, Green AND Circle, Square AND Big. All are treated as the same query. For the query “Square AND Big”, various icons were used. For instance, big red square is a positive example of the query which requires two features to be tested. Small red circle is a negative example with one feature to be tested. Both of them are used for testing the query.

### 7.2.5 Display Sequence and Delay Time

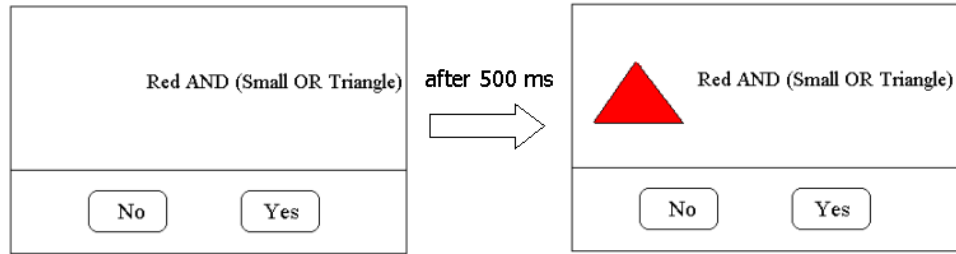
As described in Section 7.2.2, the display sequence of the stimulus and the delay time between their onset are important variables in the experiment. Which part of the stimulus should be displayed first? What delay time values should be used so that the elbow point can be captured? To answer the questions, a pilot study was done.

In the pilot study, two display sequences were initially tested: query-first (displaying the query first, as shown in Figure 7.2), and icon first (displaying the icon first). Displaying the icon first gave uninteresting results because the time for icon comprehension is small and relatively constant. Furthermore, the subject does not know which icon features require attention. Thus, a representation of the relevant features was added to the icon first condition, omitting how they were conjoined in the query. That is, presenting both the icon and the query terms first. This modification is used in the experiment, and is called **icon-first**, as shown in Figure 7.3.

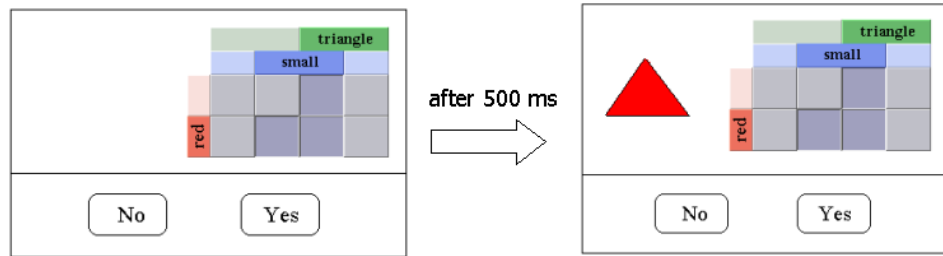
Elbow point positions were estimated for each combination of Boolean query, representation type, and display sequence based on the results of the pilot study. Delay time values were thus chosen so that the elbow point is well within their range, with different sets of delay times for different conditions. For simple Boolean queries, the delay time values are less than those for complex queries.

Once the values of the trial variables were chosen, they were used to set up the trials used in the experiment.

For the purpose of easy reference, the term **presentation condition** is taken to



(a) display query first in textual representation



(b) display query first in Karnaugh-Map representation

Figure 7.2: Query-first Example Trials: Display Query First

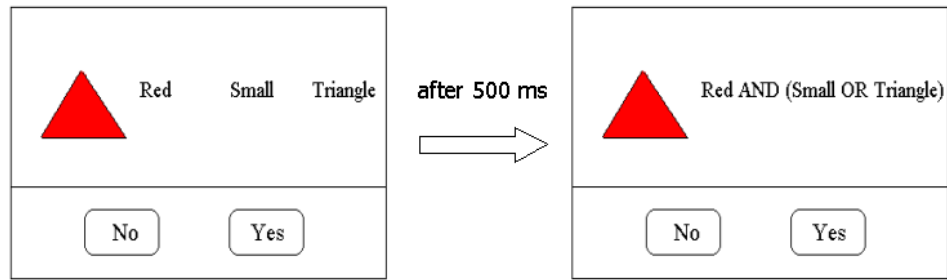
mean the combination of query representation type and display sequence type. The term **query condition** refers to the combination of query and presentation condition.

In the experiment, there are 4 presentation conditions: Karnaugh map icon-first, Karnaugh map query-first, textual expression icon-first, and textual expression query-first. All twelve queries were tested in each presentation condition, giving 48 query conditions.

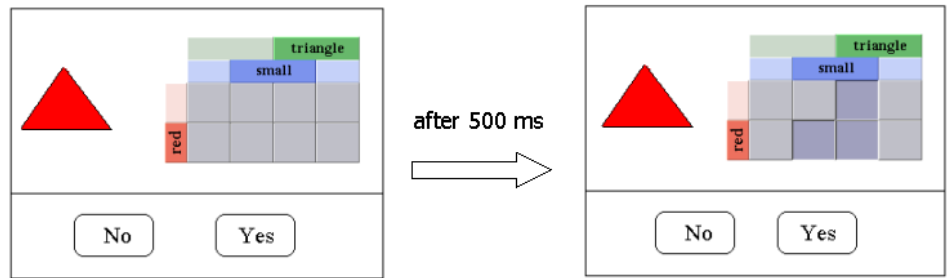
## 7.3 Conduction of the Experiment

### 7.3.1 Trial Setup

In the experiment, there were twelve Boolean queries, six icons per query, three positive and three negative, two query representation types, two display sequences, and eight delay times, resulting in 2304 trials, too many for a single session. Thus the trials were distributed over six sessions, each containing all combinations of query condition and delay time. Each session had 384 trials, plus ten warm-up trials, the only difference



(a) display icon and query terms first in textual representation



(b) display icon and query terms first in Karnaugh map representation

Figure 7.3: Icon-first Example Trials: Display Icon and Query Terms First

between the corresponding trials in different sessions being the icons.

Each subject did six sessions, one per week for six consecutive weeks. The order of trials in each session was randomised. The trials were randomly assigned to blocks of 64 trials, which normally took 10 to 15 minutes to finish. Subjects took three to ten minute breaks between blocks.

### 7.3.2 Lab Facilities

The experiment was run on a PC with an Intel Pentium 4 CPU running at 2.53GHz, with 512 MB of RAM and a 17 inch monitor. The experiment software was developed using the DJGPP and Allegro graphics package running on MS-DOS. Because MS-DOS is a single-process operating system there are no system processes to increase timing variability, which was less than one millisecond.

### 7.3.3 Subject Characteristics

This experiment studies subject behaviours in different conditions. Since different subjects may behave differently, it is necessary to analyse the experiment data for each individual subject. In order to detect patterns, each subject is required to take many trials: as explained earlier, each subject needs to finish 6 sessions, each of which contains 394 trials. This makes it hard to conduct experiment using many subjects. Therefore, 7 subjects participated in the experiment.

When choosing subjects, it is important to consider whether novice or expert users should be involved. Users who were as familiar to Boolean logic as possible were chosen, because they are likely to show minimal learning effects, and because real users will become more familiar to Boolean logic within a few hundred hours of use. Although expert users are not good representatives for the whole user class, their behaviour is more consistent, and many nuisance factors can be avoided. Thus, all seven subjects were familiar to Boolean logic at the level of computer science graduate students.

### 7.3.4 Process of the Experiment

Because the subjects were expected to have little familiarity with Karnaugh maps, they were given some instruction before the experiment. They were shown an online presentation that contained examples of textual Boolean expressions and Karnaugh maps. The presentation also gave the subjects general information on how the experiment would proceed. After the online presentation, subjects took an exercise which contains 192 trials with the computer. The exercise allowed subjects get more familiar with textual expressions and Karnaugh maps, and let them exercise using the functional keys and become familiar with the experimental environment.

On the day after their training and exercise, subjects did their first session. To give answers, subjects need to press ENTER key for answering “YES”, SPACE key for “NO”. When one of the keys was pressed, the corresponding button was highlighted to give visual feedback, indicating which choice the subject made. No feedback was provided indicating whether the response was right or wrong.

When all the trials in a session were finished, the total score of the session was presented on the screen, showing the number of correct answers in the session. In the tests, the following data were recorded for each of the trials: the response time, and

whether subject’s answer is correct or not.

## 7.4 Experiment Results

The experimental results were analysed using statistical analysis software. The results are shown in graphical form, which shows the trends more intuitively.

### 7.4.1 Error Rate

All subjects had error rates well below 10% for each session (the mean error rate was 3.29% with standard deviation 4.38%), indicating that they followed the experiment instructions, to do the task as quickly as possible without sacrificing correctness. ANOVA was performed on the scores, as shown in Table 7.2.

Source	DF	F Ratio	Prob > F
Query Complexity	7	1.038	0.4326
Representation Type	1	0.897	0.3532
Display Sequence	1	0.138	0.7136
Representation Type × Display Sequence	1	0.057	0.8127
Query Complexity × Representation Type	7	0.241	0.9701
Query Complexity × Display Sequence	7	0.106	0.9974

Table 7.2: ANOVA of Error Rate

The ANOVA indicates that error rate was not significantly affected by the query complexity, query representation type, display sequence, or any interactions between them. Thus, I expected to find the most interesting results in the response time data.

### 7.4.2 Response Time at Zero SOA

When the error rate is low the usual proxy for ease-of-use is response time at zero SOA, for which there are 2016 data points, one-eighth of the total. Plotted against query and grouped by type of representation, the results are shown in Figure 7.4.

The response time rises roughly linearly with query complexity for both representations, agreeing with Feldman’s experiments. However, while the two representations

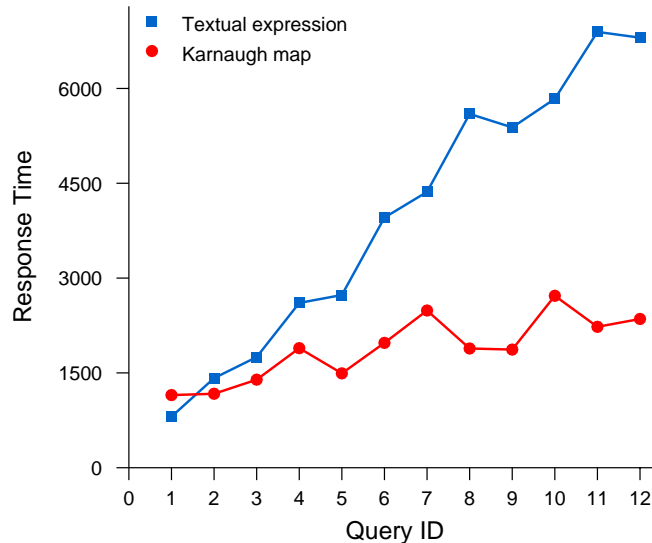


Figure 7.4: Average Response Time at Zero SOA: plotted against query, and grouped by representation.

have roughly the same response time for simple queries, they diverge substantially when queries are more complex, with the response time for the textual expression increasing faster. For the most complex queries in the experiment the response time for the textual expression is about four times the response time for the Karnaugh map.

Karnaugh map clearly scales much better with complexity than the textual expression. Why might that be so? To investigate further it is necessary to examine the results when the SOA is not zero.

### 7.4.3 Plots of Response Time Sequence

For each subject, for each query condition, there is a set of 48 (response time, delay time) pairs. Each set is called a **response time sequence**. There are 336 response time sequences in the experiment (7 subjects and 48 query conditions). They show how the subjects perform as the delay time changes. Each of them was fit to the best elbow pattern, and the parameters retained for further analysis. A typical response time sequence is shown in Figure 7.5.

Before conducting elbow data fitting, outlier removal was done. For each response

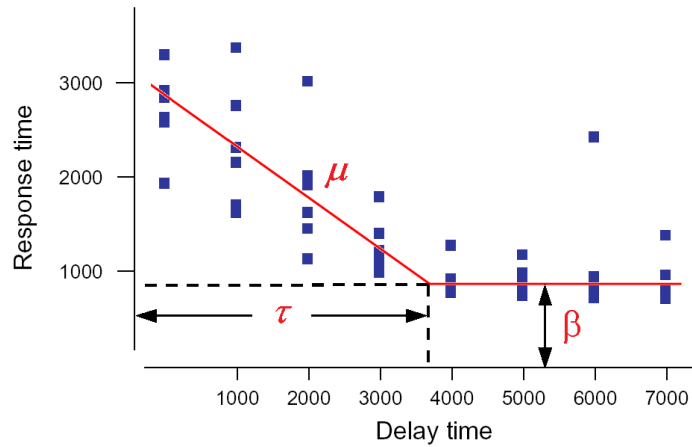


Figure 7.5: Typical Data Set: the data set is plotted with the fitted curve drawn on.

time sequence, for each delay time value, data points lying outside three standard deviations from the mean were removed as outliers.

For each response time sequence, the elbow data fitting program computes the  $\mu$ ,  $\tau$ , and  $\beta$  values. The  $\tau$  value is the time at which the two curves intersect, indicating the time spent on comprehending the first part of the stimulus; the  $\beta$  value is assumed to be sum of the time spent on comprehending the second part of the stimulus and response generation, and the  $\mu$  value is the slope of the descending part of the curve, which indicates how much the subject benefits from the delay time.

A few curves have no elbow, and the estimate of their slope,  $\mu$ , is zero. When this occurs  $\tau$  is undefined. These instances occur for interesting sets of parameters and confirm an important aspect of the result interpretation, so they are further discussed below. For the purpose of consistent data analysis, undefined  $\tau$  is valued as zero.

Figure 7.6 shows the data for all parameters of all subjects. The data is shown in detail because some of the most interesting conclusions are drawn by observing where a group of subjects deviates from the other subjects.

#### 7.4.4 Analysis of the $\mu$ Values

The most obvious feature of the data is that the Karnaugh map in the icon-first mode is qualitatively different from all the other conditions, and that the qualitative difference exists in the data of every subject.



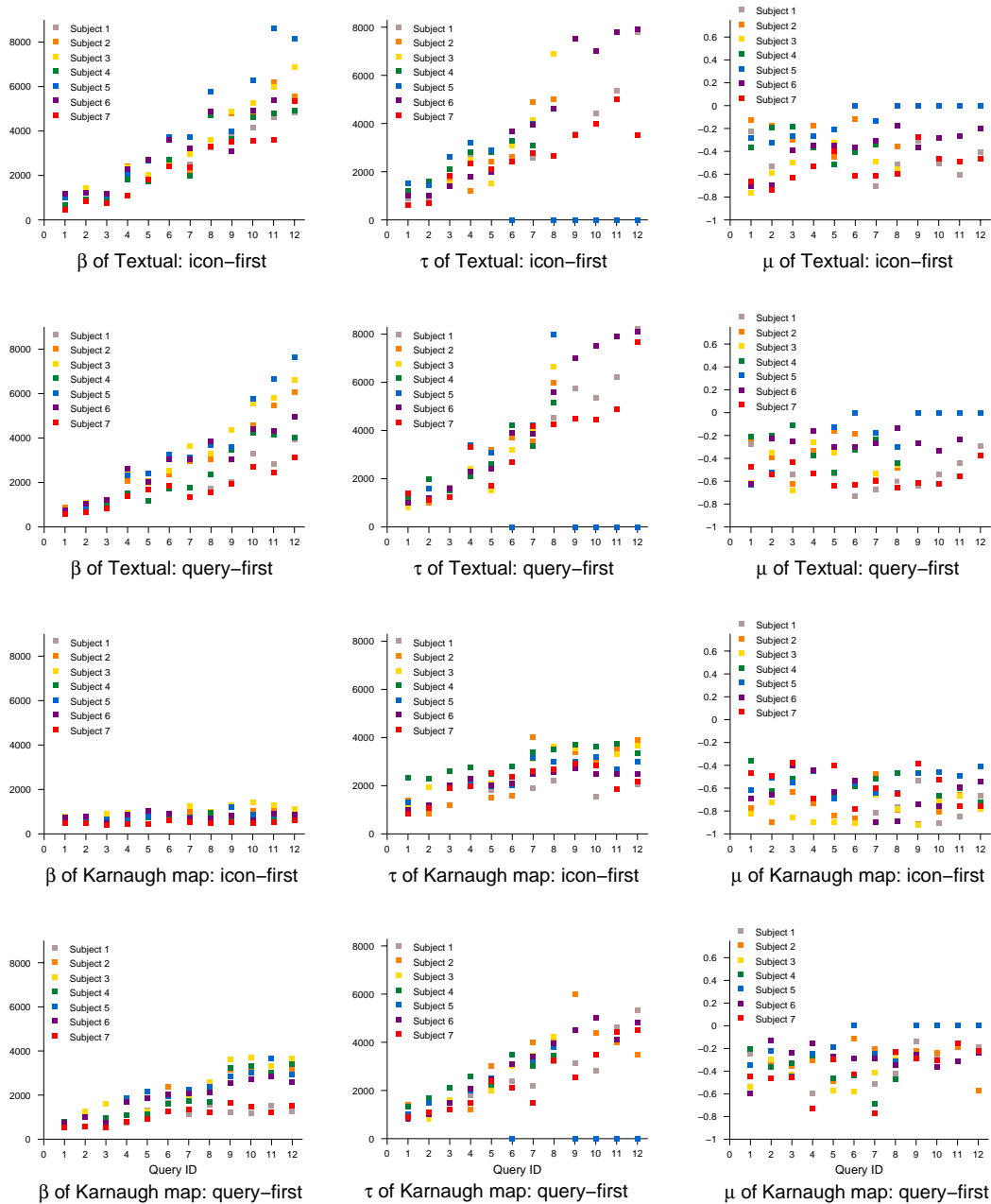


Figure 7.6: The Fitting Parameters for all Twelve of the Presentation Conditions Shown Subject by Subject.  $\beta$ , the amount of processing time, in milliseconds, when the early part of the stimulus has been completely processed, is shown in the left column.  $\tau$ , the amount of time that the early part of the stimulus continues to be processed, in milliseconds, is in the centre column.  $\mu$ , the effectiveness of early processing at reducing the amount of processing afterwards, is shown in the right column.

The key to understanding the difference lies in  $\mu$ , which is the slope of the early part of the curve. The hypothesis has  $\mu = -1$ , which occurs when time spent processing the partial stimulus subtracts an equal amount of time from processing the full stimulus. That is, if subjects can process the partial stimulus as effectively as they can process the full stimulus then  $\mu = -1$ . Contrarily, if subjects cannot usefully process the partial stimulus then  $\mu = 0$ .

The mean values of  $\mu$  in the four presentation conditions were computed, as shown in Table 7.3.

<b>Presentation Condition</b>	<b>Count</b>	<b>Mean</b>	<b>Standard Deviation</b>
Textual,icon-first	84	-0.325	0.237
Textual,query-first	84	-0.343	0.251
Karnaugh map,icon-first	84	-0.626	0.195
Karnaugh map,query-first	84	-0.293	0.184

Table 7.3: Summary of  $\mu$  in the Four Presentation Conditions

In three cases, both textual cases and Karnaugh map query-first,  $\mu$  varies between zero and -0.8, with substantial between subject consistency. There is no significant difference between the means of  $\mu$  in the three conditions. This means that in these three cases some subjects found the partial stimulus quite useful, while others found it to be no help at all.

On the most complex queries (9, 10, 11 & 12), some subjects appear to have ignored the partial display completely, producing slopes ( $\mu$ ) exactly equal to zero. Three subjects did so on Karnaugh map query- first, and four in the textual conditions. In each condition it was the same subjects. When questioned afterwards, they said that some queries were so complex that it was simply not worth trying to understand them. Interestingly, two of the subjects (2 & 3) who exhibited many zero slopes have some slopes close to -1, indicating that they process partial displays well when they can.

The anomalous condition, Karnaugh map icon-first, has values of  $\mu$  between -0.4 and -1.0, indicating that the subjects found the partial display very useful. (The average value is -0.626, which is significantly more negative than the values of the other three presentation conditions.) This indicates that subjects process the partial stimulus more efficiently in this presentation condition. This result seems to show that

subjects are processing the query in a qualitatively different way.

### 7.4.5 Analysis of the $\beta$ Values

Then  $\beta$ , the time taken to complete the task after the stimulus was complete, was analysed. The slope of  $\beta$  with query complexity was analysed using a t-test, with the hypothesis that the slope is not zero, as shown in table 7.4.

Presentation condition	slope	t-Statistic	Prob
Textual, icon-first	597	17.2	$\leq 0.001$
Textual, query-first	495	13.6	$\leq 0.001$
Karnaugh map, icon-first	22	2.57	0.012
Karnaugh map, query-first	211	7.44	$\leq 0.001$

Table 7.4: Slope of  $\beta$  with Query Complexity

Once again Karnaugh map icon-first is very different from the other conditions. Its times are small and do not change with complexity. (The slope is 22 milliseconds, which is only marginally different from zero: t-Statistic = 2.57,  $p = 0.012$ .) The ANOVA analysis of  $\beta$  in this condition showed that  $\beta$  was not significantly affected by query complexity ( $F = 1.02$ ,  $p = 0.44$ ).

In the other three conditions  $\beta$  increases substantially with complexity, (For Karnaugh map query-first the slope is 211 milliseconds: t-Statistic = 7.44,  $p < 0.001$ . For the textual representations the slopes are 597 and 495 milliseconds: t-Statistics 17.2 and 13.6,  $p < 0.001$ .) The results show that Karnaugh map handles complexity better than textual representations, which is interesting from a usability point of view. But, what is most important is the immunity from complexity of Karnaugh map icon-first. It seems to be a clue to something novel.

### 7.4.6 Analysis of the $\tau$ Values

Finally,  $\tau$ , the amount of time for which a subject continues to benefit from processing the partial stimulus, was considered.

In the query-first conditions  $\tau$  rises rapidly with query complexity and subjects continue to get information from a query for as long as eight seconds. This is not

surprising since all experiments find long response times for comprehending Boolean queries. For the textual icon-first condition,  $\tau$  rises equally fast. What is a subject doing, for example, during the sixth and seventh seconds that is giving them further information? Note, however, that the values of  $\mu$  are close to zero in all three conditions, so it is possible that subjects continue to benefit for so long because the benefit is generated very slowly. To be sure, the rapidity with which  $\beta$  rises when complexity increases indicates that the benefit was small.

The  $\tau$  value for the Karnaugh map icon-first condition is higher than the other three conditions for low complexity queries. It increases only when the number of query terms grows. But for queries with the same number of query terms, there is no significant difference between the  $\tau$  values.

An ANOVA was performed on  $\tau$  in this presentation condition, using query term number and query complexity as the variables. The result further proved that in this presentation condition,  $\tau$  is significantly affected by query term number, but not significantly affected by query complexity.

<b>Analysis of Variance</b>	<b>Source</b>	<b>DF</b>	<b>F Ratio</b>	<b>Prob &gt; F</b>
$\tau$	Query term number	2	61.97	$\leq 0.0001$
$\tau$ of 1-term queries	Query Complexity	1	0.02	0.89
$\tau$ of 2-terms queries	Query Complexity	3	0.72	0.55
$\tau$ of 3-terms queries	Query Complexity	5	0.28	0.91

Table 7.5: ANOVA of  $\tau$  in the Presentation Condition of Karnaugh Map Icon-first

This observation is interesting because this condition also has  $\mu$  values that are closest to -1, which means that the time is being used most effectively. Thus it is natural to ask what the subjects are learning, that seems to use time so effectively, that is independent of complexity, and that leaves so little processing to be done after the complete stimulus is available.

#### 7.4.7 Consistency Check

At this point, both the zero SOA and non-zero SOA results have been analysed. As shown, they are qualitatively consistent: the Karnaugh map is faster overall than the textual expression, and its response times grow more slowly with complexity. It is

necessary to check whether the two analyses are quantitatively consistent.

There are two ways for a subject to do the zero SOA trials: by focussing first on the query or by focussing first on the icon. Is the result of zero SOA a mixture of the two strategies? Or does one strategy dominate? The consistency check of the zero SOA and non-zero SOA results provides a way of answering the question: If the non-zero SOA results of one strategies is quantitatively consistent with the zero SOA results, then the subject does the zero SOA trials using this strategy.

There are two types of non-zero SOA results: the results having zero-valued  $\mu$ s, and the results having non-zero  $\mu$ s.

For the most complex queries (9, 10, 11 & 12), some subjects produced values of  $\mu$  exactly equal to zero. In these conditions, the  $\beta$  value was compared with the mean response time  $T_0$  at zero SOA. They were essentially the same, with differences varying between zero and 700 milliseconds. This result further demonstrates that when  $\mu$  equals zero, the subject is waiting for the second part of the stimulus to appear, without trying to process the first part. When the second part was presented, the stimulus was processed as in the zero SOA trials. Zero-valued  $\mu$ s appeared in three presentation conditions: textual icon-first, textual query-first, and Karnaugh map query-first. In these three conditions, subjects appear to have difficulty processing the partial stimulus when queries are complex. But this result did not obtain in the Karnaugh map icon-first condition.

When  $\mu$  is non-zero, the product  $\mu\tau$  measures the amount of useful work done when the partial stimulus was present, then  $\beta - \mu\tau$  is the expected response time to perform the task at zero SOA. For each subject and query condition, with non-zero  $\mu$ ,  $\beta - \mu\tau$  was computed and compared with the mean response time  $T_0$  at zero SOA. To measure their difference, I use the variable  $T_{diff}$ , defined as

$$T_{diff} = \beta - \mu\tau - T_0.$$

The value of  $T_{diff}$  indicates the difference between the expected response time with the practical response time at zero SOA. If one strategy was used at zero SOA, the expected response time using that strategy should be close to the measured response time  $T_0$ . Then the value of  $T_{diff}$  should be close to zero. If  $T_{diff}$  is significantly different from zero, that means at zero SOA, that strategy was not used.

The values of  $T_{diff}$  of each query condition were plotted against query id in Figure

7.7, and grouped by presentation condition.

As shown in the graph, the values of  $T_{diff}$  in the three presentation conditions, textual query-first ( $E(T_{diff}) = -33$ ), textual icon-first ( $E(T_{diff}) = 122$ ), and Karnaugh map icon-first ( $E(T_{diff}) = 203$ ), are close to zero. The slope of the curves in these three conditions are approximately flat (the slope coefficient of textual query-first is -14, textual icon-first is 18, Karnaugh map icon-first is 29).  $T_{diff}$  does not significantly grow with query complexity in these three conditions.

This indicates that the zero SOA and non-zero SOA results were quantitatively consistent in these three conditions. It is impossible to choose between icon-first and query-first when using textual expressions, and the behaviour of doing zero SOA trials using Karnaugh maps is consistent with icon-first strategy.

The values of  $T_{diff}$  of low complexity queries in the Karnaugh map query-first condition are also close to zero. This fact indicates that it is impossible to choose between icon-first and query first when using Karnaugh maps at low complexity.

But the  $T_{diff}$  curve of the Karnaugh map query-first condition is obviously different from the other three conditions. It grows substantially with query complexity (the slope coefficient is 173). For high complexity queries, the values of  $T_{diff}$  are substantially different to zero in this presentation condition. For example, the value of  $T_{diff}$  for query 12 in the Karnaugh map query-first condition is approximately 2000 ms. But for the same query, the value of  $T_{diff}$  in the Karnaugh map icon-first condition is only 410 ms.

This fact indicates that the query-first strategy is not consistent with the behaviour of doing zero SOA trials using Karnaugh maps for complex queries. When using Karnaugh maps at high complexity, icon-first must be the strategy in use. This observation confirms the general rule of thumb, which is not without exceptions, that when there are several ways of doing a tasks humans quickly learn the best one.

#### 7.4.8 Subjective Evaluation

After the tests, subjects were asked to subjectively evaluate Karnaugh maps and the textual expressions. All agree that Karnaugh maps are easier to use when the queries are complex. Several subjects mentioned that when comprehending complex textual expressions, they have difficulty remembering what they had already read, and have to reread it, which takes time. Some subjects also mentioned that finding matching

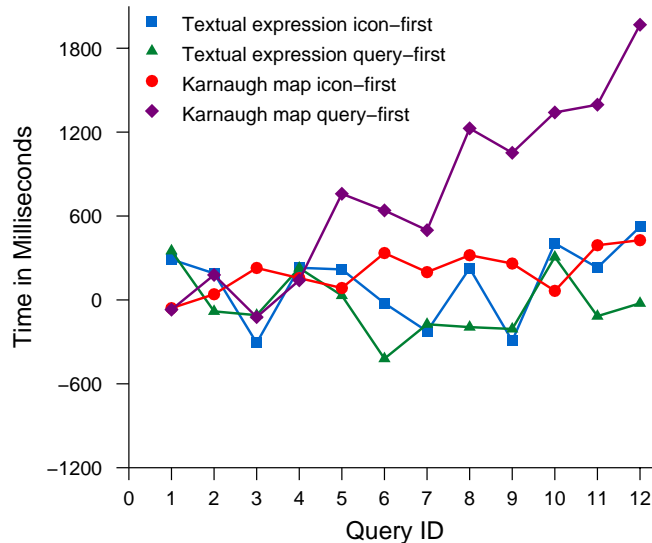


Figure 7.7: The Values of  $T_{diff}$  of Each Query Condition: they are plotted against query, and grouped by representation. The values indicate the differences between expected response time with the practical response time at zero SOA

parenthesis in textual expressions is hard.

## 7.5 Discussion

The most remarkable result in the experiment is the performance of subjects in the Karnaugh map icon-first condition. However, the benefit of this condition is apparent only because the evaluation encompassed queries of complexity higher than usual.

After finishing all the test sessions, subjects were asked how they performed so well in this condition, and the response was both obvious and insightful: seeing the icon they located its position in the Karnaugh map, and when the query appeared they inspected only that position, checking whether the cell was selected or not, which was very fast. Other subjects, given their comparable performance, must have followed the same strategy, but without being able to enunciate it. Is this result a feature special to Karnaugh map or a more general property which has been missed because evaluations failed to examine a wide enough range of complexities?

I believe the latter. Consider the two ways of doing the matching task: focussing

first on the icon, the user substitutes the icon into a query and checks whether it is consistent with the query being presented on screen, or focussing first on the query, the user solves the query to find the selected icon set, then asks if the icon is present in the set.

The first strategy works well with a simple icon and a complex query, the second with a complex icon and a simple query. To explain the latter: a complex icon has many features and the user does not know which ones are relevant, so can do little with the icon to prepare for the query. The display of relevant features with the icon ensured that it was always simple.

Beyond these simple observations, the distinction between the two styles of solution runs deeper. The icon-first mode is similar to inductive science: while the icon is displayed the user accumulates hypotheses, then searches the query to see if they are consistent. As soon as an inconsistency is discovered a negative answer is given. In the absence of inconsistency a positive answer is given. If a rich enough set of consistency checks can be created this method will operate at a high level of correctness, like experimental science. I call this the inductive method of query comprehension, which produces a large number of consistency conditions, each of which can be checked very quickly. It has both the strengths and weaknesses of inductive science: the ability to find a usually true response on the basis of a small amount of evidence is its strength; the likelihood of being wrong because of an overlooked or misunderstood aspect is its weakness. In the experiment the subjects were able to keep the error rate well below 10% using this method.

On this basis it is easy to see why Karnaugh maps perform so well. Because the icon occupies a unique location in the visual query there is a small number of consistency checks to examine, and because the site of information about consistency is localized in space the ability of visual attention to focus on a specific location makes the response easy. There is a logarithmic dependence on query complexity for operations that analyse the icon, and no dependence on query complexity for inspecting the appropriate location once the query appears, provided only that there is enough SOA for the icon to be analysed completely.

The query-first mode is similar to deductive science: while the query is displayed the user accumulates possible solutions against which the icon will later be checked. This works very well when the query is simple enough, but fails as the query gets



more complex. As shown in the experiment, when the query gets complex enough some subjects even stop trying to process the query at all, presumably falling back on another strategy that is possible only when icon and query are simultaneously present. To most of us it seems strange that we automatically assume that the brain always will deal most easily with logic using deductive methods. Presumably we are so likely to expect deduction because logic occurs in science and engineering as a branch of mathematics, and not, as Aristotle saw it, a mode of human thought.

The deductive and inductive methods are not limited to analysing Karnaugh maps and textual expressions. In fact, they can be used to analyse other visual query representations such as Venn diagrams, InfoCrystal, TEBI, Filter/Flow, expression trees or LabView. By analysing which query comprehension method is enabled and how it is enabled by a query representation, subjects' performance can be predicted so that various query representations can be analysed and compared at a theoretical level.

Different query representations support different methods of comprehending queries because they are based on different query models. Textual representation, expression trees, and flow diagrams are all based on logical expressions. When using them, subjects have to predict from the query what the results will be. Logical structures of the queries have to be understood. Therefore, these query representations are expected to have qualitatively similar performance: in particular, vulnerability to query complexity. For queries having complex logical structures, using such representations is hard.

On the other hand, Karnaugh maps, Venn diagrams, InfoCrystal, TEBI, and all the visual query representations derived from my formalism are based on truth-table. Thus they all should have similar properties.

As a practical matter, several features of Karnaugh maps that make them effective in the icon first condition can be identified: a simple rule mapping icon characteristics to a particular location on the display, a clear indication of the relevant attributes of the icon, and minimal processing to test consistency. Furthermore, Figure 7.7 also shows that the inductive method is effective when all parts of the stimulus occur simultaneously: SOA is a tool for understanding what is hidden in Figure 7.5, not a precondition for the processes it allowed us to observe.

Other visual query representations can be evaluated in terms of these features to determine whether or not they are likely to share the complexity resistance of Karnaugh map. InfoCrystal [88], for example, has rules mapping icon characteristics to display

locations and a clear indication of attribute relevance. But the mapping rules are not simple, and the consistency tests are perceptually difficult. The limitations of Infocrystal are, it seems, perceptual, and not cognitive. Thus, the concepts been introduced identify the weaknesses of query representations, and also show how they can be improved.

## 7.6 Summary

Compared with previous experiments on visual query languages, the experiment presented in this Chapter has the following features.

- By manipulating SOA, users' cognitive activities when using the query representations are separated from one another. The time components are isolated, so that users' cognitive activities are analysed in more detail.
- The reasons why one query representation performs better than another are investigated.
- Queries with a wide range of complexities are studied in the experiment. Query complexities are computed using Feldman's algorithm, which is more precise than the classification method being used in previous experiments. This allows us examine how the difficulty of understanding a query representation scales with query complexity.

As shown by the experimental results, the methodology of manipulating SOA for isolating cognitive activities is efficient, allowing us find user behaviour that was overlooked in previous studies. From the experimental results, the following conclusions are derived.

- The overall performance using Karnaugh map is better than using textual representations. This benefit increases when the query complexity grows.
- Users understand queries in two qualitatively different ways: inductive and deductive.
- Performance using the deductive method is significantly affected by query complexity.

- Karnaugh maps facilitate inductive understanding, which is relatively independent of query complexity.
- For Karnaugh maps, the inductive method is easier to use than the deductive method.

Although in the experiment only query matching tasks were used, the inductive and deductive methods found in the experiment can be used for analysing other query tasks. The inductive method reveals a process of formulating queries: with a set of items, users summarize the Boolean concept that describes them. As shown by the experiment results, Karnaugh map facilitates induction, which eliminates the necessity of formal specification of relations between the query terms.

Furthermore, the inductive and deductive concepts can be used to identify the weaknesses of query representations, and show how they can be improved.

# Chapter 8

## Conclusions

The goal of this thesis is to apply visualization techniques to design and improve user interface tools for exploring and analysing data. This goal was accomplished by formalizing selection and presentation in information visualization, and by a new experimental paradigm for examining cognitive processes in Boolean information selection. The main contributions of this thesis are:

- a new mathematical formalism which describes the basic components and their relationships in visual query interfaces for information exploration systems;
- a new visual query interface, KMVQL;
- a new experiment methodology for evaluating query languages; and
- an empirical user study comparing the comprehensibility of Karnaugh map representations with textual representations of Boolean expressions.

This chapter reviews these contributions in detail.

### 8.1 The Formalism for Visual Query Interfaces

Most topics in visual information exploration environment have been studied in isolation. The intrinsic connections between data visualization, query specification, and query result visualization have been little studied.

What is the relationship between multi-dimensional data visualization and visual query representations? Are there any relationships between widget-based query devices

and selection technique like brushing and linking? How should compound brushing be supported? How might new representations for Boolean queries be designed? Such questions, and many others, seem unrelated with one other, but the formalism shows them to be specific instances of more general underlying questions. Noticing the similarities gives ideas for designing more intuitive visual query interfaces and extending existing ones.

The formalism presented in the thesis shows the relationships among the basic components of information exploration. The formalism was summarized in Chapter 3 under the following aspects, which are restated here.

1. All entities in the query/visualization environment are represented as tables. Data sets are cardinal-valued; queries are Boolean-valued; and visual representations are icon-valued. The visualizing and querying operations are defined as mappings in the formalism, which are also tables. The fundamental operation of the formalism is applying one table to another.
2. The transformation model describes the relationship between the components. Innovative visualizations for Boolean querying can be derived by applying multi-dimensional visualization techniques to visualizing query space. Direct manipulation data selection operations are visual queries applied to visual representations. Complex logical combinations comprising multiple selections can be implemented exactly as simple queries of data sets.
3. The formalism shows how to integrate query specification with displaying query results, which helps users make sense of the data which facilitating query refinement and modification.
4. All mappings in the query/visualization environment can be visualized. Legends are examples of visualizing visualization mappings; query devices are examples of visualizing query mappings. Seeing them as results of visualizations makes it possible to apply existing visualization techniques when designing user interface widgets for visualization and querying. The formalism also provides guidelines for improving or adding visual cues to existing user interface widgets such as radio buttons, sliders, and combo boxes. Thus, more informative and powerful user interface widgets can be designed for visual querying.

5. The formalism also introduces the recursive application of querying, which supports querying a query, or visualizing, which supports visualizing a visual representation. The recursive applications allow users to ask questions that help them understand complex queries or visualizations, which provides systematic reduction of complexity.

Some results claimed in this thesis have been presented as examples. They demonstrate that the formalism is capable of providing a common description of different interfaces for creating visual queries.

A formalism is a tool to assist thinking. The formalism for information exploration captures querying and visualizing in a single conceptual framework. Thus is an important contribution to the theory of visual information exploration. The formalism has the following benefits.

- It provides a systematic integration of querying and visualization,
- It is a foundation for analysing and comparing information exploration environments.
- It provides a well-defined set of capabilities required by implementations.
- It provides concepts for designing new querying and visualizing interfaces.
- It reveals opportunities for extending existing visual query interfaces.

## 8.2 The Software Design Model and the Application

The formalism is a model that is more general than any specific implementation. To fill the gap between the formalism and implementing practical systems, a software design model is presented in Chapter 4. The software design model consists of two parts.

- User actions. The model describes the basic user actions that must be handled by a visual query interface for information exploration.
- Software domain model. The domain model describes the software modules that must be included in a visual query interface architecture as well as the responsibilities of the modules, which define clearly what modules should do what.

The software design model is an explicit instance of the formalism. It provides detailed guidelines for designers implementing new visual query interfaces.

A new interactive visual query interface, KMVQL, was implemented using the formalism and the software design model. It provides a concrete example of a practical software application based on the formalism. It shows how the elements are realized and how they interact with each other. There are many advantages to developing interactive visual query interfaces using formalisms as guides. In building the KMVQL interface, I observed the following benefits.

- Innovative visual representations for Boolean queries. In KMVQL two types of visual representations are used: Karnaugh maps and iconic representations, which are seamlessly integrated with the visualization of query results to help users comprehend the data and accomplish query refinement or modification.
- Innovative methods for specifying Boolean queries. To specify a Boolean combination of query terms, users select the corresponding cells in the Karnaugh map, without considering explicit logic operators, which makes Boolean query specification much easier.
- New query device types. KMVQL not only supports traditional interface widgets such as checkboxes and sliders, but also provides several new types of query widgets: pie-charts, x-y-plots, parallel-coordinates, and Karnaugh maps.
- Flexible management of query devices. New query devices can be added to the interface as needed; users can remove unused query devices from the interface to make the display less crowded.
- Intermediate exploration states can be saved and reloaded. Previous search results can be reused as components of new queries.

Most of these features are not supported in existing systems for visual information exploration. As shown in Chapter 4, they are important for facilitating the information exploration process. With KMVQL, users are able to complete complex tasks in shorter periods of time.

## 8.3 The Experiment Methodology

When designing a novel query interface, it is important to evaluate it, to find out how it compares to other query interfaces, and why it performs better or worse.

Chapter 6 introduces an experimental methodology for evaluating query languages. The methodology solves a specific problem of query language evaluation experiments: how to isolate the cognitive components of user responses? Isolating the cognitive components allows us to understand users' behaviours and thoughts in more detail. Thus, questions like why one query representation performs better than another can be answered.

The innovation in this methodology is the manipulation of the stimulus-onset asynchrony (SOA) of different parts of the stimulus. Manipulating SOA encourages subjects do a specific subtask first, with the effect that as the delay time(SOA) is increased, the measured response time decreases to a point of inflection, after which it stays constant, which means the relationship between response time and delay time follows an elbow curve. Using an elbow data fitting algorithm, the time taken for completing each task components is isolated.

Although the methodology was presented in the context of visual query language evaluation, it is a general method. When there are many stimulus parts that affect a subject's performance, the methodology can be applied to measure how each stimulus part affects performance. Therefore, it can be readily used in other HCI experiments, such as evaluating a new interface, evaluating a graphical representation, etc.

## 8.4 The Experiment

An experiment was conducted comparing the comprehensibility of Karnaugh map representations with textual Boolean expressions. The results show that Karnaugh maps scale very well with complexity.

The most important contribution of the experiment is the two qualitatively different methods by which humans understand queries: inductive and deductive. The inductive method provides users with efficient solutions to logical problems. Karnaugh maps facilitate inductive understanding, which is relatively independent of query complexity. This explains why Karnaugh maps increasingly outperform textual Boolean expressions



as query complexity increases.

The deductive and inductive methods can also be used to analyse other visual query representations. By analysing which query comprehension method is enabled and how it is enabled by a query representation, subjects' performance can be predicted, allowing query representations to be analysed and compared theoretically.

# Chapter 9

## Future Work

This thesis consists of four parts: the formalism, applications of the formalism, the experimental methodology for evaluating Boolean query representations, and the empirical study. The directions for future work are discussed from these four aspects.

### 9.1 Extensions of the Formalism

The formalism is based on tabular definitions for queries that use Boolean logic. But Boolean models are limited in their scope. For instance, the models assume the following.

- All terms in a query are equally weighted, so that there is no need or provision for assigning a degree of importance to terms in a query.
- For OR (ANY) searches, data items meeting only one query term are as relevant as data items meeting all the query terms.
- For AND (FINDALL) searches, data items containing one or more query terms are as irrelevant as data items containing no query terms.

Conventional Boolean models support binary assortment: data records are either relevant (“1”) or irrelevant (“0”). This approach to scoring relevance is shortsighted because for users the notion of relevance is not static but dynamic. That is, for a given user, what is irrelevant today may be relevant tomorrow. In addition, the degree of relevance varies with synonymy associations between terms.

The fuzzy set model [11, 14], the vector space model [105] and the extended Boolean model [61, 82] have been proposed answers to these limitations. They are extensions of the Boolean model because they reduce to the Boolean model when query term weights are restricted to zero or one. They have been applied in information-retrieval for combining Boolean retrieval systems with document ranking.

These extended query models may be useful for information exploration. Therefore, it is an important future research direction for my formalism to extend it beyond Boolean logic. For example, how can my formalism support queries based on fuzzy logic? Is it possible to describe weighted queries with the table model?

In addition, different data models produce different visualizations. Many existing query interfaces are based on Boolean expressions. Currently queries in the formalism are defined based on truth-tables. It is possible to formalize other models for queries that go beyond Boolean expressions and truth-tables.

Furthermore, formal descriptions of a user interface reveal its basic components and behaviours, which improves the design of new interfaces, and generates opportunities for extending existing ones. Although the formalism presented here focuses on visual query interfaces for information exploration, I believe that the principle of finding a generic model for components and their relationships can be extended to other types of graphical user interfaces.

## 9.2 Applications of the Formalism

Another area of future work related with to formalism is exploring other applications of it. The table model permits the application of queries and visualizations to all entities in the query/visualization environment. The techniques and examples presented in this thesis constitute only a small part of the possible applications. Other designs for querying and visualizing can be found by applying the formalism, which will enrich visual information exploration.

Applying visualization techniques to visualizing queries will provide interesting new designs for visual query interfaces. In a visual query interface, it is helpful to provide users with a choice of query formulation methods. In the future, more visual representations for Boolean queries will be implemented and integrated into KMVQL. There is also much interesting work to be done in developing new query devices and interactive

legends simply by applying existing visualization and interaction techniques.

Today's information exploration tools support only simple query and visualization mappings. The formalism supports recursive models, which provide a theoretical basis for more complex interactions between users and data, such as visualizing a visualization, querying a query, or visualizing a query about a visualization mapping, etc. These possibilities are the future of information exploration and require tools that support them. Developing tools to support these complex interactions is another good direction for future work.

### 9.3 The Experiment Methodology

The methodology presented in Chapter 6 focuses on evaluating the comprehensibility of query representations. The methodology uses query matching tasks, and the independent variable is response time.

In addition to response time, there are many other dimensions to the performance of a user-computer system, such as learning, recall, concentration, fatigue, and acceptability. These dimensions, as summarized by Card [16], are also important in query language evaluation experiments.

- Time. How long does it take a user to accomplish a given set of tasks using the query language?
- Errors. How many errors does a user make using the query language and how serious are they?
- Understandability. How easy is it for a user to understand a query represented in the query language?
- Usability. How easy is it for a user to formulate a query with the query language?
- Learning. How long does it take a novice user to learn how to use the query language?
- Functionality. What range of tasks can a user do with the query language?
- Recall. How easy is it for a user to recall how to use the query language on a task that she has not done for some time?

- Concentration. How many things does a user have to keep in mind while using the query language?
- Fatigue. How tired do users get when they use the query language for extended periods?
- Acceptability. How do users subjectively evaluate the query language?

Since manipulating SOA to isolate task components is a general principle, it can be readily used for other dependent variables to evaluate other aspects of a query language.

The methods of manipulating SOA are very powerful for determining how and why perceptual and cognitive efficiencies do, or do not, occur. Thus, applying the methods to other issues in applied perception is an important direction for future work. Examining the time course of pattern perception in visualization is an attractive example.

## 9.4 Empirical User Studies

Chapter 7 describes an empirical user study. There are several directions for extending the current study and conducting future experiments:

- The subjects who participated in the study are familiar to Boolean logic. In the future, it is necessary to conduct the study could be conducted with novice users to see how the results vary with learning;
- The experiment introduced in Chapter 7 studies the comprehensibility of textual Boolean expressions and Karnaugh maps. Other representations for Boolean queries can be studied and analysed based on the same methods;
- The current experiment tests queries with 1, 2, and 3 query terms. More complex queries in higher dimensions can be studied to see if the patterns found for lower dimensional queries exist for higher dimensional queries.

## 9.5 Future Work Summary

The mathematical formalism paves the way for future research in visual query techniques. More research can be done in extending the formalism beyond Boolean logic, and exploring more applications of the formalism. The experiment methodology and empirical study can also be extended to study other aspects of a query language.

# Appendix A

## A story about information exploration

In order to explain the problems with information exploration systems, we use a fictitious user, Iris, and describe the problems she met in her information seeking process.

Iris just graduated from university, the location of her working place is far from where she lives. So she wants to buy a used car because she doesn't have much money. But Iris wasn't familiar with the car market, and she has no particular preference of what kind of cars she wants. Then she seek for advice from her friend Mary.

Mary recommended Iris to use the software "AutoExplorer" to search for a desired car. "Because", Mary said, "AutoExplorer connects to a most up-to-date database of cars for-sale and it provides various commonly used interface mechanisms for information seeking."

Iris was very glad at the suggestion. She thought, "Since AutoExplorer provides various mechanisms for information seeking, there must be one that suits me." So she installed AutoExplorer on her computer and begin to use it.

When the program starts up, Iris found that in the user interface, there are several different methods that can be used to specify information needs, they are: Natural Language, SQL Commands, Form-Filling for Features, and Dynamic Query. Iris was not familiar with the meaning of "SQL Commands, Form-Filling for Features, Dynamic Query". She just thought that "Natural Language" sounds to be a very natural and easy way. So she choose "Natural Language" as the input mechanism.

## A.1 Specify queries with Natural Language

Iris had no particular preference of what kind of cars she wants, She thought “probably I need to try some general features.”. Her first thought is to search for some cars with price less than 8,000 dollars, the cars could be in Make Acura or Ford. So she input

*“I am interested in the cars with the Make Acura and Ford and the price less than 8,000 dollars.”*

To her surprise, the return of the search is “No Cars Found”. “How come? There should be a lot of cars meet this query!”, Iris thought, “Probably I should use OR instead of AND?” Then She modified her input as

*“I am interested in the cars with the Make Acura or Ford and the price less than 8,000 dollars.”*

This time, 12,000 results returned. But Iris soon found that some cars in Make “Acura” are with very high price. For example, there’s a car item with the following features: Make = Acura, Price = 25,000. “This kind of data should not be returned because I have restricted the price limit as 8,000 dollars. The result number is so huge, it even contains those void data, how can I find my desired car from such huge amount of data? I really don’t want to waste my time to go through each data one by one!”

Iris was very unsatisfied at the results. So she gave Mary a telephone call telling her the problems she met with “Natural Language” specifications. Mary said: “Yes, natural language is not accurate, the meaning of ‘and’ is ambiguous and sometimes confusing. The range of limiting factors could be different in different context.” After saying this, Mary give Iris another suggestion: “Why not try using SQL Commands? SQL is accurate and powerful.” Then she gave Iris a brief introduction of Boolean logic and how to formulate SQL Commands.

## A.2 Specify queries with SQL commands

“Although I don’t want to learn a command language, in order to find my desired car, I have to try to learn it.” Iris read online help information for SQL commands provided by AutoExplorer, placed a SQL manual at hand, and input a command:



*Select All from Cars where Make=Acura or Make=Ford and Price ≤ 8000*

Iris was depressed to see that the result returned was the same as she was using natural language, still with 12,000 items. When she checked the manual, she found that “There’re rules of precedence for the Boolean operators and I need to use parentheses here.” So she modified her query into:

*Select All from Cars where (Make=Acura or Make=Ford) and Price ≤ 8000*

This time, 7500 items returned. But for Iris, this number is still too huge. She only browsed the first five records, find none that meets her expectation. So She add another constraint into the query:

*Select All from Cars where (Make=Acura or Make=Ford) and Price ≤ 8000  
and Year ≥ 2000*

The result of this search is “No Cars Found”. “Probably the constraint is too strict. If a car is made after the year 2000, the acceptable price could be higher”, thought Iris. She decided to modify the query to express the constraint that “if a car is made before 2000, the price for it should be less than 8000, otherwise, the price should be less than 12000.” She felt it’s not easy for her to express this clearly with the mathematical formula. Her first attempt is:

*Select All from Cars where (Make=Acura or Make=Ford) and (Price ≤ 8000 and Year ≤ 2000) and not Year ≤ 2000 and Price ≤ 12000*

The result of this search is “No Cars Found”. The second attempt:

*Select All from Cars where (Make=Acura or Make=Ford) and (Price ≤ 8000) and (Year ≤ 2000) or (not (Year ≤ 2000) and (Price ≤ 12000))*

The result of this search is “Syntax Error, the parentheses don’t match”. The third try:

*Select All from Cars where (Make=Acura or Make=Ford) and (Price ≤ 8000) and (Year ≤ 2000) or (not (Year ≤ 2000) and (Price ≤ 12000))*

This time, 8900 items returned. “The number is even greater!”. The fourth try:

*Select All from Cars where (Make=Acura or Make=Ford) and ((Price ≤ 8000) and (Year ≤ 2000) or (not (Year ≤ 2000) and (Price ≤ 12000)))*

500 items returned. But Iris was so tired at the multiple tries and she was not sure whether the mathematical expression is correct or not. She had no confidence about it. “I only used the Make, price and year as query constraints, it’s already cost me too much time, how can I add more features as constraints? SQL is too hard for me!”

Iris called Mary again: “SQL is too hard for me, I have difficulty in using the logical operators and I don’t like parentheses. Are there any easier ways of specifying my queries?” Mary said: “You can try using form-filling interface or dynamic query method. With them, you don’t have to worry about the logical operators any more. The dynamic query interface even provides graphical representation for the cars database which can give you an overview of the price distribution and help you get some information about the car market.”

### **A.3 Exploring Data With Form-Filling Interface**

When Iris choose “Form-Filling for Features” in AutoExplorer, an electronic form is displayed, as illustrated in Figure A.1.

In the interface, a set of list widgets are provided. Iris was glad to find that she can specify her desired features by selecting the items provided in the widgets. “Yes, it’s much easier.” So she restrict the car features as:

Make: Acura  
Model: Legend  
Colour: Pink  
Mileage: Under 15,000  
Doors: 4 doors  
Year: 2000 to 2005  
Price: 10,000 to 20,000

When she submitted the query, the result returned from the system was: “No Cars Found: We’re sorry, but no results were found that matched your search options.”

**AutoExplorer**

**Make** Acura [Change make](#)  
**Model** Legend  
**Body Style** All Styles  
**Color** Orange  
 Pink  
 Purple  
**Mileage** Under 15,000  
**Doors** Four Door  
**Engine** 6 Cylinder  
**Fuel Type** All Types  
**Drive Type** 4 wheel drive  
**Transmission** Automatic  
**Year** 2000 to 2005  
**Price** 10,000 to 20,000  
 Cars with prices only  
**Distance** 50 miles from 23456  
**Certified**  Search for Certified Cars only.

Figure A.1: AutoExplorer: Form-Filling Query Interface

“Now, what shall I do?” Iris had to modify her query criteria so that some car information can be returned. But how? She guess that “probably the colour should be changed since it’s so rare to find a car in pink colour”. Then she changed the colour option to “All colours” and submitted the new query, but the message returned still was “No Cars Found”. It’s really frustrating because Iris had to try her luck to decide which query criteria to modify, since from the system feedback she got no information concerning how to modify her query.

But after several steps, there still no cars found for the Iris’s queries. At last, in order to obtain some car information, Iris modified her options as:

Make: Acura

Price: 5,000 to 18,000

This time, the system returned 10885 items in a long list. It’s easy to be overwhelmed by such large amount of information. So Iris had to refine her query gradually, still try her luck. “It’s really hard for me to specify proper features since I’m not familiar with the car market.” Also she found a big problem of the form-filling interface: she can’t formulate some queries involved with disjunction, such as “if a car

is made before 2000, the price for it should be less than 8000, otherwise, the price should be less than 12000.”

“The exploration process is really time consuming and not efficient.” After an hour exploration, Iris still hadn’t found any car satisfied her. She remembered that Mary told her the dynamic query interface provides graphical representation for the cars data set which can help her get some information about the car market. So Iris decided to give up exploring with the Form-filling interface to try the dynamic query method.

## A.4 Exploring Data With Dynamic Query Interface

When Iris choose “Dynamic Query” in AutoExplorer, an interface is displayed as illustrated in Figure A.2.

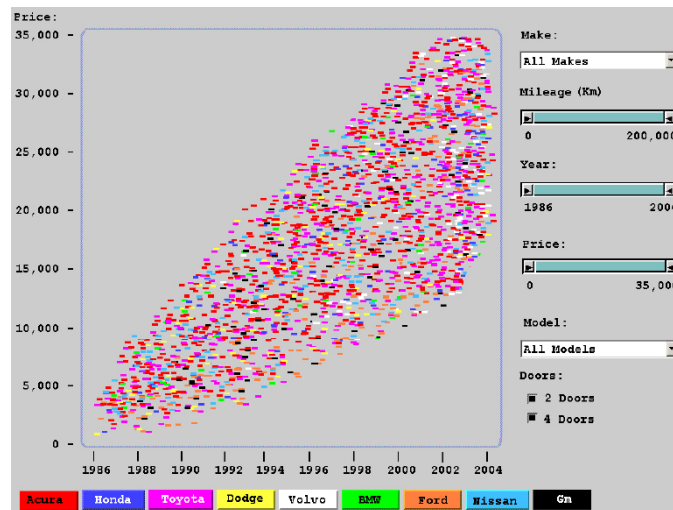


Figure A.2: AutoExplorer Dynamic Query Interface

In the dynamic query interfaces, the window is split into two parts: the left part is the visualization of data, and the right part is the control widgets window.

In the data visualization view, the cars database is visualized with X-Y-Plot: the Year attribute is mapped to X-axis and Price value is mapped to Y-axis. Each car Make is assigned a unique colour. Each car in the database is displayed as a coloured dot in the X-Y-Plot. For example, all the Acura cars are represented as red dots,

Toyota cars are pink dots, Honda cars are blue dots, and Ford cars are orange dots in the plot.

“Now I can see a trend of increasing prices when the year value grows.” Iris was glad that she obtained information from the visualization.

Unfortunately, in the Cars database, there are 9 different car Makes, the total number of cars is more than 50,000. The visual representation is filled with coloured pixels which is so cluttered that Iris can hardly get more information than just the price increasing trend. From the visualization, she can’t distinguish all the “Acura” car from other Makes such as “Toyota” cars and “Ford” cars.

Iris selected “Acura” from the “Make” list, she found that the visualization is updated immediately, only those red dots left in the plot. “Now it looks much better, I can focus on the Acura cars. But how can I make comparison the price difference of Acura car with other Makes?” Then she wanted to compare the price difference between Acura cars and Ford cars. So she selected both “Acura” and “Ford” from the “Make” list, then orange dots appeared in the plot. But the two colours are so close to each other that it’s hard to distinguish all the red dots from the orange dots.

Then Iris wanted to compare the price difference between the cars with Mileage less than 50,000 and those with Mileage value greater than 50,000. But when she operate the slider that controls the Mileage value, only those items with Mileage value less than 50,000 displayed in the plot.

Iris became a little disappointed. But she had to keep exploring with dynamic query interface, since, for Iris, it was the best so far. During the exploration process, Iris had used some query constraints such as:  $Year \geq 2000$ ,  $Mileage \leq 60000$ ,  $Price \leq 15000$ , etc. After reading the visualization for a while, Iris obtained some information about the price distribution and felt that if an Acura car’s price is less than 15000, then it might be acceptable if it meets any one of the two query criteria:  $Year \geq 2000$  or  $Mileage \leq 60000$ . She wanted to retrieve those data items and read them in more detail. But with the dynamic query interface, all the query constraints are only conjunct together. She couldn’t specify such queries with it.

“Probably I can first save the Acura car items that meet both the query criteria  $Price \leq 15000$  and  $Year \geq 2000$ , then save the items that meet both the query criteria  $Price \leq 15000$  and  $Mileage \leq 60000$ . Then put the two saved results together.” thought Iris. She searched the menu items provided in AutoExplorer, couldn’t find the

functions that can support her actions.

“Natural Language causes ambiguity, not accurate. Boolean Logic is difficult for me to use. Form-Filling Interface and dynamic query interface are so simple, can support only the conjunctions of the query constraints, and they provide little context information. None of them suits my situation!” Iris was upset, she finally decided to give up the exploration.

# Appendix B

## Elbow Data Fitting Algorithm

### B.1 Introduction

The elbow data fitting is to fit a piecewise linear function to the data. The linear function, sometimes called an elbow or knee curve, takes the form:

$$f(t) = \begin{cases} \beta - \mu(\tau - t) & \text{if } 0 \leq t \leq \tau; \\ \beta & \text{if } t \geq \tau. \end{cases}$$

The point at which the function changes slope is the elbow or knee of the curve, providing the justification for its name. There are three parameters to fit:  $(\mu, \beta, \tau)$ .

- $\mu$  - the slope of the line before the elbow point;
- $\tau$  - the x-axis position of the elbow point;
- $\beta$  - the y position of the line after the elbow point. The two segments intersect at  $(\tau, \beta)$ .

### B.2 Minimization of the Objective Function

The objective function, which is to be minimized with respect to  $\mu$ ,  $\tau$ , and  $\beta$ , is the squared error,  $E^2$ , defined by

$$E^2 = \sum_{i=1}^n (r_i - f(t_i))^2$$

Replace  $f(t_i)$  with the equation defined earlier, we get:

$$E^2 = \sum_{i=1}^m (r_i - (\beta - \mu(\tau - t_i)))^2 + \sum_{i=m+1}^n (r_i - \beta)^2$$

where  $m$  is defined by  $t_m \leq \tau \leq t_{m+1}$ . Choosing a value  $m$  and minimizing the squared error. In order to find the minima of this function we need to calculate the partial derivatives.

$$\begin{aligned} (1) \quad \frac{\partial E^2}{\partial \tau} &= -2 \sum_{i=1}^m (r_i - f(\mu, \tau, \beta, t_i)) \frac{\partial}{\partial \tau} f(\mu, \tau, \beta, t_i) \\ &= 2\mu \sum_{i=1}^m (r_i - \beta + \mu\tau - \mu t_i) \\ &= 0 \\ (2) \quad \frac{\partial E^2}{\partial \beta} &= -2 \sum_{i=1}^m (r_i - f(\mu, \tau, \beta, t_i)) \frac{\partial}{\partial \beta} f(\mu, \tau, \beta, t_i) \\ &= -2 \sum_{i=1}^m (r_i - \beta + \mu\tau - \mu t_i) + \sum_{i=m+1}^n (r_i - \beta) \\ &= 0 \\ (3) \quad \frac{\partial E^2}{\partial \mu} &= -2 \sum_{i=1}^m (r_i - f(\mu, \tau, \beta, t_i)) \frac{\partial}{\partial \mu} f(\mu, \tau, \beta, t_i) \\ &= 2 \sum_{i=1}^m (r_i - \beta + \mu\tau - \mu t_i)(\tau - t_i) \\ &= 0 \end{aligned}$$

### B.3 Formalism

We can simplify the computation of equations a little by making a few definitions. Define  $m$ ,  $u_m$  and  $s_m$  by

$$t_m < \tau < t_{m+1}, u_m = \sum_{i=1}^m t_i, s_m = \sum_{i=1}^m r_i, v_m^{tt} = \sum_{i=1}^m t_i^2, v_m^{rt} = \sum_{i=1}^m t_i r_i,$$

Based on the partial derivatives, the equations (1), (2), and (3) can be written as:



$$\begin{aligned}
s_m - m\beta + \mu(m\tau - u_m) &= 0 \text{ or } \mu = 0, \\
(s_m - m\beta + \mu(m\tau - u_m)) + s_n - s_m - (n - m)\beta &= 0, \\
v_m^{rt} - \mu v_m^{tt} &= (\beta - \mu\tau)u_m.
\end{aligned}$$

Note that the solution  $\mu = 0$  requires  $m=0$  and leaves  $\tau$  otherwise undefined. Then  $\beta = s_n/n$ , as expected, because we are fitting the data points to a constant.

Then we must solve three linear equations having coefficients depending only on  $m$  and the data:

$$\begin{pmatrix} v_m^{tt} & -u_m & u_m \\ u_m & -m & m \\ 0 & 0 & (n - m) \end{pmatrix} \begin{pmatrix} \mu \\ \mu\tau \\ \beta \end{pmatrix} = \begin{pmatrix} v_m^{rt} \\ s_m \\ s_n - s_m \end{pmatrix}$$

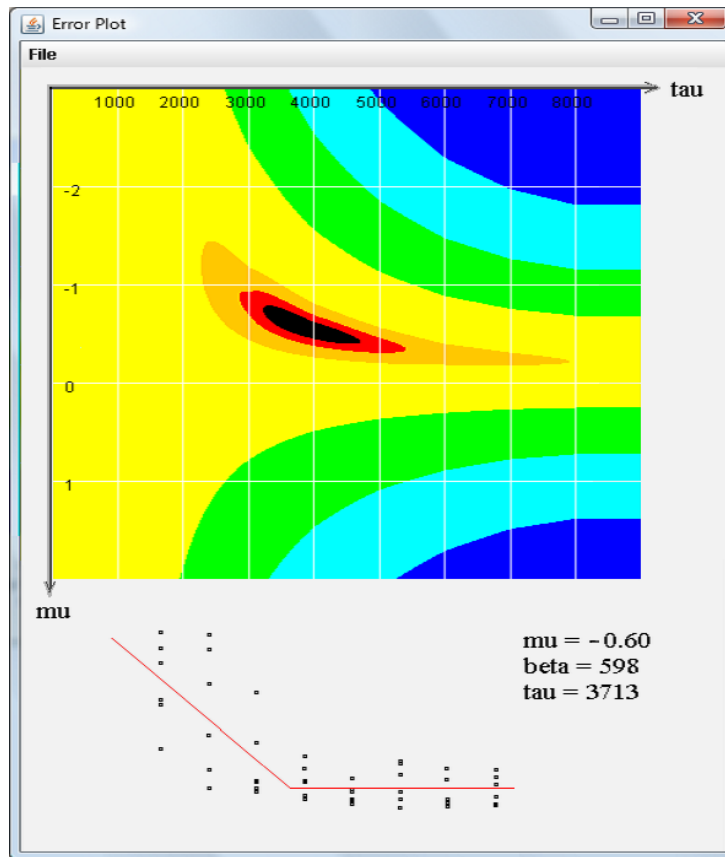
Therefore, the parameters are estimated as:

$$\begin{aligned}
\beta &= (s_n - s_m)/(n - m), \\
\mu &= (mv_m^{rt} - u_m s_m)/(mv_m^{tt} - u_m^2), \\
\mu\tau &= ((ns_m - ms_n)/(n - m) + \mu u_m)/m.
\end{aligned}$$

## B.4 The elbow fitting program

An elbow data fitting program was created to compute the  $\mu$ ,  $\tau$ , and  $\beta$  values of a response time sequence. The program displays a plot of the response time sequence, the best fitting elbow-curve, values of the parameters, and a plot of  $\mu$  and  $\tau$  showing the values of the squared errors, as shown in Figure B.1.

The plot uses colours for encoding the value of the squared errors. The region coloured in black means the squared error has minimal values when  $\mu$  and  $\tau$  reside in it. Thus by detecting the position of the black region we can find out the approximate values for  $\mu$  and  $\tau$ . It allows us to validate the parameter values.



color scale for encoding the value of the squared errors: 
Min
Max
→




Figure B.1: The Elbow Data Fitting Program

# Appendix C

## Subject Evaluation

The following comments were collected from subjects when they finished all the sessions. The subjects were asked if they had any comments about the experiment. They were asked to subjectively evaluate Karnaugh map and textual representation. When subjects refer to the 'table' or 'map', they mean Karnaugh map.

**Subject 1:** Reading textual expression is tired, especially when it is long. I had to search for the matching parenthesis and remember what I've read, which is hard. Using Karnaugh map is much easier.

**Subject 2:** When there are only one or two cells being selected, I can understand the meaning of the Karnaugh map. It is harder when there are more selected cells. But there is an easier method. That is, I can look up the map and check the selection of the cell. The most difficult part of textual expression is parenthesis. Sometimes I have to go back to find where the matching parenthesis starts.

**Subject 3:** I prefer using the maps than the expressions. The embedded parenthesis in expressions makes me dazzling.

**Subject 4:** The tricky thing about the map is that you can find out where the icon should be located. You can answer the questions without understanding what the map means.

**Subject 5:** Karnaugh map is definitely faster. When the expression is long, sometimes I forgot the previous contents and had to read it again. This slows down my response. Using Karnaugh map I can quickly find out what icons are not answers of the query. But textual expressions don't have such property.

**Subject 6:** It is easier using the table. I only need to look up the table and see

if the cell is selected.

**Subject 7:** The map form is easier. With a map, each feature only needs to be checked once. But with an expression, you have to check a feature whenever you read a keyword.

# References

- [1] Christopher Ahlberg and Ben Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings CHI'94*, pages 313–317, 1994.
- [2] Christopher Ahlberg and Staffan Truv. Exploring terra incognita in the design space of query devices. In *Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction*, pages 49–68, 1996.
- [3] Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. Dynamic queries for information exploration: an implementation and evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 619–626. ACM Press, 1992.
- [4] Peter G. Anick, Jeffrey D. Brennan, Rex A. Flynn, David R. Hanssen, Bryan Alvey, and Jeffrey M. Robbins. A direct manipulation interface for boolean information retrieval via natural language query. In *SIGIR'90, 13th International Conference on Research and Development in Information Retrieval, Brussels, Belgium, 5-7 September 1990, Proceedings*, pages 135–150, 1990.
- [5] Manoochehr Azmoodeh and Hongbo Du. Gql, a graphical query language for semantic databases. In *Proceedings of the 4th International Working Conference SSDBM on Statistical and Scientific Database Management*, pages 259–277, London, UK, 1989. Springer-Verlag.
- [6] M. J. Bates. The design of browsing and berrypicking techniques for the on-line search interface. *Online Review*, 13(5):407–431, 1989.
- [7] Richard A. Becker and William S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.
- [8] Christopher Van Berendonck and Timothy Jacobs. Bubbleworld: a new visual information retrieval technique. In *Proceedings of the Australian symposium on Information visualisation*, 47-56, 2003.

- [9] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. The University of Wisconsin Press, Madison, Wisconsin, 1967/1983.
- [10] George Boole. *An Investigation of the Laws of Thought*. Dover Publications, Incorporated, 1958.
- [11] Gloria Bordogna and Gabriella Pasi. Modeling vagueness in information retrieval. pages 207–241, 2001.
- [12] Eugene Borokhovski, Norman Segalowitz, and Guy L.Lacroix. Learning by imitation, by reinforcement and by verbal rules in problem solving. *Talk given at the International Conference on Development and Learning (ICDL04)*, 2004.
- [13] Robert Bosch, Chris Stolte, Diane Tang, John Gerth, Mendel Rosenblum, and Pat Hanrahan. Rivet: A flexible environment for computer systems visualization. pages 68–73, 2000.
- [14] D.A. Buell. A general model of query processing in information retrieval system. *Information Processing Management*, 17(5):249–262, 1981.
- [15] S. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. 1999.
- [16] Stuart K. Card, Thomas P. Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, 1980.
- [17] Matthew Chalmers and Paul Chitson. Bead: explorations in information visualization. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 330–337, 1992.
- [18] H. C. Chan, H. J. Lu, and K. K. Wei. A survey of sql language. *Journal of Database Management*, 4(4):4–15, 1993.
- [19] Hock Chan, Keng Siau, and Kwok-Kee Wei. The effect of data model, system and task characteristics on user query performance: an empirical study. *SIGMIS Database*, 29(1):31–49, 1997.
- [20] Hong Chen. Compound brushing explained. *Information Visualization*, 3(2):96–108, 2004.
- [21] Ed H. Chi, P. Barry, J. Riedl, and J. Konstan. A spreadsheet approach to information visualization. In *INFOVIS '97: Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, page 17. IEEE Computer Society, 1997.

- [22] Michael Chui. Pattern, procedurality & pictures: Factors affecting boolean query interface design for the web. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999.
- [23] William C. Cleveland and Marylyn E. McGill. *Dynamic Graphics for Statistics*. CRC Press, Inc., 1988.
- [24] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.
- [25] T. Wang D. Shasha. New techniques for best- match retrieval. In *ACM Transactions on Office Information Systems*, volume 8-2, pages 140–158, 1990.
- [26] Mehdi Dastani. The role of visual perception in data visualization. *Journal of Visual Languages and Computing*, 13(6):601–622, 2002.
- [27] Mark Derthick, James Harrison, Andrew Moore, and Steven F. Roth. Efficient multi-object dynamic query histograms. In *Proceedings of the 1999 IEEE Symposium on Information Visualization*, page 84. IEEE Computer Society, 1999.
- [28] Mark Derthick, John Kolojejchick, and Steven F. Roth. An interactive visual query environment for exploring data. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 189–198, 1997.
- [29] A. Dix and A. Patrick. Query by browsing. In *Proceedings of IDS'94: The 2nd International Workshop on User Interfaces to Databases*, pages 236–248, Lancaster, UK, 1994.
- [30] Juan C. Dursteler. Exploratory search. 2006.
- [31] Stephen G. Eick. Data visualization sliders. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 119–120. ACM Press, 1994.
- [32] Stephen G. Eick, J.L. Steffen, and E.E. Sumner. Seesoft - a tool for visualizing line oriented software statistics. In *IEEE Trans. on Software Engineering*, volume 18, page 11, 1992.
- [33] Amineh Fadhil and Volker Haarslev. Ontovql: A graphical query language for owl ontologies. In *Proceedings of the 2007 International Workshop on Description Logics (DL-2007)*, pages 267–274, 2007.
- [34] Jacob Feldman. Minimization of boolean complexity in human concept learning. *Nature*, 407:630–633, 2000.
- [35] Jacob Feldman. A catalog of boolean concepts. *J. Math. Psychol.*, 47(1):75–89, 2003.

- [36] Ken Fishkin and Maureen C. Stone. Enhanced dynamic queries via movable filters. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–420, 1995.
- [37] E. A. Fox and G. Marchionini. Toward a worldwide digital library. In *Communications of the ACM*, volume 41(4), pages 29–32, 1998.
- [38] T. Green and M. Petre. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages and Computing*, 7(2):131–174, 1996.
- [39] S. Greene, S. Devlin, P. Cannata, and L. Gomez. No ifs ands or ors: A study of database querying. *International Journal of Man-Machine Studies*, 32:303–325, 1990.
- [40] Pat Hanrahan. *Visual Thinking for Business Intelligence - A White Paper of Tableau Software*. 2005.
- [41] Tomoyuki Hansaki, Buntarou Shizuki, Kazuo Misue, and Jiro Tanaka. Findflow: visual interface for information search based on intermediate results. In *APVis '06: Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation*, pages 147–152, 2006.
- [42] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. Angular brushing of extended parallel coordinates. In *Proc. of the IEEE Symposium on Information Visualization*, page 127, 2002.
- [43] C Healey. Perception in visualization.
- [44] C Healey. *Effective Visualization of Large, Multidimensional Datasets*. PhD thesis, Department of Computer Science, University of British Columbia, 1996.
- [45] M. A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *In Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems*, 1995.
- [46] Marti A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, 2006.
- [47] Jeffrey Heer, Maneesh Agrawala, and Wesley Willett. Generalized selection via interactive query relaxation. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 959–968, 2008.



- [48] Matthias Hemmje, Clemens Kunkel, and Alexander Willett. Lyberworlda visualization user interface supporting fulltext retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 249–259. Springer-Verlag New York, Inc., 1994.
- [49] M. Hertzum and E. Frokjaer. Browsing and querying in online documentation: A study of user interfaces and the interaction process. *ACM Transactions on Computer-Human Interaction*, 3(2):136–161, 1996.
- [50] B. Hetzler and N. Miller. Four critical elements for designing information exploration systems. *Information Exploration workshop for ACM SIGCHI*, 1998.
- [51] Harry Hochheiser and Ben Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.
- [52] Bernard C. Y. Tan Hock C. Chan and Kwok kee Wei. Three important determinants of user performance for database retrieval. *Int. J. Hum.-Comput. Stud.*, 51(5):895–918, 1999.
- [53] Heike Hofmann, Arno Siebes, and Adalbert F. X. Wilhelm. Visualizing association rules with interactive mosaic plots. In *In Proc. of the 6th Int'l Conference of SigKDD*, pages 227–235, 2000.
- [54] T. J. Jankun-Kelly, Kwan-Liu Ma, and Michael Gertz. A model and framework for visualization exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):357–369, 2007.
- [55] Steve Jones. Graphical query specification and dynamic result previews for a digital library. In *In Proc. of UIST'98, ACM Symposium on User Interface Software and Technology*, pages 143–151, San Francisco, USA, 1998.
- [56] M. Karnaugh. The map method for synthesis of combinational logic circuits. *Transactions AIEE, Communications and Electronics*, 72:593–599, 1953.
- [57] D. A. Keim and H. Kriegel. Visdb: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, 14:40–49, 1994.
- [58] Robert R. Korfhage. To see, or not to see is that the query? In *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 134–141. ACM Press, 1991.
- [59] J. P. Lee and Georges Grinstein. An architecture for retaining and analyzing visual explorations of databases. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 101, 1995.

- [60] John Peter Lee. *A systems and process model for data exploration*. PhD thesis, University of Massachusetts Lowell, 1998.
- [61] Joon Ho Lee, Won Yong Kin, Myoung Ho Kim, and Yoon Joon Lee. On the evaluation of boolean operators in the extended boolean retrieval framework. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–297, 1993.
- [62] Jonathan Levy, Harold Pashler, and Erwin Boer. Central interference in driving: Is there any stopping the psychological refractory period? *Psychological Science*, 17(3):228–235, 2006.
- [63] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger. Devise: integrated querying and visual exploration of large datasets. *SIGMOD Rec.*, 26(2):301–312, 1997.
- [64] A. MacEachern. *How Maps Work: Representation, Visualization, and Design*. The Guilford Press, 1995.
- [65] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5(2):110–141, 1986.
- [66] Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, 2006.
- [67] Allen R. Martin and Matthew O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 271, 1995.
- [68] Wolfgang Martin. Interactive visual analytics - put your smart brains in the driver's seat. 2007.
- [69] Timothy P. McNamara. *Semantic priming : perspectives from memory and word recognition*. Psychology Press, 2005.
- [70] A. Michard. Graphical presentation of boolean expressions in a database query language: Design notes and an ergonomic evaluation. *Behaviour and Information Technology*, 1,3:279–288, 1982.
- [71] J.L. Morrison. A theoretical framework for cartographic generalization with the emphasis on the process of symbolization. *International Yearbook of Cartography*, 14:115–127, 1974.

- [72] Norman Murray, Norman Paton, and Carole Goble. Kaleidoquery: a visual query language for object databases. In *Proceedings of the working conference on Advanced visual interfaces*, pages 247–257. ACM Press, 1998.
- [73] Chris North and Ben Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 128–135, 2000.
- [74] R.M. Nosofsky, T.J. Palmeri, and S.C. McKinley. Rule-plus-exception model of classification learning. *Psychological Review*, 101:53–79, 1994.
- [75] Vicki L. O’Day and Robin Jeffries. Information artisans: patterns of result sharing by information searchers. In *Proceedings of the conference on Organizational computing systems*, pages 98–107. ACM Press, 1993.
- [76] A. Papantonakis and P. J. H. King. Syntax and semantics of gql, a graphical query language. *Journal of Visual Languages and Computing*, 6:3–25, 1995.
- [77] Phyllis Reisner. Human factors studies of database query languages: A survey and assessment. *ACM Computing Surveys*, 13(1):13–31, 1981.
- [78] Jonathan C. Roberts and Michael A. E. Wright. Towards ubiquitous brushing for information visualization. In *IV '06: Proceedings of the conference on Information Visualization*, pages 151–156, 2006.
- [79] S. Roth, M. Chuah, S. Kerpedjiev, J. Kolojejchick, and P. Lucas. Towards an information visualization workspace: Combining multiple means of expression. *Human-Computer Interaction Journal*, 12, 131185.
- [80] S. F. Roth, P. Lucas, J. A. Senn, C. C. Gomberg, M. B. Burks, P. J. Stroffolino, A. J. Kolojejchick, and C. Dunmire. Visage: a user interface environment for exploring information. In *INFOVIS '96: Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, page 3. IEEE Computer Society, 1996.
- [81] L.Tweedie R.Spence. The attribute explorer: information synthesis via exploration. *Interacting with Computers*, 11:137–146, 1998.
- [82] Gerard Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036, 1983.
- [83] R. Shepard, C. L. Hovland, and H. M. Jenkins. Learning and memorization of classifications. *Psychological Monographs: General and Applied*, 75(13):1–42, 1961.
- [84] Ben Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11:70–77, 1994.

- [85] Ren Sieber, Christoph Schmid, and Samuel Wiesmann. Smart legend - smart atlas! In *Proc. of the 22th Int. Conference of the ICA*, 2005.
- [86] C. Silverstein, M.Henzinger, H.Marais, , and M.Moricz. Analysis of a very large altavista query log. *Technical note 1998-014, Digital SRC*, 1998.
- [87] Robert Spence. Sensitivity encoding to support information space navigation: a design guideline. *Information Visualization*, 1(2):120–129, 2002.
- [88] Anselm Spoerri. *InfoCrystal: A Visual Tool For Information Retrieval*. PhD thesis, MIT, 1995.
- [89] Chris Stolte. *Query, Analysis, and Visualization of Multidimensional Databases*. PhD thesis, Stanford University, 2003.
- [90] Chris Stolte, Diane Tang, and Pat Hanrahan. Query, analysis, and visualization of hierarchically structured data using polaris. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 112–122. ACM Press, 2002.
- [91] Diane Tang, Chris Stolte, and Robert Bosch. Design choices when architecting visualizations. volume 3(2), pages 65–79, 2004.
- [92] Heikki Topi, Joseph S. Valacich, and Jeffrey A. Hoffer. The effects of task complexity and time availability limitations on human performance in database query tasks. *Int. J. Hum.-Comput. Stud.*, 62(3):349–379, 2005.
- [93] M. Eduard Tudoreanu and Delbert Hart. Interactive legends. In *ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference*, pages 448–453. ACM Press, 2004.
- [94] Lisa Tweedie, Bob Spence, David Williams, and Ravinder Bhogal. The attribute explorer. In *Conference companion on Human factors in computing systems*, pages 435–436. ACM Press, 1994.
- [95] Paul Velleman. *Learning Data Analysis with Data Desk*. Addison Wesley, 1993.
- [96] Matthew O. Ward. Xmdvtool: integrating multiple methods for visualizing multivariate data. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 326–333, 1994.
- [97] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, San Francisco, CA, 2000.
- [98] L. Wilkinson. *The Grammar of Graphics*. Springer, New York, 1999.

- [99] Leland Wilkinson. *The grammar of graphics*. Springer-Verlag New York, Inc., New York, NY, USA, 2005.
- [100] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. In *Proceedings of the 2007 IEEE Symposium on Information Visualization (INFOVIS '07)*. IEEE Computer Society, 2007.
- [101] G. J. Wills. Selection: 524,288 ways to say "this is interesting". In *Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, page 54. IEEE Computer Society, 1996.
- [102] K. Wilson. Evaluating information exploration interfaces. In *Workshop on Innovation in Information Exploration Environments, CHI'98 Conference on Human Factors in Computing Systems.*, 1998.
- [103] Allison Woodruff, Chris Olston, Alexander Aiken, Michael Chu, Vuk Ercegovic, Mark Lin, Mybrid Spalding, and Michael Stonebraker. Datasplash: A direct manipulation environment for programming semantic zoom visualizations of tabular data. *Journal of Visual Languages and Computing*, 12:551 – 571, 2001.
- [104] Michael A.E. Wright and Jonathan C. Roberts. Click and brush: A novel way of finding correlations and relationships in visualizations. In *Proceedings Theory and Practice of Computer Graphics*, pages 179–186, 2005.
- [105] R.R. Yager. A note on weighted queries in information retrieval systems. *Journal of the American Society for Information Science*, 38:23–24, 1987.
- [106] Degi Young and Ben Shneiderman. A graphical filter/flow model for boolean queries: An implementation and experiment. *Journal of the American Society for Information Science*, 44(6):327–339, 1993.
- [107] M.M. Zloof. Query-by-example: A database language. *IBM Systems J.*, 21(3):324–343, 1977.