Probabilistic Databases and Their Applications

Wenzhong Zhao
University of Kentucky, wzha0@cs.uky.edu

Recommended Citation
https://uknowledge.uky.edu/gradschool_diss/326

This Dissertation is brought to you for free and open access by the Graduate School at UKnowledge. It has been accepted for inclusion in University of Kentucky Doctoral Dissertations by an authorized administrator of UKnowledge. For more information, please contact UKnowledge@lsv.uky.edu.
ABSTRACT OF DISSERTATION

Wenzhong Zhao

The Graduate School
University of Kentucky
2004
Probabilistic Databases and Their Applications

ABSTRACT OF DISSERTATION

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the College of Engineering at the University of Kentucky

By

Wenzhong Zhao

Lexington, Kentucky

Co-Directors: Dr. Alexander Dekhtyar, Assistant Professor of Computer Science and Dr. Judy Goldsmith, Associate Professor of Computer Science

Lexington, Kentucky

2004
ABSTRACT OF DISSERTATION

Probabilistic Databases and Their Applications

Probabilistic reasoning in databases has been an active area of research during the last two decades. However, the previously proposed database approaches, including the probabilistic relational approach and the probabilistic object approach, are not good fits for storing and managing diverse probability distributions along with their auxiliary information.

The work in this dissertation extends significantly the initial semistructured probabilistic database framework proposed by Dekhtyar, Goldsmith and Hawkes in [20]. We extend the formal Semistructured Probabilistic Object (SPO) data model of [20]. Accordingly, we also extend the Semistructured Probabilistic Algebra (SP-algebra), the query algebra proposed for the SPO model.

Based on the extended framework, we have designed and implemented a Semistructured Probabilistic Database Management System (SPDBMS) on top of a relational DBMS. The SPDBMS is flexible enough to meet the need of storing and manipulating diverse probability distributions along with their associated information. Its query language supports standard database queries as well as queries specific to probabilities, such as conditionalization and marginalization. Currently the SPDBMS serves as a storage backbone for the project Decision Making and Planning under Uncertainty with Constraints \(^1\), that involves managing large quantities of probabilistic information. We also report our experimental results evaluating the performance of the SPDBMS.

We describe an extension of the SPO model for handling interval probability distributions. The Extended Semistructured Probabilistic Object (ESPO) framework improves the flexibility of the original semistructured data model in two important features: (i) support for interval probabilities and (ii) association of context and conditionals with individual random variables. An extended SPO

---

\(^1\)This project is partially supported by the National Science Foundation under Grant No. ITR-0325063.
(ESPO) data model has been developed, and an extended query algebra for ESPO has also been introduced to manipulate probability distributions for probability intervals.

The Bayesian Network Development Suite (BaNDeS), a system which builds Bayesian networks with full data management support of the SPDBMS, has been described. It allows experts with particular expertise to work only on specific subsystems during the Bayesian network construction process independently and asynchronously while updating the model in real-time.

There are three major foci of our ongoing and future work: (1) implementation of a query optimizer and performance evaluation of query optimization, (2) extension of the SPDBMS to handle interval probability distributions, and (3) incorporation of machine learning techniques into the BaNDeS.

KEYWORDS: Probabilistic Databases, Uncertainty Modeling, Probability Intervals, Semistructured Data, Query Algebra

Wenzhong Zhao

June 22, 2004
Probabilistic Databases and Their Applications

By

Wenzhong Zhao

Dr. Alexander Dekhtyar
(Co-Director of Dissertation)

Dr. Judy Goldsmith
(Co-Director of Dissertation)

Dr. Grzegorz W. Wasilkowski
(Director of Graduate Studies)

June 22, 2004
RULES FOR THE USE OF DISSERTATIONS

Unpublished dissertations submitted for the Doctor’s degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the dissertation in whole or in part requires also the consent of the Dean of The Graduate School of the University of Kentucky.

A library which borrows this dissertation for use by its patrons is expected to secure the signature of each user.

Name

Date
Probabilistic Databases and Their Applications

DISSERTATION

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the College of Engineering at the University of Kentucky

By

Wenzhong Zhao
Lexington, Kentucky

Co-Directors: Dr. Alexander Dekhtyar, Assistant Professor of Computer Science and: Dr. Judy Goldsmith, Associate Professor of Computer Science
Lexington, Kentucky
2004
Acknowledgments

There are so many friends and colleagues, deserved to be acknowledged, that I could not include them all in this short acknowledgement. I would like to thank everyone who has given support and advice during the writing of this dissertation. Special thanks are due to my advisors, Dr. Alexander Dekhtyar and Dr. Judy Goldsmith, for their guidance and insight throughout the research. Their encouragement, support and friendship have been the most critical factor in my enjoyable stay at the University of Kentucky. I cannot express the positive influence they have had on my education and career. Thanks go to the other committee members, Dr. Jane Hayes, Dr. Arnold Stromberg, and Dr. Anita Lee-Post, for their suggestions and comments.

I wish to acknowledge my gratitude to the fellow students in our research group, Erik Jessup, Jiangyu Li, Derek Williams, Krol Kevin Mathias, and Ionut Emil Iacob, for their time, effort, cooperation, and friendship. In addition, I wish to offer special thanks to a good friend, Lengning Liu, for his technical assistance.

Lastly, I would like to express my sincere appreciation to my family. My wife, Huifang Qin, has given me selfless support during the completion of this dissertation. Without her, I would never have had the strength or desire to finish this dissertation. Together we have accomplished the impossible. Thank you for your love and friendship. To my lovely son, Mengzhe, I thank him for his understanding my frequent absences. I would also like to thank my parents for their unconditional love and understanding when we moved so far from home. Their support has been invaluable to my success.
# Table of Contents

Acknowledgments ................................................................. iii
List of Tables ................................................................. vi
List of Figures ................................................................. vii
List of Files ................................................................. x

Chapter 1 Introduction: Probabilistic Reasoning in Databases 1
  1.1 Motivating Examples .................................................. 3
    1.1.1 Auto Insurance Risk Analysis .................................. 3
    1.1.2 Academic Advising ............................................. 5
    1.1.3 Welfare to Work ............................................... 6
    1.1.4 Political Elections ............................................. 7
  1.2 General Framework .................................................... 9
  1.3 Towards Semistructured Probabilistic Databases ................. 11
  1.4 Overview of Dissertation .......................................... 13

Chapter 2 Background ...................................................... 15
  2.1 Probability Distributions .......................................... 15
  2.2 Interval Probabilities .............................................. 17
  2.3 Bayesian Networks ................................................. 24
  2.4 Query Processing in RDBMS and in Mediators .................... 26

Chapter 3 Semistructured Probabilistic Database Framework 30
  3.1 Overview of the SPO framework ................................... 30
  3.2 Semistructured Probabilistic Object (SPO) Data Model .......... 31
  3.3 Semistructured Probabilistic Algebra ............................ 35
    3.3.1 Set Operations ............................................... 36
    3.3.2 Selection .................................................... 37
    3.3.3 Projection .................................................. 43
    3.3.4 Conditionalization ......................................... 45
    3.3.5 Cartesian Product ......................................... 47
    3.3.6 Join ....................................................... 48
  3.4 Semantics of SP-Algebra Operations ................................ 51
  3.5 Query Optimization ................................................. 56
    3.5.1 Rewrite rules ................................................. 56
    3.5.2 Cost Model .................................................. 71

Chapter 4 Semistructured Probabilistic DBMS 75
  4.1 Representation of SPOs .............................................. 76
  4.2 Architecture of SPDBMS ............................................ 78
  4.3 Data Storage for SPOs ............................................. 78
    4.3.1 Mapping SPOs to relational tables ......................... 79
    4.3.2 Algorithms for Populating SP-Relations .................... 80
  4.4 Querying the SPDBMS ............................................... 81
    4.4.1 Query Language (SPO-QL) .................................. 81
List of Tables

Table 1.1 Representation of domain letter for random variables. . . . . . . . . . . . . . . 5
Table 4.1 File size and number of tuples in Oracle database for typical data sets. . . . . 90
Table 4.2 Time distribution between Oracle and postprocessing for data set of 10,000
    SPOs with 3 participating variables and domain size of 2. . . . . . . . . . . . . 90
Table 5.1 Selection queries to ESPOs. . . . . . . . . . . . . . . . . . . . . . . . . . . . 103
Table 5.2 Different types of conditions for selection queries. . . . . . . . . . . . . . . 103
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Typical types of probabilistic information for risk analysis applications</td>
<td>4</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Sunny Hill pre-election polls</td>
<td>8</td>
</tr>
<tr>
<td>Figure 1.3</td>
<td>A general framework for decision making and planning under uncertainty.</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Marginalization and conditioning operations</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Tightness of interval probability distributions</td>
<td>20</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>A methodology for performing conditioning (or marginalization) operation on an interval probability distribution</td>
<td>24</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>A sample Bayesian network for Welfare to Work application domain</td>
<td>26</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>A typical CPT for random variable Communication Skills with different literacy levels</td>
<td>26</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Major components of query processor in relational DBMS and in mediator.</td>
<td>27</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Joint Probability Distribution of risk on Driver’s Age and License Years for Asian drivers in Lexington</td>
<td>34</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Selection on Probabilistic Table and on Probability values in SP-Algebra</td>
<td>40</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Projection on Probabilistic Table and on Probability values in SP-Algebra</td>
<td>45</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Conditionalization in SP-Algebra</td>
<td>47</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Join operations in SP-Algebra</td>
<td>50</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Selection on Probabilistic Table and on Probability values in SP-Algebra</td>
<td>52</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>XML DTD template for SPOs.</td>
<td>77</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>A typical SPO object for risk level on Driver Age and License Years, and its XML representation</td>
<td>78</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>The overall architecture of SPDBMS.</td>
<td>79</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>A set of relational schema for storing SPOs</td>
<td>80</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Pseudo-code for algorithm CREATE.</td>
<td>81</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Pseudo-code for algorithm INSERT.</td>
<td>82</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Internal representation for the SPO defined in Figure 3.1</td>
<td>83</td>
</tr>
<tr>
<td>Figure 4.8</td>
<td>The grammar of the query language for SPDBMS.</td>
<td>84</td>
</tr>
<tr>
<td>Figure 4.9</td>
<td>The query parse tree for the sample SP-Algebra query.</td>
<td>85</td>
</tr>
<tr>
<td>Figure 4.10</td>
<td>The SPDBMS query engine.</td>
<td>86</td>
</tr>
<tr>
<td>Figure 4.11</td>
<td>Steps to evaluate the selection query $\sigma_{p \cdot P &gt; 0.1}(S)$</td>
<td>87</td>
</tr>
<tr>
<td>Figure 4.12</td>
<td>Steps to evaluate the conditionalization query $\mu_{\text{cond.}DA='A'}(S)$</td>
<td>88</td>
</tr>
<tr>
<td>Figure 4.13</td>
<td>Postprocessing algorithm for the conditionalization query</td>
<td>88</td>
</tr>
<tr>
<td>Figure 4.14</td>
<td>An internal object model for SP-relation.</td>
<td>89</td>
</tr>
<tr>
<td>Figure 4.15</td>
<td>Effect of number of SPOs in an SP-relation</td>
<td>91</td>
</tr>
<tr>
<td>Figure 4.16</td>
<td>Effect of domain size of participating random variables.</td>
<td>92</td>
</tr>
<tr>
<td>Figure 4.17</td>
<td>Effect of number of variables on conditionalization operation</td>
<td>93</td>
</tr>
<tr>
<td>Figure 4.18</td>
<td>Effect of number of SPOs and selectivity on selection query on probability.</td>
<td>93</td>
</tr>
<tr>
<td>Figure 4.19</td>
<td>Effect of number of SPOs and selectivity on Cartesian product.</td>
<td>93</td>
</tr>
<tr>
<td>Figure 4.20</td>
<td>Effect of number of SPOs and selectivity on join operation.</td>
<td>93</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Simple vs. Extended context and conditionals in SPOs</td>
<td>97</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Extended Semistructured Probabilistic Object</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>Sunny Hill pre-election polls in ESPO format</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>Selection on probability table and probabilities.</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>Probabilistic selection on ( \exists P ) and ( \forall P ) conditions.</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>Projections on context.</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>Projections on conditionals.</td>
<td></td>
</tr>
<tr>
<td>5.8</td>
<td>Projection on the participating random variables.</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>Conditionalization operation.</td>
<td></td>
</tr>
<tr>
<td>5.10</td>
<td>Join operation (left join) in ESP-Algebra</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>The overall system architecture.</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>A typical eCPT for the random variable Communication_Skills, and its representation as a collection of SPOs.</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>Screenshot of the Bayesian network builder.</td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>A sample XML file which represents the W2W Bayesian network topology.</td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>Screenshot of the current probability elicitation toolkit.</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>A sample XML file communication_skills.xml which represents the CPT for variable Communication_Skills as SPOs in the W2W Bayesian network.</td>
<td></td>
</tr>
<tr>
<td>6.7</td>
<td>Add or remove a node.</td>
<td></td>
</tr>
<tr>
<td>6.8</td>
<td>Add or remove a dependency.</td>
<td></td>
</tr>
<tr>
<td>6.9</td>
<td>Algorithms for maintaining Bayesian network consistency</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>An example probabilistic relation.</td>
<td></td>
</tr>
<tr>
<td>7.2</td>
<td>An example of STOCHASTIC operator.</td>
<td></td>
</tr>
<tr>
<td>7.3</td>
<td>An example probabilistic relation.</td>
<td></td>
</tr>
<tr>
<td>7.4</td>
<td>Possible worlds for CS major.</td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td>A class hierarchy for classifying undergraduate students.</td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>Definitions for the two special elements Dist and Val.</td>
<td></td>
</tr>
<tr>
<td>7.7</td>
<td>An example of transforming an SPO into a ProTDB instance.</td>
<td></td>
</tr>
<tr>
<td>7.8</td>
<td>An example of transforming an ESPO into a PIXml instance.</td>
<td></td>
</tr>
<tr>
<td>7.9</td>
<td>An example of transforming “unit of information” of a PIXml into an ESPO.</td>
<td></td>
</tr>
<tr>
<td>7.10</td>
<td>Lower bounds do not uniquely define upper bounds for tight interval probability distribution functions.</td>
<td></td>
</tr>
<tr>
<td>7.11</td>
<td>A sample SPO, with ids added for non-leaf nodes.</td>
<td></td>
</tr>
<tr>
<td>7.12</td>
<td>A sample edge table.</td>
<td></td>
</tr>
<tr>
<td>7.13</td>
<td>Sample attribute tables, without those single-row tables for elements ( \langle \text{spo} \rangle ), ( \langle \text{path} \rangle ), ( \langle \text{context} \rangle ), ( \langle \text{variable} \rangle ), ( \langle \text{table} \rangle ), and ( \langle \text{conditional} \rangle ).</td>
<td></td>
</tr>
<tr>
<td>7.14</td>
<td>The SQL statement for finding all SPO SIDs in edge approach.</td>
<td></td>
</tr>
<tr>
<td>7.15</td>
<td>The SQL statement for finding all SPO SIDs in attribute approach.</td>
<td></td>
</tr>
<tr>
<td>A.1</td>
<td>Steps for selection on context, i.e., ( \sigma_{\text{cnt-major}=CS}(S) ).</td>
<td></td>
</tr>
<tr>
<td>A.2</td>
<td>Steps for selection on conditional, i.e., ( \sigma_{\text{cnt-Al}=B}(S) ).</td>
<td></td>
</tr>
<tr>
<td>A.3</td>
<td>Steps for selection on participating variable, i.e., ( \sigma_{\text{var-DBE}V}(S) ).</td>
<td></td>
</tr>
<tr>
<td>A.4</td>
<td>Steps for selection on probability table, i.e., ( \sigma_{\text{tbl-DB}=A}(S) ).</td>
<td></td>
</tr>
<tr>
<td>A.5</td>
<td>Steps for selection on context, i.e., ( \sigma_{\text{pro-P}&gt;0.1}(S) ).</td>
<td></td>
</tr>
<tr>
<td>A.6</td>
<td>Steps for projection on context, i.e., ( \pi_{\text{cnt-major}}(S) ).</td>
<td></td>
</tr>
<tr>
<td>A.7</td>
<td>Steps for projection on conditional, i.e., ( \pi_{\text{cnt-Al}}(S) ).</td>
<td></td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Figure A.8</td>
<td>Steps for projection on participating variable, i.e., $\pi_{var\cdot DB}(S)$</td>
<td>169</td>
</tr>
<tr>
<td>Figure A.9</td>
<td>Steps for selection on context, i.e., $\mu_{bl\cdot DB=\cdot A}(S)$</td>
<td>170</td>
</tr>
<tr>
<td>Figure B.1</td>
<td>XML schema template for SPOs.</td>
<td>171</td>
</tr>
<tr>
<td>Figure B.2</td>
<td>XML schema template for SPOs (cont.)</td>
<td>172</td>
</tr>
</tbody>
</table>
List of Files

wzhao0.pdf .pdf 1.21MB
Chapter 1

Introduction: Probabilistic Reasoning in Databases

Uncertain information occurs in many vital applications, such as multimedia databases for storing the results of image recognition, logistics databases, stock market prediction software, and applications of Bayesian networks [70]. Different techniques have been proposed to represent and handle uncertain information. In the context of databases, however, research has focused on two major approaches when dealing with uncertainty. One uses Zadeh’s fuzzy sets and possibility theory [86] to build framework of fuzzy databases, and the other adopts probability theory to establish framework of probabilistic databases. The former has been studied extensively in the last three decades. It extends the classical relational model to represent and manipulate uncertain information with imprecise data values. The fuzzy set theory and fuzzy logic [86] provide a foundation to handle data with such extended values. In this dissertation we will focus on the latter: probabilistic databases. We review related approaches in Section 7.1.

Storing and managing probabilistic information has been an active area of research in the last two decades. There have been relational [5, 14, 25, 58] and object [28, 56] data models proposed to support storage and querying of probabilistic information. Unfortunately, these approaches are not sufficiently flexible to handle different contexts in which probabilities must be dealt with in analyzing a stochastic system. For instance, consider auto insurance risk analysis, where the risk level of possible financial loss when offering a driver with an insurance policy may be represented in a variety of forms: a simple probability distribution for one aspect or a joint probability distribution for several aspects, or a simple or joint conditional probability distribution (risk level may depend on earlier driving record).

Information with different formats would require separate storage in any of the current probabilistic relational models, making even simple queries, if not impossible, hard to express. For example, when one asks a query “Find all probability distributions that involve the aspect Driver's Age”, the system has to query on all the relations that have Driver's Age as a field. Note that this may require users to know in advance the names of tables that have this field and may result in thousands of separate queries, with each for one table. Thus, we propose a semistructured probabilistic data model to alleviate this problem.

In this work, we propose an innovative approach to managing probabilistic information by treating the probability tables as the primary objects in the database. In order to make such data usable, we store significant auxiliary information along with the probability tables. The initial
Semistructured Probabilistic Object (SPO) data model and algebra were introduced by Dekhtyar, Goldsmith and Hawkes [20]. We use this initial framework as a starting point and continue developing new features for it [89, 88, 44, 87]. Our databases are flexible enough to handle diverse “shapes” of probability distributions over arbitrary numbers of discrete random variables. Our query algebra allows the user to retrieve data based on probabilities, variables, or associated information, and to transform the probability distributions according to the laws of probability theory. The major contributions of this dissertation are:

- We extended and revised the initial SPO data model and algebra of Dekhtyar, Goldsmith and Hawkes [20]. A number of features, key to the SPO framework, were modified in the current model. We have added a path expression in the SPO data model to specify the origin of a particular object, and subsequently revise the algebra as well to reflect this change. A new join operation, which complies with the laws of probability theory, has been defined to replace the one in the initial framework. We developed a set of equivalence relations and cost models for query optimization framework. In the meantime, we also proved all the theorems in the framework.

- Based on the new SPO framework, we designed and implemented a Semistructured Probabilistic Database Management System (SPDBMS) on top of a relational DBMS. We developed the XML-based implementation of the SPO data model with a meta markup language SPO-ML. A relational schema generation strategy was designed for efficiently mapping XML representations of SPOs into relational tables. A query translation mechanism which automatically generates a sequence of SQL queries for evaluating SP-Algebra queries has been developed. Extensive experiments for evaluating our database implementation were conducted.

- We further extended the standard SPO model to incorporate two important features: (i) support for interval probabilities and (ii) association of context and conditionals with individual random variables. An extended SPO (ESPO) data model has been developed, and an extended query algebra for ESPO has also been introduced to manipulate probability distributions for probability intervals. We also added proofs for several algebra operations.

- We designed and implemented a Bayesian Network Development Suite (BaNDeS) for building Bayesian networks with the full support of the SPDBMS. We split the entire Bayesian network building system into subsystems, each of which performs a specific subtask. The Bayesian network building process becomes distributed and asynchronous. This allows experts with particular expertise to work only on specific subsystems during the model construction process independently and asynchronously while updating the model in real-time.
The SPO framework described above can be applied to any applications where there are large quantities of probabilistic information. In the following sections, we give a couple of examples where the database framework can be used.

1.1 Motivating Examples

In this section we describe four motivating examples, which we will use in explaining our database frameworks and the Bayesian Network Development Suite (BaNDeS).

1.1.1 Auto Insurance Risk Analysis

In order to get car insurance, one must first fill out a complex form, giving information on driving history, insurance history, and a variety of personal matters. Based on this data, the insurer sets a policy premium for the available policies.

It is a goal for every insurance company to prevent major financial losses and maximize annual profits. As described by Russell and Norvig [74], a 1% improvement of risk assessment brings over a billion dollars annually for a typical insurance company. One way to lure customers is to lower prices. Most insurance companies try to set the insurance premium for each insurance policy holder as low as possible without giving up their profit. As a result, they attract more customers.

How can an insurer increase the likelihood of a reasonable profit? Insurance companies could try to improve their risk assessment analysis, for instance, by constructing a Bayesian network [83] which allows them to decide what the financial risk level is for each policy holder.

Statistical information about the association between financial risk and driver personal information, driving skills and vehicle information can be obtained from a database of previous claims maintained by the company. We may assume that when this information correctly reflects or approximates the true probabilities, it can help on providing better estimates for policy premiums. However, the statistical information needs to be updated periodically so that it accurately reflects the current probabilities.

Consider a database designed to assist insurance companies with the risk assessment process. Note that various types of probabilistic information may be available to an insurance company and that information needs to be stored in the database. The simplest type of probabilistic information is a probability distribution of financial risk for one aspect. The company may need the probability distribution of risk for Driver's Age or the probability distribution over different approximate values for risk given the number of years the driver has had a license.

Other types of probabilistic information, for instance joint probability distributions, may also
be useful. One might want to know the risk level of a customer who has college degree and a brand new passenger car. In this case, the company needs the probability distribution of risk for both Education Level and Vehicle Year. In other situations, the risk level can also depend on her/his past Driving Record or other aspects observed. A Driving Record value of “medium accident” may suggest to the company that the policy holder might belong to the group of higher risk level, while a Safety Equipment value of “yes” might suggest that the policy holder might belong to the lower risk level group. Other information that can affect the probability distribution may include: where the policy holder lives, such as city, state, rural/urban; the policy holder’s background such as race, employment type; vehicle information such as personal/business vehicle, etc.

The typical types of probabilistic information to be stored in a database for risk assessment support are shown in Figure 1.1. Note that the domains of the variables are finite, as shown in Table 1.1. Probability distribution $D_1$ represents a simple probability distribution over random variable Driver’s Age (DA). The second one represents a joint probability distribution over variables Driver’s Age (DA) and License Years (LY). $D_3$ is a joint probability distribution with additional auxiliary information, i.e., the driver’s job type and location. The last object represents a joint probability distribution for those drivers whose had a medium to severe accident within the last 3 years.

![Figure 1.1: Typical types of probabilistic information for risk analysis applications](image)

With all this information in hand, insurance agents may ask themselves a variety of different questions when issuing an insurance policy to a driver, including:
Table 1.1: Representation of domain letter for random variables.

<table>
<thead>
<tr>
<th>Domain Letter</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver’s Age (DA)</td>
<td>&lt;20</td>
<td>21-35</td>
<td>36-55</td>
<td>&gt;56</td>
</tr>
<tr>
<td>Education Level (EL)</td>
<td>high school</td>
<td>college</td>
<td>advanced degree</td>
<td>none of above</td>
</tr>
<tr>
<td>Driver Gender (DG)</td>
<td>male</td>
<td>female</td>
<td>married</td>
<td></td>
</tr>
<tr>
<td>Marital Status (MS)</td>
<td>single</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>License Years (LY)</td>
<td>&lt;1</td>
<td>1-3</td>
<td>3-10</td>
<td>&gt;10</td>
</tr>
<tr>
<td>Driving Record (DR)</td>
<td>severe</td>
<td>medium</td>
<td>minor</td>
<td>none</td>
</tr>
<tr>
<td>Vehicle Type (VT)</td>
<td>heavy truck</td>
<td>light truck</td>
<td>passenger car</td>
<td>motorcycle</td>
</tr>
<tr>
<td>Vehicle Make (VM)</td>
<td>Toyota</td>
<td>BMW</td>
<td>Ford</td>
<td>GM</td>
</tr>
<tr>
<td>Vehicle Age (VA)</td>
<td>&lt;1</td>
<td>1-5</td>
<td>5-10</td>
<td>&gt;10</td>
</tr>
<tr>
<td>Safety Equipment (SE)</td>
<td>yes</td>
<td>no</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- “What would the likelihood of risk level be if we issue an insurance policy to an Asian driver?”
- “What probability of financial loss will be if we issue an insurance policy to a young driver?”
- “How likely will drivers who had severe accidents cause new financial loss?”
- “What information is available about the risk of granting insurance policy to a driver with less than one year of driving experience?”
- “What outcomes have probability over 0.1?”

While finding answers to those questions from the data is a matter of statistical analyses, once the results are obtained, they need to be stored somewhere for reuse. These results may also be integrated with results from other analyses. This particular part of efficient management of diverse probability distributions is what this dissertation aimed to. We propose a database framework specially designed for storing and managing large quantities of probabilistic information.

1.1.2 Academic Advising

This example comes from our work on modeling academic advising as a process of uncertain inference [21]. Consider a database, designed to assist faculty members with the academic advising process. Typically, an advisor sees each advisee once every semester to suggest the set of courses to take next semester. While individual advising styles of the faculty differ, many advisors try to suggest courses that fulfill degree requirements and for which the student has the highest chance (probability) of success.
Statistical information about student performance in different classes can be extracted from the student transcript database maintained by every university. This information, e.g., 33% of students with an A in Data Structures got an A in Databases, captures historical trends and can be used to predict future performance of students. Under the assumption that this information correctly reflects (approximates) the true probabilities, it can be used to assist the advisor in her recommendation.

Note that the type of probabilistic information available to the advisor in this example varies greatly. A simple form is a probability distribution of student performance in one course. The advisor can compare the probability distribution for Databases with the probability distribution for Operating Systems in order to choose the course that has a higher probability of success.

Another type of probabilistic information that can be useful in this situation is a joint probability distribution. When the advisor needs to consider the entire list of courses for the student to take next semester, she is interested in the student’s success in all courses at once. This brings up another type of probabilistic information that can be useful, a joint probability distribution.

Furthermore, we notice that student’s success in the future classes can depend intrinsically on her current grades. A C in a Data Structures class may suggest to the advisor that the student might not do well in Algorithms, while and A in Logic suggests a good chance of success in Artificial Intelligence. Other possible information that can affect the probability distribution may include some general information about the student/course background such as student’s major, college, graduate/undergraduate status, professor who teaches the course, etc.

### 1.1.3 Welfare to Work

Consider decision-theoretic planning in a federally mandated, locally administered program such as Welfare to Work (W2W). W2W is a program of services offered to US welfare recipients to help them re-enter the work force. The broad goals are federally mandated; the mandates are interpreted and refined by state governments, and services available to the W2W clients depend on local service providers. There is a high turn-over both at the level of individual service providers such as case workers, and at the level of agencies engaged to provide services. To enter the W2W programs, clients must go to a specific office and register for the program. Case managers then determine the client’s eligibility through a series of assessments, such as economic need, family structure, etc. Regulations change frequently at all levels of the regulatory hierarchy.

W2W case workers in the locality we are studying often have case loads of 40 to 50 clients. Based on their knowledge of the individual client’s preferences and goals, on the current regulations, and on the currently available services, they must recommend appropriate choices of services for each client. Case workers meet their clients in a weekly basis to assess the level of work, progress
in learning, and attitude towards employment, review and revise the actions and supports which are contracted between case managers and their clients. The W2W program is analogous to the Academic Advising project in that there is, an advisor who tries to find the best plans for either their W2W clients or their students. The case managers in W2W have the similar role of the academic advisors in the Academic Advising project, while W2W clients have the similar role of the students. When case managers develop goals and plans for the clients, they must balance state and federal requirements that imposed on the employment activities. A lot of the work case managers do with the W2W clients is based on the case managers’ assessment of the likelihood that the combined activities suggested for a client will lead to improved employability of the client (or other success measures for the program and/or the particular client).

### 1.1.4 Political Elections

Imagine the town of Sunny Hill is holding elections for the mayor, state representative and state senator. Together with these races, residents of Sunny Hill need to vote on two ballot initiatives: whether or not to build a new park downtown and whether or not to legalize AI conferences. Candidates from three parties, Donkey, Elephant and Rhino, are vying for the elected offices and each candidate takes a position on each ballot initiative, with the Donkey party candidates generally supporting both, Elephant party candidates opposing both, and Rhino party candidate opposing the park but supporting legal AI.

The public, the candidates and their campaigns, as well as the election commissioner are kept aware of the voting trends in Sunny Hill by polls in the weeks preceding the elections. The polls are conducted among diverse groups such as representative samples of the entire town population, of likely voters, women, residents of specific neighborhoods, members of specific parties, etc. Among the questions asked on the polling surveys are current preferences of the participant for each race and for the initiatives, together with some demographic information and some supplementary questions such as whether the participant saw a specific and highly-charged infomercial. Poll data, such as the collection shown in Figure 1.2, is assumed to have a margin of error. A typical statement is, “The straight Donkey ticket for the election is preferred by 36% of respondents +/- 3%”. Although the actual polling data indicates statistical information about respondents, it can be interpreted probabilistically as “The probability that a resident of Sunny Hills will vote straight Donkey ticket in the elections is between 33% and 39% based on the October 26 poll.” (See the top line of Poll7 table in Figure 1.2.)

Figure 1.2 shows a small sample of a wide variety of distributions that may be produced by the pollsters. Poll1 contains the distribution of the vote in the mayoral race and the two ballot initiatives
<table>
<thead>
<tr>
<th>id</th>
<th>Poll1</th>
<th>gender: men</th>
<th>party: Donkey</th>
<th>date: October 26</th>
<th>senate = Donkey</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mayor</td>
<td>park</td>
<td>legalization</td>
<td>l</td>
<td>u</td>
</tr>
<tr>
<td>Donkey yes</td>
<td>yes</td>
<td>0.44</td>
<td>0.52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Donkey yes</td>
<td>no</td>
<td>0.12</td>
<td>0.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Donkey no</td>
<td>yes</td>
<td>0.08</td>
<td>0.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Donkey no</td>
<td>no</td>
<td>0.04</td>
<td>0.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant yes</td>
<td>yes</td>
<td>0.05</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant yes</td>
<td>no</td>
<td>0.01</td>
<td>0.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant no</td>
<td>yes</td>
<td>0.03</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant no</td>
<td>no</td>
<td>0.06</td>
<td>0.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino yes</td>
<td>yes</td>
<td>0.02</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino yes</td>
<td>no</td>
<td>0.01</td>
<td>0.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino no</td>
<td>yes</td>
<td>0.03</td>
<td>0.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino no</td>
<td>no</td>
<td>0.01</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>Poll2</th>
<th>date: October 23</th>
<th>gender: male</th>
<th>respondents: 238, {senate}</th>
<th>respondents: 195, {legalization}</th>
<th>overlap: 184</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>senate legaliz-</td>
<td>l</td>
<td>u</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino yes</td>
<td>0.04</td>
<td>0.11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino no</td>
<td>0.1</td>
<td>0.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Donkey yes</td>
<td>0.22</td>
<td>0.27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Donkey no</td>
<td>0.09</td>
<td>0.16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant yes</td>
<td>0.05</td>
<td>0.13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant no</td>
<td>0.21</td>
<td>0.26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>Poll3</th>
<th>locality: Sunny Hill</th>
<th>date: October 26</th>
<th>mayor = Donkey</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>park</td>
<td>legalization</td>
<td>l</td>
<td>u</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>0.56</td>
<td>0.62</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>0.14</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>0.21</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>0.03</td>
<td>0.07</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>Poll4</th>
<th>locality: South Side</th>
<th>date: October 12</th>
<th>sample: 323</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mayor</td>
<td>l</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>Donkey</td>
<td>0.2</td>
<td>0.26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant</td>
<td>0.42</td>
<td>0.49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino</td>
<td>0.25</td>
<td>0.33</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>Poll5</th>
<th>locality: Downtown</th>
<th>date: October 12</th>
<th>sample: 275</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mayor</td>
<td>l</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>Donkey</td>
<td>0.48</td>
<td>0.55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant</td>
<td>0.25</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino</td>
<td>0.2</td>
<td>0.24</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>Poll6</th>
<th>locality: West End</th>
<th>date: October 12</th>
<th>sample: 249</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mayor</td>
<td>l</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>Donkey</td>
<td>0.38</td>
<td>0.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant</td>
<td>0.34</td>
<td>0.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino</td>
<td>0.15</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>Poll7</th>
<th>locality: Sunny Hills</th>
<th>date: October 26</th>
<th>sample: 249</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>representative</td>
<td>l</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>Donkey</td>
<td>0.33</td>
<td>0.39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elephant</td>
<td>0.32</td>
<td>0.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhino</td>
<td>0.25</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.2: Sunny Hill pre-election polls
by men affiliated with the Donkey party who intended to vote Donkey in the Senate race, as indicated in a survey conducted on October 26. Poll2 is a joint distribution of the expected vote for the senate race and the ballot of legalization based on a survey of a representative sample of male residents of Sunny Hill taken on October 23. Poll3 contains information about the expected vote on the ballot initiatives based on a survey taken on October 26. Poll4, Poll5 and Poll6 contain information about the expected vote distribution in the mayoral race of the residents of four different parts of Sunny Hill based on the surveys taken on the same day. Finally, Poll7 contains the information of state representative race. Sample sizes are also provided for convenience.

These and other similar distributions are used by campaign managers and the elections commissioner to gain insight into the political trends in Sunny Hill. This statistical and probabilistic information should be stored in a way that facilitates future recall. There are many possible questions that the commissioner and her/his analysts might ask, including:

- “What information is available about voter attitudes on October 26?”
- “What are other voting intentions of people who vote Donkey for mayor?”
- “What information is known about voter intentions in the mayoral race?”
- “What voting patterns are likely to occur with probability between 0.2 and 0.3?”
- “With what probability are voters likely to vote for a Donkey mayor and Elephant Senator?”
- “How do people who intend to vote Donkey for mayor plan to vote for the park construction ballot initiative?”

While analyzing poll data to find answers to these questions is fairly straightforward, these results should be stored somewhere as well for future comparison. We present an extended SPO framework for convenient storage and retrieval of complex statistical and probabilistic information.

### 1.2 General Framework

Decision support systems would be useful in all of the above examples. These systems could assist case managers to automatically generate feasible plans for their clients in the W2W program, help advisors to find possible plans for their students, and etc. In this section, we describe a general framework which supports decision making and planning under uncertainty.
Bayesian Networks [70] are becoming an increasingly important area of research, especially in the Artificial Intelligence community. A Bayesian network is a compact, graphical representation of a joint probability distribution. Bayesian networks could offer assistance in a wide range of endeavors. The use of Bayesian networks in a number of applications is examined [41, 74]. Recently many new methods have been developed to learn Bayesian networks from data [50, 51, 37], and these techniques are remarkably effective for many data analysis problems, for instance, characterizing gene regulatory networks from high throughput biological data [38, 66]. Researchers are exploiting the Bayesian network approach for the construction of decision support systems or expert systems [65].

The work described here is a part of the larger project “Decision Making and Planning under Uncertainty with Constraints”. This project develops software tools for eliciting, modeling, and managing complex and probabilistic information, and algorithms for building domain models and suggesting feasible plans. It will develop database-driven systems for supporting decision making and planning under uncertainty. In our general framework, we adopt the Bayesian network representation for modeling application domains. The framework is an integrated solution for building decision support systems, and it consists of three major parts. The overall architecture of the framework is shown in Figure 1.3. The left part is responsible for model construction and data management. It extracts information from data, constructs models for application domains, elicits probabilistic information, stores and manipulates the data. The middle part considers construction of situation-specific models. It elicits client demographic data and preferences, and configures a smaller Bayesian network model based on those decision variables most likely to impact the client. The network will use conditional probability tables most appropriate for that client. The situation-specific network will then be passed to the decision-theoretic planning part of the system, which constructs feasible plans for the client.

The process of building a smaller network specific to a planning query is called knowledge-based model construction (KBMC). This notion was first introduced by Breese, Goldman and Wellman [11], and in part inspired this work. Such networks are also referred to as situation-specific network fragments [60]. Several suggested methods for KBMC require on-the-fly construction of conditional probabilities from simple components plus a standard combination method. For instance, Ngo and Haddawy [68] limit their system to NOISY-OR tables, while Glesner and Koller [43] allow NOISY-OR, NOISY-AND, MIN, and MAX. We refer the reader for recent surveys to Peter Haddawy’s introduction to the 1999 AI Magazine special issue on Bayesian Reasoning [47] and to Kathy Laskey’s class notes [59].

The idea is to use information about a specific instance in order to specialize or personalize
the model being used to reason about that instance. One of the key features of KBMC is that the network constructed for an instance can be significantly smaller than the non-specific network, and can be constructed on demand. Situation-specific networks improve planning in two ways: (i) their smaller size means faster turnaround time for the planning algorithm (and quite possibly better overall plans) and (ii) situation-specific conditional probability tables may reflect current situations much better than generic ones, resulting in a more appropriate plan.

To support KBMC, our knowledge construction tools have to deal with an enormous amount of probabilistic data efficiently. We cannot rely on ad-hoc data management methods as most of other tools currently do. In order to facilitate efficient and effective KBMC, it is necessary to provide a database capable of storing and manipulating probabilities and contextual information for those probabilities. It is also necessary to provide a means of generating queries to that database that will retrieve those variables deemed likely to impact a client, based on demographic and preference data for the client.

### 1.3 Towards Semistructured Probabilistic Databases

There are large quantities of uncertain information in decision support systems, such as those for solving problems arising in our motivating examples. However, when trying to store the data
described in the above examples using one of the probabilistic database models, relational or object, a number of problems will be encountered [49]. Probabilistic relational database models [5, 25, 58] lack the flexibility to store all of our data in a straightforward manner. For instance, in the risk analysis application the aspects are viewed as random variables. As such, it is natural to represent each aspect as a database attribute that can take values from its domain. However, with such an interpretation, a joint probability distribution of values in two aspects will have a schema different from the joint probability distribution of three aspects, and therefore, will have to be stored in a separate relation. For the same reason, joint probability distributions with same number of aspects but different aspects have to be stored in separate relations. In such a database, expressing queries like “Find all probability distributions that include Driver’s Age as a random variable” is very inconvenient, if at all possible.

Probabilistic object models [28, 56] are also not a good fit for storing this kind of data. In the framework of Eiter, et al. [28], a probabilistic object is a “real” object, some of whose properties are uncertain and probabilistically described. For our application, however, the probability distribution is the primary object that needs to be stored.

As seen from the examples above, probability distributions are complex objects. Yet we want a way to store probability distributions of varying shapes and sizes, and to access them readily. Probabilistic relational DBMS could not deal with both storage and retrieval of the probability distributions constructed during the analysis. Such probability distributions are hard to store in probabilistic relational databases. General XML databases could not manipulate probability distribution correctly although they can store those probability distributions in general.

This work develops a probabilistic database framework for collecting, storing and managing a large quantity of probabilistic or statistical information. It provides a data model and query language to store, query and manipulate probability distribution objects. The examples indicate the importance of the following features:

- probability distributions along with their associated, non-probabilistic information are treated as single objects;
- probability distributions with different structures (e.g., different number/type of random variables involved) can be stored in the same “relations”;
- query language facilities for retrieval of full distributions and retrieval of parts of distributions (individual rows of the probability tables) based on their properties are provided;
- query language facilities for manipulations and transformations of probability distributions according to the laws of probability theory are provided;
probability distributions are correctly handled during query processing.

The semistructured data model [1, 12, 80] has gained wide acceptance recently as the means of representing data which do not conform to a rigid structure of schema. In particular, the similarity between the semistructured data model and the underlying data model for eXtensible Markup Language (XML) [10], the emerging open standard for data storage and transmission over the Internet, makes this approach attractive. However, currently available XML databases, such as native XML databases [63, 82], do not generally handle probability distributions.

1.4 Overview of Dissertation

The dissertation is organized as follows. In Chapter 2, we review concepts related to probabilistic distributions, interval probabilities, Bayesian networks and query processing in relational database management systems and in middleware or mediators.

Chapter 3 describes the semistructured probabilistic database framework: we introduce SPO data model, SP-Algebra, and discuss our initial steps towards designing a query optimization framework. We describe the initial SPO data model and algebra of Dekhtyar, Goldsmith and Hawkes [20] due to its close relationship with the research described in this dissertation, and include all the changes and extensions introduced to it. A number of features, key to the SPO framework, were modified in the current model. For instance, a path expression is added in the SPO data model to specify the origin of a particular object. A new definition of the join operation replaces the one in the initial framework so that the operation complies with the laws of probability theory. We describe a set of equivalence relations and cost model for query optimization framework.

Chapter 4 describes in detail the actual implementation of the semistructured probabilistic database management system (SPDBMS). We discuss the XML-based implementation of the SPO data model and show how SPOs are represented in XML with the help of a meta language SPO-ML. We describe our relational schema generation strategies for efficiently mapping XML representations of SPOs into relational tables. We also discuss our query translation mechanism which automatically generates a sequence of SQL queries for evaluating SP-Algebra queries, as well as the preliminary experimental results of our pilot implementation.

In Chapter 5, we discuss the extension of SPO framework to support probability intervals. The extended framework allows one to associate of context or conditionals with individual random variables. We describe the extended data model and query algebra.

We describe our Bayesian network development suite (BaNDeS), a system which has been specially designed and implemented for building Bayesian networks with the full support of the
SPDBMS, in Chapter 6.

We describe related work on probabilistic databases and probability intervals, and discuss the physical storage techniques for storing XML data in relational databases in Chapter 7. Finally, we conclude the work developed and outline the ongoing and future work in Chapter 8.
Chapter 2

Background

In this section we briefly introduce the concepts related to probability distributions, interval probabilities, Bayesian networks and the principles of query processing in relational database management systems and in mediators.

2.1 Probability Distributions

In the chapters below, we frequently discuss manipulation of probability distributions of discrete random variables. We briefly introduce those basic concepts in this section. The presentation below follows [27].

What is a random variable? We can think of a random variable as a variable with a domain of possible values. A formal mathematical definition of random variable can be found in [73], "A random variable is a function from the sample space of an experiment to the set of real numbers".

There are two types of random variables, discrete and continuous. A discrete random variable is one which may take on only a countable number of distinct values. In this work we only consider discrete random variables with a domain of finite number of distinct values.

Let $X$ be a discrete random variable with the domain of values $\{x_1, x_2, \ldots, x_n\}$. The probability distribution of the $X$ is a function that gives the probability $p(x_i)$ that the random variable takes the value of $x_i$. We have $p(x_i) = P(X = x_i)$ for each value $x_i$. We require that the following two conditions be met:

a. $0 \leq p(x_i) \leq 1$ for $i = 1, 2, \ldots, n$, and

b. $\sum_{i=1}^{n} p(x_i) = 1$.

A joint probability distribution of discrete random variables $X_1, X_2, \ldots, X_m$ is defined as the probability of co-occurrence of multiple events, i.e., $P(X_1 = x_{1i}, X_2 = x_{2i}, \ldots, X_m = x_{mi})$. The domain of the set of random variables is defined as the Cartesian product of all individual domains, i.e., $D(X_1) \times D(X_2) \times \ldots \times D(X_m)$. Note that the sum of all probabilities still equals 1.

Given a joint probability distribution over $X$ and $Y$ with $X = \{X_1, X_2, \ldots, X_m\}$ and $Y = \{Y_1, Y_2, \ldots, Y_k\}$, one may want to obtain the probability distribution over a subset of random variables $X$ regardless of the rest of the random variables. This operation is called marginalization.

\footnote{A mediator is software layer which facilitates the communication between different applications. It functions both as a server to its front end and as a client to its back end.}
Summing a discrete joint probability distribution over a set of random variables effectively computes the marginal probability distribution. For instance, this operation gives

\[ P(X) = \sum_y (X, Y = y). \]

Given a joint probability distribution over X and Y with \( X = \{X_1, X_2, \ldots, X_m\} \) and \( Y = \{Y_1, Y_2, \ldots, Y_k\} \), in other situations one may also derive the probability distribution of a subset of random variables given the rest of the random variables have been set to specific values. This can be obtained with a conditioning operation, and the resulting distribution is called a conditional probability distribution. Bayes’ Rule described below can be used to determine a conditional probability

\[ P(X = x | y) = \frac{P(y | X = x)P(X = x)}{P(y)}. \]

For probability distributions the following expression can be used to compute the conditional probability distribution:

\[ P(X | Y) = \frac{P(X, Y)}{P(Y)}. \]

Here we give an example of how to perform marginalization and conditioning operations.

**Example 2.1** Suppose we have a joint probability distribution over two random variables \( X \) and \( Y \), as shown in the left side of Figure 2.1. Based on this joint probability distribution, we can perform marginalization to obtain a marginal probability distribution. A marginal probability distribution over random variable \( X \) can be computed by projecting out the random variable \( Y \) and summing up the probabilities of the rows whose other attributes have the same values, and is shown in the middle of Figure 2.1. A conditional probability distribution over \( X \) given \( Y \) takes the value of \( y_1 \) can also be computed by remove the rows which do not satisfy the condition that \( Y \) takes the value of \( y_1 \), followed by a normalization operation. The final results is shown in the right side of Figure 2.1.

![Figure 2.1: Marginalization and conditioning operations](image)
2.2 Interval Probabilities

Probability intervals occur in many situations. In general, when computing the probability of a conjunction of two events, knowing the point probabilities of the original events does not immediately guarantee uniqueness of the probability of the conjunction. The latter probability depends also on the known relationship between the two original events. When no such relationship is known, the probability of conjunction can only be computed to lie in an interval [8]:
\[
\max(p(a) + p(b) - 1, 0) \leq p(a \land b) \leq \min(p(a), p(b)).
\]
In some applications, it may be infeasible to obtain the exact probabilities or the point probabilities obtained will not be robust, so intervals better represent our knowledge of the domain.

Here we assume that the probability space is \( \mathcal{P} = C[0,1] \), the set of all subintervals of the interval \([0,1]\). The rest of this section formally introduces the possible worlds semantics for the probability distributions over \( \mathcal{P} \) and the notions of consistency and tightness of the distributions. The possible worlds approach to describing interval probabilities has been adopted by a number of researchers. In particular, the semantics described here was first introduced by de Campos, Huete and Moral [17]. Similar notions are also found in the work of Weichselberger [85]. In database literature, a specialized version of possible worlds semantics for interval probabilities appeared in [22]. The description of the semantics in this dissertation follows the work in [17], but we use a different notation in the context of probabilistic databases [88, 44, 87]. The theorems have been formulated before, however, we give our own proofs (derived independently). We discuss related work in more detail in Section 7.2.

**Definition 2.1** Let \( V \) be a set of random variables. A probabilistic interpretation (p-interpretation) over \( V \) is a function \( I_V : \text{dom}(V) \rightarrow [0,1] \), such that \( \sum_{\bar{x} \in \text{dom}(V)} I_V(\bar{x}) = 1 \).

Given a set of random variables \( V \), a p-interpretation over it is any valid point probability distribution. Our main idea is that a probability distribution function (pdf) \( P : \text{dom}(V) \rightarrow C[0,1] \) represents a set of possible point probability distributions (a.k.a., p-interpretations). This corresponds to de Campos, et al.’s instance [17].

In the rest of the paper we adopt the following notation. Given a probability distribution \( P : \text{dom}(V) \rightarrow C[0,1] \), for each \( \bar{x} \in \text{dom}(V) \) we write \( P(\bar{x}) = [l_{\bar{x}}, u_{\bar{x}}] \). Whenever we enumerate \( \text{dom}(V) \) as \( \{\bar{x}_1, \ldots, \bar{x}_m\} \), we write \( P(\bar{x}_i) = [l_i, u_i], 1 \leq i \leq m \).

**Definition 2.2** Let \( V \) be a set of random variables and \( P : \text{dom}(V) \rightarrow C[0,1] \) be a complete interval probability distribution function over \( V \). A probabilistic interpretation \( I_V \) satisfies \( P (I_V \models P) \) iff \( \forall \bar{x} \in \text{dom}(V) (l_{\bar{x}} \leq I_V(\bar{x}) \leq u_{\bar{x}}) \).
Let $V$ be a set of random variables and $P' : X \to C[0,1]$ an incomplete interval probability distribution function over $X \subseteq \text{dom}(V)$. A probabilistic interpretation $I_V$ satisfies $P'$ ($I_V \models P'$) iff $(\forall \bar{x} \in X)(I_\bar{x} \leq I_V(\bar{x}) \leq u_\bar{x})$.

Basically, if a p-interpretation $I_V$ satisfies an interval probability distribution function, then given $P$, $I_V$ is a possible point probability distribution.

Example 2.2 Consider a random variable $v$ with domain $\{a, b, c\}$. Let probability distribution functions $P_1$, $P_2$ and $P_3$ and p-interpretations $I_1$, $I_2$, $I_3$ and $I_4$ be defined in this table.

\[
\begin{array}{|c|c|c|}
\hline
& P_1 & P_2 & P_3 \\
\hline
P_1(a) & [0.2, 0.3] & P_2(a) = [0.3, 0.6] & P_3(a) = [0.4, 0.5] \\
P_1(b) & [0.3, 0.45] & P_2(b) = [0.3, 0.4] & P_2(b) = [0.4, 0.5] \\
P_1(c) & [0.3, 0.5] & P_2(c) = [0.4, 0.5] & P_3(c) = [0.4, 0.5] \\
\hline
I_1 & I_2 & I_3 & I_4 \\
\hline
I_1(a) & 0.3 & 0.5 & 0.25 & 0.7 \\
I_1(b) & 0.3 & 0.4 & 0.45 & 0.3 \\
I_1(c) & 0.4 & 0.1 & 0.3 & 0 \\
\hline
\end{array}
\]

P-interpretation $I_1$ satisfies both $P_1$ and $P_2$. P-interpretation $I_2$ satisfies $P_2$ but not $P_1$ while $I_3$ satisfies $P_1$ but not $P_2$. Finally, $I_4$ satisfies neither $P_1$ nor $P_2$. None of the p-interpretations $I_1$, $I_2$, $I_3$, $I_4$ satisfies $P_3$.

We can now specify the consistency criterion for interval probability distribution functions.

Definition 2.3 An interval probability distribution function $P : \text{dom}(V) \to C[0,1]$ is consistent iff there exists a p-interpretation $I_V$, such that $I_V \models P$.

Example 2.3 Consider the interval probability distribution functions $P_1$, $P_2$ and $P_3$ described in Example 2.2. As that example shows, $I_1 \models P_1$ and $I_1 \models P_2$, and thus, both $P_1$ and $P_2$ are consistent interval probability distributions.

On the other hand, none of the p-interpretations from Example 2.2 satisfies $P_3$. One notices that any p-interpretation $I$ satisfying $P_3$ must have $I(a) \geq 0.4$, $I(b) \geq 0.4$ and $I(c) \geq 0.4$, hence $I(a) + I(b) + I(c) \geq 1.2$, which contradicts the constraint $I(a) + I(b) + I(c) = 1$ on p-interpretations. Therefore, no p-interpretation would satisfy $P_3$ and thus, $P_3$ is inconsistent.

The following theorem specifies the necessary and sufficient conditions for an interval probability distribution function to be consistent.
Theorem 2.1 Let $V$ be a set of random variables and $P : \text{dom}(V) \rightarrow C[0,1]$ be a complete interval probability distribution function over $V$. Let $\text{dom}(V) = \{\bar{x}_1, \ldots, \bar{x}_m\}$ and $P(\bar{x}_i) = [l_i, u_i]$. $P$ is consistent iff the following two conditions hold: (1) $\sum_{i=1}^{m} l_i \leq 1$; (2) $\sum_{i=1}^{m} u_i \geq 1$.

Let $P' : X \rightarrow C[0,1]$ be an incomplete interval probability distribution function over $V$. Let $X = \{\bar{x}_1, \ldots, \bar{x}_m\}$ and $P'(\bar{x}_i) = [l_i, u_i]$. $P'$ is consistent iff $\sum_{i=1}^{m} l_i \leq 1$.

Proof of Theorem 2.1

Let $P$ be an interval probability distribution function over $V$ and let $\text{dom}(V) = \{\bar{x}_1, \ldots, \bar{x}_m\}$. Remember that we denote $P(\bar{x}_i)$ as $[l_i, u_i]$. Consider now two functions $f_l, f_u : \text{dom}(V) \rightarrow [0,1]$ such that $f_l(\bar{x}_i) = l_i$ and $f_u(\bar{x}_i) = u_i$ for all $1 \leq i \leq m$.

First we prove $P$ is consistent if $\sum_{i=1}^{m} l_i \leq 1$ and $\sum_{i=1}^{m} u_i \geq 1$.

If $\sum_{i=1}^{m} l_i = 1$ then $f_l$ is a p-interpretation and $f_l \models P$. Therefore $P$ is consistent.

If $\sum_{i=1}^{m} u_i = 1$ then $f_u$ is a p-interpretation and $f_u \models P$ Therefore $P$ is consistent.

Consider now the case when $\sum_{i=1}^{m} l_i < 1$ and $\sum_{i=1}^{m} u_i > 1$. Let $\sum_{i=1}^{m} l_i = L$ and $\sum_{i=1}^{m} u_i = U$. We know that $L < 1 < U$.

Consider a function $I : \text{dom}(V) \rightarrow [0,1]$ such that

$$I(\bar{x}_i) = \frac{1 - L}{U - L} u_i + \left(1 - \frac{1 - L}{U - L}\right) l_i.$$  

We now show that $I$ is a p-interpretation and $I \models P$. Let $\alpha = \frac{1 - l_i}{U - L}$. As $L < 1 < U$, $0 < \alpha < 1$ and we can rewrite the definition of $I$ as $I(\bar{x}_i) = l_i + \alpha (u_i - l_i)$. Then $l_i \leq I(\bar{x}_i) \leq u_i$. Thus, if $I$ is a p-interpretation then $I \models P$.

To show that $I$ is a p-interpretation we need $\sum_{i=1}^{m} I(\bar{x}_i) = 1$. This can be demonstrated as follows:

$$\sum_{i=1}^{m} I(\bar{x}_i) = \sum_{i=1}^{m} \left(\frac{1 - L}{U - L} u_i + \left(1 - \frac{1 - L}{U - L}\right) l_i\right)$$

$$= \alpha \sum_{i=1}^{m} u_i + (1 - \alpha) \sum_{i=1}^{m} l_i$$

$$= \alpha U + (1 - \alpha) L$$

$$= \frac{1 - L}{U - L} U + \left(1 - \frac{1 - L}{U - L}\right) L$$

$$= \frac{(1 - L) U + (U - L - 1 + L) L}{U - L}$$

$$= \frac{U}{U - L} = 1.$$  

To complete the proof, we prove that if $P$ is consistent then $\sum_{i=1}^{m} l_i \leq 1$ and $\sum_{i=1}^{m} u_i \geq 1$.

If $P$ is consistent, then there exists a p-interpretation $I : \text{dom}(V) \rightarrow [0,1]$, such that $I \models P$.

Then, for each $\bar{x}_i$, $1 \leq i \leq m$, we have $l_i \leq I(\bar{x}_i) \leq u_i$. But then,

$$\sum_{i=1}^{m} l_i \leq \sum_{i=1}^{m} I(\bar{x}_i) \leq \sum_{i=1}^{m} u_i.$$  

19
As $I$ is a $p$-interpretation, $\sum_{i=1}^{m} I(\bar{x}_i) = 1$, we immediately get $\sum_{i=1}^{m} l_i \leq 1$ and $\sum_{i=1}^{m} u_i \geq 1$. 

Consistency is not the only property of interval probability distribution functions that is of interest. Another property of interval probability distribution functions, tightness, is also important.

**Example 2.4** Consider the interval probability distribution $P$ as shown on Figure 2.2 (left). Assume that $P$ is complete. It is easy to see that $P$ is consistent (indeed, the sum of lower bound of probability intervals adds up to 0.4 and the the sum of the upper bounds adds up to 1.5). In fact, there are many different $p$-interpretations satisfying $P$. Of particular interest to us are the $p$-interpretations that satisfy $I(\bar{x}_1) = 0.1; I(\bar{x}_2) = 0.1; I(\bar{x}_3) = 0.1; I(\bar{x}_4) = 0.7$ satisfies $P$ and hits the lower bounds of probability intervals provided by $P$ for $\bar{x}_1, \bar{x}_2$ and $\bar{x}_3$. Similarly, $I(\bar{x}_1) = 0.2; I(\bar{x}_2) = 0.2; I(\bar{x}_3) = 0.3; I(\bar{x}_4) = 0.3$ satisfies $P$ and hits the upper bounds of probability intervals for $\bar{x}_1, \bar{x}_2$ and $\bar{x}_3$. Thus, every single number in the probability intervals for $\bar{x}_1, \bar{x}_2$ and $\bar{x}_3$ is reachable by different $p$-interpretations satisfying $P$.

However, the same is not true for $\bar{x}_4$. If some $p$-interpretation $I$ satisfies $P$, then $I(\bar{x}_4) \neq 0.1$. Indeed, we know that $I(\bar{x}_1) + I(\bar{x}_2) + I(\bar{x}_3) + I(\bar{x}_4) = 1$ and if $I(\bar{x}_4) = 0.1$ then $I(\bar{x}_1) + I(\bar{x}_2) + I(\bar{x}_3) = 0.9$. However, the maximum values for $\bar{x}_1, \bar{x}_2$ and $\bar{x}_3$ allowed by $P$ are 0.2, 0.2 and 0.3 respectively, and they add up to only 0.7.

Similarly, no $p$-interpretation $I$ satisfying $P$ can have $I(\bar{x}_4) = 0.8$. Indeed, in this case, $I(\bar{x}_1) + I(\bar{x}_2) + I(\bar{x}_3) = 1 - 0.8 = 0.2$. However, the smallest values for $\bar{x}_1, \bar{x}_2$ and $\bar{x}_3$ allowed by $P$ are all 0.1 and they add up to 0.3.

<table>
<thead>
<tr>
<th>$X$</th>
<th>$l$</th>
<th>$u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{x}_1$</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>$\bar{x}_2$</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>$\bar{x}_3$</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>$\bar{x}_4$</td>
<td>0.1</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$X$</th>
<th>$l$</th>
<th>$u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{x}_1$</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>$\bar{x}_2$</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>$\bar{x}_3$</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>$\bar{x}_4$</td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Figure 2.2: Tightness of interval probability distributions.

The notion of “reachability” discussed above can be formalized as follows.
Definition 2.4 Let $P : X \rightarrow \mathcal{C}[0,1]$ be an interval probability distribution function over a set of random variables $V$. Let $X = \{\tilde{x}_1, \ldots, \tilde{x}_m\}$ and $P(\tilde{x}_i) = [l_i, u_i]$. A number $\alpha \in [l_i, u_i]$ is reachable by $P$ at $\tilde{x}_i$ iff there exists a p-interpretation $I_V \models P$, such that $I(\tilde{x}_i) = \alpha$.

Proposition 2.1 Let $P : X \rightarrow \mathcal{C}[0,1]$ be an interval probability distribution function over a set of random variables $V$. If for some $\tilde{x} \in X$ there exist $\alpha, \beta, l_{\tilde{x}} \leq \alpha \leq \beta \leq u_{\tilde{x}}$ which are both reachable by $P$ at $\tilde{x}$, then any $\gamma \in [\alpha, \beta]$ is reachable by $P$ at $\tilde{x}$.

Intuitively points unreachable by an interval probability distribution function represent “dead weight”; they do not provide any additional information about any possible point probability distributions.

Definition 2.5 Let $P : X \rightarrow \mathcal{C}[0,1]$ be an interval probability distribution over a set $V$ of random variables. $P$ is called tight iff $(\forall \tilde{x} \in X)(\forall \alpha \in [l_{\tilde{x}}, u_{\tilde{x}}])$ $\alpha$ is reachable by $P$ at $\tilde{x}$.

Example 2.5 As shown in Example 2.4, the interval probability distribution function $P$ shown on the left-hand side of Figure 2.2 is not tight. On the other hand, interval probability distribution function $P'$ shown on the right-hand side of Figure 2.2 is tight. Its tightness follows from the fact that p-interpretations $I_1$ and $I_2$ from Example 2.4 both satisfy it, and now, both upper and lower bounds for $\tilde{x}_4$ are reachable.

Function $P'$ has another important distinction with respect to $P$. Indeed, one can show that for any p-interpretation $I$, $I \models P$ iff $I \models P'$, i.e., the sets of p-interpretations that satisfy $P$ and $P'$ coincide. Hence, one can say that $P'$ is a tight equivalent of $P$.

We want to replace interval probability distributions that are not tight with their tight equivalents. This will be done using the tightening operator.

Definition 2.6 Given an interval probability distribution $P$, an interval probability distribution $P'$ is its tight equivalent iff (i) $P'$ is tight and (ii) For each p-interpretation $I$, $I \models P$ iff $I \models P'$.

Proposition 2.2 Each complete interval probability distribution $P$ has a unique tight equivalent.

Definition 2.7 A tightening operator $\mathcal{T}$ takes as input an interval probability function $P : X \rightarrow \mathcal{C}[0,1]$ and returns its tight equivalent $P' : X \rightarrow \mathcal{C}[0,1]$. 21
Our next goal is to compute the result of applying the tightening operator to an interval probability distribution function efficiently. First we notice that if $P$ is tight then $\mathcal{T}(P) = P$.

The theorem below specifies an efficient procedure for computing the results of tightening an interval probability distribution function.

**Theorem 2.2** Let $P : \text{dom}(V) \rightarrow C[0,1]$ be a complete interval probability distribution function over a set of random variables $V$. Let $\text{dom}(V) = \{\bar{x}_1, \ldots, \bar{x}_m\}$ and $P(\bar{x}_i) = [l_i, u_i]$. Then $(\forall 1 \leq i \leq m)$

$$\mathcal{T}(P)(\bar{x}_i) = \left[\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i), \min(u_i, 1 - \sum_{j=1}^{m} l_j + l_i)\right].$$

**Proof of Theorem 2.2**

Let $P'(\bar{x}_i) = \left[\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i), \min(u_i, 1 - \sum_{j=1}^{m} l_j + l_i)\right]$. We need to prove two statements: $P \equiv P'$ and $P'$ is tight.

First we prove $P \equiv P'$.

Notice that for all $1 \leq i \leq m$, $[\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i), \min(u_i, 1 - \sum_{j=1}^{m} l_j + l_i)] \subseteq [l_i, u_i]$.

Indeed, $l_i \leq \max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i)$ and $\min(u_i, 1 - \sum_{j=1}^{m} l_j + l_i) \leq u_i$.

Now, because $P$ is consistent, $(\forall 1 \leq i \leq m)$, $l_i \leq u_i$ and $\sum_{j=1}^{m} l_j \leq 1 \leq \sum_{j=1}^{m} u_j$. But then $1 - \sum_{j=1}^{m} u_j \leq 0$ and hence $1 - \sum_{j=1}^{m} u_j + u_i \leq u_i$, and therefore $\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i) \leq u_i$.

Similarly, we obtain $l_i \leq \min(u_i, 1 - \sum_{j=1}^{m} l_j + l_i)$. Finally, for $1 \leq j \leq m$, $l_j \leq u_j$, $\sum_{j=1}^{m} l_j - l_i \leq \sum_{j=1}^{m} u_j - u_i$ and therefore $1 - \sum_{j=1}^{m} u_j + u_i \leq 1 - \sum_{j=1}^{m} l_j + l_i$. Therefore,

$$l_i \leq \max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i) \leq \min(u_i, 1 - \sum_{j=1}^{m} l_j + l_i) \leq u_i.$$

This means that $(\forall I : \text{dom}(V) \rightarrow [0,1])(I \models P' \Rightarrow I \models P)$.

We now need to show the inverse: $(\forall I : \text{dom}(V) \rightarrow [0,1])(I \models P \Rightarrow I \models P')$.

Let $I$ be a $p$-interpretation over $V$ and let $I \models P$. Therefore, $(\forall 1 \leq i \leq m)(l_i \leq I(\bar{x}_i) \leq u_i)$. We need to show $\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i) \leq I(\bar{x}_i) \leq \min(u_i, 1 - \sum_{j=1}^{m} l_j + l_i)$.

We show $\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i) \leq I(\bar{x}_i)$. The other inequality can be proven similarly.

We know that $l_i \leq I(\bar{x}_i)$, so if $\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i) = l_i$ then the inequality holds.

Assume now that $\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i) = 1 - \sum_{j=1}^{m} u_j + u_i$. Since $l_i \geq 0$ and $1 - \sum_{j=1}^{m} u_j + u_i \geq 0$, we can have $\sum_{j=1}^{m} u_j - u_i \leq 1$.

Assume that the inequality does not hold, i.e., $I(\bar{x}_i) < 1 - \sum_{j=1}^{m} u_j + u_i$. We know that for all $1 \leq j \leq m$, $I(\bar{x}_j) \leq u_i$. Therefore $\sum_{j=1}^{m} I(\bar{x}_j) = \sum_{j=1, j \neq i}^{m} I(\bar{x}_j) + I(\bar{x}_i) \leq \sum_{j=1}^{m} u_j - u_i + I(\bar{x}_i) < \sum_{j=1}^{m} u_j - u_i + 1 - \sum_{j=1}^{m} u_j + u_i = 1.$

22
As $I$ is a p-interpretation, $\sum_{j=1}^{m} I(x_j)$ must be equal to 1. This contradicts with $I(x_i) \geq 1 - \sum_{j=1}^{m} u_j + u_i$.

Next we show that $P'$ is tight.

We show that for all $1 \leq i \leq m$, every point $a \in P'(x_i)$ is reachable. By Proposition 2.1, it is sufficient to prove that the end points of the $P'(x_i)$ interval are reachable.

Recall that $P'(x_i) = [\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i), \min(u_i, 1 - \sum_{j=1}^{m} l_j + l_i)]$. We show that $\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i)$ is reachable. Similar reasoning can be applied to show the reachability of the upper bound.

We show that there exists a p-interpretation $I$ such that $I \models P$ and $I(x_i) = \max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i)$. As we have shown that $P \equiv P'$, it follows that $I \models P'$.

First, suppose $\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i) = l_i$. Then $1 - l_i \leq \sum_{j=1}^{m} u_j - u_i$. Because $\sum_{j=1}^{m} l_j \leq 1$, we get $\sum_{j=1}^{m} l_j - l_i \leq 1 - l_i \leq \sum_{j=1}^{m} u_j - u_i$.

But then, by reasoning similar to that in the proof of Theorem 2.1, there exist numbers $a_1, \ldots, a_m$, such that $a_i = l_i$ and $(\forall 1 \leq j \leq m) (l_j \leq a_i \leq u_j)$ and $a_1 + \ldots + a_m = 1$. Let $I$ be a p-interpretation such that $I(x_j) = a_j$ for all $1 \leq j \leq m$. Then $I(x_i) = l_i$, $\sum_{j=1}^{m} I(x_j) = 1$ and $I \models P$. Therefore $I \models P'$ and $l_i = \max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i)$ is reachable.

Now suppose $\max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i) = 1 - \sum_{j=1}^{m} u_j + u_i$.

Consider the function $I : dom(V) \rightarrow [0, 1]$, such that $I(x_i) = 1 - \sum_{j=1}^{m} u_j + u_i$ and $I(x_j) = u_j$ for all $1 \leq j \leq m$, $j \neq i$. If $I$ is a p-interpretation then $I \models P$ as $l_i \leq 1 - \sum_{j=1}^{m} u_j + u_i \leq u_i$ and $u_j \in [l_j, u_j]$. To prove that $I$ is a p-interpretation we must show that $\sum_{j=1}^{m} I(x_j) = 1$.

Indeed, $\sum_{j=1}^{m} I(x_j) = \sum_{j=1, j \neq i}^{m} I(x_j) + I(x_i) = \sum_{j=1}^{m} u_j - u_i + 1 - \sum_{j=1}^{m} u_j + u_i = 1$. This proves the reachability of $1 - \sum_{j=1}^{m} u_j + u_i = \max(l_i, 1 - \sum_{j=1}^{m} u_j + u_i)$, which in turn proves the theorem.

In particular, we are more interested in operations such as marginalization and conditioning for interval probability distributions. We proposed a way to perform those two operations. The ideas for conditioning an interval probability distribution is shown in the Figure 2.3, and consist of the following steps:

(i) First represent the original interval probability distribution in a set of point probability distributions by using p-interpretation. Note that it is an infinite set in most cases.

(ii) Then conditioning each point probability distribution on the given conditions to form a set of conditional probability distributions for point probabilities.

(iii) Finally compute both lower bound and upper bound for each row (event) in the probability distribution. For each event, we take the minimum and maximum as the lower bound and upper
bound, respectively.

A set of point PDs

$\{I | I \text{ satisfies } P\}$

$P$-Interpretation

Conditioning point PDs

$P'$-Interpretation

Generalization

A set of conditional point PDs

$\{I' | I' \text{ satisfies } P'\}$

Figure 2.3: A methodology for performing conditioning (or marginalization) operation on an interval probability distribution

This approach works perfectly with the marginalization operation. However, we note that Jaffray [54] has shown that conditioning interval probabilities is a dicey matter. As shown in Figure 2.3, the set of point probability distributions representing $P'$ will contain point probability distributions $I'$ which do not correspond to any $I$ representing $P$. The consequence is that the conditioning operation is no longer commutative. But conditioning operation is frequently used in applications, and our approach to conditioning, to our knowledge, gives the best possible solution for conditioning interval probability distributions. We include conditionalization in our interval probabilistic database framework with caution.

2.3 Bayesian Networks

In the last two decades, Bayesian networks have become a popular model for representing uncertain expert knowledge [50]. The Bayesian network approach combines two different mathematical theories: graph theory and probability theory. A Bayesian network is a compact, graphical representation of a joint probability distribution. It encodes the probabilistic relationships between random variables, and explicitly represents the conditional independence between random variables.

Consider a finite set $V = \{v_1, \ldots, v_n\}$ of discrete random variables where each variable $v_i$
takes on a value \( x_i \) from the domain \( \text{dom}(v_i) \). A Bayesian network over \( V \) consists of two components: a qualitative part which represents the network structure \( S \), and a quantitative part which contains a set of conditional probability distributions \( \mathcal{P} \) with each for one random variable. The network structure can be represented as a directed acyclic graph (DAG). That is, all edges in the graph are directed and there is no cycle when edge directions are followed. Nodes represent random variables in the specific application domain, and edges represent conditional dependencies among the variables. A conditional probability table (CPT) is associated with each node and describes the dependency between a node and its parents. A CPT lists the probability that the child node takes on each of its different values for each combination of values of its parents.

Formally, a conditional probability table (CPT) is defined as follows.

**Definition 2.8** A conditional probability table \( T \) is a tuple \( T = (v, \mathcal{D}, C) \), where

- \( v \) is the random variable for which \( T \) is defined;
- \( C = (v_1, v_2, \ldots, v_m) \) is the set of \( v \)'s parents in the Bayesian network;
- \( \mathcal{D} : \text{dom}(v) \times \text{dom}(C) \rightarrow [0, 1] \) is the conditional probability function, where \( \text{dom}(C) = \text{dom}(v_1) \times \ldots \times \text{dom}(v_m) \).

Now we are ready to define a Bayesian network.

**Definition 2.9** A Bayesian network \( BN \) is a tuple \( BN = (S, \mathcal{P}) \), where

- \( S = (V, E) \) is a directed acyclic graph, with (i) \( V = \{v_1, v_2, \ldots, v_n\} \) is a set of random variables in the Bayesian network. (ii) \( E = \{(u, v) | u, v \in V\} \) is a set of edges in the Bayesian network.
- \( \mathcal{P} : V \rightarrow T \) is a set of conditional probability tables, with each corresponding to a specific random variable in the network.

**Example 2.6** Figure 2.4 illustrates a fragment of a Bayesian network model for the W2W application domain. The random variable \text{CommunicationSkills} in the network has one parent node, which is the variable \text{Literacy}. Its CPT is shown in Figure 2.5, from which we can see that the variables \text{CommunicationSkills} and \text{Literacy} have a domain of values ‘Excellent’, ‘Good’, ‘So-so’, ‘Poor’ and a domain of values ‘Good’, ‘Fair’, ‘Poor’, respectively. Each row in the CPT represents a conditional probability distribution over the variable \text{CommunicationSkills} given that the variable \text{Literacy} takes a specific value from its domain.
Let a node $v_i$ in $\mathcal{S}$ denote the random variable $v_i$ and $\text{parent}(v_i)$ denotes the parent nodes of $v_i$, from which dependency edges come to the node $v_i$. The conditional distribution $\mathcal{P}$ of the variable $v_i$ can be represented as the following: $\mathcal{P} = \{P(v_i|\text{parent}(v_i))\}$. The joint probability function of random variables $\{v_1, \ldots, v_n\}$ in a Bayesian network $\mathcal{S}$ is calculated by the multiplication of local conditional probabilities of all the nodes. In this case, the joint probability function of $\{v_1, \ldots, v_n\}$ is given as follows: $P(v_1, \ldots, v_n) = \prod_{i=1}^{n} P(v_i|\text{parent}(v_i))$.

There are three main research problems in probabilistic reasoning using Bayesian networks: learning, inference and planning. The idea of learning with Bayesian networks is to construct Bayesian networks from available data by using some learning algorithms. Bayesian network inference involves computing the posterior marginal probabilities of some query nodes. There is also a recent development in planning with Bayesian networks (see a survey by Boutilier, et al. [9]).

### 2.4 Query Processing in RDBMS and in Mediators

A typical relational database management system is composed of a storage management layer, an access layer, a query processing layer, a transaction management layer, and a language and
interface layer, among which the query processing layer is the core of any relational database management systems. When an application client sends a SQL query to a RDBMS, the query processor gets the query from the upper layer, parses it, executes it and finally sends the result(s) back to the upper layer. The query optimizer is where the database makes decision among different execution plans that are equivalent. Although these different execution plans all implement the same query, they may differ greatly in terms of their execution efficiency. Good query optimization can reduce query execution time dramatically. A typical query processor for relational databases consists of the following components (as shown in Figure 2.6.(a)) (For detailed information, please read [40, 72]).

(a) Query processing in relational DBMS

(b) Query processing in mediator

Figure 2.6: Major components of query processor in relational DBMS and in mediator.

a) query parser. This component is responsible for syntax checking. It checks incoming SQL queries for correct syntax, and parses them into parse trees if everything goes fine.

b) query preprocessor. This component is responsible for semantic checking, i.e., to check
for correct relation name or attribute name. It also checks for possible view expansion if there are views involved in a query. A complete parse tree of relational algebra will be generated at the end of this stage.

   c) **logical query plan generator.** This component generates various logical query plans which are equivalent to the initial logical query plan according to some rewrite rules.

   d) **logical query plan selector.** This component chooses the best logical query plan based on the cost model chosen and statistics available, and possibly with the help of heuristics.

   e) **physical query plan enumerator.** According to the best logical plan given from the previous stage, this component transforms logical operators to physical operators. By choosing different orders of join operations, implementation methods of operators, etc., it enumerates all possible physical query plans.

   f) **physical query plan selector.** This component chooses the best physical query plan based on the cost model chosen and statistics available.

   g) **query execution engine.** This component executes the query based on the physical query plan passed to it.

Recently, a new type of data management software, middleware or mediators (i.e., [39, 79, 33, 61, 4]), has emerged. A mediator is software layer which facilitates the communication between application clients and application servers. It promotes loose coupling between the applications connecting to it. There are different architectures proposed for query processing in mediators because each serves different purpose. A mediator determines which mediator query operations need to be executed by the underlying RDBMS engine. It pushes as much computation as possible to the underlying RDBMS. These operations are then translated into a sequence of SQL statements. A mediator enumerates all possible combinations and chooses the best possible plan (including the order of executing SQL statements) based on the rewrite rules and cost model with the help of the statistics maintained by the mediator query engine. Then the final query plan, which is a sequence of SQL statements, is executed by the underlying RDBMS. Post-processing and assembly process are applied after the mediator gets the tuple streams from the underlying RDBMS. The common components (as shown in Figure 2.6.(b)) are summarized in the following.

   a) **query parser.** This component is responsible for syntax checking. It checks incoming mediator queries for correct syntax, and parses them into parse trees if everything goes well.

   b) **pre-plan generator.** This component generates various mediator query pre-plans which are equivalent to the initial mediator query plan according to the predefined rewrite rules.

   c) **query plan selector.** This component chooses the best mediator query plan based on the cost model chosen and statistics available, and possibly with the help of heuristics. The statistics
can be learned periodically through a thread to the underlying relational database.

d) query plan translator. According to the preferred mediator plan given from the previous stage, this component transforms each mediator operator to a sequence of SQL statements.

e) relational DBMS engine. This component executes the sequence of SQL statements which are passed to it from the previous stage.
Chapter 3

Semistructured Probabilistic Database Framework

The database framework described here has been designed specifically for storing and manipulating probability distributions with diverse structures. It has two major components: a Semistructured Probabilistic Object (SPO) data model and a Semistructured Probabilistic Algebra (SP-Algebra). The SPO data model allows one to represent probability distributions of different “shapes” along with associated information, and the SP-Algebra allows one to efficiently retrieve probabilistic information stored in the database. The probability distributions are handled correctly according to the laws of probability theory.

This chapter is organized as follows. Section 3.1 gives an overview of the database framework. In Section 3.2 we describe our SPO data model. We discuss our SP-Algebra and the semantics of SP-Algebra operations in Section 3.3 and Section 3.4, respectively. Finally, we introduce our work on query optimization and outline its major components: rewrite rules and cost models in Section 3.5.

3.1 Overview of the SPO framework

We start with the SPO data model.

Given a universe \( \mathcal{V} \) of random variables and a set \( \mathcal{R} \) of context attributes, a **Semistructured Probabilistic Object** (SPO) \( S \) over \( \langle \mathcal{V}, \mathcal{R} \rangle \) is a tuple \( S = \langle T, V, P, C, \omega \rangle \), where \( T \), the **context** of \( S \), is a relational tuple over a context schema \( R^* \) over \( \mathcal{R} \); \( V = \{ v_1, \ldots, v_q \} \subseteq \mathcal{V} \) is a set of **random variables** that participate in \( S \); \( P : \text{dom}(V) \rightarrow [0, 1] \) is the **probability table** of \( S \); \( C \) is called the **conditional** of \( S \); \( \omega \) is called a **path** expression, which represents the origin of an SPO.

A **semistructured probabilistic relation** (SP-relation) is defined as a collection of SPOs, and a **semistructured probabilistic database** (SP-database) is a collection of SP-relations.

SP-Algebra extends the definitions of standard relational operations and includes **selection** (\( \sigma \)), **projection** (\( \pi \)), **conditionalization** (\( \mu \)), **Cartesian product** (\( \times \)) and **join** (\( \bowtie \)) operations. Given an SPO \( S = \langle T, V, P, C, \omega \rangle \), the selection operation may query information stored in each of its components: context, participating variables, conditional information, probability table and probabilities. Each type of selection operation requires its own language of selection conditions. Given a selection condition on context, participating variables or conditionals, an SPO will either match the condition and be returned in the result of selection operation, or will not match and be omitted from
the result. Selection conditions that operate on the contents of the probability table (these are either conditions of the rows or on the probability values) are processed in a slightly different manner. Such selection operations return only the rows of the probability table that match the conditions, while keeping context and conditional parts intact.

The projection operation may apply to context, participating variables or conditionals. Projection on context or conditionals is similar to classical relational projection: it removes the unwanted context or conditional entries from the SPO. Given a list \( F \) of random variables, \( \pi_F(S) \), the projection on participating variables computes the marginal probability distribution of random variables \( F \cap V \) from the joint probability distribution for \( V \) stored in \( P \) (the context and conditional parts are preserved).

Given a constraint \( v = a \) where \( v \) is a random variable and \( a \in \text{dom}(v) \), conditionalization operation \( \mu_{\text{thd}.v=a}(S) \)\(^1\) computes from \( P \) the conditional probability distribution of random variables \( V - \{v\} \) given \( v = a \). The resulting SPO will have the same context, and the new condition \( \{v, \{a\}\} \) will be added to its conditional part.

Cartesian product and join compute joint probability distributions from the probability tables of the participating SPOs. Cartesian product is applicable only to a pair of SPOs which do not share common variables, while join applies only to a pair of SPOs that do share common variables.

In the sections below, we will describe in detail the new SPO data model and algebra operations. We will base our examples on the auto insurance risk analysis application described in Section 1.1.1.

### 3.2 Semistructured Probabilistic Object (SPO) Data Model

Consider a universe \( V \) of random variables \( \{v_1', \ldots, v_n'\} \). With each random variable \( v \in V \) we associate \( \text{dom}(v) \), the set of its possible values. Given a set \( V = \{v_1, \ldots, v_q\} \subseteq V \), \( \text{dom}(V) \) will denote \( \text{dom}(v_1) \times \cdots \times \text{dom}(v_q) \).

Let \( \mathcal{R} = (A_1, \ldots, A_n) \) be a collection of regular relational attributes. For \( A \in \mathcal{R} \), \( \text{dom}(A) \) will denote the domain of \( A \). We define a semistructured schema \( R^* \) over \( \mathcal{R} \) as a multiset of attributes from \( \mathcal{R} \). For example, if \( \mathcal{R} = \{\text{city}, \text{job}, \text{race}\} \), the following are valid semistructured schemas over \( \mathcal{R} \): \( R_1^* = \{\text{city}, \text{job}\} \); \( R_2^* = \{\text{city}, \text{city}, \text{job}, \text{race}\} \); \( R_3^* = \{\text{job}, \text{job}, \text{job}\} \).

Let \( \mathcal{P} \) denote a probability space used in the framework to represent probabilities of different events. Examples of such probability spaces include (but are not limited to) the interval \([0, 1]\) and

---

\(^1\)We use pre-\( x \) \( \text{cnt}, \text{var}, \text{end}, \text{thd}, \) or \( \text{pro} \) to specify a particular part of an object. \( \text{cnt}: \) context, \( \text{var}: \) participating variable, \( \text{end}: \) conditional, \( \text{thd}: \) probability table, and \( \text{pro}: \) probability.
the set $\mathbb{C}[0,1]$ of all subintervals of $[0, 1]$ [67, 23, 58]. For each probability space $\mathcal{P}$ there should exist a notion of a consistent probability distribution over $\mathcal{P}$.\footnote{For $\mathcal{P} = [0, 1]$ the consistency constraint states that the sum of probabilities in a complete probability distribution must add up to exactly 1.}

We are ready to define the key notion of our framework: Semistructured Probabilistic Objects (SPOs). We extended the initial SPO data model [20] by adding a fifth element called path expression in the current data model to specify the origin for each object. First we give a brief description of the new framework, then describe it in details in the sections that follow.

**Definition 3.1** A Semistructured Probabilistic Object (SPO) $S$ is defined as a tuple $S = (T, V, P, C, \omega)$, where

- $T$ is a relational tuple over some semistructured schema $R^*$ over $R$. We will refer to $T$ as the context of $S$.
- $V = \{v_1, \ldots, v_q\} \subseteq \mathcal{V}$ is a set of random variables that participate in $S$. We require that $V \neq \emptyset$.
- $P : \text{dom}(V) \rightarrow \mathcal{P}$ is the probability table of $S$. Note that $P$ need not be complete, but it must be consistent w.r.t. $\mathcal{P}$.
- $C = \{(u_1, X_1), \ldots, (u_s, X_s)\}$, where $\{u_1, \ldots, u_s\} = U \subseteq \mathcal{V}$ and $X_i \subseteq \text{dom}(u_i), 1 \leq i \leq n$, such that $V \cap U = \emptyset$. We refer to $C$ as the conditional of $S$.
- $\omega$, called the path expression, is an expression of Semistructured Probabilistic Algebra (SP-Algebra).

An explanation of this definition is in order. For our data model to possess the ability to store all the probability distributions mentioned in Section 1.1 (see Figure 1.1), the following information needs to be stored in a single object:

1. **Participating random variables.** These variables determine the probability distribution described in an SPO.

2. **Probability Table.** If only one random variable participates, it is a simple probability distribution table; otherwise the distribution will be joint. Probability table may be complete, when the information about the probability of every instance is supplied, or incomplete.

It is convenient to visualize the probability table $P$ as a table of rows of the form $(\bar{x}, \alpha)$, where $\bar{x} \in \text{dom}(V)$ and $\alpha = P(\bar{x})$. Thus, we will speak about rows and columns of the probability table where it makes explanations more convenient.
3. **Conditional.** A probability table may represent a distribution, conditioned by some prior information. The conditional part of its SPO stores the prior information in one of two forms: “random variable \( u \) has value \( x \)” or “the value of random variable \( u \) is restricted to a subset \( X \) of its values”. In our definition, this is represented as a pair \((u, X)\). When \( X \) is a singleton set, we get the first type of the condition.

4. **Context.** It provides supporting information for a probability distribution – information about the known values of certain parameters, which are not considered to be random variables by the application.

5. **Origin** or **path.** Each SPO in an SP-Database can either be inserted into the database directly, or can be a result of one or more SP-Algebra operations over already existing SPOs. When an SPO is inserted into the database, a unique identifier is assigned as its path. Whenever an SPO is created as a result of an SP-Algebra operation, its path is extended by the description of this operation. An SPO inserted into the database is called a base SPO. In Section 3.3 we introduce the syntax for complex path expressions that are formed when SP-Algebra operations are performed on SPOs.

Intuitively, a Semistructured Probabilistic Object represents a (possibly complex) probability distribution and the information associated with it. The actual distribution is described by the participating random variables and probability table parts of the object. The conditional part, when non-empty, indicates that the object represents a conditional probability distribution and specifies the conditions. The context contains any non-stochastic information associated with the distribution. Finally its path tells us how this object have been constructed. If the path is atomic (single unique identifier), than the object had been constructed from scratch and inserted into the database. Complex paths indicate which database objects participated in its creation and what SP-Algebra operations have been applied to obtain it. As examples throughout the dissertation will show, knowing how an object was constructed may help in the interpretation of its probability table.

**Example 3.1** Consider the joint probability distribution of risk level based on *Driver’s Age* (DA) and *License Years* (LY) for *Asian* drivers in *Lexington* who had either a severe or medium accident within the last 3 years, as defined in Figure 1.1. An SPO representing this probability distribution is shown in Figure 3.1.

We can break this information into our five constituent parts as follows:

- **participating random variables:** \( V = \{DA, LY\} \).
Figure 3.1: Joint Probability Distribution of risk on Driver’s Age and License Years for Asian drivers in Lexington.

**probability table:** as shown in Figure 3.1, the probability table defines a complete and consistent probability distribution.

**conditional:** there is a single conditional $DR \in \{A, B\}$ associated with this distribution, which is stored in an SPO as $C = \{(DR, \{A, B\})\}$.

**context:** information about where the driver lives and the driver’s race is not represented by random variables in our universe. They are, therefore, represented as relational attributes city and race, respectively. Thus, city: Lexington and race: Asian are the context of the probabilistic information in this example.

**path:** assuming that this information is being added to the database, we associate with this SPO a unique identifier $S1$ that will serve as its path.
3.3 Semistructured Probabilistic Algebra

Let us fix the universe of random variables \( \mathcal{V} \), the universe of context attributes \( \mathcal{R} \), and the probability space \( \mathcal{P} \). In the remainder of this section we will assume that \( \mathcal{P} = [0, 1] \).

A finite collection \( \mathcal{S} = \{S_1, \ldots, S_n\} \) of semistructured probabilistic objects over \( \langle \mathcal{V}, \mathcal{R}, \mathcal{P} \rangle \) is called a semistructured probabilistic relation (SP-relation). A finite collection \( \mathcal{D}_\mathcal{S} = \{S_1, \ldots, S_m\} \) is called a semistructured probabilistic database (SP-database).

One important difference between semistructured probabilistic databases and classic relational or relational probabilistic databases is that each table in a relational database has a specified schema, whereas all SP-relations are “schema-less”: any collection of SPOs can form an SP-relation. Thus, the division of a semistructured probabilistic database into relations is a matter of the logic of a particular application. For example, if the SP-database is built from the information supplied by three different experts, this information can be arranged into three semistructured probabilistic relations according to the origin of each object inserted in the database. Alternatively, the information can be arranged in SP-relations by the date it was obtained.

The key to the successful management of probabilistic information is how we can retrieve the stored information in SPDs efficiently. In probabilistic relational databases, Barbara et al. [5], Dey and Sarkar [25] and Lakshmanan et al. [58] define probabilistic relational algebras by extending the classical relational algebra. They add probability-specific (and probability theory compliant) manipulation of the probabilistic attributes in the relations. We also define a new semistructured probabilistic algebra for SPDs, in order to capture properly the manipulation of probabilities.

In the remainder of this section we introduce such an algebra, called the Semistructured Probabilistic Algebra (SP-Algebra). This algebra contains three standard set operations, union, intersection and difference and extends the definitions of standard relational operation selection, projection, Cartesian product and join to account for the appropriate management and maintenance of probabilistic information within SPOs. In addition, a new operation, conditionalization (see also [25]), is defined in the SP-Algebra. This operation is specific to the probabilistic databases and results in the construction of SPOs that represent conditional probability distributions of the input SPOs.

Before proceeding with the description of individual operations, we need to make an important distinction among the notions of equality, equivalence and containment of SPOs. Two SPOs \( S \) and \( S' \) are equal if all their components, including the paths are equal. At the same time, only the first four components of any SPO, context, participating variables, probability table and conditional information, represent the real content of the object. The path merely records how the object was obtained in the database. It is possible to obtain, as a result of SP-Algebra operations, two SPOs
with the same first four components but different paths. Such objects will not, technically, be equal. However, they would represent exactly the same probability distribution, and in many cases, we could substitute one such object with another without any loss. We reserve the notion of equivalence of SPOs for such situations. In some cases, only three components of two SPOs, context, participating variables and conditional information, are exactly same, but all rows in the probability table in one SPO are also contained in the probability table in the other. We use the notion of containment for such cases.

**Definition 3.2** Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ be two SPOs. $S$ is equivalent to $S'$, denoted $S \equiv S'$ iff $T = T'$, $V = V'$, $P = P'$ and $C = C'$.

The notion of equivalence can be extended to semistructured probabilistic relations.

**Definition 3.3** Let $S$ and $S'$ be two SP-relations. Then $S$ is equivalent to $S'$, denoted $S \equiv S'$, iff $\forall S \in S$ there exists at least one equivalent SPO in $S'$, and $\forall S' \in S'$ there exists at least one equivalent SPO in $S$.

**Definition 3.4** Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ be two SPOs. $S$ is contained in $S'$, denoted $S \subseteq S'$ iff $T = T'$, $V = V'$, $C = C'$, and all rows in $P$ are also contained in $P'$.

**Proposition 3.1** Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ be two SPOs. If $S$ is equivalent to $S'$, then $S$ is contained in $S'$ ($S \subseteq S'$) and $S'$ is contained in $S$ ($S' \subseteq S$).

3.3.1 Set Operations

Semistructured Probabilistic relations are sets of SPOs. Because of it, the definitions of union, intersection and difference of SP-relations are straightforward.

**Definition 3.5** Let $S$ and $S'$ be two SP-relations. Then,

- Union: $S \cup S' = \{ S | S \in S$ or $S \in S' \}$.
- Intersection: $S \cap S' = \{ S | S \in S$ and $S \in S' \}$.
- Difference: $S - S' = \{ S | S \in S$ and $S \notin S' \}$.
- Containment Elimination: $\delta(S) = \{ S | S \in S, \forall S' \in S - \{ S \}, \neg S \subseteq S' \}$.
We note two features of the set operations in SP-Algebra. Classical relational algebra has a restriction on applicability of the set operations: they are defined only on pairs of relations with matching schemas. Because SP-relations are schema-less and represent logical rather than syntactic grouping of probability distributions in an SP-database, set operations are applicable to any pair of SP-relations. Another note is that set operations do not leave their imprint on the path component of individual SPOs.

3.3.2 Selection

Given an SPO $S = \langle T, V, P, C, \omega \rangle$, a selection query may be issued to any part of the object, save for the path. Each part requires its own language of selection conditions.

Given an SPO $S$, selection on context, participating random variables and conditionals will result in either $S$ being selected or not in its entirety (as is the case with selection on classical relations). It is also possible to select a subset of rows of the probability table based either on the values of random variables or on the probability values of individual rows in the probability table. Such selection operations may return only part of the original probability table $P$, while keeping context, conditionals and participating random variables intact. For any selection operation, the path expression of the result will be updated to indicate the selection operation.

The five different types of selections are illustrated in the following example.

Example 3.2 Consider the SPO $S$ described in Example 3.1. Five different types of selection queries are illustrated below.

1. “Find all probability distributions related to Asian drivers.” $S.T$ contains tuple race : Asian, therefore $S$ matches the selection condition.

2. “Find all probability distributions that involve the Driver Age aspect.” As DA is one of the participating random variables of $S$, $S$ matches the selection condition.

3. “Find all probability distributions related to drivers who had a medium or severe accident within the last 3 years.” The conditional part of $S$ contains expression $DR \in \{A, B\}$ which matches the selection condition (“medium or severe accident within the last 3 years”).

4. “What information is available about the risk when granting insurance to a driver with less than one year of driving experience?” $S.P$ contains four entries that relate to the probability for drivers with less than one year of driving experience. This part of $S.P$ should
be returned as a result together with the $T, V$ and $C$ parts of $S$. The remainder of the $S.P$ should not be returned.

As an alternative, consider the query, “what is the risk when granting insurance to a 30-year-old driver with 5 years of driving experience?” The answer to this query on $S$ contains only one line from $S.P$, for the appropriate information of drivers.

5. “What outcomes have probability over 0.1?” In the probability table of $S$, there are five possible outcomes that have the probability greater than 0.1. In the result of executing this query on $S$, $S.P$ should contain exactly these five rows, with $S.T$, $S.V$ and $S.C$ remaining unchanged.

a. Selection on Context, Participating variables or Conditionals

Here, we define the three selection operations that do not alter the content of the selected objects. We start by defining the acceptable languages for selection conditions for the three types of selects.

Recall that the universe $\mathcal{R}$ of context attributes consists of a finite set of attributes $A_1, \ldots, A_n$ with domains $\text{dom}(A_1), \ldots, \text{dom}(A_n)$. With each attribute $A \in \mathcal{R}$ we associate a set $\text{Pr}(A)$ of allowed predicates. We assume that equality and inequality are allowed for all $A \in \mathcal{R}$.

**Definition 3.6**

1. An atomic context selection condition is an expression $c$ of the form $A Q x$ ($Q(A, x)$), where $A \in \mathcal{R}$, $x \in \text{dom}(A)$ and $Q \in \text{Pr}(A)$.

2. An atomic participation selection condition is an expression $c$ of the form $v \in V$, where $v \in \mathcal{V}$ is a random variable.

3. An atomic conditional selection condition is one of the following expressions: $u = \{x_1, \ldots, x_h\}$ or $u \ni x$ where $u \in \mathcal{V}$ is a random variable and $x, x_1, \ldots, x_h \in \text{dom}(u)$.

Complex selection conditions can be formed as Boolean combinations of atomic selection conditions.

**Definition 3.7** Let $S = (T, V, P, C, \omega)$ be an SPO and let $c : Q(A, x)$ be an atomic context selection condition. Let $\omega' = \sigma_c(\omega)$ and let $S' = (T, V, P, C, \omega')$. Then $\sigma_c(S) = \{S'\}$ iff

- $A \in S.T$;
- For some instance $A^*$ of $A$ in $T$, $(S.T.A^*, x) \in Q$;
otherwise \( \sigma_c(S) = \emptyset \).

**Definition 3.8** Let \( S = \langle T, V, P, C, \omega \rangle \) be an SPO and let \( c : v \in V \) be an atomic participation selection condition. Let \( \omega' = \sigma_c(\omega) \) and let \( S' = \langle T, V, P, C, \omega' \rangle \). Then \( \sigma_c(S) = \{S'\} \iff v \in V \).

**Definition 3.9**

1. Let \( S = \langle T, V, P, C, \omega \rangle \) be an SPO and let \( c : u = \{x_1, \ldots, x_h\} \) be an atomic conditional selection condition. Let \( \omega' = \sigma_c(\omega) \) and let \( S' = \langle T, V, P, C, \omega' \rangle \). Then \( \sigma_c(S) = \{S'\} \iff C \ni (u, X) \) and \( X = \{x_1, \ldots, x_h\} \).

2. Let \( c : u \ni x \) be an atomic conditional selection condition. Then \( \sigma_c(S) = \{S'\} \iff C \ni (u, X) \) and \( X \ni x \).

**b. Selection on Probability Table or Probabilities**

Selection operations considered in the previous sections were simple in that their result on a semistructured probabilistic relation was always a subset of the relation.

The two types of selections introduced here are more complex. The result of each operation applied to an SPO can be a non-empty part of the original SPO. In particular, both operations preserve the context, participating random variables and conditionals in an SPO, but may return only a subset of the rows of the probability table. In both selection on probability table and selection on probabilities, the selection condition will indicate which rows are to be included and which are to be omitted.

**Definition 3.10** An atomic probabilistic table selection condition is an expression of the form \( v = x \) where \( v \in V \) and \( x \in \text{dom}(v) \). Probabilistic table selection conditions are Boolean combinations of atomic probabilistic table selection conditions.

**Definition 3.11** Let \( S = \langle T, V, P, C, \omega \rangle \) be an SPO, \( V = \{v_1, \ldots, v_k\} \) and let \( c : v = x \) be an atomic probabilistic table selection condition.

If \( v \in V \), then (assume \( v = v_i, 1 \leq i \leq k \)), the result of selection from \( S \) on \( c, \sigma_c(S) \) is a semistructured probabilistic object \( S' = \langle T, V, P', C, \omega' \rangle \), where

\[
P'(y_1, \ldots, y_1, \ldots, y_k) = \begin{cases} 
P(y_1, \ldots, y_1, \ldots, y_k) & \text{if } y_1 = x; \\ 
\text{undefined} & \text{if } y_1 \neq x, 
\end{cases}
\]

and \( \omega' = \sigma_c(\omega) \).
Definition 3.12 An atomic probabilistic selection condition is an expression of the form \( P \circ \circ \alpha \), where \( \alpha \in [0, 1] \) and \( \circ \circ \in \{=, \neq, \leq, \geq, <, >\} \). Probabilistic selection conditions are Boolean combinations of atomic probabilistic selection conditions.

Definition 3.13 Let \( S = \langle T, V, P, C, \omega \rangle \) be an SPO and let \( c : P \circ \circ \alpha \) be an atomic probabilistic selection condition. Let \( \tilde{x} \in \text{dom}(V) \). The result of selection from \( S \) on \( c \) is defined as follows:

\[
\sigma_{P \circ \circ \alpha}(S) = S' = \langle T, V', P', C, \omega' \rangle,
\]

where

\[
P'(\tilde{x}) = \begin{cases} 
P(\tilde{x}) & \text{if } P(\tilde{x}) \circ \circ \alpha; \\
\text{undefined} & \text{otherwise,}
\end{cases}
\]

and \( \omega' = \sigma_c(\omega) \).

Example 3.3 Figure 3.2 shows two examples of selection queries on an SPO. The central object is obtained from the original SPO (left) as the result of the query, “Find all information about the risk when granting insurance to a 19-year-old driver”, denoted \( \sigma_{DA=A}(S) \). In the probability table of the resulting SPO, only the rows that have the value of the \( DA \) random variable equal to \( A \) remain.

![Figure 3.2: Selection on Probabilistic Table and on Probability values in SP-Algebra](image)

The rightmost object in the figure is the result of the query “Find all combinations whose probability is greater than 0.11.” This query can be written as \( \sigma_{P>0.11}(S) \). The probability table of the resulting object will contain only those rows from the original probability table where the probability value was greater than 0.11.

Proposition 3.2 Let \( c \) be an arbitrary selection condition and let \( S \) be an SPO. Then \( \sigma_c(S) \subseteq S \).
SP-Algebra operations can be extended to a semistructured probabilistic relation, as described in the following proposition.

**Proposition 3.3** Any SP-Algebra operation on a semistructured probabilistic relation is equivalent to the union of the SP-Algebra operation on each SPO in the SP-relation.

- Let $\mathcal{S}$ be a semistructured probabilistic relation and $\gamma$ be one of the three unary SP-Algebra operators. Then $\gamma(\mathcal{S}) = \bigcup_{\mathcal{S}\in\mathcal{S}} (\gamma(\mathcal{S}))$.

- Let $\mathcal{S}$ and $\mathcal{S}'$ be two semistructured probabilistic relations and $\otimes$ be one of the two binary SP-Algebra operators. Then $\mathcal{S} \otimes \mathcal{S}' = \bigcup_{\mathcal{S}\in\mathcal{S}} \bigcup_{\mathcal{S}'\in\mathcal{S}'} (\mathcal{S} \otimes \mathcal{S}')$.

The semantics of atomic selection conditions can be extended to their Boolean combinations in a straightforward manner: $\sigma_{c\land c'}(\mathcal{S}) \equiv \sigma_c(\sigma_c(\mathcal{S}))$ and $\sigma_{c\lor c'}(\mathcal{S}) \equiv \sigma_c(\mathcal{S}) \lor \sigma_c(\mathcal{S})$.

The interpretation of negation in the context selection condition requires some additional explanation. In order for a selection condition of a form $\neg Q(A, x)$ to succeed on some SPO $\mathcal{S} = (T, V, P, C, \omega)$, attribute $A$ must be present in $\mathcal{S}$. If $A$ is not in $\mathcal{S}$, the selection condition does not get evaluated and the result will be $\emptyset$. Therefore, the statement $\mathcal{S} \in \sigma_c(\mathcal{S}) \lor \mathcal{S} \in \sigma_{\neg c}(\mathcal{S})$ is not necessarily true. This also applies to conditional selection conditions.

Different selection operations commute, as shown in the following theorem.

**Theorem 3.1** Let $c$ and $c'$ be two selection conditions and let $\mathcal{S}$ be a semistructured probabilistic relation. Then $\sigma_c(\sigma_{c'}(\mathcal{S})) \equiv \sigma_{c'}(\sigma_{c}(\mathcal{S}))$.

**Proof of Theorem 3.1**

Let $c$ and $c'$ be two atomic selection conditions. There are 5 types of atomic selection conditions, namely context, participation, conditional, table and probability. Selection on context, participation or conditional will result in entire SPOs being selected, while selection on table or probability will select only parts of the relevant SPOs. We could partition the conditions into two groups,

- Group I, containing context, participation and conditional conditions, and

- Group II, containing table and probability conditions.

First we prove that $\sigma_c(\sigma_{c'}(\mathcal{S})) \equiv \sigma_{c'}(\sigma_{c}(\mathcal{S}))$ for a single SPO $\mathcal{S}$.

**Case 1.** Both conditions $c$ and $c'$ are in Group I.
There are four possible combinations for whether each condition is satisfied:

a) $S$ satisfies $c$ but not $c'$, or
b) $S$ satisfies $c'$ but not $c$, or
c) $S$ satisfies both $c$ and $c'$, or
d) $S$ does not satisfy either $c$ or $c'$.

By the definition of selection on atomic selection conditions in Group $I$, we know selection on these conditions will result in the entire SPO being selected, or none of it.

For case (a), since $S$ does not satisfy $c'$, $\sigma_c(S)$ returns empty and subsequently $\sigma_c(\sigma_{c'}(S))$ will return empty. Since $\sigma_c(S)$ returns $S$, we see that $\sigma_{c'}(\sigma_c(S)) = \sigma_{c'}(S)$ will also return empty for the same reason. Thus, $\sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S))$ holds for case (a). The same applies to case (b). Similarly, for case (d). For case (c), $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(S)$ returns $S$, and $\sigma_{c'}(\sigma_c(S)) = \sigma_{c'}(S)$ returns $S$ too. Thus, $\sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S))$ holds for case (c).

So $\sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S))$ holds for all these cases.

**Case 2.** Condition $c$ is in Group $I$ and condition $c'$ is in Group $II$.

There are only two possible combinations for whether each condition is satisfied, assuming that condition $c'$ is always partially satisfied:

a) $S$ does not satisfy $c$, or
b) $S$ satisfies $c$.

By the definition of selection on atomic selection conditions in both Group $I$ and Group $II$, we know selection on conditions in Group $I$ will result in the entire SPO being selected or not, while selection on conditions in Group $II$ will preserve all the context, participating random variables and conditionals in the original SPO, but produce only a part of the probability table.

Let $\sigma_{c'}(S) = S'$, where $S'$ has the part of the probability table that satisfies the condition $c'$ and retains all the context, participating random variables and conditionals in $S$.

For case (a), $\sigma_c(\sigma_{c'}(S)) = \sigma_c(S')$ will return empty since $S'$ does not satisfy the condition $c$ either. Since $\sigma_c(S)$ returns $S$, we see that $\sigma_{c'}(\sigma_c(S)) = \sigma_{c'}(S)$ will also return empty. This proves $\sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S))$ for case (a). For case (b), $\sigma_c(\sigma_{c'}(S)) = \sigma_c(S')$ will return $S'$, since $S'$ should also satisfy the condition $c$. Since $\sigma_c(S)$ returns $S$, so $\sigma_{c'}(\sigma_c(S)) = \sigma_{c'}(S)$ will also return $S'$. This proves that $\sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S))$ holds for case (b).

So $\sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S))$ holds for both cases.

**Case 3.** Both $c$ and $c'$ are conditions in Group $II$.

Assume that both conditions $c$ and $c'$ are partially satisfied by $S$. By the definition of selection
on atomic selection conditions in Group $II$, we know that selection on these conditions will result in part of the probability table and will preserve all the context, participating random variables and conditionals in the original SPO. In other words, all the components in the original SPO except the probability table and the path will be preserved.

Let $S = \langle T, V, P, C, \omega \rangle$. Then $S' = \sigma_c(\sigma_{c'}(S)) = \langle T, V, P', C, \omega' \rangle$ and $S'' = \sigma_{c'}(\sigma_c(S)) = \langle T, V, P'', C, \omega'' \rangle$ with $P' = \varrho_c(\varrho_{c'}(P))$ and $P'' = \varrho_{c'}(\varrho_c(P))$, where $\varrho$ is the relational selection operator. Since the relational selection operator $\varrho$ is commutative, $\varrho_c(\varrho_{c'}(P)) = \varrho_{c'}(\varrho_c(P))$. Therefore we have $P' = P''$ or $S' \equiv S''$. So $\sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S))$ holds for this case.

This proves $\sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S))$ for a single SPO. Now let an SP-relation $S = \cup_{S \in \mathcal{S}}(S)$. Since the union operator is commutative, i.e., $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S)) = \cup_{S \in \mathcal{S}}(\sigma_c(\sigma_{c'}(S)))$ and $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S)) = \cup_{S \in \mathcal{S}}(\sigma_c(\sigma_{c'}(S))) \equiv \cup_{S \in \mathcal{S}}(\sigma_{c'}(\sigma_c(S)))$.

This proves $\sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S)).$ □

### 3.3.3 Projection

Just as with selection, the results of projection operation differ, depending on which parts of an SPO are to be projected out. Projections on context and conditionals are similar to the traditional relational algebra projection: either a context attribute or a conditional is removed from an SPO object, which does not change otherwise. These operations change the semantics of the SPO and thus must be used with caution. (Note that removing attributes from the relations in a relational database system also changes the semantics of the data.)

**Definition 3.14** Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO and let $\mathcal{L} \subset \mathcal{A}$ be a set of context attributes. The projection of $S$ onto $\mathcal{L}$, denoted $\pi_{\mathcal{L}}(S)$ is an SPO $S' = \langle T', V, P, C, \omega' \rangle$ where

- $T' = \{(A, x) | (A, x) \in T, A \in \mathcal{L}\}$, i.e., $T'$ contains all entries from $T$ for attributes from the list $\mathcal{L}$ only.
- $\omega' = \pi_{\mathcal{L}}(\omega)$.

**Definition 3.15** Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO and let $\mathcal{F}$ be a set of conditionals. The projection of the conditional part of $S$ onto $\mathcal{F}$, denoted $\pi_{c: \mathcal{F}}(S)$ is an SPO $S' = \langle T, V, P, C', \omega' \rangle$ where

- $C' = \{(u, X) | (u, X) \in C, u \in \mathcal{F}\}$.  

\(^3\)The list of variables is also used in the projection onto the participating random variables. The syntax $c : \mathcal{F}$ is chosen to distinguish between the two types of projection operations.
• $\omega' = \pi_{\mathcal{F}\omega}(\omega)$.

A somewhat more difficult and delicate operation is the projection on the set of participating random variables. Removal of a random variable from the SPO’s participant set entails that information related to this random variable has to be removed from the probability table as well. Informally, this corresponds to removing one random variable from consideration in a joint probability distribution, which is usually called marginalization. The result of this operation is a new marginal probability distribution that needs to be stored in the probability table component of the resulting SPO.

This computation is performed in two steps. First, the columns for random variables that are to be projected out are removed from the probability table. In the remainder of the table, there can now be duplicate rows: rows that have all values (except for the probability value) coincide. All duplicate rows of the same type are then collapsed (coalesced) into one, with the new probability value computed as the sum of the values in the collapsed rows.

A formal definition of this procedure is given below.

**Definition 3.16** Let $S = \langle T, V, P, C, \omega \rangle$ be an SPO, $V = \{v_1, \ldots, v_q\}$, $q > 1$ and $\mathcal{L}_V \subset V$, $\mathcal{L}_V \neq \emptyset$. The projection of $S$ on $\mathcal{L}_V$, denoted $\pi_{\mathcal{L}_V}(S)$, is defined to be an object $S' = \langle T, \mathcal{L}_V, P', C, \omega' \rangle$ where $P' : \text{dom}(\mathcal{L}_V) \rightarrow [0,1]$ and for each $\bar{x} \in \text{dom}(\mathcal{L}_V)$,

$$P' (\bar{x}) = \sum_{\bar{y} \in \text{dom}(V - \mathcal{L}_V) : P(\bar{x}, \bar{y}) \text{ is defined}} P(\bar{x}, \bar{y})$$

and $\omega' = \pi_{\mathcal{L}_V}(\omega)$.

Notice that projection on the set of participants is allowed only if the set of participants is not a singleton and if at least one random variable remains in the resulting set.

**Example 3.4** Figure 3.3 illustrates how projection on the set of participating random variables works. First, the columns of random variables to be projected out are removed from the probability table (step I). Next, the remaining rows are coalesced (step II). After the Vehicle Type (VT) random variable has been projected out, the interim probability table has three rows $(B, A)$ with probabilities 0.07, 0.02 and 0.04 respectively. These rows are combined into one row with probability value set to $0.07 + 0.02 + 0.04 = 0.13$. Similar operations are performed on the other rows.
### 3.3.4 Conditionalization

Conditionalization is an operation specific to probabilistic algebras. Similarly to the projection operation, conditionalization reduces the probability distribution table. The difference is that the result of conditionalization is a conditional probability distribution. Given a joint probability distribution, conditionalization answers the following general query, "What is the probability distribution of the remaining random variables if the value of some random variable $v$ in the distribution is restricted to subset $X$ of its values?"

Informally, the conditionalization operation proceeds on a given SPO as follows. The input to the operation is one participating random variable of the SPO, $v$, and a subset of its values $X$. The first step of conditionalization consists of removing from the probability table of the SPO all rows whose $v$ values are not from the set $X$. Then the $v$ column is removed from the table. The remaining rows are coalesced (if needed) in the same manner as in the projection operation and the probability values are normalized. Finally, $(v, X)$ is added to the set of conditionals of the resulting SPO.

The formal definition of conditionalization is given below. Note that if the original table is incomplete, there is no meaningful way to normalize a conditionalized probability distribution. Thus, we restrict this operation to situations where normalization is well-defined.
Definition 3.17  An SPO \( S = \langle T, V, P, C, \omega \rangle \) is **conditionalization-compatible** with an atomic conditional selection condition \( v = \{x_1, \ldots, x_h\} \) iff

- \( v \in V \);
- \( P \) on \( \{x_1, \ldots, x_h\} \) for \( v \) is a complete function.

Definition 3.18  Let \( S = \langle T, V, P, C, \omega \rangle \) be an SPO which is conditionalization-compatible with an atomic conditional selection condition \( c : v = \{x_1, \ldots, x_h\} \).

The result of **conditionalization** of \( S \) by \( c \), denoted \( \mu_c(S) \), is defined as follows:

\[
\mu_c(S) = \langle T', V', P', C', \omega' \rangle,
\]

where

- \( V' = V - \{v\} \);
- \( C' = C \cup \{(v, \{x_1, \ldots, x_h\}\)};
- \( P' : \text{dom}(V') \longrightarrow [0, 1] \).

Let

\[
N = \sum_{\vec{y} \in \text{dom}(V')} \sum_{\vec{x} \in \{x_1, \ldots, x_h\}} P(\vec{x}, \vec{y}).
\]

For any \( \vec{y} \in \text{dom}(V') \),

\[
P'(\vec{y}) = \frac{\sum_{\vec{x} \in \{x_1, \ldots, x_h\}} P(\vec{x}, \vec{y})}{N};
\]

- \( \omega' = \mu_c(\omega) \).

Conditionalization can be extended to a semistructured relation in a straightforward manner. Given a relation \( S \), \( \mu_c(S) \) will consist of \( \mu_c(S) \) for each \( S \in S \) that is conditionalization-compatible with \( c \). SPOs not conditionalization-compatible with \( c \) will not be included in \( \mu_c(S) \).

**Example 3.5**  Consider the SPO \( S \) defined in Example 3.1 describing the joint probability distribution of risk on Driver’s Age (DA) and License Years (LY) for Asian drivers in Lexington who had either a severe or medium accident within the last 3 years. We derive the probability distribution for drivers with less than one year of driving experience. Figure 3.4 depicts the work of the conditionalization operation \( \mu_{\text{tbl.LY=A}}(S) \). The original object is shown to the left. As \( S.P \) is a complete distribution, \( S \) is conditionalization compatible with \( \text{LY = A} \). The first step of conditionalization...
consists of removing all rows that do not satisfy the conditionalization condition from $S.P$ (result depicted in the center). Then, on step II, the $LY$ column is dropped from the table, probability values in the remaining rows are normalized and $LY = A$ is added to the list of conditionals. The rightmost object in Figure 3.4 shows the final result.

<table>
<thead>
<tr>
<th>$\omega$: S race: Asian</th>
<th>$\omega$: $\mu_{LY=A}(S)$ (step I) race: Asian</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA</td>
<td>LY</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>A</td>
</tr>
<tr>
<td>F</td>
<td>B</td>
</tr>
<tr>
<td>F</td>
<td>C</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>DR $\in { A, B }$</td>
<td>DR $\in { A, B }$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\omega$: $\mu_{LY=A}(S)$ (step II) race: Asian</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>F</td>
</tr>
<tr>
<td>DR $\in { A, B }$</td>
</tr>
</tbody>
</table>

Figure 3.4: Conditionalization in SP-Algebra.

### 3.3.5 Cartesian Product

Sometimes an SP database has only simple probability distributions for some random variables. In order to get a joint probability distribution, either a Cartesian product or join operation has to be performed on the SPOs storing these distributions. Intuitively, a Cartesian product or join of two probabilistic distributions is the joint probability distribution of random variables involved in both original distributions. Cartesian product is defined only on pairs of compatible SPOs. Here, we will restrict ourselves to the assumption of independence between the probability distributions in Cartesian products. This restriction allows us to represent the result as a point probability distribution⁴.

---

⁴In general, given two events $a$ and $b$ and their point probabilities $p(a)$ and $p(b)$, the probability $p(a \land b)$ of their conjunction lies in the interval $[\max(0, p(a) + p(b) - 1), \min(p(a), p(b))]$. One can obtain a point probability for
Two SPOs are *compatible* for Cartesian product if their participating variables are disjoint, but their conditionals coincide. When the sets of participating variables are not disjoint, we will use the join operation instead of Cartesian product to find, for instance, the Driver's Age joint probability distribution.

**Definition 3.19** Two SPOs \( S = \langle T, V, P, C, \omega \rangle \) and \( S' = \langle T', V', P', C', \omega' \rangle \) are Cartesian product-compatible (cp-compatible) *iff* \( V \cap V' = \emptyset \) and \( C = C' \).

We can now define the Cartesian product.

**Definition 3.20** Let \( S = \langle T, V, P, C, \omega \rangle \) and \( S' = \langle T', V', P', C', \omega' \rangle \) are two cp-compatible SPOs. Then, the result of their Cartesian product (under assumption of independence), denoted \( S \times S' \), is defined as follows:

\[
S \times S' = S'' = \langle T'', V'', P'', C'', \omega'' \rangle,
\]

where

- \( T'' = \langle T, T' \rangle \);
- \( V'' = V \cup V' \);
- \( P'' : \text{dom}(V'') \to [0, 1] \).

For all \( \bar{z} \in \text{dom}(V'') \); \( \bar{z} = (\bar{x}, \bar{y}) \); \( \bar{x} \in \text{dom}(V) \), \( \bar{y} \in \text{dom}(V') \):

\[
P''(\bar{z}) = P(\bar{x}) \cdot P'(\bar{y}).
\]

- \( C'' = C = C' \).
- \( \omega'' = \omega \times \omega' \).

**3.3.6 Join**

Join is also defined only on pairs of compatible SPOs. Two SPOs are *join-compatible* if they share some participating variables (these will be the “join attributes”) and their conditionals coincide.
Definition 3.21 Two SPOs $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ are join-compatible iff $V \cap V' \neq \emptyset$ and $C = C'$.

Given two join-compatible SPOs $S$ and $S'$, we can break the set $V \cup V'$ into three non-empty disjoint parts: $V_1 = V - V', V'_1 = V' - V$ and $V_c = V \cap V'$. The information about the probability distribution of random variables in $V_c$ can be found in both $S$ and $S'$. The join operation must take this into consideration when the joint probability distribution for variables in $V \cup V'$ is computed. The key to computing the joint distribution correctly is the following statement.

Based on probability theory, we give a new definition of join operation in the current framework. We want the join of $S$ and $S'$ to contain the joint probability distribution of the set $V_1 \cup V_c \cup V'_1$. Since $P(\bar{y})$ could be obtained either from $S$ or from $S'$, there exist two families of join operations, called left join, $\Join$, and right join, $\Join$, with the following definitions. The only difference between the two join operations is the probability distribution.

Definition 3.22 Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C', \omega' \rangle$ be two join-compatible SPOs. Let $V = V_1 \cup V_c$ and $V' = V'_1 \cup V_c$, i.e., $V_c = V \cap V'$. We define the operations of left join of $S$ and $S'$, denoted $S \Join S'$, and right join of $S$ and $S'$, denoted $S \Join S'$ as follows:

$$S \Join S' = S'' = \langle T'', V'', P'', C'', \omega'' \rangle;$$

$$S \Join S' = S''' = \langle T''', V''', P''', C''', \omega''' \rangle,$$

where

- $T'' = T \cup T'$;
- $V'' = V_1 \cup V_c \cup V'_1$;
- $P'', P''' : \text{dom}(V'') \rightarrow [0, 1]$.

For all $\bar{w} \in \text{dom}(V'')$: $\bar{w} = (\bar{x}, \bar{y}, \bar{z})$; $\bar{x} \in \text{dom}(V_1)$, $\bar{y} \in \text{dom}(V_c)$, $\bar{z} \in \text{dom}(V'_1)$:

$$P''(\bar{w}) = P(\bar{x}, \bar{y}) \cdot P'(\bar{y}, \bar{z}) / P'(\bar{y});$$

$$P'''(\bar{w}) = P(\bar{x}, \bar{y}) \cdot P'(\bar{y}, \bar{z}) / P(\bar{y}).$$

- $C'' = C = C'$.
- $\omega'' = \omega \Join \omega'$; $\omega''' = \omega \Join \omega'$. 

49
Two join-compatible SPOs are join-consistent if probability distributions over the set of shared participating variables are identical for both SPOs.

**Definition 3.23** Let \( S = \langle T, V, P, C, \omega \rangle \) and \( S' = \langle T', V', P', C', \omega' \rangle \) be two join-compatible SPOs with \( V \cap V' = V_c \). Then, \( S \) and \( S' \) are join-consistent iff \( P(\tilde{g}) = P'(\tilde{g}) \) for any \( \tilde{g} \in \text{dom}(V_c) \).

**Example 3.6** Consider two simple SPOs \( S \) and \( S' \) as presented in Figure 3.5. \( S \) and \( S' \) share one random variable (LY) and their conditional parts coincide (DR = B). Hence, \( S \) and \( S' \) are join-compatible.

<table>
<thead>
<tr>
<th>( \omega: S )</th>
<th>race: Asian</th>
<th>LY</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA</td>
<td>A</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DR = B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \omega: \pi_{LY}(S) )</td>
<td>race: Asian</td>
<td>LY</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>( \omega: S \times S' )</td>
<td>race: Asian</td>
<td>LY</td>
<td>VT</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DR = B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \omega: \pi_{LY}(S') )</td>
<td>city: Lexington</td>
<td>LY</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \omega: S \times S' )</th>
<th>race: Asian</th>
<th>city: Lexington</th>
<th>LY</th>
<th>VT</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>A</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>A</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>A</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.5: Join operations in SP-Algebra**

The results of the two join operations of \( S \) and \( S' \), \( S \times S' \) and \( S \times S' \), are presented in the rest of Figure 3.5. In the resulting SPOs, the context will be a union of the contexts of the two original objects and the conditional part will be the same as in \( S \) and \( S' \). The probability table is formed first by projecting the shared variable set from one of the original SPOs. For left join, first we do projection on all the variables in \( V_c \), in this case \( LY \), against the right operand SPO \( S' \), and save the result in a temporary SPO object \( temp \).

\[
\text{temp} = \pi_{V_c}(S'),
\]

50
as shown in the center of the figure. Then, we formulate the probability table by using the definition for left join, which is 

\[ P'(\bar{x}, \bar{y}, \bar{z}) = P(\bar{x}, \bar{y}) \times P'(\bar{y}, \bar{z})/\text{temp}.P'(\bar{y}). \]

The right join can be computed in the same manner.

Respective join results are shown in the last two columns in Figure 3.5. One can see that these two SPOs are not join-consistent.

3.4 Semantics of SP-Algebra Operations

The problem of determining the meaning of the results of the operations of SP-Algebra is complicated by the fact that at any moment, SP-databases can contain SPOs of two types. In the SPOs of the first type, the probabilities of all rows are exact, while in the SPOs of the second type, the probabilities of some rows may represent the lower bounds on the probability of those instances. We start this section by defining the two types of SPOs formally, discussing their properties and the effects that different SP-Algebra operations have on the SPOs in light of all this.

**Definition 3.24** An SPO \( S = (T, V, P, C, \omega) \) is a Type I SPO iff

\[ \sum_{\bar{x} \in \text{dom}(V)} P(\bar{x}) = 1. \]

Otherwise, \( S \) is a Type II SPO.

When \( S \) is a Type I SPO, its probability table is complete: the probabilities of all rows add up to exactly 1. The probability table may contain a row for every instance \( \bar{x} \in \text{dom}(V) \), or it may omit some of the instances. However, because the probabilities of the rows present in the table add up to 1, we know that the probabilities of all omitted rows are 0, and these can be added to the probability table of \( S \). Basically, when \( S \) is a Type I SPO, we are guaranteed that for all \( \bar{x} \in \text{dom}(V) \), \( P(\bar{x}) \) is the exact point probability of instance \( \bar{x} \).

The nature of Type II SPOs is somewhat more complex. The fact that the sum of probabilities in all rows of the probability table is less than 1 means that the probability table is missing some information. This can either be missing instances: some \( \bar{x} \in \text{dom}(V) \) has a non-zero probability but is not included in the probability table of \( S \), or underestimation: all possible instances are present, but the probabilities add up to less than 1, which means that information about the probabilities of some (possibly all) instances presents only a lower bound on the true probability of the instance in the distribution.
It is important to note here that SP-Algebra operations allow for Type II SPOs to occur in the SP-database, even if all original SPOs in the database were Type I. We illustrate this on the following example.

**Example 3.7** Consider the SPO S: the left-most SPO depicted in Figure 3.6. It is clear that \( \text{dom}(\text{DA}) = \text{dom}(\text{LY}) = \{A, B, C\} \), which means that not all instances are present in the probability table \( P \) of \( S \). However, because the probabilities of all rows present in \( P \) add up to exactly 1, \( S \) is a Type I SPO and the probabilities of all instances not in \( P \) are 0. We also can be assured that each probability is exact.

Consider now the central SPO \( S' = \sigma_{\text{pro}, P \leq 0.2}(S) \) in Figure 3.6. Here, only the rows with probability value less than or equal to 0.2 are selected from the probability table of \( S \). There are 3 such rows, for a combined probability of 0.35. Therefore, \( S' \) is a Type II SPO. We note here that, despite being of Type II, the probability of each row is exact. Consider now the SPO \( S'' = \pi_{\text{var}, \text{LY}}(S') = \pi_{\text{var}, \text{LY}}(\sigma_{\text{pro}, P \leq 0.2}(S)) \) shown on the right side of Figure 3.6. The projection operation leads to removal of the DA random variable from \( S'' \). However, because each row of \( P' \) had a different value for \( \text{LY} \), \( P'' \) will have three rows, one for each value of \( \text{LY}: A, B \) and \( C \). While the probability table \( P'' \) of \( S'' \) has no missing rows, the probabilities add up to the same value of 0.35 as in \( P' \), and therefore \( S'' \) is also a Type II SPO. More importantly, the rows for \( A \) and \( C \) contain incomplete probability — applying \( \pi_{\text{var}, \text{LY}} \) to \( S \) we can see that the probability of getting the grade of \( A \) in \( \text{LY} \) is 0.43 and the probability of getting the grade of \( C \) is 0.37. Therefore, the probability values in the probability table \( P'' \) represent only the lower bounds on the probabilities of these rows.

<table>
<thead>
<tr>
<th>( \omega: S )</th>
<th>( \omega: \sigma_{\text{pro}, P \leq 0.2}(S) )</th>
<th>( \omega: \pi_{\text{var}, \text{LY}}(\sigma_{\text{pro}, P \leq 0.2}(S)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>race: Asian</td>
<td>race: Asian</td>
<td>race: Asian</td>
</tr>
<tr>
<td>DA</td>
<td>LY</td>
<td>P</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>0.1</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>0.2</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>0.33</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>0.22</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>0.15</td>
</tr>
<tr>
<td>DR = B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.6: Selection on Probabilistic Table and on Probability values in SP-Algebra
The difference in the meaning of probability values for Type I and Type II SPOs causes us to apply extra caution when interpreting the results of SP-Algebra operations. In particular, when considering a specific SP-Algebra operation applied to an SPO or a pair of SPOs, it is important for us to know the type of the input objects and be able to determine the type of the result. The following proposition identifies the set of "safe" operations in SP-Algebra: operations that given Type I SPOs are guaranteed to produce Type I results.

**Proposition 3.4** Let $S$ and $S'$ be two Type I SPOs. Then, the following SPOs are also Type I:

1. $\sigma_c(S)$, where $c$ is a selection condition on context, participating random variables or conditional.
2. $\pi_{L}(S)$, $\pi_{F}(S)$ and $\pi_{L'}(S)$, where $F$ is a list of context attribute names and $F$, $L \subset V$.
3. $\mu_c(S)$, where $c$ is a conditional selection condition.
4. $S \times S'$.
5. $S \bowtie S'$ and $S \times S'$.

Two operations missing from the list in Proposition 3.4 are selection on probabilities and selection on probability table. Example 3.7 shows how selection on probabilities can produce a Type II SPO from Type I input; selection on probability table can be used instead to obtain the same result.

The following statements specify the semantics of the SP-Algebra operations producing Type I results.

**Theorem 3.2** Let $S = \langle T, V, P, C, \omega \rangle$ be a Type I SPO and let $\emptyset \neq L \subset V$. Let $S' = \langle T, L', P', C, \omega' \rangle = \pi_{L'}$. Then $P'$ contains the correct marginal probability distribution of random variables in $L'$ given the probability distribution $P$.

**Proof of Theorem 3.2**

Let $P(\vec{x}, \vec{y})$ be a probability distribution with $(\vec{x}, \vec{y}) \in \text{dom}(V)$. By the definition of projection in Section 3.3.3, we know that $P' : \text{dom}(L') \to [0, 1]$ and for each $\vec{x} \in \text{dom}(L')$,

$$P'(\vec{x}) = \sum_{\vec{y} \in \text{dom}(V - L') : P(\vec{x}, \vec{y}) \text{is defined}} P(\vec{x}, \vec{y}).$$

Note that this sum is exactly the marginal probability, for any $\vec{x} \in \text{dom}(L')$. \qed
Theorem 3.3 Let $S = \langle T, V, P, C, \omega \rangle$ be a Type I SPO and let $c$ be a conditional selection condition involving variable $v \in V$. Let $S' = \langle T, V - \{v\}, P', C', \omega' \rangle = \mu_c$. Then $P'$ contains the correct conditional probability distribution of random variables $V - \{v\}$ from the distribution $P$ given condition $c$ on $v$.

Proof of Theorem 3.3

Let $P(\bar{x}, \bar{y})$ be a probability distribution with $(\bar{x}, \bar{y}) \in \text{dom}(V)$. By the definition of conditionalization in Section 3.3.4, we have $P' : V' \rightarrow [0, 1]$. Let

$$N = \sum_{\bar{y} \in \text{dom}(V')} \sum_{\bar{x} \in \{x_1, \ldots, x_h\}} P(\bar{x}, \bar{y}).$$

For any $\bar{y} \in \text{dom}(V'),$

$$P'(\bar{y}) = \frac{\sum_{\bar{x} \in \{x_1, \ldots, x_h\}} P(\bar{x}, \bar{y})}{N}.$$

We can see that $N$ represents the sum of the probabilities of those rows which satisfy the conditional selection condition $\bar{x} \in \{x_1, \ldots, x_h\}$. Then we know that $P'(\bar{y})$ computes the conditional probability for any $\bar{y} \in \text{dom}(V - \{v\})$.

Theorem 3.4 Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C, \omega' \rangle$ be two Cartesian product-compatible SPOs. Let $S'' = \langle T'', V'', P'', C, \omega'' \rangle = S \times S'$. Then $P''$ is the correct joint probability distribution of random variables in $V$ and $V'$ under the assumption of independence between them, given distributions $P$ of $V$ and $P'$ of $V'$.

Proof of Theorem 3.4

Let $P(\bar{x})$ and $P'(\bar{y})$ be two probability distributions with $\bar{x} \in \text{dom}(V)$ and $\bar{y} \in \text{dom}(V')$. Since we assume that the variables in the two SPOs are independent, the definition of Cartesian product in Section 3.3.5, $P''(\bar{x}, \bar{y}) = P(\bar{x}) \cdot P'(\bar{y})$, correctly computes the joint probability distribution by multiplying the probabilities of the two individual events.

Theorem 3.5 Let $S = \langle T, V, P, C, \omega \rangle$ and $S' = \langle T', V', P', C, \omega' \rangle$ be two join-compatible SPOs. Let $S'' = \langle T'', V'', P'', C, \omega'' \rangle = S \times S'$ and $S''' = \langle T''', V''', P''', C, \omega''' \rangle = S \times S'$. Then $P''$ and $P'''$ are the correct joint probability distributions of random variables in $V$ and $V'$ under the assumption of independence between them, given distributions $P$ of $V$ and $P'$ of $V'$.

Lemma 3.1 Let $\bar{z} \in \text{dom}(V_1), \bar{y} \in \text{dom}(V_c), \bar{z} \in \text{dom}(V_1'),$ and let $V_1, V_c$ and $V_1'$ be all disjoint. Under the assumption of independence between variables in $V_1$ and $V_1'$,

$$P(\bar{x}, \bar{y}, \bar{z}) = P(\bar{x}, \bar{y}) \cdot P(\bar{y}, \bar{z}) / P(\bar{y}) = P(\bar{x}, \bar{y}) \cdot P(\bar{z} | \bar{y}) = P(\bar{z}, \bar{y}) \cdot P(\bar{x} | \bar{y}).$$
Proof of Lemma 3.1

By the definition of conditional probability, we have
\[
P(\bar{x}, \bar{y}, \bar{z}) = P((\bar{x}, \bar{z}), \bar{y})
\]
\[
= P((\bar{x}, \bar{z})|\bar{y}) \cdot P(\bar{y})
\]
\[
= P(\bar{x}, \bar{y}) \cdot P(\bar{y})
\]
Here we assume that \( \bar{x} \) and \( \bar{z} \) are conditional independent given \( \bar{y} \), then
\[
P(\bar{x}, \bar{y}, \bar{z}) = P(\bar{x}|\bar{y}) \cdot P(\bar{z}|\bar{y}) \cdot P(\bar{y})
\]
\[
= P(\bar{x}, \bar{y}) \cdot P(\bar{z}|\bar{y})
\]
\[
= P(\bar{z}, \bar{y}) \cdot P(\bar{x}|\bar{y}).
\]

\[\square\]

Proof of Theorem 3.5

Let \( P(\bar{x}, \bar{y}) \) and \( P'(\bar{y}, \bar{z}) \) be two probability distributions with \( \bar{x} \in \text{dom}(V_1), \bar{y} \in \text{dom}(V_c) \) and \( \bar{z} \in \text{dom}(V'_{1'}) \) By assuming that the variables \( \bar{x} \) and \( \bar{z} \) are independent, Lemma 3.1 gives \( P(\bar{x}, \bar{y}, \bar{z}) = P(\bar{x}, \bar{y}) \cdot P'(\bar{y}, \bar{z})/P'(\bar{y}). \) So the definition of left join in Section 3.3.6 computes the joint probability distribution of random variables in \( V'' \).

\[\square\]

Theorem 3.6 Let \( S \) and \( S' \) be two join-compatible SPOs. The left join \( S \Join S' \) and right join \( S \Join S' \) are equivalent iff the two SPOs are join-consistent.

Proof of Theorem 3.6

First, we prove that if two SPOs are join-consistent, then the join operations \( S \Join S' \) and \( S \Join S' \) are equivalent. From the definition of join-consistent, we know that \( P(\bar{y}) = P'(\bar{y}) \) for any \( \bar{y} \in \text{dom}(V_c) \) and \( V_c \neq \emptyset \). Then the probability distribution \( P(\bar{x}, \bar{y}, \bar{z}) \) will be identical, which implies the the two join operations \( S \Join S' \) and \( S \Join S' \) are equivalent.

Second, consider the other direction. If the join operations \( S \Join S' \) and \( S \Join S' \) are equivalent, i.e. \( P''(\bar{w}) = P''(\bar{w}) \) for any \( \bar{w} \in \text{dom}(V'') \), then by the definition of the join operations we have
\[
P'(\bar{y}) = P(\bar{x}, \bar{y}) \cdot P'(\bar{y}, \bar{z})/P''(\bar{w}),
\]
and
\[
P(\bar{y}) = P(\bar{x}, \bar{y}) \cdot P'(\bar{y}, \bar{z})/P''(\bar{w}).
\]
so the two probability distributions are identical, i.e. \( P(\bar{y}) = P'(\bar{y}) \) for any \( \bar{y} \in \text{dom}(V_c) \), which implies that the two SPOs are join-consistent.

\[\square\]
3.5 Query Optimization

In this section we will describe our work on query optimization. Query optimization is a necessary part of any useful databases. Our approach to query optimization tries to follow the principles of query optimization for standard relational databases, see a survey in [45]. However, some of the results may differ greatly due to the semistructured nature of SP-relations. The theory of semistructured probabilistic query optimization consists of two parts: rewrite rules and cost models. We have successfully established a number of rewrite rules and a set of cost models for SP-Algebra queries in the SP-database, which can be implemented in the query optimizer for future implementations of our database framework.

The rewrite rules obtained so far are listed in Section 3.5.1. As you will notice, there are cases in which no general rewrite rules are available. The cost models for different operations are described in Section 3.5.2.

3.5.1 Rewrite rules

For traditional relational query optimizers, cost models are established based on disk I/Os as well as CPUs if necessary. However, since SPDBMS uses a two-tier architecture and builds on top of a relational DBMS, it is hard for the SPDBMS engine to compute disk I/Os (not mention to CPUs) due to the lack of information on the underlying storage structures. So basically we need to adopt other measures to establish our cost models, i.e., the number of tuples returned by the underlying RDBMS or the number of SPOs (and the average size of SPOs in an SP-relation) in the intermediate SP-relations, or both.

Following are the query rewrite rules that hold in SP-Algebra. Note that no useful query rewrite rules have been found for join operations.

**Theorem 3.7** Let \( c \) and \( c' \) be two selection conditions and let \( S \) be an SP-relation. Then the following rules are valid.

1. \( \sigma_c(\sigma_{c'}(S)) \equiv \sigma_{c'}(\sigma_c(S)) \) (Same as Theorem 3.1)
2. \( \sigma_c \land c'(S) \equiv \sigma_c(\sigma_c(S)) \)

**Proof of Theorem 3.7.2**

Let \( c \) and \( c' \) be two atomic selection conditions. There are 5 types of atomic selection conditions, namely context, participation, conditional, table and probability. Selection on context, participation or conditional will result in entire SPOs being selected, while selection on table or
probability will select only parts of the relevant SPOs. We could partition the conditions into two groups,

- Group I, containing context, participation and conditional conditions, and
- Group II, containing table and probability conditions.

First we prove that \( \sigma_c \land c'(S) \equiv \sigma_c(\sigma_{c'}(S)) \) for a single SPO \( S \).

**Case 1.** Both conditions \( c \) and \( c' \) are in Group I.

There are four possible combinations for whether each condition is satisfied:

a) \( S \) satisfies \( c \) but not \( c' \), or
b) \( S \) satisfies \( c' \) but not \( c \), or
c) \( S \) satisfies both \( c \) and \( c' \), or
d) \( S \) does not satisfy either \( c \) or \( c' \).

By the definition of selection on atomic selection conditions in Group I, we know selection on these conditions will result in the entire SPO being selected, or none of it.

For case (a), since \( S \) does not satisfy \( c' \), \( \sigma_{c'}(S) \) returns empty and subsequently \( \sigma_c(\sigma_{c'}(S)) \) will return empty. \( \sigma_c \land c'(S) \) will also return empty. Thus, \( \sigma_c(\sigma_{c'}(S)) \equiv \sigma_c(\sigma_{c'}(S)) \) holds for case (a). The same applies to case (b). Similarly, for case (d). For case (c), \( \sigma_c(\sigma_{c'}(S)) = \sigma_c(S) \) returns \( S \), and \( \sigma_c \land c'(S) \) returns \( S \) too. Thus, \( \sigma_c \land c'(S) \equiv \sigma_c(\sigma_{c'}(S)) \) holds for case (c).

So \( \sigma_c \land c'(S) \equiv \sigma_c(\sigma_{c'}(S)) \) holds for all these cases.

**Case 2.** Condition \( c \) is in Group I and condition \( c' \) is in Group II.

There are only two possible combinations for whether each condition is satisfied, assuming that condition \( c' \) is always partially satisfied:

a) \( S \) does not satisfy \( c \), or
b) \( S \) satisfies \( c \).

By the definition of selection on atomic selection conditions in both Group I and Group II, we know selection on conditions in Group I will result in the entire SPO being selected or not, while selection on conditions in Group II will preserve all the context, participating random variables and conditionals in the original SPO, but produce only a part of the probability table.

Let \( \sigma_{c'}(S) = S' \), where \( S' \) has the part of the probability table that satisfies the condition \( c' \) and retains all the context, participating random variables and conditionals in \( S \).

For case (a), \( \sigma_c(\sigma_{c'}(S)) = \sigma_c(S') \) will return empty since \( S' \) does not satisfy the condition \( c \) either. \( \sigma_c \land c'(S) \) will also return empty. This proves \( \sigma_c \land c'(S) \equiv \sigma_c(\sigma_{c'}(S)) \) for case (a).
For case (b), $\sigma_c(\sigma_{c'}(S)) = \sigma_c(S')$ will return $S'$, since $S'$ should also satisfy the condition $c$. $\sigma_{c \land c'}(S) = \sigma_{c'}(S)$ will also return $S'$. This proves that $\sigma_{c \land c'}(S) \equiv \sigma_{c'}(\sigma_c(S))$ holds for case (b).

So $\sigma_{c \land c'}(S) \equiv \sigma_c(\sigma_{c'}(S))$ holds for both cases.

**Case 3.** Both $c$ and $c'$ are conditions in Group II.

Assume that both conditions $c$ and $c'$ are partially satisfied by $S$. By the definition of selection on atomic selection conditions in Group II, we know that selection on these conditions will result in part of the probability table and will preserve all the context, participating random variables and conditionals in the original SPO. In other words, all the components in the original SPO except the probability table and the path will be preserved.

Let $S = \langle T, V, P, C, \omega \rangle$. Then $S' = \sigma_{c \land c'}(S) = \langle T, V, P', C, \omega' \rangle$ and $S'' = \sigma_{c'}(\sigma_c(S)) = \langle T, V, P'', C, \omega'' \rangle$ with $P' = \theta_{c \land c'}(P)$ and $P'' = \theta_{c'}(\theta_c(P))$, where $\theta$ is the relational selection operator. Since for relational selection operator $\theta$ we have $\theta_{c \land c'}(P) = \theta_c(\theta_{c'}(P))$. Therefore, we have $P' = P''$ or $S' \equiv S''$. So $\sigma_{c \land c'}(S) \equiv \sigma_{c'}(\sigma_c(S))$ holds for this case.

This proves $\sigma_{c \land c'}(S) \equiv \sigma_{c'}(\sigma_c(S))$ for a single SPO. Now let an SP-relation $S = \bigcup_{S \in S}(S)$. Since the union operator is commutative, i.e., $\sigma_{c \land c'}(S) = \sigma_{c \land c'}(\bigcup_{S \in S}(S)) = \bigcup_{S \in S}(\sigma_{c \land c'}(S))$ and $\sigma_{c}(\sigma_{c'}(S)) = \sigma_{c}(\bigcup_{S \in S}(\sigma_{c'}(S))) = \bigcup_{S \in S}(\sigma_{c}(\sigma_{c'}(S))) \equiv \bigcup_{S \in S}(\sigma_{c \land c'}(S))$.

This proves $\sigma_{c \land c'}(S) \equiv \sigma_{c'}(\sigma_c(S))$. \hfill $\square$

**Theorem 3.8** Let $F$ and $G$ be two projection lists and let $S$ be an SP-relation. Then the following rules are valid.

1. $\pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S))$
2. $\pi_F \cap \pi_G(S) \equiv \pi_G(\pi_F(S))$

**Proof of Theorem 3.8.1**

First we prove that $\pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S))$ for a single SPO $S$. Let $S = \langle T, V, P, C, \omega \rangle$.

Projection can be performed on context attributes, conditional, and random variables. There are nine possible combinations of types of projection lists $F$ and $G$.

**Case 1.** Both $F$ and $G$ are lists of context attributes.

With the definition of projection on context attributes, we have $\pi_F(S) = \langle T1', V, P, C, \omega 1' \rangle$, where $T1' = \{(A, x) | (A, x) \in T, A \in F\}$, or $T1'$ contains all entries from $T$ for attributes from the list $F$ only.
\[ \pi_G(\pi_F(S)) = \langle T''', V, P, C, \omega1'' \rangle, \text{ where } T'' = \{(A, x)| (A, x) \in T', A \in G \}, \text{ or } T'' \text{ contains all entries from } T' \text{ for attributes from the list } G \text{ only.} \]

Similarly, \( \pi_G(S) = \langle T''', V, P, C, \omega1'' \rangle, \text{ where } T'' = \{(A, x)| (A, x) \in T, A \in G \}, \text{ or } T'' \text{ contains all entries from } T \text{ for attributes from the list } G \text{ only.} \)

It can be seen that both \( T'' \) and \( T'' \) contain all entries from \( T \) for attributes from the list \( F \cap G \) (same as \( G \cap F \) only, or \( T'' = T'' \). So \( \pi_F(\pi_G(S)) \) is equivalent to \( \pi_G(\pi_F(S)) \), or \( \pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S)) \).

**Case 2.** Both \( F \) and \( G \) are lists of conditionals.

With the definition of projection on conditional, we have \( \pi_F(S) = \langle T, V, P, C', \omega1' \rangle, \text{ where } C' = \{(u, X)| (u, X) \in C, u \in F \}, \text{ or } C' \text{ contains all entries from } C \text{ for conditional from the list } F \text{ only.} \)

\[ \pi_G(\pi_F(S)) = \langle T, V, P, C'', \omega1'' \rangle, \text{ where } C'' = \{(u, X)| (u, X) \in C', u \in F \}, \text{ or } C'' \text{ contains all entries from } C' \text{ for conditional from the list } G \text{ only, or } C'' \text{ contains all entries from } C \text{ for conditional from the list } F \text{ only.} \]

Similarly, \( \pi_G(S) = \langle T, V, P, C', \omega2' \rangle, \text{ where } C' = \{(u, X)| (u, X) \in C, u \in G \}, \text{ or } C' \text{ contains all entries from } C \text{ for conditional from the list } G \text{ only, or } C' \text{ contains all entries from } C \text{ for conditional from the list } F \text{ only.} \)

\[ \pi_F(\pi_G(S)) = \langle T, V, P, C'', \omega2'' \rangle, \text{ where } C'' = \{(u, X)| (u, X) \in C', u \in G \}, \text{ or } C'' \text{ contains all entries from } C' \text{ for conditional from the list } G \text{ only, or } C'' \text{ contains all entries from } C \text{ for conditional from the list } F \text{ only.} \]

It can be seen that both \( C'' \) and \( C'' \) contain all entries from \( C \) for conditional from the list \( F \cap G \) (same as \( G \cap F \) only, or \( C'' = C'' \). So \( \pi_F(\pi_G(S)) \) is equivalent to \( \pi_G(\pi_F(S)) \), or \( \pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S)) \).

**Case 3.** Both \( F \) and \( G \) are lists of random variables.

With the definition of projection on random variables, we have \( \pi_F(S) = \langle T, V', P', C, \omega1' \rangle, \text{ where } V' = V \cap F, \text{ and } P' : \text{dom}(V') \rightarrow [0, 1] \text{ and for each } \bar{x} \in \text{dom}(V'), \)

\[ P'(\bar{x}) = \sum_{\bar{y} \in \text{dom}(V' - V')} P(\bar{x}, \bar{y}). \]

\[ \pi_G(\pi_F(S)) = \langle T, V'', P'', C, \omega1'' \rangle, \text{ where } V'' = V' \cap G = V \cap F \cap G, \text{ and } P' :
\(\text{dom}(V1'') \rightarrow [0, 1]\) and for each \(\tilde{x}' \in \text{dom}(V1'').\)

\[
P1''(\tilde{x}') = \sum_{\tilde{y}' \in \text{dom}(V1' - V1''); P1'(\tilde{x}', \tilde{y}') \text{is defned}} P1'(\tilde{x}', \tilde{y}').
\]

By substituting \(P1'\) (note that \(\tilde{x} = (\tilde{x}', \tilde{y}'))\), we get

\[
P1''(\tilde{x}') = \sum_{\tilde{y}' \in \text{dom}(V1' - V1''); P1'(\tilde{x}', \tilde{y}') \text{is defned}} \sum_{\tilde{y} \in \text{dom}(V - V1'); P(\tilde{x}', \tilde{y}, \tilde{y}) \text{is defned}} P(\tilde{x}', \tilde{y}', \tilde{y}).
\]

Let

\[
\tilde{y}'' = (\tilde{y}', \tilde{y}) \in \text{dom}(V - V1'),
\]

so we have

\[
P1''(\tilde{x}') = \sum_{\tilde{y}' \in \text{dom}(V1' - V1''); P1'(\tilde{x}', \tilde{y}') \text{is defned}} P(\tilde{x}', \tilde{y}'').
\]

Similarly, \(\pi_G(S) = \langle T, V2', P2', C, \omega 2' \rangle\), where \(V2' = V \cap G\), and \(P2' : \text{dom}(V2') \rightarrow [0, 1]\) and for each \(\tilde{x} \in \text{dom}(V2')\),

\[
P2'(\tilde{x}) = \sum_{\tilde{y} \in \text{dom}(V - V2'); P(\tilde{x}, \tilde{y}) \text{is defned}} P(\tilde{x}, \tilde{y}).
\]

\(\pi_F(\pi_G(S)) = \langle T, V2'', P2'', C, \omega 2'' \rangle\), where \(V2'' = V2' \cap F = V \cap G \cap F\), and \(P2'' : \text{dom}(V2'') \rightarrow [0, 1]\) and for each \(\tilde{x}' \in \text{dom}(V2'')\),

\[
P2''(\tilde{x}') = \sum_{\tilde{y}' \in \text{dom}(V2' - V2''); P2'(\tilde{x}', \tilde{y}'; \tilde{y}) \text{is defned}} P2'(\tilde{x}', \tilde{y}').
\]

By substituting \(P2'\) (note that \(\tilde{x} = (\tilde{x}', \tilde{y}')\)), we get

\[
P2''(\tilde{x}') = \sum_{\tilde{y}' \in \text{dom}(V2' - V2''); P2'(\tilde{x}', \tilde{y}') \text{is defned}} \sum_{\tilde{y} \in \text{dom}(V - V2'); P(\tilde{x}', \tilde{y}, \tilde{y}) \text{is defned}} P(\tilde{x}', \tilde{y}', \tilde{y}).
\]

Let

\[
\tilde{y}'' = (\tilde{y}', \tilde{y}) \in \text{dom}(V - V2''),
\]

we have

\[
P2''(\tilde{x}') = \sum_{\tilde{y}'' \in \text{dom}(V - V2''); P(\tilde{x}', \tilde{y}'') \text{is defned}} P(\tilde{x}', \tilde{y}'').
\]

It can be seen that \(V1'' = V2'' = V \cap F \cap G\) and then \(P1''(\tilde{x}') = P2''(\tilde{x}')\) for each \(\tilde{x}' \in \text{dom}(V \cap F \cap G)\). So \(\pi_F(\pi_G(S))\) is equivalent to \(\pi_G(\pi_F(S))\), or \(\pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S))\).

**Cases 4 & 5.** \(F\) is a list of context attributes, and \(G\) is a list of conditionals.
With the definition of projection on context attributes, we have \( \pi_F(S) = \langle T1', V, P, C, \omega 1' \rangle \), where \( T1' = \{(A, x)| (A, x) \in T, A \in F\} \), or \( T1' \) contains all entries from \( T \) for attributes from the list \( F \) only.

With the definition of projection on conditional, we have \( \pi_G(\pi_F(S)) = \langle T1', V, P, C1', \omega 1'' \rangle \), where \( C1' = \{(u, X)| (u, X) \in C, u \in G\} \), or \( C1' \) contains all entries from \( C \) for conditional from the list \( G \) only.

Similarly, with the definition of projection on conditional, we have \( \pi_G(S) = \langle T, V, P, C2', \omega 2' \rangle \), where \( C2' = \{(u, X)| (u, X) \in C, u \in G\} \), or \( C2' \) contains all entries from \( C \) for conditional from the list \( G \) only.

With the definition of projection on context attributes, we have \( \pi_F(\pi_G(S)) = \langle T2', V, P, C2', \omega 2'' \rangle \), where \( T2' = \{(A, x)| (A, x) \in T, A \in F\} \), or \( T2' \) contains all entries from \( T \) for attributes from the list \( F \) only.

It can be seen that \( T1' = T2' \) and \( C1' = C2' \). So \( \pi_F(\pi_G(S)) \) is equivalent to \( \pi_G(\pi_F(S)) \), or \( \pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S)) \).

The same applies to the case where \( F \) is a list of conditionals, and \( G \) is a list of context attributes.

**Cases 6 & 7.** \( F \) is a list of context attributes, and \( G \) is a list of random variables.

With the definition of projection on context attributes, we have \( \pi_F(S) = \langle T1', V, P, C, \omega 1' \rangle \), where \( T1' = \{(A, x)| (A, x) \in T, A \in F\} \), or \( T1' \) contains all entries from \( T \) for attributes from the list \( F \) only.

With the definition of projection on random variables, we have \( \pi_G(\pi_F(S)) = \langle T1', V1', P1', C, \omega 1'' \rangle \), where \( V1' = V \cap F \), and \( P1': \text{dom}(V1') \rightarrow [0, 1] \) and for each \( \bar{x} \in \text{dom}(V1') \),

\[ P1'(\bar{x}) = \sum_{\bar{y} \in \text{dom}(V-V1') \cap \text{dom}(P(\bar{x}, \bar{y}))} P(\bar{x}, \bar{y}). \]

Similarly, with the definition of projection on random variables, we have \( \pi_G(S) = \langle T, V2', P2', C, \omega 2' \rangle \), where \( V2' = V \cap G \), and \( P2': \text{dom}(V2') \rightarrow [0, 1] \) and for each \( \bar{x} \in \text{dom}(V2') \),

\[ P2'(\bar{x}) = \sum_{\bar{y} \in \text{dom}(V-V2') \cap \text{dom}(P(\bar{x}, \bar{y}))} P(\bar{x}, \bar{y}). \]

With the definition of projection on context attributes, we have \( \pi_F(\pi_G(S)) = \langle T2', V2', P2', C, \omega 2'' \rangle \), where \( T2' = \{(A, x)| (A, x) \in T, A \in F\} \), or \( T2' \) contains all entries from \( T \) for attributes from the list \( F \) only.
It can be seen that \( T1' = T2', \ V1' = V2' \) and \( P1' = P2' \). So \( \pi_F(\pi_G(S)) \) is equivalent to \( \pi_G(\pi_F(S)) \), or \( \pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S)) \).

The same applies to the case where \( F \) is a list of \textit{random variables}, and \( G \) is a list of \textit{context attributes}.

**Cases 8 & 9.** \( F \) is a list of \textit{conditionals}, and \( G \) is a list of \textit{random variables}.

With the definition of projection on conditional, we have \( \pi_F(S) = \langle T, V, P, C1', \omega 1' \rangle \), where \( C1' = \{(u, X)|(u, X) \in C, u \in F\} \), or \( C1' \) contains all entries from \( C \) for conditional from the list \( F \) only.

With the definition of projection on random variables, we have \( \pi_G(\pi_F(S)) = \langle T, V1', P1', C1', \omega 1'' \rangle \), where \( V1' = V \cap G \), and \( P1' : \text{dom}(V1') \mapsto [0, 1] \) and for each \( \bar{x} \in \text{dom}(V1') \),

\[
P1'(\bar{x}) = \sum_{\bar{y} \in \text{dom}(V-V1'); P(\bar{x}, \bar{y}) \text{ is defned}} P(\bar{x}, \bar{y}).
\]

Similarly, with the definition of projection on random variables, we have \( \pi_G(S) = \langle T, V2', P2', C, \omega 2' \rangle \), where \( V2' = V \cap G \), and \( P2' : \text{dom}(V2') \mapsto [0, 1] \) and for each \( \bar{x} \in \text{dom}(V2') \),

\[
P2'(\bar{x}) = \sum_{\bar{y} \in \text{dom}(V-V2'); P(\bar{x}, \bar{y}) \text{ is defned}} P(\bar{x}, \bar{y}).
\]

With the definition of projection on conditional, we have \( \pi_F(\pi_G(S)) = \langle T, V2', P2', C2', \omega 2'' \rangle \), where \( C2' = \{(u, X)|(u, X) \in C, u \in G\} \), or \( C2' \) contains all entries from \( C \) for conditional from the list \( G \) only.

It can be seen that \( C1' = C2' \), \( V1' = V2' \) and \( P1' = P2' \). So \( \pi_F(\pi_G(S)) \) is equivalent to \( \pi_G(\pi_F(S)) \), or \( \pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S)) \).

The same applies to the case where \( F \) is a list of \textit{random variables}, and \( G \) is a list of \textit{conditional}.

This proves \( \pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S)) \) for a single SPO. Now let an SP-relation \( S = \cup_{S \in S}(S) \).

Since the union operator is commutative, i.e., \( \pi_F(\pi_G(S)) = \pi_F(\pi_G(\cup_{S \in S}(S))) = \cup_{S \in S}(\pi_F(\pi_G(S))) \) and \( \pi_G(\pi_F(S)) = \pi_G(\pi_F(\cup_{S \in S}(S))) = \cup_{S \in S}(\pi_G(\pi_F(S))) \equiv \cup_{S \in S}(\pi_F(\pi_G(S))) \).

This proves \( \pi_F(\pi_G(S)) \equiv \pi_G(\pi_F(S)) \).

\[\Box\]

**Proof of Theorem 3.8.2**

First we prove that \( \pi_{F \cap G}(S) \equiv \pi_G(\pi_F(S)) \) for a single SPO \( S \). Let \( S = \langle T, V, P, C, \omega \rangle \).
Projection can be performed on context attributes, conditionals, and random variables. There are nine possible combinations of types of projection lists $F$ and $G$.

**Case 1.** Both lists $F$ and $G$ are lists of context attributes.

With the definition of projection on context attributes, we have $\pi_F(S) = \langle T1', V, P, C, \omega l' \rangle$, where $T1' = \{(A, x)|(A, x) \in T, A \in F\}$, or $T1'$ contains all entries from $T$ for attributes from the list $F$ only.

$$\pi_G(\pi_F(S)) = \langle T1'', V, P, C, \omega l'' \rangle,$$
where $T1'' = \{(A, x)|(A, x) \in T1', A \in G\}$, or $T1''$ contains all entries from $T1'$ for attributes from the list $G$ only, or $T1''$ contains all entries from $T$ for attributes from the list $F \cap G$ only.

Similarly, $\pi_F \cap_G(S) = \langle T2', V, P, C, \omega l' \rangle$, where $T2' = \{(A, x)|(A, x) \in T, A \in F \cap G\}$, or $T2'$ contains all entries from $T$ for attributes from the list $F \cap G$ only.

It can be seen that both $T1''$ and $T2'$ contain all entries from $T$ for attributes from the list $F \cap G$ only, or $T1'' = T2'$. So $\pi_F \cap_G(S)$ is equivalent to $\pi_G(\pi_F(S))$, or $\pi_F \cap_G(S) \equiv \pi_G(\pi_F(S))$.

**Case 2.** Both lists $F$ and $G$ are lists of conditionals.

With the definition of projection on conditional, we have $\pi_F(S) = \langle T, V, P, C1', \omega l' \rangle$, where $C1' = \{(u, X)|(u, X) \in C, u \in F\}$, or $C1'$ contains all entries from $C$ for conditionals from the list $F$ only.

$$\pi_G(\pi_F(S)) = \langle T, V, P, C1'', \omega l'' \rangle,$$
where $C1'' = \{(u, X)|(u, X) \in C1', u \in F\}$, or $C1''$ contains all entries from $C1'$ for conditionals from the list $G$ only, or $C1''$ contains all entries from $C$ for conditionals from the list $F \cap G$ only.

Similarly, $\pi_F \cap_G(S) = \langle T, V, P, C2', \omega l' \rangle$, where $C2' = \{(u, X)|(u, X) \in C, u \in F \cap G\}$, or $C2'$ contains all entries from $C$ for conditionals from the list $F \cap G$ only.

It can be seen that both $C1''$ and $C2'$ contain all entries from $C$ for conditionals from the list $F \cap G$ only, or $C1'' = C2'$. So $\pi_F \cap_G(S)$ is equivalent to $\pi_G(\pi_F(S))$, or $\pi_F \cap_G(S) \equiv \pi_G(\pi_F(S))$.

**Case 3.** Both lists $F$ and $G$ are lists of random variables.

With the definition of projection on random variables, we have $\pi_F(S) = \langle T, V1', P1', C, \omega l' \rangle$, where $V1' = V \cap F$, and $P1' : dom(V1') \rightarrow [0, 1]$ and for each $\bar{x} \in dom(V1')$,

$$P1'(\bar{x}) = \sum_{\bar{y} \in dom(V-V1')} P(\bar{x} \bar{y})$$

$$\pi_G(\pi_F(S)) = \langle T, V1'', P1'', C, \omega l'' \rangle,$$
where $V1'' = V1' \cap G = V \cap F \cap G$, and $P1'' : dom(V1'') \rightarrow [0, 1]$ and for each $\bar{x}'' \in dom(V1'')$.}

63
\[ P1''(\vec{x}') = \sum_{\vec{y}' \in dom(V'1' - V1') \cap P1'(\vec{x'},\vec{y}')} P1'(\vec{x}',\vec{y}'). \]

By substituting \( P1' \) (note that \( \vec{x} = (\vec{x'},\vec{y}') \)), so we get

\[ P1''(\vec{x}') = \sum_{\vec{y}' \in dom(V'1' - V1') \cap P1'(\vec{x'},\vec{y}')} \sum_{\vec{g} \in dom(\vec{y}' \cap \vec{g}')} P(\vec{x}',\vec{y}',\vec{g}). \]

Let

\[ \vec{y}' = (\vec{y'}, \vec{y}) \in dom(V - V1''), \]

we have

\[ P1''(\vec{x}') = \sum_{\vec{y}' \in dom(V - V1'')} P(\vec{x}',\vec{y'}). \]

Similarly, \( \pi_F \cap G(S) = \langle T', V2', P2', C, \omega2' \rangle \), where \( V2' = V \cap F \cap G \) and \( P2' : dom(V2') \rightarrow [0, 1] \) and for each \( \vec{x} \in dom(V2') \),

\[ P2'(\vec{x}) = \sum_{\vec{y} \in dom(V - V2')} P(\vec{x},\vec{y}). \]

It can be seen that \( V1'' = V2' = V \cap F \cap G \) and then \( P1''(\vec{x}) = P2'(\vec{x}) \) for each \( \vec{x} \in dom(V \cap F \cap G) \). So \( \pi_F \cap G(S) \) is equivalent to \( \pi_G(\pi_F(S)) \), or \( \pi_F \cap G(S) \equiv \pi_G(\pi_F(S)) \).

**Cases 4 & 5.** \( F \) is a list of context attributes, and \( G \) is a list of conditionals.

With the definition of projection on context attributes, we have \( \pi_F(S) = \langle T1', V, P, C, \omega1' \rangle \), where \( T1' = \{(A, x)|(A, x) \in T, A \in F\} \), or \( T1' \) contains all entries from \( T \) for attributes from the list \( F \) only.

With the definition of projection on conditional, we have \( \pi_G(\pi_F(S)) = \langle T1', V, P, C1', \omega1'' \rangle \), where \( C1' = \{(u, X)|(u, X) \in C, u \in G\} \), or \( C1' \) contains all entries from \( C \) for conditionals from the list \( G \) only.

Similarly, with the definitions of projection on context attributes and projection on conditional, we have \( \pi_F \cap G(S) = \langle T2', V, P, C2', \omega2' \rangle \), where \( T2' = \{(A, x)|(A, x) \in T, A \in F\} \), or \( T2' \) contains all entries from \( T \) for attributes from the list \( F \) only; \( C2' = \{(u, X)|(u, X) \in C, u \in G\} \), or \( C2' \) contains all entries from \( C \) for conditionals from the list \( G \) only.

It can be seen that \( T1' = T2' \) and \( C1' = C2' \). So \( \pi_F \cap G(S) \) is equivalent to \( \pi_G(\pi_F(S)) \), or \( \pi_F \cap G(S) \equiv \pi_G(\pi_F(S)) \).

The same applies to the case where \( F \) is a list of conditionals, and \( G \) is a list of context attributes.
Cases 6 & 7. $F$ is a list of context attributes, and $G$ is a list of random variables.

With the definition of projection on context attributes, we have $\pi_F(S) = \langle T_1', V, P, C, \omega_1' \rangle$, where $T_1' = \{(A, x) | (A, x) \in T, A \in F\}$, or $T_1'$ contains all entries from $T$ for attributes from the list $F$ only.

With the definition of projection on random variables, we have $\pi_G(\pi_F(S)) = \langle T_1', V_1', P_1', C, \omega_1'' \rangle$, where $V_1' = V \cap F$, and $P_1' : \text{dom}(V_1') \rightarrow [0, 1]$ and for each $\bar{x} \in \text{dom}(V_1')$,

$$P_1'(\bar{x}) = \sum_{\bar{g} \in \text{dom}(V-V_1'); P(\bar{x}, \bar{g}) \text{ is defined}} P(\bar{x}, \bar{g}).$$

Similarly, with the definitions of projection on context attributes and projection on random variables, we have $\pi_F \cap G(S) = \langle T_2', V_2', P_2', C, \omega_2' \rangle$, where $T_2' = \{(A, x) | (A, x) \in T, A \in F\}$, or $T_2'$ contains all entries from $T$ for attributes from the list $F$ only; $V_2' = V \cap G$, and $P_2' : \text{dom}(V_2') \rightarrow [0, 1]$ and for each $\bar{x} \in \text{dom}(V_2')$,

$$P_2'(\bar{x}) = \sum_{\bar{g} \in \text{dom}(V-V_2'); P(\bar{x}, \bar{g}) \text{ is defined}} P(\bar{x}, \bar{g}).$$

It can be seen that $T_1' = T_2'$, $V_1' = V_2'$ and $P_1' = P_2'$. So $\pi_F \cap G(S)$ is equivalent to $\pi_G(\pi_F(S))$, or $\pi_F \cap G(S) \equiv \pi_G(\pi_F(S))$.

The same applies to the case where $F$ is a list of random variables, and $G$ is a list of context attributes.

Cases 8 & 9. $F$ is a list of conditionals, and $G$ is a list of random variables.

With the definition of projection on conditional, we have $\pi_F(S) = \langle T, V, P, C_1', \omega_1' \rangle$, where $C_1' = \{(u, X) | (u, X) \in C, u \in F\}$, or $C_1'$ contains all entries from $C$ for conditional from the list $F$ only.

With the definition of projection on random variables, we have $\pi_G(\pi_F(S)) = \langle T, V_1', P_1', C_1', \omega_1'' \rangle$, where $V_1' = V \cap G$, and $P_1' : \text{dom}(V_1') \rightarrow [0, 1]$ and for each $\bar{x} \in \text{dom}(V_1')$,

$$P_1'(\bar{x}) = \sum_{\bar{g} \in \text{dom}(V-V_1'); P(\bar{x}, \bar{g}) \text{ is defined}} P(\bar{x}, \bar{g}).$$

Similarly, with the definitions of projection on random variables and projection on conditional, we have $\pi_F \cap G(S) = \langle T, V_2', P_2', C_2', \omega_2' \rangle$, where $C_2' = \{(u, X) | (u, X) \in C, u \in F\}$, or $C_2'$ contains all entries from $C$ for conditional from the list $F$ only; $V_2' = V \cap G$, and $P_2' : \text{dom}(V_2') \rightarrow [0, 1]$ and for each $\bar{x} \in \text{dom}(V_2')$. 
Proof of Theorem 3.9.1

First we prove that \( \mu_c(\mu_{c'}(S)) \equiv \mu_{c'}(\mu_c(S)) \) for a single SPO, where \( S = \langle T, V, P, C, \omega \rangle \) be an SPO which is conditionalization-compatible; \( c : v = \{ x_1, \ldots, x_h \} \) and \( c' : v' = \{ x_1', \ldots, x_h' \} \) are two atomic conditional selection conditions.

With the definition of conditionalization, we have \( \mu_c(S) = \langle T, V1', P1', C1', \omega 1' \rangle \), where \( V1' = V - \{ v \}; C1' = C \cup \{ (v, \{ x_1, \ldots, x_h \}) \}; P1' : dom(V1') \rightarrow [0, 1] \).

Let

\[
N1' = \sum_{\bar{y} \in dom(V1')} \sum_{\bar{x} \in \{ x_1, \ldots, x_h \}} P(\bar{x}, \bar{y}).
\]

For any \( \bar{y} \in dom(V1') \),

\[
P1'(\bar{y}) = \frac{\sum_{\bar{x} \in \{ x_1, \ldots, x_h \}} P(\bar{x}, \bar{y})}{N1'}.
\]

\[
\mu_c(\mu_{c'}(S)) = \langle T, V1'', P1'', C1'', \omega 1'' \rangle \), where \( V1'' = V1' - \{ v' \} = V - \{ v \} - \{ v' \}; C1'' = C1' \cup \{ (v', \{ x_1', \ldots, x_h' \}) \} = C \cup \{ (v, \{ x_1', \ldots, x_h' \}) \} \cup \{ (v', \{ x_1', \ldots, x_h' \}) \}; P1'' : dom(V1'') \rightarrow [0, 1] \).
Let
\[ N_{1''} = \sum_{\bar{y}'' \in \text{dom}(V_{1''})} \sum_{\bar{x}'' \in \{x_1, \ldots, x_h\}} P_{1''}(\bar{x}'', \bar{y}''). \]
For any \( \bar{y}'' \in \text{dom}(V_{1''}) \),
\[ P_{1''}(\bar{y}'') = \frac{\sum_{\bar{x}'' \in \{x_1, \ldots, x_h\}} P_{1''}(\bar{x}'', \bar{y}'')} {N_{1''}}. \]
By substituting \( P_{1''} \) (note that \( \bar{y}'' = (\bar{x}'', \bar{y}'') \)), we get
\[ N_{1''} = \sum_{\bar{y}'' \in \text{dom}(V_{1''})} \sum_{\bar{x}'' \in \{x_1, \ldots, x_h\}} \sum_{\bar{x}'' \in \{x_1, \ldots, x_h\}} P(\bar{x}, \bar{x}'', \bar{y}''). \]
For any \( \bar{y}'' \in \text{dom}(V_{1''}) \),
\[ P_{1''}(\bar{y}'') = \frac{\sum_{\bar{x}'' \in \{x_1, \ldots, x_h\}} \sum_{\bar{x}'' \in \{x_1, \ldots, x_h\}} P(\bar{x}, \bar{x}'', \bar{y}'')} {N_{1''}}. \]
Let
\[ N_{1'''} = N_{1'} \times N_{1''} = \sum_{\bar{y}''' \in \text{dom}(V_{1''})} \sum_{\bar{x}''' \in \{x_1, \ldots, x_h\}} P(\bar{x}, \bar{x}'', \bar{y}''). \]
Then for any \( \bar{y}''' \in \text{dom}(V_{1''}) \),
\[ P_{1'''}(\bar{y}''' ) = \frac{\sum_{\bar{x}''' \in \{x_1, \ldots, x_h\}} P(\bar{x}, \bar{x}'', \bar{y}''')} {N_{1'''}}. \]
Similarly, \( \mu_v(S) = (T, V_{2'}, P_{2'}, C_{2'}, \omega_{2'}) \), where \( V_{2'} = V - \{v\} \); \( C_{2'} = C \cup \{v', \{x_1, \ldots, x_h\}\} \); \( P_{2'} : \text{dom}(V_{2'}) \rightarrow [0, 1] \).
Let
\[ N_{2'} = \sum_{\bar{y}'' \in \text{dom}(V_{2'})} \sum_{\bar{x}'' \in \{x_1, \ldots, x_h\}} P(\bar{x}', \bar{y}''). \]
For any \( \bar{y}'' \in \text{dom}(V_{2'}) \),
\[ P_{2'}(\bar{y}'') = \frac{\sum_{\bar{x}'' \in \{x_1, \ldots, x_h\}} P(\bar{x}', \bar{y}'')} {N_{2'}}. \]
\( \mu_v(\mu_v'(S)) = (T, V_{2''}, P_{2''}, C_{2''}, \omega_{2''}) \), where \( V_{2''} = V_{2'} - \{v\} = V - \{v, v'\}; C_{2''} = C_{2'} \cup \{v', \{x_1, \ldots, x_h\}\} = C \cup \{v', \{x_1, \ldots, x_h\}\} \cup \{v, \{x_1, \ldots, x_h\}\} \); \( P_{2''} : \text{dom}(V_{2''}) \rightarrow [0, 1] \).
Let
\[ N_{2''} = \sum_{\bar{y}''' \in \text{dom}(V_{2''})} \sum_{\bar{x}''' \in \{x_1, \ldots, x_h\}} P_{2'}(\bar{x}, \bar{y}'''). \]
For any \( \bar{y}''' \in \text{dom}(V_{2''}) \),
\[ P_{2''}(\bar{y}''' ) = \frac{\sum_{\bar{x}''' \in \{x_1, \ldots, x_h\}} P_{2'}(\bar{x}, \bar{y}''')} {N_{2''}}. \]
By substituting $P_2'$ (note that $\tilde{y} = (\tilde{x}, \tilde{y}')$), we get
\[
N_{2''} = \frac{\sum_{\tilde{y}' \in \text{dom}(V_2')} \sum_{\tilde{x} \in \{x_1', \ldots, x_h\}} \sum_{x \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}')}{N_2'}.
\]

For any $\tilde{y}' \in \text{dom}(V_2')$,
\[
P_{2''}(\tilde{y}') = \frac{\sum_{\tilde{x} \in \{x_1', \ldots, x_h\}} \sum_{x \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}')}{N_2'}.
\]

Let
\[
N_{2'''} = N_{2'} \times N_{2''} = \sum_{\tilde{y}' \in \text{dom}(V_2')} \sum_{\tilde{x} \in \{x_1', \ldots, x_h\}} \sum_{x \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}').
\]

Then for any $\tilde{y}' \in \text{dom}(V_2')$,
\[
P_{2'''}(\tilde{y}') = \frac{\sum_{\tilde{x} \in \{x_1', \ldots, x_h\}} \sum_{x \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}')}{N_{2'''}},
\]

It can be seen that $V_1'' = V_2''$, $P_1'' = P_2''$, and $C_1'' = C_2''$. So $\mu_{c'}(\mu_{c'}(S))$ is equivalent to $\mu_{c'}(\mu_{c'}(S))$ or $\mu_{c'}(\mu_{c'}(S)) \equiv \mu_{c'}(\mu_{c'}(S))$.

Now let an SP-relation $S = \cup_{S \in \mathcal{S}}(S)$. Since the union operator is commutative, i.e. $\mu_{c'}(\mu_{c'}(S)) = \mu_{c'}(\cup_{S \in \mathcal{S}}(S)) = \cup_{S \in \mathcal{S}}(\mu_{c'}(S))$ and $\mu_{c'}(\mu_{c'}(S)) = \mu_{c'}(\cup_{S \in \mathcal{S}}(S)) = \cup_{S \in \mathcal{S}}(\mu_{c'}(S))$, we get $\mu_{c'}(\mu_{c'}(S)) \equiv \mu_{c'}(\mu_{c'}(S))$.

This proves $\mu_{c'}(\mu_{c'}(S)) \equiv \mu_{c'}(\mu_{c'}(S))$. $\square$

**Proof of Theorem 3.9.2**

First we prove that $\mu_{c \land c'}(S) \equiv \mu_{c'}(\mu_{c'}(S))$ for a single SPO $S$, where $S = \langle T, V, P, C, \omega \rangle$ be an SPO which is conditionalization-compatible; $c : v = \{x_1, \ldots, x_h\}$ and $c' : v' = \{x'_1, \ldots, x'_h\}$ are two atomic conditional selection conditions.

With the definition of conditionalization, we have $\mu_{c}(S) = \langle T, V_2', P_2', C_2', \omega \rangle$, where $V_2' = V - \{v\}$; $C_2' = C \cup \{(v', \{x'_1, \ldots, x'_h\})\}$; $P_2' : \text{dom}(V_2') \rightarrow [0, 1]$.

Let
\[
N_{2'} = \sum_{\tilde{y} \in \text{dom}(V_2')} \sum_{\tilde{x} \in \{x_1', \ldots, x_h\}} P(\tilde{x}, \tilde{y}).
\]

For any $\tilde{y} \in \text{dom}(V_2')$,
\[
P_2'(\tilde{y}) = \frac{\sum_{\tilde{x} \in \{x_1', \ldots, x_h\}} P(\tilde{x}, \tilde{y})}{N_{2'}}.
\]

$\mu_{c}(\mu_{c'}(S)) = \langle T, V_2'', P_2'', C_2'', \omega \rangle$, where $V_2'' = V_2' - \{v\} = V - \{v\} - \{v\}$; $C_2'' = C_2' \cup \{(v', \{x'_1, \ldots, x'_h\})\} = C \cup \{(v', \{x'_1, \ldots, x'_h\})\} \cup \{(v, \{x_1, \ldots, x_h\})\}$; $P_2'' : \text{dom}(V_2'') \rightarrow [0, 1]$.

68
Let
\[ N2'' = \sum_{\tilde{y}' \in \text{dom}(V2'')} \sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} P2'(	ilde{x}, \tilde{y}'). \]
For any \( \tilde{y}' \in \text{dom}(V2'') \),
\[ P2''(\tilde{y}') = \frac{\sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} P2'(	ilde{x}, \tilde{y}')} {N2''}. \]
By substituting \( P2' \) (note that \( \tilde{y} = (\tilde{x}, \tilde{y}') \)), so we get
\[ N2'' = \frac{\sum_{\tilde{y}' \in \text{dom}(V2'')} \sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} \sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}')} {N2'}. \]
For any \( \tilde{y}' \in \text{dom}(V2'') \),
\[ P2''(\tilde{y}') = \frac{\sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}')} {N2''}. \]
Let
\[ N2''' = N2' \times N2'' = \sum_{\tilde{y}' \in \text{dom}(V2'')} \sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} \sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}'). \]
Then for any \( \tilde{y}' \in \text{dom}(V2'') \),
\[ P2''(\tilde{y}') = \frac{\sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}')} {N2'''}. \]
Similarly, \( \mu_c \land c'(S) = \langle T, V1', P1', C1', \omega1' \rangle \), where \( V1' = V - \{v\} - \{v'\} \); \( C1' = C \cup \{(v, \{x_1, \ldots, x_h\}) \cup \{(v', \{x_1', \ldots, x_h'\}) \}; P1' : \text{dom}(V1') \rightarrow [0, 1]. \)
Let
\[ N1' = \sum_{\tilde{y}' \in \text{dom}(V1')} \sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} \sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}'). \]
Then for any \( \tilde{y}' \in \text{dom}(V1') \),
\[ P1'(\tilde{y}') = \frac{\sum_{\tilde{x} \in \{x_1, \ldots, x_h\}} P(\tilde{x}, \tilde{x}', \tilde{y}')} {N1'} \]
It can be seen that \( V1' = V2'', P1' = P2'', \) and \( C1' = C2'' \). So \( \mu_c \land c'(S) \) is equivalent to \( \mu_c(\mu_c'(S)) \). or \( \mu_c(\mu_c'(S)) \equiv \mu_c'(\mu_c(S)). \)

Now let an SP-relation \( S = \cup_{S \in S}(S). \) Since the union operator is commutative, i.e. \( \mu_c(\mu_c'(S)) = \mu_c(\cup_{S \in S}(S)) \) \( \mu_c(\cup_{S \in S}(S)) \) and \( \mu_c'(\cup_{S \in S}(S)) = \mu_c(\cup_{S \in S}(S)) \) \( \mu_c(\mu_c'(S)) \) \( \mu_c(\cup_{S \in S}(S)) \equiv \cup_{S \in S}(S) \).
This proves \( \mu_c \land c'(S) \equiv \mu_c(\mu_c'(S)). \)
\[ \square \]
Theorem 3.10  Let \( S, \mathcal{R} \) and \( B \) be three SP-relations. Then the following rules are valid.

1. \( S \times \mathcal{R} \equiv \mathcal{R} \times S \)
2. \( (S \times \mathcal{R}) \times B \equiv S \times (\mathcal{R} \times B) \)

Proof of Theorem 3.10.1

First we prove that \( S \times R \equiv R \times S \) for a single pair of SPOs \( S \) and \( R \), where \( S = (T, V, P, C; \omega) \) and \( R = (T', V', P', C', \omega') \) are two cp-compatible SPOs.

With the definition of Cartesian product, we have \( S \times R = S1' = (T1', V1', P1', C1', \omega1') \), where \( T1' = T \cup T' \); \( C1' = C = C' \); \( V1' = V \cup V' \); \( P1' : \text{dom}(V1') \rightarrow [0, 1] \).

For all \( \tilde{z} \in \text{dom}(V1') \); \( \tilde{z} = (\tilde{x}, \tilde{y}) \); \( \tilde{x} \in \text{dom}(V) \), \( \tilde{y} \in \text{dom}(V') \):

\[
P1'(\tilde{z}) = P'(\tilde{x}) \cdot P'(\tilde{y}).
\]

Similarly, \( R \times S = S2' = (T2', V2', P2', C2', \omega2') \), where \( T2' = T \cup T' \); \( C2' = C = C' \); \( V2' = V \cup V' ; P2' : \text{dom}(V2') \rightarrow [0, 1] \).

For all \( \tilde{z} \in \text{dom}(V2') \); \( \tilde{z} = (\tilde{x}, \tilde{y}) \); \( \tilde{y} \in \text{dom}(V) \), \( \tilde{x} \in \text{dom}(V') \):

\[
P2'(\tilde{z}) = P'(\tilde{x}) \cdot P(\tilde{y}).
\]

It can be seen that \( T1' = T2' \); \( C1' = C2' \); \( V1' = V2' \); and \( P1' = P2' \). So \( S \times R \) is equivalent to \( R \times S \), or \( S \times R \equiv R \times S \).

Now let \( S = \bigcup_{S \in \mathcal{S}}(S) \) and \( R = \bigcup_{R \in \mathcal{R}}(R) \) be two SP-relations. Since the union operator is commutative, i.e. \( S \times R = \bigcup_{S \in \mathcal{S}}(S) \times \bigcup_{R \in \mathcal{R}}(R) = \bigcup_{S \in \mathcal{S}} \bigcup_{R \in \mathcal{R}}(S \times R) \) and \( R \times S = \bigcup_{R \in \mathcal{R}}(R) \times \bigcup_{S \in \mathcal{S}}(S) = \bigcup_{R \in \mathcal{R}} \bigcup_{S \in \mathcal{S}}(R \times S) = \bigcup_{S \in \mathcal{S}} \bigcup_{R \in \mathcal{R}}(R \times S) \equiv \bigcup_{S \in \mathcal{S}} \bigcup_{R \in \mathcal{R}}(S \times R) \).

This proves \( S \times \mathcal{R} \equiv \mathcal{R} \times S \).

\( \Box \)

Proof of Theorem 3.10.2

First we prove that \( (S \times R) \times B \equiv S \times (R \times B) \) for a single triple of SPOs \( S, \mathcal{R} \) and \( B \), where \( S = (T, V, P, C, \omega) \); \( R = (T', V', P', C', \omega') \); and \( B = (T'', V'', P'', C'', \omega'') \) are three cp-compatible SPOs.

With the definition of Cartesian product, we have \( S \times R = S1' = (T1', V1', P1', C1', \omega1') \), where \( T1' = T \cup T' \); \( C1' = C = C' \); \( V1' = V \cup V' \); \( P1' : \text{dom}(V1') \rightarrow [0, 1] \).

For all \( \tilde{z} \in \text{dom}(V1') \); \( \tilde{z} = (\tilde{x}, \tilde{y}) \); \( \tilde{x} \in \text{dom}(V) \), \( \tilde{y} \in \text{dom}(V') \):

\[
P1'(\tilde{z}) = P'(\tilde{x}) \cdot P'(\tilde{y}).
\]
\[(S \times R) \times B = S1'' = \langle T1'', V1'', P1'', C1'', \omega 1'' \rangle, \text{ where } T1'' = T1' \cup T'' = T \cup T' \cup T'\prime; C1'' = C1' = C'' = C = C'; V1'' = V1' \cup V'' = V \cup V' \cup V''; P1'' : \text{dom}(V1'') \rightarrow [0, 1].\]

For all \(\tilde{z}' \in \text{dom}(V1'')\); \(\tilde{z}' = (\tilde{x}', \tilde{y})\); \(\tilde{x}' \in \text{dom}(V1')\), \(\tilde{w} \in \text{dom}(V'')\):

\[P1''(\tilde{z}') = P1'(\tilde{x}') \cdot P''(\tilde{w}).\]

By substituting \(P1'\) (note that \(\tilde{x}' = (\tilde{x}, \tilde{y})\)), we get

\[P1''(\tilde{z}') = P(\tilde{x}) \cdot P'(\tilde{y}) \cdot P''(\tilde{w}).\]

Similarly, \(R \times B = S2' = \langle T2', V2', P2', C2', \omega 2' \rangle\), where \(T2' = T' \cup T'' = T \cup T' \cup T'\prime; C2' = C' = C'' = C'' = C = C'' = C'' = C = C''; V2' = V'' \cup V'; P2' : \text{dom}(V2') \rightarrow [0, 1].\]

For all \(\tilde{z} \in \text{dom}(V2')\); \(\tilde{z} = (\tilde{y}, \tilde{w})\); \(\tilde{y} \in \text{dom}(V')\), \(\tilde{w} \in \text{dom}(V'')\):

\[P2'(\tilde{z}) = P(\tilde{y}) \cdot P'(\tilde{w}).\]

\[S \times (R \times B) = S2'' = \langle T2'', V2'', P2'', C2'', \omega 2'' \rangle, \text{ where } T2'' = T \cup T' \cup T'' = T \cup T' \cup T'\prime; C2'' = C = C' = C'' = C'' = C'' = C = C'' = C'' = C' = C'' = C'' = C'' = C = C'' = C'' = C = C'' = C'' = C = C''.\]

For all \(\tilde{z}'' \in \text{dom}(V2'')\); \(\tilde{z}'' = (\tilde{x}, \tilde{w}'')\); \(\tilde{x} \in \text{dom}(V)\), \(\tilde{w}'' \in \text{dom}(V2')\):

\[P2''(\tilde{z}'') = P(\tilde{x}) \cdot P2'(\tilde{w}'').\]

By substituting \(P2'\) (note that \(\tilde{w}'' = (\tilde{y}, \tilde{w})\)), we get

\[P2''(\tilde{z}'') = P(\tilde{x}) \cdot P'(\tilde{y}) \cdot P''(\tilde{w}).\]

It can be seen that \(T1'' = T2'', C1'' = C2'', V1'' = V2'', \text{ and } P1'' = P2''. \) So \((S \times R) \times B\) is equivalent to \(S \times (R \times B)\), or \((S \times R) \times B \equiv S \times (R \times B).\)

Now let \(S = \cup_{S \in S}(S), R = \cup_{R \in R}(R)\), and \(B = \cup_{B \in B}(B)\) be three SP-relations. Since the union operator is commutative, i.e. \((S \times R) \times B = (\cup_{S \in S}(S) \times \cup_{R \in R}(R)) \times \cup_{B \in B}(B) = \cup_{S \in S} \cup_{R \in R} \cup_{B \in B} ((S \times R) \times B)\) and \(S \times (R \times B) = \cup_{S \in S}(S) \times (\cup_{R \in R}(R) \times \cup_{B \in B}(B)) = \cup_{S \in S} \cup_{R \in R} \cup_{B \in B} ((S \times R) \times B)\).

This proves \((S \times R) \times B \equiv S \times (R \times B).\)

\[\square\]

### 3.5.2 Cost Model

The two parameters of the cost model for traditional query optimizer are disk I/Os and CPUs. In order to get relatively accurate estimates on those two parameters, a query engine needs to maintain certain statistics. Typical statistics maintained by a traditional query optimizer are listed as follows:
a) \( T(R) \), number of tuples in a relation \( R \).
b) \( V(R, a) \), the number of different values in the column of relation \( R \) for attribute \( a \). If there are great many values, then some type of histograms will be maintained.
c) Information about indexes.

These statistics, however, are not sufficient for SP-Algebra queries in an SP-database. Since we use the two-tier architecture, for complex SP-Algebra queries we may use other parameters to establish the cost model, i.e., use the number of tuples returned by the underlying RDBMS or the number of SPOs in the intermediate SP-relations. We propose the following statistics for an SP-database engine:

a) \( NS(S) \): number of SPOs in the SP-relation \( S \);
b) \( CON(S) \): total number of relational attributes in \(<\text{context}>\) of the SP-relation \( S \);
c) \( V(S, a) \): distinct values on sub-element \(<a>\) in \(<\text{context}>\) of the SP-relation \( S \);
d) \( VAR(S) \): total number of random variables in the SP-relation \( S \);
e) \( DM(S) \): average (or median) size of domain for random variables in the SP-relation \( S \);
f) \( NV(S) \): average (or median) number of random variables in each SPO of the SP-relation \( S \).
g) \( TSR \): average (or median) ratio of the number of rows in SPOs in the new SP-relation to the number of rows in SPOs in the original SP-relation.

Based on the statistics provided above, we define an initial set of cost models for various SP-Algebra queries. Note that the cost models are based on the number of SPOs \( NS(S) \) in the intermediate SP-relations, and the ratio of rows \( TSR \) as well in cases where the number of rows in the probability table changes.

(i) Selection operations:

- Select on context, \( R = \sigma_{\text{cont.} \cdot a = c}(S) \), then \( NS(R) = NS(S) \times \frac{1}{CON(S) \times V(S, a)} \).

By assuming uniform distributions for context attributes, the number of the SPOs selected in the original SP-relation is proportional to \( \frac{1}{CON(S) \times V(S, a)} \).

- Select on conditional, \( R = \sigma_{\text{cond.} \cdot v = x}(S) \), then \( NS(R) = NS(S) \times \frac{1}{VAR(S) \times DM(S)} \).

By assuming uniform distributions for random variables and variable values, the number of the SPOs selected in the original SP-relation is proportional to \( \frac{1}{VAR(S) \times DM(S)} \).

- Select on random variable, \( R = \sigma_{\text{var.} \cdot v \in \mathcal{V}}(S) \), then \( NS(R) = NS(S) \times \frac{NV(S) - 1}{(VAR(S) - 1)} \times \frac{1}{VAR(S)} \).

\[ 72 \]
By assuming uniform distributions for random variables, the number of the SPOs selected in the original SP-relation is proportional to \( \frac{(NV(S) - 1)}{(VAR(S))} \).

- Select on probability table, \( R = \sigma_{tbl.v=x}(S) \), then \( NS(R) = NS(S) * \frac{(NV(S) - 1)}{(VAR(S))} \).

By assuming uniform distributions for random variables and variable values, the number of the SPOs selected in the original SP-relation is proportional to \( \frac{(NV(S) - 1)}{(VAR(S))} \) and \( TSR = \frac{1}{DM(S)} \).

- Select on probability value, where \( c \in [0, 1] \).
  \[
  R = \sigma_{pro.p>c}(S) \text{ or } R = \sigma_{pro.p\geq c}(S), \text{ then } NS(R) = NS(S) \text{ and } TSR = 1 - c.
  \]
  \[
  R = \sigma_{pro.p\leq c}(S) \text{ or } R = \sigma_{pro.p< c}(S), \text{ then } NS(R) = NS(S) \text{ and } TSR = c.
  \]
  \[
  R = \sigma_{pro.p=c}(S), \text{ then } NS(R) = 1.
  \]
  \[
  R = \sigma_{pro.p\neq c}(S), \text{ then } NS(R) = NS(S).
  \]

By assuming uniform distributions for probability values, the number of the SPOs selected in the original SP-relation is proportional to the values specified above.

(ii) Projection operations:

- Project on context, \( R = \pi_{ctx.a}(S) \), then \( NS(R) = NS(S) * \frac{1}{CON(S)} \).

By assuming uniform distributions for context attributes, the number of the SPOs selected in the original SP-relation is proportional to \( \frac{1}{CON(S)} \).

- Project on conditional, \( R = \pi_{xld.v}(S) \), then \( NS(R) = NS(S) * \frac{1}{VAR(S)} \).

By assuming uniform distributions for random variables and variable values, the number of the SPOs selected in the original SP-relation is proportional to \( \frac{1}{VAR(S)} \).

- Project on random variable, \( R = \pi_{var.v}(S) \), then \( NS(R) = NS(S) * \left( \frac{NV(S) - 1}{VAR(S)} \right) \).

By assuming uniform distributions for random variables, the number of the SPOs selected in the original SP-relation is proportional to \( \frac{(NV(S) - 1)}{(VAR(S))} \).

(iii) Conditionalization operation:

- Conditionalize on probability table, \( R = \mu_{tbl.v=x}(S) \), then \( NS(R) = NS(S) * \left( \frac{NV(S) - 1}{VAR(S)} \right) \).
By assuming uniform distributions for random variables and variable values, the number of the SPOs selected in the original SP-relation is proportional to \( \frac{(N_{V(\mathcal{S})}-1)}{(V_{ARS(\mathcal{S})}-1)} \) and \( TSR = \frac{1}{DM(\mathcal{S})} \).

(iv) Cartesian product and join operations:

For Cartesian product and join operations, it is hard to estimate the size of an intermediate SP-relation. The major reason for this is that the operation (either Cartesian product or join) is only defined between two compatible SPOs, and it’s difficult to predict the number of pairs of SPOs which are compatible in two arbitrary SP-relations.
Chapter 4

Semistructured Probabilistic DBMS

In the previous chapter we introduced our SPO database framework. The framework was developed in the logical level, meaning that our definitions of the data model and SP-Algebra do not rely on any specific representation. Here we describe a specific implementation of the database framework — a semistructured probabilistic database management system (SPDBMS), which was built on top of a relational DBMS.

In order to implement the database framework, first we need a format to represent our SPOs. Since there are several applications in our project, and we need to transfer data between those applications and between the database and applications. Therefore, we require the data media being transferred to be able to cross platforms and cross machines from different vendors. XML is a good candidate for that. That is one of the major reasons why we choose XML to encode our SPOs. Second we need to establish a set of relational schema which allow us to store our SPOs in relational tables in the underlying relation database management system. Note that a SP-relation is a finite collection of SPOs and a SP-database is a finite collection of SP-relations. One important difference between semistructured probabilistic databases and classical relational or relational probabilistic databases is that different semistructured probabilistic objects conform to the same “schema” template and can be stored in the same SP-relation, whereas relational tuples over different schemas must be stored in separate relations.

Once the data is stored in the database, a query algebra-like query language (we call SPO-QL) has been designed to query the information stored in the database. A query processing engine is also developed to process the queries passed to the database. Among the most important components are the query optimizer, query translator, and post-processor. The query optimizer determines whether a query operation will be executed by the underlying relational DBMS or by SPDBMS itself; The query translator translates SP-algebra into a sequence of SQL statements which can be executed directly by the underlying RDBMS; The post-processor performs operations which can not be executed by the RDBMS, for instance, normalizing probability distributions for condition-alization operation. Instead of using Document Object Model (DOM), we designed a new data structure — an internal object model to represent intermediate XML objects. This object model is a compact representation of SP-relations at run time, and it facilitates the efficient retrieval of stored information.

This chapter is organized as follows. We discuss the XML-based implementation of the SPO
data model in Section 4.1, and show how SPOs are represented in XML with a meta-language SPO-ML. In Section 4.2 we introduce the overall architecture of our SPDBMS. In Section 4.3 we discuss our relational schema generation strategies for efficiently mapping XML representations of SPOs into relational tables, and the algorithms for creating SP-relation sand inserting SPOs into existing SP-relations. Then we present our query language and how to process SP-algebra within the SPDBMS query engine in Section 4.4. We describe our query translation mechanism, which automatically generates a sequence of SQL statements for evaluating SP-Algebra queries. Finally, we report on the results of the experiments testing the performance of the SPDBMS in Section 4.5.

4.1 Representation of SPOs

One of the major considerations in the design of the SPDBMS was the idea that it would take over the data management routine from complex AI applications dealing with uncertain data. Applications such as support of Bayesian network construction typically consist of different components, some of which serve to extract and/or elicit the probability tables while others serve to support the construction and further use of Bayesian networks given the data. If SPDBMS takes on the role of the data backbone of such an application, the issue of representation of SPOs for the purpose of passing the information between different components of the system becomes important. The representation mechanism must be transparent and easy to use by diverse applications.

Extensible Markup Language (XML) [10] provides us with such a representation framework. It affords us the benefit of using clear APIs for parsing and processing XML data, together with open source software implementing these tasks, relieving the SPDBMS from the need to do its own syntactic parsing. This makes SPO data encoded in XML easy to pass from component to component.

To be more specific, we represent SPOs in XML using a markup meta-language we call SPO-ML. Among a number of possible XML encodings of SPOs we choose the approach that is independent of any application domain. Thus, the actual DTD/XML schema of the markup language does not depend on the application domain, namely the pair \( \langle V, R \rangle \). SPO-ML simply represents the general markup rules for any application domain.

For example, consider an arbitrary application domain with a universe \( V \) of random variables \( \{v_1, v_2, \ldots, v_n\} \) and a collection of context attributes \( R = \{A_1, A_2, \ldots, A_k\} \), we can use the same markup language as shown on the template DTD in Figure 4.1 ¹.

¹DTD representation of SPO-ML is chosen here for its simplicity and succinctness. We also maintain and use the corresponding XML schemas. See Appendix B for details.
Figure 4.1: XML DTD template for SPOs.

Figure 4.2 (right) shows the SPO-ML encoding of the SPO shown on the left side. The top layer of the XML document for the SPO consists of three elements `<context>`, `<table>` and `<conditional>`. The path is represented as an attribute for the `<spo>` element. The contents of context and conditional parts are fairly straightforward. The probability table consists of a `<variable>` element, which contains a sequence of names of all the participating random variables in the SPO, and a collection of `<variable>` elements, each of which consists of a sequence of values for corresponding participating random variables and the probability for this row.

Semistructured Probabilistic Objects are complex structures and not all their properties can be captured by XML validity checks. An SPO representation in SPO-ML should satisfy the following extra validity constraints. First, all `<row>` elements inside the `<table>` elements have to have exactly the same number of `<val>` elements. Second, the content of `<P>` elements inside `<row>` elements is expected to be real numbers between 0 and 1, and their sum must be less than or equal to 1. These additional constraints mean that validation on an XML representation of an SPO is a two-step process: first the XML is validated against the appropriate DTD/XML schema, and then the additional constraints are verified. While the validity of the SPO-ML documents is checked by a validating parser, these extra checks are performed by the SPDBMS itself.
Figure 4.2: A typical SPO object for risk level on Driver Age and License Years, and its XML representation.

4.2 Architecture of SPDBMS

We have implemented a prototype semistructured probabilistic database system on top of a RDBMS in Java, JDK1.3. Figure 4.3 depicts the overall architecture of our system. The core of the system is the SPDBMS application server which processes query requests from a variety of client applications. The application server provides a JDBC-like API, through which client applications can send standard database management instructions, such as CREATE DATABASE, DROP DATABASE, CREATE SP-RELATION, DROP SP-RELATION, INSERT INTO SP-RELATION, DELETE FROM SP-RELATION, as well as SP-Algebra queries to the server.

4.3 Data Storage for SPOs

A relational database system has been used as a backend to store SPO-ML encoded data by mapping the XML schema onto a set of relational tables. While numerous techniques for converting XML documents into relational databases exist [24, 34, 55, 81], as shown in [81] none of the proposed translation schemes is monotonically better than others. These schemes are proposed for
storage of arbitrary XML with unknown structure. In the case of storing SPO-ML \(<spo>\) elements in relational databases, we can take advantage of our knowledge of the general structure of these elements when designing the translation mechanism. This consideration lead us to adopt a translation scheme, described below, that is specific to the structure of SPO-ML object in lieu of a generic mechanism.

### 4.3.1 Mapping SPOs to relational tables

The SPO-ML to relational database translation works as follows. SPOs are stored in a set of relational tables with schema defined in Figure 4.4.

In order to improve data integrity and query performance, besides primary keys and foreign keys, we also created indexes for the last three type of relations, for instance, multicoloumn index
**RELATION (rid integer, name varchar, schema varchar)** contains SP-relation level information. It connects all other tables by using the table naming convention that every table uses the unique identifier of the corresponding SP-relation rid as a prefix for its name. The attributes name and schema represent the SP relation name and the corresponding schema URL, respectively.

**rid_SPO (id integer, path varchar, head varchar, numvar integer)** contains SPO level information. The association between this table and other tables is established by the unique identifier of an SPO id. The attribute head stores the prolog of an XML document and numvar stores the number of participating random variables in an SPO.

**rid_SPO_CONS (id integer, type char, elemname varchar, elemvalue varchar, idref varchar)** contains all the information about SPO context and conditional. The attribute id is a foreign key, and type tells whether it’s a context or conditional. The attributes elemname and elemvalue give the pair of element name and element value.

**rid_SPO_VAR (id integer, position integer, varname varchar)** contains all the information of SPO participating variables. The attributes position and varname represent a pair of position and variable name.

**rid_SPO_num (id integer, var_1 char, ... , var_num char, p decimal)** contains all the information of the probability tables for SPOs which have num participating random variables. The attribute p stores the probability value.

---

*a* represents keys to the relational tables.

*b* num is a variable, which equals the number of participating variables in a particular SPO. Thus, num may vary from 1 to |\mathcal{V}|.

---

Figure 4.4: A set of relational schema for storing SPOs

on (id, elemname) for relation rid_SPO_CONS, multicolumn index on (id, varname) for relation rid_SPO_VAR and multicolumn index on (id, var_1, ..., var_num) for relation rid_SPO_num.

### 4.3.2 Algorithms for Populating SP-Relations

The database system stores SPOs from each SP-relation in a separate set of relational tables. The CREATE algorithm, as shown in Figure 4.5, starts by checking whether the SP-relation specified already exists. If not, it stores the SP-relation name and the schema url, and generates a unique identifier rid for the SP-relation. It also creates all the empty tables with the schema defined above and associates them with the SP-relation. In particular, all the relation names have a suffix rid. In order to store SPOs in an SP-relation, the SPOs must be validated against a schema
template and parsed into a Document Object Model (DOM) tree and then decomposed into five components, Head, Path, Context, Table and Conditional, according to the predefined relational schema generation technique. The INSERT algorithm, as shown in Figure 4.6, gets a unique SPO object identifier id, then stores the XML prolog information, the path and the number of participating random variables in the SPO in the table rid_SPO. It stores SPO context and conditional in the table rid_SPO_CONS, the probability table in the table rid_SPO_num and participating random variables in the table rid_SPO_VAR, respectively. Figure 4.7 shows the resulting tables after storing the SPO defined in Figure 3.1 in an SP-relation.

<table>
<thead>
<tr>
<th>Algorithm CREATE (String SPR, String schema)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Query the table RELATION to obtain the unique SP-relation identifier rid for SP-relation SPR.</td>
</tr>
<tr>
<td>2. If SPR exists, return false.</td>
</tr>
<tr>
<td>3. Query the table RELATION to obtain the next available unique SP-relation identifier rid.</td>
</tr>
<tr>
<td>4. Insert the SP-relation name SPR and the schema URL schema into the table RELATION.</td>
</tr>
<tr>
<td>5. Create all the empty tables with relational schema defined above, namely rid_SPO, rid_SPO_CONS, rid_SPO_VAR and rid_SPO_num, where num varies from 1 to 4.</td>
</tr>
<tr>
<td>6. return true.</td>
</tr>
</tbody>
</table>

Figure 4.5: Pseudo-code for algorithm CREATE.

4.4 Querying the SPDBMS

The SP-Algebra operations described in previous sections have been implemented. The query language allows us to navigate through the entire database with structured queries, including any combination of SELECTION, PROJECTION, CONDITIONALIZATION, CARTESIAN PRODUCT and JOIN. It also allows for conjunctive, disjunctive queries and negation operations.

4.4.1 Query Language (SPO-QL)

Currently we are using a query algebra-like query language for the SPDBMS. We list the language grammar in Figure 4.8. Note that, here we only allow the left join for the join operation since right join can be easily expressed in terms of left join, i.e. $S_1 \times S_2$ is equivalent to $S_2 \times S_1$. 

81
Algorithm INSERT (String SPR, InputStream xmlDoc)
1. Query the table RELATION to obtain the unique SP-relation identifier rid for SP-relation SPR.
2. If SPR does not exist, return false.
3. Validate and parse the XML document xmlDoc into XML DOM tree.
4. Extract the XML prolog and the list of SPO subtrees.
5. Dissemble each SPO subtree into five components, namely context, participating variables, table, conditional and path.
6. Insert XML prolog, path and the number of participating random variables num in the SPO into the table rid_SPO.
7. Insert context and conditional into the table rid_SPO_CONS.
8. Insert participating variables into the table rid_SPO_VAR.
9. If num is greater than 4 and table rid_SPO_num does not exist in the database, then a new table rid_SPO_num will be created before inserting the probability table.
10. Insert the values of the probability table into the table rid_SPO_num based on the number of random variables num in the SPO.
11. return true.

Figure 4.6: Pseudo-code for algorithm INSERT.

Also we use the traditional join symbol \(\bowtie\) instead to represent the left join in the query language.

4.4.2 Query Parsing

In the current implementation the query processing proceeds as follows. Structured queries are first parsed and then transformed into a query tree. A typical SP-algebra query is

\[
\pi_{\text{var.LY}}(\mu_{\text{tbl.DA} \bowtie B}(R_1) \bowtie_{\sigma_{\text{pro.P} > 0.2}}(R_2))
\]

and its query parse tree is shown in Figure 4.9. The leaf nodes and internal nodes of the query tree denote the local SP-relations and SP-algebra operators, respectively.

The SPDBMS query engine determines which operations get executed by the underlying relational DBMS engine and which operations get executed by itself, and pushes as much query computation as possible to the underlying RDBMS. The SPDBMS query engine is shown in Figure 4.10. Operations executed by the RDBMS are translated in a straightforward manner into a sequence.
of corresponding SQL statements that can be executed directly by the underlying RDBMS. The returned tuple stream is then transformed into an instance of the internal object model (for short, IOM, discussed below), and subsequent queries are performed against the IOM instance in the postprocessor. Finally the XML generator does the assembly of the resulting XML document.

As multiple equivalent query trees can be generated from a single SP-algebra query, the query tree must be carefully chosen to reflect the query optimization strategy adopted by the query engine\(^2\). As part of the query optimization (discussed later), the query tree may be simplified by a set of heuristic rules to reduce its processing cost.

### 4.4.3 SP-Algebra Query Translation

Here we give examples of query translations for two types of probabilistic queries, illustrating how to map from an SP-Algebra query to a sequence of SQL statements; other translations can be found in Appendix A. First, consider selection on probabilities. Given a selection condition \( P \) op \( \ddot{x} \) and an SP-relation \( \mathcal{S} = \{S_1, \ldots, S_n\} \), this operation returns SPOs that have at least one row with probability that satisfies the selection condition. Selection preserves the context, participating random variables and conditionals in the original SPOs, but returns only those rows of the proba-

\(^2\)Work on query optimization in SPDBMS is underway, but the current version of the SPDMBS server does not have this feature.

---

<table>
<thead>
<tr>
<th>id</th>
<th>var_1</th>
<th>var_2</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>A</td>
<td>A</td>
<td>0.09</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>B</td>
<td>0.12</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>C</td>
<td>0.03</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>F</td>
<td>0.005</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>A</td>
<td>0.12</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>B</td>
<td>0.16</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>C</td>
<td>0.13</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>F</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>A</td>
<td>0.03</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>B</td>
<td>0.08</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>C</td>
<td>0.11</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>F</td>
<td>0.045</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>A</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>B</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>C</td>
<td>0.02</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>F</td>
<td>0.04</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>schema_url</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First</td>
<td>schema_url</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>id</th>
<th>path</th>
<th>head</th>
<th>numvar</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>S</td>
<td>null</td>
<td>2</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>id</th>
<th>position</th>
<th>varname</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>DA</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>LY</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>elemname</th>
<th>elemvalue</th>
<th>idref</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>cont</td>
<td>race</td>
<td>Asian</td>
<td>null</td>
</tr>
<tr>
<td>2</td>
<td>cond</td>
<td>DR</td>
<td>A</td>
<td>null</td>
</tr>
<tr>
<td>2</td>
<td>cond</td>
<td>DR</td>
<td>B</td>
<td>null</td>
</tr>
</tbody>
</table>

---

Figure 4.7: Internal representation for the SPO defined in Figure 3.1.
Figure 4.8: The grammar of the query language for SPDBMS.

Selection conditions are represented in a conjunctive normal form (CNF).

Our second example is the operation of conditionalization. This operation computes conditional probability distributions and thus, is specific to probabilistic algebras. Given the constraint $v = a$ and an SP-relation $S = \{S_1, \ldots, S_n\}$, conditionalization $\mu_{tk,v=a}$ first selects SPOs in which $v$ is a participating random variable. For each selected SPO, conditionalization preserves the context, conditions the joint probability distribution, replaces the probability table with a new conditional probability distribution over the remaining random variables, and adds the condition $(v, \{a\})$ to the conditional part of each of the resulting SPOs. Consider a sample query $\sigma_{pr.a,P>0.1}(S)$. In order to perform the conditionalization operation, first a sequence of SQL statements, as shown in Figure 4.12, need to be issued against the underlying RDBMS to retrieve all the SPOs satisfying the condition. This, however, is not enough for the conditionalization operation since this operation changes the probability tables in the results according to the semantics of the operation. These com-
putations are performed by the postprocessing process, and the postprocessing algorithm is shown in Figure 4.13.

4.4.4 Internal Object Model (IOM)

When a client application sends queries to the SP-database, the underlying RDBMS could not handle them all and there always are intermediate XML objects. In a typical XML database system, a generic DOM tree (or XML view) is used to represent those XML objects. In our approach, since we know in advance our DTD/XML schema template we use an internal object model instead, as shown in Figure 4.14, for efficient manipulation. Basically, an SP-relation object consists of a set of SPO objects and each SPO object consists of three objects, namely Context, Table and Conditional. The component Path is included in SPO object and the component Variable is included in the Table object.

4.5 Experiments

In this section we present the results of tests conducted with the prototype system. The algorithms for storing and querying SPO objects in a relational database are completely independent of
the underlying RDBMS. All features specific to a RDBMS are implemented in the special database adapter layer, so the system can port to any relational database with little modification.

4.5.1 System Environment

The current system uses Oracle8i as the RDBMS back-end. To avoid network delays during tests, both the application server and Oracle DB server were running on the same machine, a 440 MHz Sun Ultra 10 running Solaris OS with 1GB of main memory. For all the experiments we allocate 256M memory for the JVM and the timing was done on the server side.
step 1. Get the SPO ID list based on given probability value:
SELECT DISTINCT id
FROM rid_SPO_1
WHERE p > 0.1
...
UNION ALL
SELECT DISTINCT id
FROM rid_SPO_i
WHERE p > 0.1
...
UNION ALL
SELECT DISTINCT id
FROM rid_SPO_max
WHERE p > 0.1

step 2. Get the variable list for each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}

step 4. Retrieve probability table
SELECT *
FROM rid_SPO_1
WHERE id IN {ID list}
  AND p > 0.1
...
UNION ALL
SELECT *
FROM rid_SPO_i
WHERE id IN {ID list}
  AND p > 0.1
...
UNION ALL
SELECT *
FROM rid_SPO_max
WHERE id IN {ID list}
  AND p > 0.1

Figure 4.11: Steps to evaluate the selection query $\sigma_{p > 0.1}(S)$
Figure 4.12: Steps to evaluate the conditionalization query \( \mu_{\text{cond}, DA='A'}(S) \)

- step 1. Store the retrieved SPOs in predefined data structures;
- step 2. Remove the entire column for attribute DA;
- step 3. Compute the sum of all the probabilities;
- step 4. Divide all the probabilities by the computed sum;
- step 5. Add a new condition (i.e., DA = 'A') in the conditional part;
- step 6. Assemble the final SPOs into an XML document.

Figure 4.13: Postprocessing algorithm for the conditionalization query

In order to ensure consistency, each experiment consists of 20 runs, and each point on a graph represents the average running time for the 20 runs. We also restarted the application server for each experiment to minimize the time difference consumed by garbage collection. Most test data
sets used in the experiments are generated randomly by a custom data generator\(^5\). However, for Cartesian product and join, we generated specific data sets in order to control the selectivity for each query. Each data set was generated based on the following three parameters: number of SPOs per SP-relation, number of participating random variables in an SPO, and size of the domain of participating random variables. The first parameter affects the number of objects to be stored in the database while the other two affect the size of individual objects. The number of SPOs specifies the total number of individual SPOs in an XML document, which are inserted into an SP-relation during the test. Throughout the experiments, we used a fixed number of context and conditional elements in a single SPO. So the last two parameters specify the internal structure of each SPO and consequently the size of each SPO. Table 4.1 shows some typical data sets with corresponding file size and number of tuples in the underlying Oracle database. Please also note that both file size and number of tuples increase linearly with the number of SPOs, increase nearly exponentially with number of variables, and increase polynomially with number of domain size for random variables.

### 4.5.2 Experimental Results

We examined the running time for each type of atomic SP-Algebra query. Most queries are generated randomly at runtime by a custom query generator\(^3\) in the client application running on

---

\(^3\)Both the SPO data generator and the SP-Algebra query generator utilize a linear congruential pseudo-random number generator, which comes with Sun JDK1.3 package.
Table 4.1: File size and number of tuples in Oracle database for typical data sets.

<table>
<thead>
<tr>
<th>number of SPOs</th>
<th>1,000</th>
<th>1,000</th>
<th>1,000</th>
<th>10,000</th>
<th>10,000</th>
<th>10,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of variables</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>size of domain</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>size of original XML file (MB)</td>
<td>0.38</td>
<td>1.64</td>
<td>1.10</td>
<td>3.81</td>
<td>16.4</td>
<td>11.0</td>
</tr>
<tr>
<td>number of tuples in Oracle DB</td>
<td>10,000</td>
<td>24,000</td>
<td>22,000</td>
<td>100,000</td>
<td>240,000</td>
<td>220,000</td>
</tr>
</tbody>
</table>

Another machine, an 864 MHz Pentium III running Linux with 512MB of main memory. During the experiment, we have collected both the total running time and the time consumed by the Oracle DB server for executing SQL statements. The Oracle server consumes 75 - 95% of the total execution time for most queries, and the percentage increases with the size of the XML files. A typical case is shown in Table 4.2.

Table 4.2: Time distribution between Oracle and postprocessing for data set of 10,000 SPOs with 3 participating variables and domain size of 2.

<table>
<thead>
<tr>
<th>Test type</th>
<th>Total time/sec</th>
<th>Oracle time/sec</th>
<th>Postprocessing/sec</th>
<th>%Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select on context</td>
<td>1.193</td>
<td>1.053</td>
<td>0.140</td>
<td>88</td>
</tr>
<tr>
<td>Select on conditional</td>
<td>0.955</td>
<td>0.845</td>
<td>0.110</td>
<td>88</td>
</tr>
<tr>
<td>Select on variable</td>
<td>1.081</td>
<td>0.959</td>
<td>0.122</td>
<td>88</td>
</tr>
<tr>
<td>Select on table</td>
<td>0.736</td>
<td>0.661</td>
<td>0.075</td>
<td>89</td>
</tr>
<tr>
<td>Project on conditional</td>
<td>1.080</td>
<td>0.958</td>
<td>0.122</td>
<td>88</td>
</tr>
<tr>
<td>Project on variable</td>
<td>0.502</td>
<td>0.471</td>
<td>0.031</td>
<td>93</td>
</tr>
<tr>
<td>Conditionalization</td>
<td>0.769</td>
<td>0.619</td>
<td>0.150</td>
<td>80</td>
</tr>
</tbody>
</table>

To study the effects of the number of SPOs in an SP-relation on the query running time, different experiments were conducted with the number of SPOs varying from 10 to 10,000. The results are plotted in Figure 4.15. It can be observed that all types of unary SP-Algebra queries scale well as the size of SP-relations increases: The reason is that, as the number of SPOs in an SP-relation increases, both the number of disk page I/Os and postprocessing time increase. Moreover, the running time increases sublinearly with the number of SPOs for large SP-relations, but at a much slower rate for SP-relations of small size. In Figure 4.16, the effects of domain size for participating variables are shown. Notice that the number of tuples considered grows polynomially (i.e., quadratically in this case, the number of variables equals 2) with the size of the domain. The running time increases with the size of domain, but at a slower rate – especially considering that the size of XML files increases quadratically with the size of domain. In particular, the running time increases 50% – 150% as the size of domain increases from 2 to 5.

Figure 4.17 shows the effects of the number of variables in SPOs on the time for the conditionalization operation. The effect of running time on the size of the SP-relation and query selectivity
for selection on probability is shown in Figure 4.18. We can see that the running time increases with selectivity faster at lower selectivity and increases with the number of SPOs linearly. As we can see, running time increases with the number of participating variables, and at a relatively faster rate for larger SP-relations than for smaller ones. However, the increase in execution time is smaller
compared to the quadratic increase in size of XML files with the number of random variables. The same effect has been seen for selection and projection operations as well. Finally, Figure 4.19 shows the dependence of running time on the size of the SP-relation and query selectivity for the Cartesian product operation. The graph shows that the running time for Cartesian product increases with the
number of SPOs at a nearly quadratic rate, and also increases with the selectivity, but at a much slower rate. One reason is that the number of SPOs output increases quadratically with the number of SPOs in the initial SP-relations. The same effect can be seen for the join operation, as shown in Figure 4.20.

Figure 4.17: Effect of number of variables on conditionalization operation.

Figure 4.18: Effect of number of SPOs and selectivity on selection query on probability.

Figure 4.19: Effect of number of SPOs and selectivity on Cartesian product.

Figure 4.20: Effect of number of SPOs and selectivity on join operation.

4.6 Summary of the SPDBMS

Based on the SPO framework described in Section 3, a semistructured probabilistic database management system (SPDBMS) has been designed and implemented on top of a relational DBMS. We discussed the XML-based implementation of the SPO data model and showed how SPOs are
represented in XML with a meta-language SPO-ML. We described our relational schema generation strategies for efficiently mapping XML representations of SPOs into relational tables. We presented our query language and translation mechanism which automatically generates a sequence of SQL queries for evaluating SP-Algebra queries. We described the overall architecture of the SPDBMS and its query engine. An internal object model was designed to represent intermediate XML objects and it facilitates the efficient retrieval of stored information. The experimental results of our database management system indicate that the SPDBMS performs well in terms of response time and scalability even without query optimization.
Chapter 5

Extended Semistructured Probabilistic Database Framework

One of the restrictions of the SPO framework described in Chapter 3 is the scope of the context and conditionals in an SPO. Every single attribute of context and every conditional entry relates to the entire probability distribution in SPOs. It would be convenient to specify contextual properties of individual random variables as well as to be able to associate conditionals with single random variables or proper subsets of the set of all participating random variables of an SPO. Another major difference of the extended SPO framework from SPO framework is the use of probability intervals to represent uncertainty. We will denote $C[0,1]$ as the set of all subintervals of interval $[0,1]$.

In what follows, we extend the definition of an SPO to handle scopes of context and conditionals, as well as interval probabilities. We give an overview of the extended framework followed by the detailed description.

5.1 Overview of the ESPO framework

We also start with the Extended SPO data model.

Let $C[0,1]$ be a set of all subintervals of the interval $[0,1]$. An Extended Semistructured Probabilistic Object (ESPO) $S$ is a tuple $S = (T^+, V, P, C^+, \omega)$, where $T^+$ is extended context over some schema $R$ and $V; V = \{v_1, \ldots, v_q\} \subseteq \mathcal{V}$ is a set of random variables that participate in $S; P : \text{dom}(V) \rightarrow C[0,1]$ is the probability table of $S$. $P$ must be consistent (see Definition 2.3 in Section 2.2); $C^+ = \{(u_1, X_1, V_1), \ldots, (u_n, X_n, V_n)\}$ is a set of extended conditionals over $V$, and $\{u_1, \ldots, u_n\} \cap V = \emptyset; \omega$, called a path expression or path of $S$ is an expression in the Extended Semistructured Probabilistic Algebra.

As we can see, the Extended SPO model extends the flexibility of the SPO model with the two important features: (i) support for interval probabilities and (ii) association of context and conditionals with individual random variables.

We describe the ESPO data model in Section 5.2 (We refer the readers to Section 2.2 for the underlying semantics for interval probability distributions.). In Section 5.3, we describe the Extended Semistructured Probabilistic Algebra (ESP-algebra). Besides similar algebra operations to those described in 3.3, we include new operations for selection and projection operations to handle queries on the extended context and extended conditionals. These are relatively straightforward.
However, it is nontrivial to extend the algebra operations to handle interval probability distributions correctly based on the laws of probability theory, especially for conditionalization and join operations. Dekhtyar and Goldsmith [19] solved the problem of computing the result of conditionalization of interval probability distributions. We present here their final formula, while referring the reader to [19] for a more detailed discussion of its derivation and discussion. Please note, however, that as Jaffray [54] pointed out, interval conditional probability estimates will not be perfect, and the unfortunate consequence of this is that conditionalizing is not commutative. The same problem occurs for join operation since join operation utilizes the result of conditionalization operation.

In the sections that follow, we will assume that all ESPOs under consideration have consistent and tight probability distribution functions. Using the tightening operator according to Theorem 2.2 will allow us to replace any probability distribution function that is not tight with its tight equivalent.

**Definition 5.1** An Extended Semistructured Probabilistic Object \(S = \langle T^+, V, P, C^+, \omega \rangle\) is consistent iff \(P\) is consistent. Also, \(S\) is tight iff \(P\) is tight.

We describe in details the data model and query algebra for the extended version of the SPO framework.

### 5.2 Extended Semistructured Probabilistic Object (ESPO) Data model

**Definition 5.2** Let \(\mathcal{R}\) be a context schema. Let \(V\) be a set of random variables. An extended context over \(\mathcal{R}\) and \(V\) is a tuple \(T^+ = \langle (A_1, a_1, V_1), \ldots, (A_n, a_n, V_n) \rangle\), where (i) \(A_i \in \mathcal{R}, 1 \leq i \leq n\); (ii) \(a_i \in \text{dom}(A_i), 1 \leq i \leq n\); (iii) \(V_i \subseteq V, 1 \leq i \leq n\).

Intuitively, extended context is organized as follows. Given the context \(T\) of a simple SPO and the set of participating random variables \(V\), we associate with each context value the set of random variables for which it provides additional information.

**Example 5.1** Consider the context attributes of the Simple SPO in Figure 5.1 (top). The \texttt{ld}, \texttt{population} and \texttt{date} attributes relate to the entire object. At the same time, we would like to represent the fact that 323 survey respondents indicated the intention to vote on the park construction question while 342 respondents responded to the question about their vote on the AI conference legalization. Without extended context, we can include both \texttt{responses:323} and \texttt{responses:342} in the SPO but we cannot associate the occurrences of the attributes with individual random variables. The SPO...
with extended context in the bottom left of the Figure 5.1 shows how extended context alleviates this problem.

![Figure 5.1: Simple vs. Extended context and conditionals in SPOs](image)

We note that in Example 5.1 the other two context attributes, population and date have the scope over the entire set of random variables participating in the SPO. This can be represented explicitly, by specifying the entire set. However, we assume that whenever the scope is not specified for a context attribute, the scope of the attribute is the entire set of participating random variables (as we did here).

Similarly to context, we extend the conditionals.

**Definition 5.3** Let \( C = \{(u_1, X_1), \ldots, (u_n, X_n)\} \) be a set of conditionals and \( V \) be a set of random variables s.t., \( V \cap \{u_1, \ldots, u_n\} = \emptyset \). The set of extended conditionals \( C^+ \) is defined as \( C^+ = \{(u_1, X_1, V_1), \ldots, (u_n, X_n, V_n)\} \), where \( V_i \subseteq V, 1 \leq i \leq n \).
Extended conditionals are more subtle than extended context, but they are useful in a number of situations.

**Example 5.2** To continue the previous example, consider now the conditional part of the SPO on the top of Figure 5.1. The condition `senate=Donkey` applies to the entire distribution. In the SPO model only such conditions can be expressed. As [20, 44] note, this leads to significant restrictions put on query algebra operations of join and Cartesian product: these two operations are defined only for pairs of SPOs with identical conditional parts. By extending conditionals to specify scope, as shown in the bottom right SPO on Figure 5.1, we can extend the expressive power of the framework. The particular SPO in question could have originated as a result of a Cartesian product (See Section 5.3.4) of an SPO containing information about park initiative votes of people who prefer the Donkey party candidate for the Senate and an SPO containing voter preferences on the AI conference legalization initiative. In the SPO model of [20] this operation would not have been possible.

We can now give the definition of an Extended SPO (ESPO).

**Definition 5.4** Let $C[0,1]$ be a set of all subintervals of the interval $[0,1]$ An Extended Semistructured Probabilistic Object (ESPO) $S$ is a tuple $S = (T^+, V, P, C^+, \omega)$, where

- $T^+$ is extended context over some schema $R$ and $V$;
- $V = \{v_1, \ldots, v_q\} \subseteq V$ is a set of random variables that participate in $S$. We require that $V \neq \emptyset$;
- $P : dom(V) \rightarrow C[0,1]$ is the probability table of $S$. $P$ must be consistent (see Definition 2.3 in Section 2.2);
- $C^+ = \{(u_1, X_1, V_1), \ldots, (u_n, X_n, V_n)\}$ is a set of extended conditionals over $V$, and $\{u_1, \ldots, u_n\} \cap V = \emptyset$, and
- $\omega$, called a path expression or path of $S$ is an expression in the Extended Semistructured Probabilistic Algebra.

The exact syntax and construction of paths will be explained in Section 5.3.

**Example 5.3** Figure 5.2 shows the anatomy of ESPOs. The object in the figure represents a joint probability distribution of votes in the Senate race and for the AI conference legalization ballot
initiative for male voters who chose to vote Donkey for mayor. The distribution is based on the
survey that took place on October 23; 238 respondents indicated their vote in the Senate race, 195
in the legalization vote, with 184 of the respondents giving both answers.

<table>
<thead>
<tr>
<th>ω:</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>date:</td>
<td>October 23</td>
</tr>
<tr>
<td>gender:</td>
<td>male</td>
</tr>
<tr>
<td>respondents:</td>
<td>238, {senate}</td>
</tr>
<tr>
<td>respondents:</td>
<td>195, {legalization}</td>
</tr>
<tr>
<td>overlap:</td>
<td>184</td>
</tr>
<tr>
<td>senate</td>
<td>legalization</td>
</tr>
<tr>
<td>Rhino yes</td>
<td>[0.04, 0.11]</td>
</tr>
<tr>
<td>Rhino no</td>
<td>[0.1, 0.15]</td>
</tr>
<tr>
<td>Donkey yes</td>
<td>[0.22, 0.27]</td>
</tr>
<tr>
<td>Donkey no</td>
<td>[0.09, 0.16]</td>
</tr>
<tr>
<td>Elephant yes</td>
<td>[0.05, 0.13]</td>
</tr>
<tr>
<td>Elephant no</td>
<td>[0.21, 0.26]</td>
</tr>
<tr>
<td>mayor = Donkey {senate, legalization}</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.2: Extended Semistructured Probabilistic Object

We extend the notion of equivalence for ESPOs.

**Definition 5.5** Let \( S = \langle T^+, V, P, C^+, ω \rangle \) and \( S' = \langle T'^+, V', P', C'^+, ω' \rangle \) be two ESPOs. We say that \( S \) is equivalent to \( S' \), denoted \( S \equiv S' \), iff \( T^+ = T'^+ \), \( V = V' \), \( P = P' \) and \( C^+ = C'^+ \).

**Note:** When representing ESPOs we assume that a lack of random variables after a context attribute or a conditional indicates that it is associated with all participating random variables. Therefore, strictly speaking, we did not need to explicitly include the list of associations for the mayor=Donkey conditional in Figure 5.2; we did it to make a point that the conditional part has extended syntax.

### 5.3 Extended Probabilistic Semistructured Algebra

We have described the ESPO data model and the underlying semantics for interval probability distributions (See Section 2.2). We are now in position to define the Extended Probabilistic Semistructured Algebra (ESP-Algebra). As in Section 3.2, we will give definitions for five major
operations on the objects: selection, projection, Cartesian product, join and conditionalization. In [87] we have described how these operations can be defined in a query algebra for interval probability distributions only (without context and conditionals) in a generic way. Here, we ground the operations described in [87] in the ESPO data model.

The first four operations are extensions of the standard relational algebra operations. However, these operations will be expanded significantly in comparison both with classical relational algebra [72] and with the definitions in [20]. The fifth operation, conditionalization, is specific to probabilistic databases and represents the procedure of constructing an ESPO containing a conditional probability distribution given an ESPO for a joint probability distribution. First proposed as a database operation by Dey and Sarkar [25] for a relational model with point probabilities, this operation had been extended to non-1NF databases in [20].

In the sections below, we will describe each algebra operation. We will base our examples on the elections in Sunny Hill that we have described in Section 1.1.4. Figure 1.2 shows different probability distributions representing a variety of polling data. These polling data can be easily represented in our ESPO format, as shown in Figure 5.3. We assume that all these objects have been inserted in the database in their current form.

In a relational data model, a relation is defined as a collection of data tuples over the same set of attributes. In our model, an Extended Semistructured Probabilistic relation (ESP-relation) is a set of ESPOs and an Extended Semistructured Probabilistic database (ESP-database) is a set of ESP-relations. Grouping ESPOs into relations is not done based on structure, as is the case in the relational databases; ESPOs with different structures can co-exist in the same ESP-relation. In the examples below we will consider ESP-relation $S = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$ consisting of ESPOs from Figure 5.3.

### 5.3.1 Selection

There is a variety of data stored in a single Extended SPO; for each individual part of the object we need to define a specific version of the selection operation, namely, selection based on context, random variables, conditionals, probabilities and probability table. The first three types of operations, described in Section 5.3.1, when applied to an ESP-relation produce a subset of that relation, but individual ESPOs do not change (except for their paths): they either satisfy the query and are returned or do not satisfy it. On the other hand, selections on probabilities or on probability tables (described in section 5.3.1) may lead to changes in the ESPOs being returned: only parts of the probability tables may “survive” such selection operations. Different types of selections are illustrated in the following example.
### Table 5.3: Sunny Hill pre-election polls in ESPO format

<table>
<thead>
<tr>
<th>ω: S₁</th>
<th>gender: men</th>
<th>party: Donkey</th>
<th>date: October 26</th>
</tr>
</thead>
<tbody>
<tr>
<td>mayor</td>
<td>park yes yes</td>
<td>legalization</td>
<td>l</td>
</tr>
<tr>
<td>Donkey</td>
<td>0.44</td>
<td>0.52</td>
<td></td>
</tr>
<tr>
<td>Donkey</td>
<td>0.12</td>
<td>0.16</td>
<td></td>
</tr>
<tr>
<td>Donkey</td>
<td>0.08</td>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td>Donkey</td>
<td>0.04</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>Elephant</td>
<td>yes yes</td>
<td>0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>Elephant</td>
<td>yes no</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>Elephant</td>
<td>no yes</td>
<td>0.03</td>
<td>0.04</td>
</tr>
<tr>
<td>Elephant</td>
<td>no no</td>
<td>0.06</td>
<td>0.08</td>
</tr>
<tr>
<td>Rhino</td>
<td>yes yes</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>Rhino</td>
<td>yes no</td>
<td>0.01</td>
<td>0.03</td>
</tr>
<tr>
<td>Rhino</td>
<td>no yes</td>
<td>0.03</td>
<td>0.05</td>
</tr>
<tr>
<td>Rhino</td>
<td>no no</td>
<td>0.01</td>
<td>0.04</td>
</tr>
<tr>
<td>senate = Donkey</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ω: S₂</th>
<th>date: October 23</th>
<th>gender: male</th>
<th>respondents: 238, {senate}</th>
</tr>
</thead>
<tbody>
<tr>
<td>respondents: 195, {legalization}</td>
<td>overlap: 184</td>
<td></td>
<td></td>
</tr>
<tr>
<td>senate</td>
<td>legalization</td>
<td>l</td>
<td>u</td>
</tr>
<tr>
<td>Rhino</td>
<td>yes</td>
<td>0.04</td>
<td>0.11</td>
</tr>
<tr>
<td>Rhino</td>
<td>no</td>
<td>0.1</td>
<td>0.15</td>
</tr>
<tr>
<td>Donkey</td>
<td>yes</td>
<td>0.22</td>
<td>0.27</td>
</tr>
<tr>
<td>Donkey</td>
<td>no</td>
<td>0.09</td>
<td>0.16</td>
</tr>
<tr>
<td>Elephant</td>
<td>yes</td>
<td>0.05</td>
<td>0.13</td>
</tr>
<tr>
<td>Elephant</td>
<td>no</td>
<td>0.21</td>
<td>0.26</td>
</tr>
<tr>
<td>mayor = Donkey {senate, legalization}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ω: S₃</th>
<th>locality: Sunny Hill</th>
<th>date: October 26</th>
</tr>
</thead>
<tbody>
<tr>
<td>park</td>
<td>legalization</td>
<td>l</td>
</tr>
<tr>
<td>yes yes</td>
<td>0.56</td>
<td>0.62</td>
</tr>
<tr>
<td>yes no</td>
<td>0.14</td>
<td>0.2</td>
</tr>
<tr>
<td>no yes</td>
<td>0.21</td>
<td>0.25</td>
</tr>
<tr>
<td>no no</td>
<td>0.03</td>
<td>0.07</td>
</tr>
<tr>
<td>mayor = Donkey</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ω: S₄</th>
<th>locality: South Side</th>
<th>date: October 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample: 323</td>
<td>mayor</td>
<td>l</td>
</tr>
<tr>
<td>Donkey</td>
<td>0.2</td>
<td>0.26</td>
</tr>
<tr>
<td>Elephant</td>
<td>0.42</td>
<td>0.49</td>
</tr>
<tr>
<td>Rhino</td>
<td>0.25</td>
<td>0.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ω: S₅</th>
<th>locality: Downtown</th>
<th>date: October 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample: 275</td>
<td>mayor</td>
<td>l</td>
</tr>
<tr>
<td>Donkey</td>
<td>0.48</td>
<td>0.55</td>
</tr>
<tr>
<td>Elephant</td>
<td>0.25</td>
<td>0.3</td>
</tr>
<tr>
<td>Rhino</td>
<td>0.2</td>
<td>0.24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ω: S₆</th>
<th>locality: West End</th>
<th>date: October 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample: 249</td>
<td>mayor</td>
<td>l</td>
</tr>
<tr>
<td>Donkey</td>
<td>0.38</td>
<td>0.42</td>
</tr>
<tr>
<td>Elephant</td>
<td>0.34</td>
<td>0.4</td>
</tr>
<tr>
<td>Rhino</td>
<td>0.15</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ω: S₇</th>
<th>locality: Sunny Hills</th>
<th>date: October 26</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample: 249</td>
<td>representative</td>
<td>l</td>
</tr>
<tr>
<td>Donkey</td>
<td>0.33</td>
<td>0.39</td>
</tr>
<tr>
<td>Elephant</td>
<td>0.32</td>
<td>0.37</td>
</tr>
<tr>
<td>Rhino</td>
<td>0.25</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Figure 5.3: Sunny Hill pre-election polls in ESPO format
Example 5.4 Table 5.1 lists some examples of queries that should be expressible as selection queries on ESPOs. For each question we describe the desired output of the selection operation.

Questions 1 – 3 and 5 in the example above do not involve the extensions of the SPO data model suggested in Section 5.2. To deal just with these kinds of queries, we could adapt the definitions from our original SPO algebra of [20]. Questions 4, 6 and 7, however, involve the extensions to the SPO model. Thus, the selection definitions from [20] need to be revised to incorporate new types of queries (like 6 and 7) and new formats for already defined queries (like 4).

a. Selection on Context, Random Variables and Conditionals

In this section, we define the selection operations that do not alter the content of the selected objects. We start by defining the acceptable languages for selection conditions for these types of selects.

Recall that the universe $\mathcal{R}$ of context attributes consists of a finite set of attributes $A_1, \ldots, A_n$ with domains $\text{dom}(A_1), \ldots, \text{dom}(A_n)$. With each attribute $A \in \mathcal{R}$ we associate a set $\text{Pr}(A)$ of allowed predicates. We assume that equality and inequality are allowed for all $A \in \mathcal{R}$.

Definition 5.6

1. An atomic context selection condition is an expression $c$ of the form “$A \, Q \, x (Q(A, x))$”, where $A \in \mathcal{R}$, $x \in \text{dom}(A)$ and $Q \in \text{Pr}(A)$.

2. An atomic participation selection condition is an expression $c$ of the form “$v \in V$”, where $v \in \mathcal{V}$ is a random variable.

3. An atomic conditional selection condition is one of the following expressions: “$u = \{x_1, \ldots, x_h\}$” or “$u \ni x$” where $u \in \mathcal{V}$ is a random variable and $x, x_1, \ldots, x_h \in \text{dom}(u)$. We will slightly abuse notation and write “$u = x$” instead of “$u = \{x\}$”.

4. An extended atomic context selection condition is an expression $c/V$ where $c$ is an atomic context selection condition and $V \subseteq \mathcal{V}$ is a set of random variables.

5. An extended atomic conditional selection condition is an expression $c/V$ where $c$ is an atomic conditional selection condition and $V \subseteq \mathcal{V}$ is a set of random variables.

Example 5.5 The table below contains some examples of selection conditions of different types for the Sunny Hill pre-election polls database.

Complex selection conditions can be formed as Boolean combinations of atomic selection conditions. The definitions below formalize the selection operation on a single Extended SPO.
Table 5.1: Selection queries to ESPOs.

<table>
<thead>
<tr>
<th>#</th>
<th>Query</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“What information is available about voter attitudes on October 26?”</td>
<td>Set of ESPOs that have date: October 26 in their context.</td>
</tr>
<tr>
<td>2</td>
<td>“What are other voting intentions of people who choose to vote Donkey for mayor?”</td>
<td>Set of ESPOs which have as a conditional mayor=Donkey.</td>
</tr>
<tr>
<td>3</td>
<td>“What information is known about voter intentions in the mayoral race?”</td>
<td>Set of ESPOs that contain mayor in the set of participating random variables</td>
</tr>
<tr>
<td>4</td>
<td>“What voting patterns are likely to occur with probability between 0.2 and 0.3?”</td>
<td>In the probability table of each ESPO, the rows with probability values guaranteed to be between 0.2 and 0.3 are found. If such rows exist, they form the probability table of the ESPO that is returned by the query.</td>
</tr>
<tr>
<td>5</td>
<td>“With what probability are voters likely to choose a Donkey mayor and Elephant Senator?”</td>
<td>Set of all ESPOs that contain mayor and senate random variables, with the probability tables of each containing only the rows where mayor=Donkey and senate=Elephant.</td>
</tr>
<tr>
<td>6</td>
<td>“Find all distributions based on more than 200 responses about senate vote.”</td>
<td>Set of ESPOs that contain senate random variable and responses = X with X &gt; 200 is associated with it in the context.</td>
</tr>
<tr>
<td>7</td>
<td>“How do people who intend to vote Donkey for mayor plan to vote for the park construction ballot initiative?”</td>
<td>Set of ESPOs that contain park random variable and conditional mayor=Donkey is associated with it.</td>
</tr>
</tbody>
</table>

Table 5.2: Different types of conditions for selection queries.

<table>
<thead>
<tr>
<th>Selection Condition Type</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>date = October26; sample &gt; 300</td>
</tr>
<tr>
<td>Extended Context</td>
<td>respondents &gt; 200/{senate}; gender = men/{mayor, park}</td>
</tr>
<tr>
<td>Participation</td>
<td>mayor ∈ V; park ∈ V</td>
</tr>
<tr>
<td>Conditional</td>
<td>mayor = Donkey; senate ⊆ Rhino; senate = {Rhino, Donkey}</td>
</tr>
<tr>
<td>Extended Conditional</td>
<td>mayor = Donkey/{park}; senate ⊆ Rhino/{park, mayor};</td>
</tr>
</tbody>
</table>

**Definition 5.7** Let $S = \langle T^+, V, P, C^+, \omega \rangle$ be an ESPO and let $c = Q(A, x)$ be an atomic context selection condition. Let $S' = \langle T^+, V, P, C^+, \omega' \rangle$ where $\omega' = \ \sigma_c(\omega)$”. Then $\sigma_c(S) = \{S'\}$ iff there exists a tuple $(A, a, V) \in T^+$ such that $(a^*, x) \in Q$; otherwise $\sigma_c(S) = \emptyset$.

**Definition 5.8** Let $S = \langle T^+, V, P, C^+, \omega \rangle$ be an ESPO and let $c : v \in V$ be an atomic participation
selection condition. Let \( S' = \langle T^+, V, P, C^+, \omega' \rangle \) where \( \omega' = \sigma_c(\omega)'' \). Then \( \sigma_c(S) = \{ S' \} \text{ iff } v \in V.

**Definition 5.9** Let \( S = \langle T^+, V, P, C^+, \omega \rangle \) be an ESPO and let \( c : u = \{ x_1, \ldots, x_h \} \) be an atomic conditional selection condition. Let \( S' = \langle T^+, V, P, C^+, \omega' \rangle \) where \( \omega' = \sigma_c(\omega)'' \). Then \( \sigma_c(S) = \{ S' \} \text{ iff } C^+ \ni (u, X) \text{ and } X = \{ x_1, \ldots, x_h \}.

Let \( c : u \ni x \) be an atomic conditional selection condition. Then \( \sigma_c(S) = \{ S' \} \text{ iff } C^+ \ni (u, X) \text{ and } X \ni x.

**Definition 5.10** Let \( S = \langle T^+, V, P, C^+, \omega \rangle \) be an ESPO and let \( c = Q(A, x)/V \) be an extended atomic context selection condition. Let \( S' = \langle T^+, V, P, C^+, \omega' \rangle \) where \( \omega' = \sigma_c(\omega)'' \). Then \( \sigma_c(S) = \{ S' \} \text{ iff there exists a tuple } (A, a, V) \in T^+ \text{ such that } (i) \ (a^*, x) \in Q; (ii) V \subseteq V^* \); otherwise \( \sigma_c(S) = \emptyset. \)

**Definition 5.11** Let \( S = \langle T^+, V, P, C^+, \omega \rangle \) be an ESPO and let \( c : u = \{ x_1, \ldots, x_h \}/V \) be an extended atomic conditional selection condition. Let \( S' = \langle T^+, V, P, C^+, \omega' \rangle \) where \( \omega' = \sigma_c(\omega)'' \). Then \( \sigma_c(S) = \{ S' \} \text{ iff } C^+ \ni (u, X, V') \text{, } X = \{ x_1, \ldots, x_h \}, \text{ and } V \subseteq V'. \)

Let \( c : u \ni x/ \) be an extended atomic conditional selection condition. Then \( \sigma_c(S) = \{ S' \} \text{ iff } C^+ \ni (u, X, V'), X \ni x \text{ and } V \subseteq V'. \)

The semantics of atomic selection conditions discussed so far can be extended to their Boolean combinations in a straightforward manner: \( \sigma_{c \land c'}(S) = \sigma_c(\sigma_{c'}(S)) \) and \( \sigma_{c \lor c'}(S) = \sigma_c(S) \lor \sigma_{c'}(S). \)

The interpretation of *negation* in the context selection condition requires some additional explanation. In order for a selection condition of the form \( \neg Q(A, x) \) to succeed on an ESPO \( S = \langle T^+, V, P, C^+, \omega \rangle \), attribute \( A \) must be present in \( T^+ \). If \( A \) is not present in the context of \( S \), the selection condition does not get evaluated and the result will be \( \emptyset. \) Therefore, the statement \( S \in \sigma_c(S) \lor S \in \sigma_{\neg c}(S) \) is not necessarily true. This also applies to conditional selection conditions.

Finally, for an ESP-relation \( S, \sigma_c(S) = \bigcup_{S \in S} (\sigma_c(S)). \)

We note here that whenever an ESPO satisfies any of the selection conditions described above, four of its five components, namely, context, participating variables, probability table and conditional, are returned intact. The only part of the ESPO that changes is its path: The new path expression reflects the fact that the selection query had been applied to the object.
Example 5.6 Consider our ESP-relation \( S \) (Figure 5.3). Below are some possible queries to this relation and their results (we specify the unique ids of the ESPOs that match the query).

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>context</td>
<td>( \sigma_{\text{cont.date}=\text{October}26}(S) )</td>
<td>{ ( S_1, S_3, S_7 ) }</td>
</tr>
<tr>
<td>Q2</td>
<td>participation</td>
<td>( \sigma_{\text{var.major} \in \text{v}}(S) )</td>
<td>{ ( S_1, S_4, S_5, S_6, S_7 ) }</td>
</tr>
<tr>
<td>Q3</td>
<td>conditionals</td>
<td>( \sigma_{\text{cond.senate}=(\text{Donkey})}(S) )</td>
<td>{ ( S_1, S_3 ) }</td>
</tr>
<tr>
<td>Q4</td>
<td>ext. context</td>
<td>( \sigma_{\text{cont.respondents}&gt;200/\text{[senate]}(S) )</td>
<td>{ ( S_2 ) }</td>
</tr>
<tr>
<td>Q5</td>
<td>ext. context</td>
<td>( \sigma_{\text{cond.gender}=[\text{men}/\text{[mayor.park]}]}(S) )</td>
<td>{ ( S_1 ) }</td>
</tr>
<tr>
<td>Q6</td>
<td>ext. conditional</td>
<td>( \sigma_{\text{cond.mayor}=(\text{Donkey})/\text{[senate]}(S) )</td>
<td>{ ( S_2 ) }</td>
</tr>
<tr>
<td>Q7</td>
<td>ext. conditional</td>
<td>( \sigma_{\text{cond.mayor}=(\text{Donkey})/\text{[senate.house]}(S) )</td>
<td>{ }</td>
</tr>
</tbody>
</table>

b. Selection on Probabilities and Probability Tables

The two types of selections introduced in this section are more complex. The result of a selection operation of either type depends on the content of the probability table, which can be considered as a relation (each row being a single record). In the process of performing the probabilistic selection or selection on the probability table (see questions 4 and 5, Example 5.4, respectively), each row of the probability table is examined individually to determine whether it satisfies the selection condition. It is retained in the answer if it does and is thrown out if it does not. Thus, a possible result of either of these two types of selection operation is an ESPO with an incomplete probability table. As the selection condition relates only to the content of the probability table of an ESPO, its context, participating random variables, and conditionals are preserved. We start by defining selection on probability tables.

Definition 5.12 An atomic probabilistic table selection condition is an expression of the form \( v = x \) where \( v \in V \) and \( x \in \text{dom}(v) \). Probabilistic table selection conditions are Boolean combinations of atomic probabilistic table selection conditions.

Definition 5.13 Let \( S = \langle T^+, V, P, C^+, \omega \rangle \) be an ESPO, \( V = \{v_1, \ldots, v_k\} \), and let \( c : v = x \) be an atomic probabilistic table selection condition.

If \( v \in V \), then (assuming \( v = v_i, 1 \leq i \leq k \)) the result of selection from \( S \) on \( c \) \( \sigma_c(S) \) is a semistructured probabilistic object \( S' = \langle T^+, V, P', C^+, \omega' \rangle \), where \( \omega' = \sigma_c(\omega) \) and

\[
P'(y_1, \ldots, y_i, \ldots, y_k) = \begin{cases} 
P(y_1, \ldots, y_i, \ldots, y_k) & \text{if } y_i = x; \\
\text{undefined} & \text{if } y_i \neq x.
\end{cases}
\]

Example 5.7 Consider the ESPO \( S_1 \) from Figure 5.3. The leftmost ESPO of Figure 5.4 shows the result of the selection query on probability table: \( \sigma_{\text{bl.park}=\text{yes}}(S_1) \) (find the probability of all
voting outcomes where respondents support the park ballot initiative). Following Definition 5.13, the result of this query is computed as follows: the context, list of conditionals and participating random variables remain the same, while the probability table now contains only the rows that satisfy the selection condition and the path changes to reflect the selection operation.

We note that if the same query is applied to the entire relation $S$, the resulting relation will contain two ESPOs constructed from $S_1$ and $S_2$: only those ESPOs have participating random variable park (and rows for park=yes).

We are now ready to describe the last type of the selection operation: selection on probabilities.

Example 5.8 The following queries are examples of the types of probabilistic selection queries that need to be expressible in ESP-Algebra.

1. Find all rows where the lower bound is equal to 0.1;
2. Find all rows where the upper bound is greater than 0.4;
3. Find all rows where the probability is guaranteed to be greater than 0.2;
4. Find all rows where the probability can be less than 0.2.

The first two queries refer to the lower and upper bounds as supplied by the $P$ function. The last two queries refer to the point probability value as associated with a row by a p-interpretation. The third query specifies a for-all condition, which is true iff the condition is true for all p-interpretations satisfying $P$. The fourth query specifies an exists condition, which is true if at least one p-interpretation satisfying $P$ is satisfies the condition. Our constraint language will allow for all four types of atomic conditions to be expressed.

Definition 5.14 An atomic probabilistic selection condition is an expression of one of the forms:

(i) $l \text{ op } \alpha$; (ii) $u \text{ op } \alpha$; (iii) $\forall P \text{ op } \alpha$; (iv) $\exists P \text{ op } \alpha$, where $\alpha \in [0, 1]$ and \text{ op } \in \{=, \neq, \leq, \geq, <, >\}. Probabilistic selection conditions are Boolean combinations of atomic probabilistic selection conditions.

Example 5.9 While the precise semantics of probabilistic selection conditions is determined in Definition 5.15 below, the following conditions match the probabilistic queries from Example 5.8:

(1) $l = 0.1$; (2) $u > 0.4$; (3) $\forall P > 0.2$; (4) $\exists P < 0.2$. 

106
Definition 5.15 Let \( S = \langle T^+, V, P, C^+, \omega \rangle \) be an ESPO. Let \( c : l \ p \ a \ (c : u \ p \ a) \) be a probabilistic atomic selection condition. Let \( \bar{x} \in \text{dom}(V) \). The result of selection from \( S \) on \( c \) is defined as follows: \( \sigma_P \ p \ a(S) = S' = \langle T^+, V, P', C^+, \omega' \rangle \), where \( \omega' = \sigma_{\bar{x}}(\omega) \) and

\[
P'(\bar{x}) = \begin{cases} 
P(\bar{x}) & \text{if } l_\bar{x} \ p \ a (u_\bar{x} \ p \ a); \\
\text{undefined} & \text{otherwise.} \end{cases}
\]

Let \( c : \forall P \ p \ a \) be a probabilistic atomic selection condition. The result of selection from \( S \) on \( c \) is defined as follows: \( \sigma_P \ p \ a(S) = S' = \langle T^+, V, P', C^+, \omega' \rangle \), where \( \omega' = \sigma_{\bar{x}}(\omega) \) and
Let $c : \exists P \ op \ \alpha$ be a probabilistic atomic selection condition. The result of selection from $S$ on $c$ is defined as follows: $\sigma_P \ op_\alpha(S) = S' = \langle T^+, V, P', C^+, \omega' \rangle$, where $\omega' = \sigma_c(\omega)$ and $P'(\overline{x}) = \begin{cases} P(\overline{x}) & \text{if } (\forall I \models P)(I(\overline{x}) \ op \ \alpha); \\ \text{undefined} & \text{otherwise.} \end{cases}$

**Example 5.10** The center and the rightmost ESPOs on Figure 5.4 represent the results of selections on probabilities: $\sigma_{u>0.14}(S_2)$ and $\sigma_{l<0.11}(S_2)$ respectively. In both cases, the results of the selection keep the same context, conditionals and participating random variables, while the probability table is modified to retain only the rows where the upper (lower) bound on the probability interval satisfies the selection condition.

Notice that the result of $\sigma_{u>0.14}(S)$ would contain seven ESPOs: every object in $S$ contains rows where upper bound on probability is greater than 0.14. The result of $\sigma_{l<0.11}(S)$ will contain two ESPOs constructed from $S_1$ and $S_2$: only those had rows with lower probability less than 0.11.

While evaluation of the probabilistic selection conditions on lower and upper bounds is fairly straightforward, evaluation of the probabilistic selection conditions referring to p-interpretations may seem to be complex. As it turns out, these conditions can be expressed via the conditions on upper and lower bounds as specified in the following proposition.

**Proposition 5.1** The following equivalences hold:

<table>
<thead>
<tr>
<th>$\forall$-conditions</th>
<th>$\exists$-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{(\forall P=0)}(S) \equiv \sigma_{=\alpha \land u=\alpha}(S)$</td>
<td>$\sigma_{(\exists P=0)}(S) \equiv \sigma_{\leq \alpha \land u=\alpha}(S)$</td>
</tr>
<tr>
<td>$\sigma_{(\forall P \geq 0)}(S) \equiv \sigma_{\geq \alpha}(S)$</td>
<td>$\sigma_{(\exists P \geq 0)}(S) \equiv \sigma_{\geq \alpha}(S)$</td>
</tr>
<tr>
<td>$\sigma_{(\forall P &gt; 0)}(S) \equiv \sigma_{&gt; \alpha}(S)$</td>
<td>$\sigma_{(\exists P &gt; 0)}(S) \equiv \sigma_{&gt; \alpha}(S)$</td>
</tr>
<tr>
<td>$\sigma_{(\forall P \leq 0)}(S) \equiv \sigma_{\leq \alpha}(S)$</td>
<td>$\sigma_{(\exists P \leq 0)}(S) \equiv \sigma_{\leq \alpha}(S)$</td>
</tr>
<tr>
<td>$\sigma_{(\forall P &lt; 0)}(S) \equiv \sigma_{&lt; \alpha}(S)$</td>
<td>$\sigma_{(\exists P &lt; 0)}(S) \equiv \sigma_{&lt; \alpha}(S)$</td>
</tr>
<tr>
<td>$\sigma_{(\forall P \neq 0)}(S) \equiv \sigma_{\neq \alpha \land u=\alpha}(S)$</td>
<td>$\sigma_{(\exists P \neq 0)}(S) \equiv \sigma_{\neq \alpha \land u=\alpha}(S)$</td>
</tr>
</tbody>
</table>

**Example 5.11** Figure 5.5 shows some selections on probabilities that use p-interpretation notation. The first query, $\sigma_{\forall P<0.11}(S_1)$, finds all rows in the probability table of $S_1$ for which all p-interpretations have probability less than 0.11. By Proposition 5.1, this query is equivalent to $\sigma_{u<0.11}(S_1)$. The second query, $\sigma_{\exists P<0.04}(S_1)$, asks for rows of the probability table of $S_1$ in which at least one satisfying p-interpretation can have probability less than or equal to 0.04. By Proposition 5.1, it is equivalent to $\sigma_{l<0.04}(S_1)$. 

108
Different selection operations (both described in this section and in Section 5.3.1) commute, as shown in the following theorem:

**Theorem 5.1** Let \( c \) and \( c' \) be two selection conditions and let \( S \) be a semistructured probabilistic relation. Then \( \sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S)) \).

**Proof of Theorem 5.1**

*Here we prove that different selection operations commute.*

Let \( c \) and \( c' \) be two atomic selection conditions. There are 5 types of atomic selection conditions, namely context, participation, conditional, table and probability. Selection on context, participation or conditional results in entire SPOs being selected, while selection on table or probability selects only parts of the relevant SPOs. We could partition the conditions into two groups,

- Group I, containing context, extended context, participation, conditional and extended conditional conditions, and
- Group II, containing table and probability conditions.

First we prove \( \sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S)) \) for a single SPO \( S \), and we consider here all the possible cases for each pair of condition groups.

**Case 1.** Both conditions \( c \) and \( c' \) are in Group I.

There are three possible combinations for whether each condition is satisfied: (a) \( S \) satisfies \( c \) but not \( c' \), or (b) \( S \) satisfies \( c' \) but not \( c \), or (c) \( S \) satisfies both \( c \) and \( c' \), or (d) \( S \) does not satisfy...
either $c$ or $c'$.

By the definition of selection on atomic selection conditions in Group $I$, we know selection on these conditions results in the entire SPO being selected, or none of it. For case a), since $S$ does not satisfy $c'$, $\sigma_{c'}(S)$ returns empty and subsequently $\sigma_c(\sigma_{c'}(S))$ returns empty. Since $\sigma_c(S)$ returns $S$, we see that $\sigma_{c'}(\sigma_c(S)) = \sigma_c(S)$ also returns empty for the same reason. Thus, $\sigma_c(\sigma_{c'}(S)) = \sigma_c(\sigma_{c'}(S))$ holds for case (a). The same applies to case (b). Similarly, for case (d).

For case c), $\sigma_c(\sigma_{c'}(S)) = \sigma_c(S)$ returns $S$, and $\sigma_{c'}(\sigma_c(S)) = \sigma_{c'}(S)$ returns $S$ too. This proves that $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for case (c).

So $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for all the cases.

**Case 2.** Condition $c$ is in Group $I$ and condition $c'$ is in Group $II$.

There are only two possible combinations for whether each condition is satisfied, assuming that condition $c'$ is always partially satisfied: a) $S$ does not satisfy $c$, or b) $S$ satisfies $c$.

By the definition of selection on atomic selection conditions in both Group $I$ and Group $II$, we know selection on conditions in Group $I$ results in the entire SPO being selected or not, while selection on conditions in Group $II$ preserves all the context, participating random variables and conditionals in the original SPO, but produce only a part of the probability table.

Let $\sigma_{c'}(S) = S'$, where $S'$ has part of the probability table which satisfies the condition $c'$ and retains all the context, participating random variables and conditionals in $S$.

For case a), $\sigma_c(\sigma_{c'}(S)) = \sigma_c(S')$ returns empty since $S'$ does not satisfy the condition $c$ either. Since $\sigma_c(S)$ returns empty, subsequently $\sigma_{c'}(\sigma_c(S))$ also returns empty. This proves $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ for case (a).

For case (b), $\sigma_c(\sigma_{c'}(S)) = \sigma_c(S')$ returns $S'$ since $S'$ should satisfy the condition $c$ too. Since $\sigma_c(S)$ returns $S$, so $\sigma_{c'}(\sigma_c(S)) = \sigma_{c'}(S)$ also returns $S'$. This proves that $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for case (b).

So $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ holds for both cases.

**Case 3.** Both $c$ and $c'$ are conditions in Group $II$.

First we prove $\sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S))$ for a single SPO $S$. Assume that both conditions $c$ and $c'$ are partially satisfied by $S$. By the definition of selection on atomic selection conditions in Group $II$, we know selection on these conditions results in part of the probability table and preserves all the context, participating random variables and conditionals in the original SPO. In other words, all the components in the original SPO except the probability table are preserved.

Let $S = \langle T, V, P, C \rangle$. Then $S' = \sigma_c(\sigma_{c'}(S)) = \langle T, V, P', C \rangle$ and $S'' = \sigma_{c'}(\sigma_c(S)) = \langle T, V, P'', C \rangle$ with $P' = \theta_c(\varphi_{c'}(P))$, and $P'' = \theta_{c'}(\varphi_c(P))$ where $\varphi$ is the relational selection operator. Since the relational selection operator $\varphi$ is commutative, $\theta_c(\varphi_{c'}(P)) = \varphi_{c'}(\theta_c(P))$. Therefore we
have \( P' = P'' \) or \( S' = S'' \). So \( \sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S)) \) holds for this case.

Now let an SP-relation \( S = \bigcup_{S \in \mathcal{S}} S \). Since the union operator is commutative, \( \sigma_{c'}(\sigma_c(S)) = \sigma_c(\sigma_{c'}(\mathcal{S}(S))) = \mathcal{S}(\sigma_c(\sigma_{c'}(S))) \), and \( \sigma_c(\sigma_{c'}(S)) = \mathcal{S}(\sigma_c(\sigma_{c'}(S))) \).

So this proves \( \sigma_c(\sigma_{c'}(S)) = \sigma_{c'}(\sigma_c(S)) \). \( \square \)

5.3.2 Projection

Projection in classical relational algebra removes columns from the relation and, if needed, collapses duplicate tuples. ESPOs consist of four different components that can be affected by projection operations. We distinguish between three different types of projection here: on context, on conditionals and on participating random variables, the latter affecting probability table as well.

There are two issues that need to be addressed when defining projection on context. First, contexts may contain numerous copies of relational attributes. Hence, projecting out a particular attribute from a context of an ESPO should result in all copies if this attribute being projected out. The second issue is the fact that in extended context, different attributes are associated with different participating random variables. Thus, it would be desirable to be able to take these associations into account when performing projections.

To address these two issues we define two types of projection on context. The first operation will be similar to standard relational projection, while the second operation will work by removing associations between context attributes and random variables.

**Definition 5.16** Let \( F = \{A_1, \ldots, A_k\} \) be a set of context attributes and \( S = \langle T^+, V, P, C^+, \omega \rangle \) be an ESPO. Projection of \( S \) on \( F \), denoted \( \pi_F(S) \), is an ESPO \( S' = \langle T'^+, V, P, C^+, \omega' \rangle \), where \( T'^+ = \{(A, a, V^*)(A, a, V^*) \in T^+, A \in F\} \) and \( \omega' = \pi_F(\omega) \).

**Definition 5.17** Let \( F^+ = \{(A_1, V_1), \ldots, (A_k, V_k)\} \) be a set of pairs where for \( 1 \leq i \leq k \), \( A_i \) a context attribute and \( V_i \subseteq V \). Let \( S = \langle T^+, V, P, C^+, \omega \rangle \) be an ESPO. Projection of \( S \) on \( F^+ \), denoted \( \pi_{F^+}(S) \), is an ESPO \( S' = \langle T'^+, V, P, C^+, \omega' \rangle \), where \( T'^+ = \{(A, a, V^*)(A, a, V^*) \in T^+, A = A_i \in \{A_1, \ldots, A_k\}, \text{ for some } 1 \leq i \leq k, \text{ and } \emptyset \neq V' = V^* \cap V_i \} \) and \( \omega' = \pi_{F^+}(\omega) \).

Given an ESPO \( S \) and a set of pairs \( F^+ \) as described in Definition 5.17, the projection operation will proceed as follows. The set of context attributes to keep which comes from \( F^+ \) specifies for each attribute the list of random variables for which it is allowed to be kept. The projection operation (i) removes from the input ESPO \( S \) all attributes not in \( F^+ \) and (ii) for each instance \((a, V^*) \in T^+\) of attribute \( A_i \) such that \((A_i, V_i) \in F^+ \) it removes all references in \( V^* \) that are not in \( V_i \). If
$V^* \cap V_i = \emptyset$, then $(a, V^*)$ is omitted from the projection. Projections on context are illustrated in the example below.

**Example 5.12** Figure 5.6 contains the results of two projection queries applied to the ESPO $S_3$ from Figure 5.3, $\pi_{date}(S_3)$ (left) and $\pi_{\{locality, park, date, \{park\}\}}(S_3)$ (right). The first operation results in the removal of all context attributes other than date from the context of the ESPO. The second operation removes associations between the context attributes locality and date and all random variables but park. In both cases, conditionals, participating random variables and probability table are not affected.

<table>
<thead>
<tr>
<th>$\omega$</th>
<th>$\pi_{\text{cnt, date}}(S_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>date: October 26</td>
<td>park legal</td>
</tr>
<tr>
<td>yes yes</td>
<td>0.56</td>
</tr>
<tr>
<td>yes no</td>
<td>0.14</td>
</tr>
<tr>
<td>no yes</td>
<td>0.21</td>
</tr>
<tr>
<td>no no</td>
<td>0.03</td>
</tr>
</tbody>
</table>

$\omega$ | $\pi_{\{\text{cnt, locality}, \{park\}, \text{cnt, date}, \{park\}\}}(S_3)$ |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>locality: Sunny Hill park</td>
<td>date: October 26 park</td>
</tr>
<tr>
<td>park legal</td>
<td>$l$</td>
</tr>
<tr>
<td>yes yes</td>
<td>0.56</td>
</tr>
<tr>
<td>yes no</td>
<td>0.14</td>
</tr>
<tr>
<td>no yes</td>
<td>0.21</td>
</tr>
<tr>
<td>no no</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Figure 5.6: Projections on context.

The projection operations on conditionals can be defined similarly. We note here that, while syntactically these operations are similar, projecting conditionals out of ESPOs is a more dangerous operation from the probability theory point of view. Basically, it is taking a conditional probability distribution $P(Y|X)$ and making a decision to “forget” the conditioning information $X$, and refer to the distribution as $P(Y)$ from then on. If unconditional distribution $P(Y)$ is also available, this may lead to some confusion. Projecting out conditionals is sometimes necessary: When a specific random variable is removed from consideration in the entire database, it needs to be projected out from both lists of participating random variables (see below) as well as from the conditionals of affected ESPOs. Similarly, if the users switch to a sub-database, if all ESPOs which contain the same conditional part, that conditional part can be removed for convenience. With this in mind we present the projection on conditionals.

**Definition 5.18** Let $U = \{u_1, \ldots, u_k\} \subseteq \mathcal{V}$ be a set of random variables and $S = \langle T^+, V, P, C^+, \omega \rangle$
be an ESPO. Projection of $S$ on $U$, denoted $\pi_{C:U}(S)$ is an ESPO $S' = (T^+, V, P, C^{+'}, \omega')$, where $C^{+'} = \{(u, X, V^*)|(u, X, V^*) \in C^+, \text{ and } u \in U\}$ and $\omega' = "\pi_{C:U}(\omega)"$.

**Definition 5.19** Let $U^+ = \{(u_1, V_1)\ldots ,(u_k, V_k)\}$ be a set of pairs where for all $1 \leq i \leq k$, $u_i \in \mathcal{V}$ and $V_i \subseteq \mathcal{V}$. Let $S = (T^+, V, P, C^{+'}, \omega)$ be an ESPO. Projection of $S$ on $U^+$, denoted $\pi_{C:U^+}(S)$ is an ESPO $S' = (T^+, V, P, C^{+'}, \omega')$, where $C^{+'} = \{(u, X, V')|(u, X, V^*) \in T^+, \text{ and } u = u_i \in \{u_1, \ldots, u_k\}\}$, and $\emptyset \neq V' = V^* \cap V_i$; $\omega' = "\pi_{C:U}(\omega)"$.

The following example illustrates how projection operations on conditionals work.

**Example 5.13** Figure 5.7 shows the results of two different projection on conditionals operations, $\pi_{C:0}(S_2)$ (left) and $\pi_{C: \{\text{mayor}/ \{\text{senate}\}\}}(S_2)$ (right). The first projection removes all conditionals in $S_2$, leaving the conditional component empty. The second projection severs the association between the mayor = Donkey conditional and the random variable legalization. Context, random variables and probability table are left intact.

<table>
<thead>
<tr>
<th>$\omega'$</th>
<th>$\pi_{C:0}(S_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>date:</td>
<td>October 23</td>
</tr>
<tr>
<td>gender:</td>
<td>male</td>
</tr>
<tr>
<td>respondents:</td>
<td>238, {senate}</td>
</tr>
<tr>
<td>respondents:</td>
<td>195, {legalization}</td>
</tr>
<tr>
<td>overlap:</td>
<td>184</td>
</tr>
<tr>
<td>senate</td>
<td>legalization</td>
</tr>
<tr>
<td>Rhino yes</td>
<td>0.04 0.11</td>
</tr>
<tr>
<td>Rhino no</td>
<td>0.1 0.15</td>
</tr>
<tr>
<td>Donkey yes</td>
<td>0.22 0.27</td>
</tr>
<tr>
<td>Donkey no</td>
<td>0.09 0.16</td>
</tr>
<tr>
<td>Elephant yes</td>
<td>0.05 0.13</td>
</tr>
<tr>
<td>Elephant no</td>
<td>0.21 0.26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\omega'$</th>
<th>$\pi_{C: {\text{mayor}/ {\text{senate}}}}(S_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>date:</td>
<td>October 23</td>
</tr>
<tr>
<td>gender:</td>
<td>male</td>
</tr>
<tr>
<td>respondents:</td>
<td>238, {senate}</td>
</tr>
<tr>
<td>respondents:</td>
<td>195, {legalization}</td>
</tr>
<tr>
<td>overlap:</td>
<td>184</td>
</tr>
<tr>
<td>senate</td>
<td>legalization</td>
</tr>
<tr>
<td>Rhino yes</td>
<td>0.04 0.11</td>
</tr>
<tr>
<td>Rhino no</td>
<td>0.1 0.15</td>
</tr>
<tr>
<td>Donkey yes</td>
<td>0.22 0.27</td>
</tr>
<tr>
<td>Donkey no</td>
<td>0.09 0.16</td>
</tr>
<tr>
<td>Elephant yes</td>
<td>0.05 0.13</td>
</tr>
<tr>
<td>Elephant no</td>
<td>0.21 0.26</td>
</tr>
<tr>
<td>mayor = Donkey{senate}</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.7: Projections on conditionals.

Now we are ready to define the most intricate projection operation, projection on the set of random variables. When defining this operation, we need to keep in mind the following: (i) projection is only allowed if at least one random variable remains in the resulting set of participating random variables, to be defined below.

---

1Symbol ‘$C$’ is used in the notation to distinguish the projection operation from the projection on the set of participating random variables, to be defined below.
variables \(^2\); (ii) projecting out a random variable \(v\) should result in removal of \(v\) from the extended context and conditionals; (iii) projecting out a random variable \(v\) should remove this variable from the probability table, i.e., the underlying probability distribution function will change. Out of these notes, the last is of the most importance.

**Definition 5.20** Let \(S = \langle T^+, V, P, C^+, \omega \rangle\) be an ESPO, and let \(V^* \subseteq V\). Projection of \(S\) on \(V^*\), denoted \(\pi_{V^*}(S)\) is defined as follows:

1. \(V^* \cap V = \emptyset\): \(\pi_{V^*}(S) = \emptyset\).
2. \(V^* \cap V = V' \neq \emptyset\): \(\pi_{V^*}(S) = S' = \langle T'^+, V', P', C'^+, \omega' \rangle\), where
   - \(T'^+ = \{(A, a, W')|(A, a, W) \in T^+ \text{ and } W \cap V^* \neq \emptyset \text{ and } W' = W \cap V^*\}\);
   - \(C'^+ = \{(u, X, W')|(u, X, W) \in C^+ \text{ and } W \cap V^* \neq \emptyset \text{ and } W' = W \cap V^*\}\);
   - \(P': \text{dom}(V') \rightarrow C[0,1]\);
   - For all \(\tilde{x}' \in \text{dom}(V')\) and \((\tilde{x}'', \tilde{x}'') \in \text{dom}(V),\)
     \[P'(\tilde{x}') = \min_{l \models P} \sum_{(\tilde{x}'', \tilde{x}'') \in \text{dom}(V)} I(\tilde{x}', \tilde{x}'') \cdot \max_{l \models P} \sum_{(\tilde{x}'', \tilde{x}'') \in \text{dom}(V)} I(\tilde{x}', \tilde{x}''),\]
   - \(\omega' = "\pi_{V^*}(\omega)"\).

This definition requires a careful explanation. Let \(S = \langle T^+, V, P, C^+, \omega \rangle\) be an ESPO, and let \(V^* \subseteq V\) be the set of projection random variables. The computation of \(\pi_{V^*}(S)\) proceeds as follows. First, we check if the intersection of \(V\), the set of participating random variables of \(S\) and \(V^*\) is empty, and if it is, we return empty set as the answer. If \(V' = V \cap V^*\) is not empty, we build the projection as follows:

(i) the new set of participating random variables is \(V'\);

(ii) the new context \(T'^+\) and conditionals \(C'^+\) are produced from \(T^+\) and \(C^+\) respectively, by eliminating all random variables not from \(V'\) from the extensions (associations). Context entries (conditionals) from \(T^+ (C^+)\) associated only with variables not from \(V'\) will be eliminated from \(T'^+ (C'^+)\);

\(^2\)We want our query algebra to be closed: ESPOs in — ESPOs out. Removing all random variables from the ESPO basically collapses it. The object returned by such an operation will no longer satisfy our definition of an ESPO. Because of that, we do not consider such operations.
(iii) finally, the new probability table function is defined as follows. The function must range over $\text{dom}(V')$. As $V' \subseteq V$, with each value $\tilde{x}' \in \text{dom}(V')$, a set of values $(\tilde{x}', \tilde{x}'') \in \text{dom}(V)$ is associated, where $\tilde{x}''$ ranges over $\text{dom}(V-V')$. Given a p-interpretation $I \models P$, for each $\tilde{x}' \in \text{dom}(V')$ we can compute the probability assigned to it by $P$ as $I(\tilde{x}') = \sum_{\tilde{x}'' \in \text{dom}(V-V')} I(\tilde{x}', \tilde{x}'')$. Now, we know that the probability of $\tilde{x}'$ has to range between the minimal and maximal value of $I(\tilde{x}')$, for all $I \models P$. This interval, $[\min_{I \models P} I(\tilde{x}'), \max_{I \models P} I(\tilde{x}')]$ is defined to be the value of the new probability distribution function $P'$ on $\tilde{x}'$.

While the computation of the new set of participating random variables, context and conditionals according to Definition 5.20 is straightforward, computing the new probability table requires solving a number of optimization problems (finding mins and maxs of $\sum I(\tilde{x}', \tilde{x}'')$ for all $\tilde{x}'$), which seems a fairly tedious task. However, it turns out that these optimization problems have analytical solutions.

**Theorem 5.2** Let $S = \langle T^+, V, P, C^+, \omega \rangle$ be an ESPO and $V^* \subseteq V$. Let $V \cap V^* \neq \emptyset$ and $S' = \langle T'^+, V', P', C'^+, \omega' \rangle = \pi_{V^*}(S)$. Let $P''(x') = \left[ \sum_{(\tilde{x}', \tilde{x}'') \in \text{dom}(V)} l(\tilde{x}', \tilde{x}''), \min(1, \sum_{(\tilde{x}', \tilde{x}'') \in \text{dom}(V)} u(\tilde{x}', \tilde{x}'')) \right]$. Then, $P' = \mathcal{T}(P'')$.

**Proof of Theorem 5.2**

Here we only give proof for the lower bound. The same approach applies to the upper bound.

From the definition of projection given above, we know the lower bound for $\pi_U(P)(\tilde{x})$ is

$$\min_{I \models P} \left( \sum_{g \in \text{dom}(V-U)} I(\tilde{x}, \tilde{y}) \right).$$

Let $l'(\tilde{x}) = \sum_{g \in \text{dom}(V-U)} l(\tilde{x}, \tilde{y})$, and $u'(\tilde{x}) = \sum_{g \in \text{dom}(V-U)} u(\tilde{x}, \tilde{y})$. For any p-interpretation $I$ over $V$ such that $I$ satisfies $P (I \models P)$, we have $l(\tilde{x}, \tilde{y}) \leq I(\tilde{x}, \tilde{y}) \leq u(\tilde{x}, \tilde{y})$, for all instances $(\tilde{x}, \tilde{y}) \in \text{dom}(V)$.

Thus the summation over all $\tilde{y} \in \text{dom}(V-U)$ gives

$$u'(\tilde{x}) = \sum_{\tilde{y} \in \text{dom}(V-U)} u(\tilde{x}, \tilde{y}) \geq \sum_{\tilde{y} \in \text{dom}(V-U)} I(\tilde{x}, \tilde{y}) \geq \sum_{\tilde{y} \in \text{dom}(V-U)} l(\tilde{x}, \tilde{y}) = l'(\tilde{x}).$$

By the definition of p-interpretation, we have

$$\sum_{(\tilde{w}, \tilde{y}) \in \text{dom}(V)} I(\tilde{w}, \tilde{y}) = \sum_{\tilde{w} \in \text{dom}(V-U)} \left[ \sum_{\tilde{y} \in \text{dom}(V-U)} I(\tilde{w}, \tilde{y}) \right] = 1.$$

The above equation can be rearranged as
\[
\sum_{\bar{g} \in \text{dom}(V-U)} I(\bar{x}, \bar{y}) = 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} \left[ \sum_{\bar{g} \in \text{dom}(V-U)} I(\bar{w}, \bar{y}) \right].
\]

By using the relationship given above, \(I(\bar{x}, \bar{y}) \leq u_{(x)}\) for any \((\bar{y}) \in \text{dom}(V-U)\), we get
\[
\sum_{\bar{g} \in \text{dom}(V-U)} I(\bar{x}, \bar{y}) \geq 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} \left[ \sum_{\bar{g} \in \text{dom}(V-U)} u_{(\omega, \bar{y})} \right] = 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u'_{(\omega)}.
\]

Thus, the lower bound for \(\pi_U (P(\bar{x}))\) is
\[
\min_{I \models P} \left( \sum_{\bar{g} \in \text{dom}(V-U)} I(\bar{x}, \bar{y}) \right) = \max (l'_{(x)}, 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u'_{(\omega)}).
\]

On the other hand, the lower bound for \(P''(\bar{x})\) is \(l''_{(x)} = \min_{\bar{g} \in \text{dom}(V-U)} l_{(x, \bar{y})} = \min (l'_{(x)}, 1),\) and the upper bound for \(P''(\bar{x})\) is \(u''_{(x)} = \min_{\bar{g} \in \text{dom}(V-U)} u_{(x, \bar{y})} = \min (u'_{(x)}, 1).\) By the definition of the tightening operation, we know the lower bound for \(\tau(P'')(\bar{x})\) is
\[
\max (l''_{(x)}, 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u''_{(\omega)}).
\]

Now let’s prove that for any \(\bar{x} \in \text{dom}(U),\) the two lower bounds \(\max (l'_{(x)}, 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u'_{(\omega)})\) and \(\max (l''_{(x)}, 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u''_{(\omega)})\) are equal for all possible cases. Please note here that \(l'_{(x)} \leq 1\) always holds, and consequently \(l''_{(x)} = l'_{(x)}\) holds for any \(\bar{x} \in \text{dom}(U)\).

First we show that if \(u'_{(\omega)} \leq 1\) holds for all \(\bar{w} \in \text{dom}(U)\) such that \(\bar{w} \neq \bar{x},\) then \(u''_{(\omega)} = \min (u'_{(\omega)}, 1) = u'_{(\omega)}\) and therefore \(1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u'_{(\omega)} = 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u'_{(\omega)}\) Then the two lower bound formulas are identical.

To complete the argument that the two lower bounds are equivalent, we also consider the case that there exists \(\bar{w} \in \text{dom}(U), \bar{w} \neq \bar{x}\) and \(u'_{(\omega)} > 1.\) Then \(1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u'_{(\omega)}\) and \(u''_{(\omega)} = \min (u'_{(\omega)}, 1) = 1.\) Therefore \(1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u''_{(\omega)} \leq 0.\) But then \(\max (l''_{(x)}, 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u''_{(\omega)}) = l''_{(x)} = l'_{(x)},\) and \(\max (l'_{(x)}, 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u'_{(\omega)}) = l'_{(x)},\) so the two lower bound expressions produces the same values.

Therefore, we obtain the following equality: \(\max (l''_{(x)}, 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u''_{(\omega)}) = \max (l'_{(x)}, 1 - \sum_{\omega \in \text{dom}(U) \wedge \omega \neq \bar{x}} u'_{(\omega)}),\) which means that the lower bound for \(\tau(P'')(\bar{x})\) is equal to the lower bound for \(\pi_U (P(\bar{x})).\)

The projection operation on a set of random variables is illustrated in the example below.

**Example 5.14** Figure 5.8 illustrates the process of computing projection \(\pi_{\{\text{noate}\} (S_2)\) on participating random variables. The first step of this operation is the removal of all other random variables from the probability table. Next, the duplicate rows of the new probability table are collapsed and
the probability intervals are added. After that, the operation on tightening is performed to find the true intervals as we can observe that neither of the three lower bounds is reachable. We then exclude respondents: 195 from the context as it is not associated with senate variable and disassociate legalization with conditionals.

$$\omega: \quad S_2$$

<table>
<thead>
<tr>
<th></th>
<th>senate</th>
<th>legalization</th>
<th>l</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rhino</td>
<td>yes</td>
<td></td>
<td>0.04</td>
<td>0.11</td>
</tr>
<tr>
<td>Rhino</td>
<td>no</td>
<td></td>
<td>0.1</td>
<td>0.15</td>
</tr>
<tr>
<td>Donkey</td>
<td>yes</td>
<td></td>
<td>0.22</td>
<td>0.27</td>
</tr>
<tr>
<td>Donkey</td>
<td>no</td>
<td></td>
<td>0.09</td>
<td>0.16</td>
</tr>
<tr>
<td>Elephant</td>
<td>yes</td>
<td></td>
<td>0.05</td>
<td>0.13</td>
</tr>
<tr>
<td>Elephant</td>
<td>no</td>
<td></td>
<td>0.21</td>
<td>0.26</td>
</tr>
</tbody>
</table>

mayor = Donkey{senate, legalization}

$$\omega: \quad \pi_{\{\text{senate}\}}(S_2)$$

<table>
<thead>
<tr>
<th></th>
<th>senate</th>
<th>l</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rhino</td>
<td></td>
<td>0.04</td>
<td>0.11</td>
</tr>
<tr>
<td>Rhino</td>
<td></td>
<td>0.1</td>
<td>0.15</td>
</tr>
<tr>
<td>Donkey</td>
<td></td>
<td>0.22</td>
<td>0.27</td>
</tr>
<tr>
<td>Donkey</td>
<td></td>
<td>0.09</td>
<td>0.16</td>
</tr>
<tr>
<td>Elephant</td>
<td></td>
<td>0.05</td>
<td>0.13</td>
</tr>
<tr>
<td>Elephant</td>
<td></td>
<td>0.21</td>
<td>0.26</td>
</tr>
</tbody>
</table>

mayor = Donkey{senate, legalization}

Figure 5.8: Projection on the participating random variables.

### 5.3.3 Conditionalization

Conditionalization was first considered as an operation of a relational algebra related to a probabilistic data model by Dey and Sarkar [25]; Classical relational algebra has no prototype of it. Intuitively, conditionalization is the operation of computing a conditional probability distribution, given a joint probability distribution. To simplify the definition below, we will employ the following
notation. Let $V = \{v_1, \ldots, v_n\}$ be a set of random variables and let $v \in V$ and $V' = V - \{v\}$. Let $I : \text{dom}(V) \to [0, 1]$ be a p-interpretation. Let $X = \{x_1, \ldots, x_m\} \subset \text{dom}(v)$ and $\bar{y} \in \text{dom}(V')$. Then $I[X](\bar{y})$ denotes the following sum: $I[X](\bar{y}) = \sum_{i=1}^{m} I(\bar{y}, x_i)$. With this notation in mind, we define conditionalization as follows.

**Definition 5.21** Let $S = \langle T^+, V, P, C^+, \omega \rangle$ be an ESPO, $|V| > 1$, $v \in V$ and $c : v = \{x_1, \ldots, x_m\}$ be a conditional selection condition. Then, the result of conditionalization of $S$ on $c$, denoted $\mu_c(S)$ is the ESPO $S' = \langle T^+, V', P', C^+, \omega' \rangle$, where

- $V' = V - \{v\}$. Without loss of generality, we will assume further that $V = \{v_1, \ldots, v_n\}$, $v = v_n$ and therefore $V' = \{v_1, \ldots, v_{n-1}\}$.
- $C^+ = C^+ \cup \{(v, X, V')\}$, where $X = \{x_1, \ldots, x_m\}$.
- $P' : \text{dom}(V') \to C[0,1]$ is defined as
  
  $$
P'(\bar{y}) = \left[ \min_{I' \models P} \left( \frac{I_X(\bar{y})}{\sum_{\bar{y}' \in \text{dom}(V')} I_X(\bar{y}')} \right), \max_{I' \models P} \left( \frac{I_X(\bar{y})}{\sum_{\bar{y}' \in \text{dom}(V')} I_X(\bar{y}')} \right) \right].$

- $\omega' = \mu_c(\omega)$.

From the definition above, it follows that in order to compute the result of conditionalization of an ESPO (in particular, in order to compute the resulting probability distribution) a number of non-linear optimization problems have to be solved. As it turns out, the new probability distribution can be computed directly (i.e., both minimization and maximization problems that need to be solved have analytical solutions). Dekhtyar and Goldsmith [19] solved the problem of computing the result of conditionalization of interval probability distributions. We present here their final formula, while referring the readers to [19] for a more detailed discussion of its derivation.

**Theorem 5.3** Let $S = \langle T^+, V, P, C^+, \omega \rangle$ be an ESPO, $c : v = \{x_1, \ldots, x_m\}$ be a conditional selection condition and $v \in V$. Let $V' = V - \{v\}$, $X = \{x_1, \ldots, x_m\}$ and $\bar{y} \in \text{dom}(V')$. The result of the conditionalization is denoted $S' = \mu_c(S) = \langle T^+, V', P', C^+, \omega' \rangle$. If we define $l[X]_{\bar{y}}$ and $u[X]_{\bar{y}}$ as follows:

$$l[X]_{\bar{y}} = \max \left( \sum_{x \in X} l(\bar{y}, x) ; 1 - \sum_{\bar{y}' \neq \bar{y} \text{ or } x' \not\in X} u(\bar{y}', x') \right),$$

^2We note, however, that Jaffray [54] has shown that conditioning interval probabilities is a dicey matter: the set of point probability distributions represented by $P'(\bar{y})$ will contain distributions $I'$ which do not correspond to any $I$ in $P$. 

118
\[ u[X]_g = \min \left( 1 - \sum_{\bar{y} \neq g \text{ or } x' \in X} l(\bar{y}, x') + \sum_{x \in X} u(\bar{g}, x) \right), \]
then the following expression correctly computes the lower and upper bounds of the conditional probability distribution for the resulting ESPO object.

\[
P^t(\bar{g}) = \left[ \frac{\min \left( 1 - \sum_{x' \in X} l(\bar{g}, x') + \sum_{x \in X} u(\bar{g}, x) + l[X]_g \right)}{\max \left( \sum_{x \neq g} x \in X l(\bar{y}, x) + u[X]_g \right) - \sum_{x' \in X} l(\bar{g}, x')} \right].
\]

The proof of this theorem can be found in [19]. The following example will illustrate how the conditionalization operation works.

**Example 5.15** Consider the ESPO \( S_2 \) in Figure 5.3. In this example, we illustrate the process of computing the conditionalization \( \mu_{\text{legalization}=\text{yes}}(S_2) \), as shown in Figure 5.9. First we collapse all the rows that do not satisfy the condition \text{legalization} = \text{yes} into one row. Next, we do a tightening operation on the new probability distribution. Then what we need to do is a normalization operation, which means that we must find the minimum and maximum values of the expressions of the form

\[
l[w, \text{yes}] = l(\text{R}hino, \text{yes}) + l(\text{D}onkey, \text{yes}) + l(\text{E}lephant, \text{yes}) \quad \text{for} \quad w \in \{\text{R}hino, \text{D}onkey, \text{E}lephant\} \quad \text{over all p-interpretations} \quad I \models P.
\]

Let us determine the lower bound for \( x = \text{R}hino \). Consider the following function \( f \) of three variables: \( f(x, y, z) = \frac{x}{x+y+z} \). For positive \( x \), \( y \) and \( z \), we could rewrite the function as \( f(x, y, z) = \frac{1}{1 + \frac{x}{y+z}} \). So, in order to minimize \( f \) we need to minimize \( x \) and maximize \( y+z \). In this case, we need to minimize \( l(\text{R}hino, \text{yes}) \) and maximize \( l(\text{D}onkey, \text{yes}) + l(\text{E}lephant, \text{yes}) \), i.e., \( l(\text{R}hino, \text{yes}) = 0.04 \) and \( l(\text{Donkey, yes}) + l(\text{Elephant, yes}) = \min(0.27 + 0.13, 1 - 0.04 - 0.49) = 0.4 \). Then the minimum value of \( l(\text{R}hino, \text{yes}) + l(\text{Donkey, yes}) + l(\text{Elephant, yes}) \) is \( \frac{0.04 + 0.4}{0.04 + 0.4} = 0.09 \).

Similarly, we can determine the upper bound for \( x = \text{R}hino \). We need to maximize \( l(\text{R}hino, \text{yes}) \) and minimize \( l(\text{Donkey, yes}) + l(\text{Elephant, yes}) \), i.e., \( l(\text{R}hino, \text{yes}) = 0.11 \) and \( l(\text{Donkey, yes}) + l(\text{Elephant, yes}) = \max(0.22 + 0.05, 1 - 0.11 - 0.57) = 0.32 \). Then the maximum value of \( l(\text{R}hino, \text{yes}) + l(\text{Donkey, yes}) + l(\text{Elephant, yes}) \) is \( \frac{0.11 + 0.32}{0.11 + 0.32} = 0.36 \). We can apply similar operations for \( x = \text{D}onkey \) and \( x = \text{E}lephant \). After that, the tightening operation is performed to find the true intervals. Finally, we exclude respondents:195 from the context as it is associated with \text{legalization}.
Figure 5.9: Conditionalization operation.
variable and add \texttt{legalization=yes} to the conditionals. The resulting ESPO is shown in the bottom of Figure 5.9.

5.3.4 Cartesian Product and Join

The Cartesian product of two ESPOs can be viewed as the joint probability distribution of the random variables from both objects. As only point probabilities were used in [20], we made there an assumption of independence between the random variables in the SPOs being combined. As probability distribution functions considered here are interval, this restriction will be removed. Also, the use of extended context and extended conditionals in ESPOs will allow us to make Cartesian product compatibility much less restrictive.

a. Probabilistic Conjunctions

Probabilistic conjunctions are interval functions (operations) that are used to compute the probability of a conjunction of two events given the probabilities of individual events. Typically, each probabilistic conjunction operation would have an underlying assumption about the relationship between the events involved, such as independence, ignorance, positive or negative correlation. Probabilistic conjunctions had been introduced by Lakshmanan, et al. [58], where they have also been used in defining the Cartesian product operation. Our definitions are borrowed from [58] and [22].

First, recall the standard “truth-ordering” on intervals: 1. \( [L_1, U_1] \leq [L_2, U_2] \iff (L_1 \leq L_2 \wedge U_1 \leq U_2). 2. [L_1, U_1] \geq [L_2, U_2] \iff (L_1 \geq L_2 \wedge U_1 \geq U_2).

\textbf{Definition 5.22 (probabilistic conjunction/disjunction strategy)} A probabilistic conjunction \( \otimes \) is a binary operation \( \otimes : C[0,1] \times C[0,1] \rightarrow C[0,1] \) that obeys the following postulates:

<table>
<thead>
<tr>
<th>Postulates</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Commutativity</td>
<td>([l_1, u_1] \otimes [l_2, u_2] = [l_2, u_2] \otimes [l_1, u_1])</td>
</tr>
<tr>
<td>2. Associativity</td>
<td>([(l_1, u_1) \otimes (l_2, u_2)] \otimes (l_3, u_3) = [(l_1, u_1) \otimes (l_2, u_2 \otimes (l_3, u_3))])</td>
</tr>
<tr>
<td>3. Monotonicity</td>
<td>([l_1, u_1] \otimes (l_2, u_2) \leq [l_1, u_1] \otimes (l_3, u_3) \text{ if } [l_2, u_2] \leq [l_3, u_3])</td>
</tr>
<tr>
<td>4. Bottomline</td>
<td>([l_1, u_1] \otimes [l_2, u_2] \leq \min(l_1, l_2), \min(u_1, u_2))</td>
</tr>
<tr>
<td>5. Identity</td>
<td>([l_1, u_1] \otimes [1, 1] = [l_1, u_1])</td>
</tr>
<tr>
<td>6. Annihilator</td>
<td>([l_1, u_1] \otimes [0, 0] = [0, 0])</td>
</tr>
<tr>
<td>7. Ignorance</td>
<td>([l_1, u_1] \otimes (l_2, u_2) \leq \max(0, l_1 + l_2 - 1), \min(u_1, u_2))</td>
</tr>
</tbody>
</table>

The following are some sample probabilistic conjunction operations ([22, 58]).
b. Cartesian Product

As different probabilistic conjunction operations compute the probabilities of conjunction of two events in different ways, there is no unique Cartesian product operation. Rather, for each probabilistic conjunction $\otimes_\alpha$, we define a Cartesian product operation $\otimes_\alpha$.

**Definition 5.23** Let $S = \langle T^+, V, P, C^+, \omega \rangle$ and $S' = \langle T'^+, V', P', C'^+, \omega' \rangle$ be two ESPOs. Let $V = \{v_1, \ldots, v_n\}$, $V' = \{v'_{l_1}, \ldots, v'_{l_m}\}$, $U = \{u \in \mathcal{V} | (u, X, V^*) \in C^+\}$, $U' = \{u' \in \mathcal{V} | (u', X', V^*) \in C'^+\}$. $S$ and $S'$ are **Cartesian product-compatible** iff (i) $V \cap V' = \emptyset$; (ii) $U \cap V' = \emptyset$, and (iii) $V \cap U' = \emptyset$.

Cartesian product compatibility of two ESPOs means that the joint probability distribution of the random variables from both objects is meaningful. In particular, we require the sets of participating random variables to be disjoint (leaving the other case to be handled by the join operation). We also want the set of random variables found in the conditionals of one ESPO to be disjoint from the participating variables of the other. Thus, for example, the Cartesian product of the probability distribution of mayor votes for respondents who will vote Donkey for senator cannot be combined with the probability distribution of senator votes. We can now define the Cartesian product.

**Definition 5.24** Let $S = \langle T^+, V, P, C^+, \omega \rangle$ and $S' = \langle T'^+, V', P', C'^+, \omega' \rangle$ be two Cartesian-product compatible ESPOs. Let $V = \{v_1, \ldots, v_n\}$, $V' = \{v'_{l_1}, \ldots, v'_{l_m}\}$, $U = \{u \in \mathcal{V} | (u, X, V^*) \in C^+\}$, $U' = \{u' \in \mathcal{V} | (u', X', V^*) \in C'^+\}$. Let $\otimes_\alpha$ be some probabilistic conjunction. The **Cartesian product** of $S$ and $S'$ under probabilistic conjunction $\otimes_\alpha$, denoted $S \times_\alpha S'$, is defined as $S \times_\alpha S' = S'' = \langle T''^+, V'', P'', C''^+, \omega'' \rangle$, where

- $V'' = V \cup V'$;
- $T''^+ = \{(A, a, V^*) | (A, a, V^*) \in T^+ \text{ and } \text{no } (A, a, V_s) \in T'^+ \text{ or } (A, a, V^*) \in T'^+ \text{ and } \text{no } (A, a, V_*) \in T' + \text{ or } (A, a, V_1^*) \in T' + \text{ and } (A, a, V_2^*) \in T' + \text{ and } V^* = V_1^* \cup V_2^* \}$;
- $C''^+ = \{(u, X, V^*) | (u, X, V^*) \in C^+ \text{ and } \text{no } (u, X, V_*) \in T'^+ \text{ or } (u, X, V^*) \in T'^+ \text{ and } \text{no } (u, X, V_1^*) \in T' + \text{ and } (u, X, V_2^*) \in T' + \text{ and } V^* = V_1^* \cup V_2^* \}$;
\[ P'' : dom(V'') \rightarrow C[0,1] \] is defined as follows. Let \( \tilde{x} \in dom(V) \), \( \tilde{x}' \in dom(V') \) (hence \((\tilde{x}, \tilde{x}') \in dom(V''))). Then, \( P''((\tilde{x}, \tilde{x}')) = P(\tilde{x}) \otimes_{\alpha} P'(\tilde{x}') \).

- \( \omega'' = \omega \times_{\alpha} \omega' \).

In Cartesian product the context and the conditionals of the two initial ESPOs are united; if a particular context record or a conditional appears in both ESPOs then their association lists are merged. The new set of participating variables is the union of the two original sets. Finally, the probability interval for each instance (row) of the new probability table is computed by applying the probabilistic conjunction operation to the appropriate rows of the two original tables.

c. Join

Join in ESP-Algebra is similar to Cartesian product in that it computes the joint probability distribution of the input ESPOs. The difference is that join is applicable to the ESPOs that have common participating random variables. Let \( S = \langle T^+, V, P, C^+, \omega \rangle \) and \( S' = \langle T^{+'}, V', P', C^{+'}, \omega' \rangle \), and let \( V^* = V \cap V' \neq \emptyset \) and participating random variables of \( S \) are not conditioned in \( S' \) and vice versa. If these conditions are satisfied, we call \( S \) and \( S' \) join-compatible.

**Definition 5.25** Let \( S = \langle T^+, V, P, C^+, \omega \rangle \) and \( S' = \langle T^{+'}, V', P', C^{+'}, \omega' \rangle \) be two ESPOs. Let \( V = \{ v_1, \ldots, v_n \} \), \( V' = \{ v'_1, \ldots, v'_m \} \), \( U = \{ u \in V \setminus \{ v \} \} \), \( U' = \{ u' \in V' \setminus \{ v' \} \} \), \( C^+ \) and \( C^{+'} \). \( S \) and \( S' \) are join-compatible iff (i) \( V \cap V' \neq \emptyset \); (ii) \( U \cap U' = \emptyset \), and (iii) \( V \cap U' = \emptyset \).

Consider three value vectors \( \tilde{x} \in dom(V - V^*) \), \( \tilde{y} \in dom(V^*) \) and \( \tilde{z} \in dom(V' - V^*) \). The join of \( S \) and \( S' \) is the joint probability distribution \( P''(\tilde{x}, \tilde{y}, \tilde{z}) \) of \( V \) and \( V' \), or, more specifically, of \( V - V^* \), \( V^* \) and \( V' - V^* \). To construct this joint distribution, we recall from probability theory that under assumption \( \alpha \) about the relationship between the random variables in \( V \) and \( V' \) and independence between variables in \( V - V^* \) and in \( V' - V^* \), we have \( p(\tilde{x}, \tilde{y}, \tilde{z}) = p(\tilde{x}, \tilde{y}) \otimes_{\alpha} p(\tilde{z} | \tilde{y}) \) and, symmetrically, \( p(\tilde{x}, \tilde{y}, \tilde{z}) = p(\tilde{x} | \tilde{y}) \otimes_{\alpha} p(\tilde{y}, \tilde{z}) \). \( p(\tilde{x}, \tilde{y}) \) is stored in \( P \), the probability table of \( S \), and \( p(\tilde{z} | \tilde{y}) \) is the conditional probability that can be computed by conditioning \( p(\tilde{y}, \tilde{z}) \) (stored in \( P' \)) on \( \tilde{y} \). The second equality can be exploited in the same manner.

This gives rise to two families of join operations, left join \( (\kappa_\alpha) \) and right join \( (\times_\alpha) \) defined as follows.

**Definition 5.26** Let \( S = \langle T^+, V, P, C^+, \omega \rangle \) and \( S' = \langle T^{+'}, V', P', C^{+'}, \omega' \rangle \) be two join-compatible ESPOs. Let \( V^* = V \cap V' \neq \emptyset \) and \( V_1 = V - V^* \) and \( V'_1 = V' - V^* \). We define the operations of left join of \( S \) and \( S' \), denoted \( S \kappa_\alpha S' \) and right join of \( S \) and \( S' \), denoted \( S \times_\alpha S' \) under assumption \( \alpha \) as follows:
for all of the three random variables. We may apply a join operation to these ESPOs since they are join-compatible.

Example 5.16 Consider the two ESPOs \( S_2 \) and \( S_3 \) in Figure 5.3. They are joint probability distributions for (senate, legalization) and (park, legalization), respectively. However, in some circumstances we may want to combine these two ESPOs and obtain the joint probability distribution for all of the three random variables. We may apply a join operation to these ESPOs since they are join-compatible according the definition. We’ll illustrate how to obtain the left join under the assumption of independence: \( S_2 \bowtie_{\text{ind}} S_3 \), as follows.

The join operation combines three operations: conditionalization, selection and Cartesian product. First, we need to calculate the results for conditionalization of the left operand (i.e., \( S_2 \)) on the set of common variables (in this case, legalization), as shown in the top right part of Figure 5.10. Second, we do selections on probability table for all the possible values of the common variables, namely, \( \sigma_{\text{legalization}=\text{yes}}(S_3) \) and \( \sigma_{\text{legalization}=\text{no}}(S_3) \). The resulting ESPOs are shown.
### Figure 5.10: Join operation (left join) in ESP-Algebra

<table>
<thead>
<tr>
<th>( \omega: \quad S_2 )</th>
<th>( \omega: \quad \mu_{\text{legalization}=\text{yes}}(S_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>date: October 23</td>
<td>date: October 23</td>
</tr>
<tr>
<td>gender: male</td>
<td>gender: male</td>
</tr>
<tr>
<td>senate legalisation</td>
<td>( l \quad u )</td>
</tr>
<tr>
<td>Rhino yes</td>
<td>0.04 0.11</td>
</tr>
<tr>
<td>Rhino no</td>
<td>0.1 0.15</td>
</tr>
<tr>
<td>Donkey yes</td>
<td>0.22 0.27</td>
</tr>
<tr>
<td>Donkey no</td>
<td>0.09 0.16</td>
</tr>
<tr>
<td>Elephant yes</td>
<td>0.05 0.13</td>
</tr>
<tr>
<td>Elephant no</td>
<td>0.21 0.26</td>
</tr>
<tr>
<td>mayor = Donkey</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \omega: \quad S_3 )</th>
<th>( \omega: \quad \mu_{\text{legalization}=\text{no}}(S_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>locality: Sunny Hill</td>
<td></td>
</tr>
<tr>
<td>date: October 26</td>
<td>date: October 23</td>
</tr>
<tr>
<td>park legalisation</td>
<td>( l \quad u )</td>
</tr>
<tr>
<td>yes yes</td>
<td>0.56 0.62</td>
</tr>
<tr>
<td>yes no</td>
<td>0.14 0.2</td>
</tr>
<tr>
<td>no yes</td>
<td>0.21 0.25</td>
</tr>
<tr>
<td>no no</td>
<td>0.03 0.07</td>
</tr>
<tr>
<td>major = Donkey</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \omega: \quad \sigma_{\text{legalization}=\text{yes}}(S_3) )</th>
<th>( \omega: \quad \sigma_{\text{legalization}=\text{no}}(S_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>locality: Sunny Hill</td>
<td></td>
</tr>
<tr>
<td>date: October 26</td>
<td>date: October 26</td>
</tr>
<tr>
<td>park legalisation</td>
<td>( l \quad u )</td>
</tr>
<tr>
<td>yes yes</td>
<td>0.56 0.62</td>
</tr>
<tr>
<td>yes no</td>
<td>0.14 0.2</td>
</tr>
<tr>
<td>no yes</td>
<td>0.21 0.25</td>
</tr>
<tr>
<td>no no</td>
<td>0.03 0.07</td>
</tr>
<tr>
<td>major = Donkey</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Figure 5.10: Join operation (left join) in ESP-Algebra
in the bottom left part. Third, Cartesian product operations on corresponding ESPOs \(^4\) are applied based on the values of the common variables, namely, \(\mu_{\text{legalization}=\text{yes}}(S_2) \times \sigma_{\text{legalization}=\text{yes}}(S_3)\) and \(\mu_{\text{legalization}=\text{no}}(S_2) \times \sigma_{\text{legalization}=\text{no}}(S_3)\). In this particular example, we assume that the random variables in the two ESPOs are independent when we apply probability conjunctions. Finally, we union all the resulting ESPOs and apply a tightening operation on it. The final result is shown in the bottom right part of the figure.

\(^4\)Prior to applying Cartesian product operation, we project legalization = yes and legalization = no out of the conditionals.
Chapter 6

Building Bayesian Networks with SPDBMS

Bayesian networks have been used in various applications, such as medical diagnosis and financial planning. Bayesian networks \emph{per se} are not designed with parameter changes or probability updates in mind. However, modeling a large and complex problem domain such as those that arise in medical, social science and military applications may involve many knowledge engineers and domain experts in a complicated iterative process resulting in frequent refinement of model structure and probability distributions. Software is needed to conveniently accommodate such changes, and to efficiently produce the most up-to-date and appropriate Bayesian network on demand.

The Bayesian Network Development Suite (BaNDeS) described here addresses exactly this problem. It is built on the idea of a central repository that stores information about all currently available Bayesian networks in the database, including network structures and conditional probability tables (CPTs). This repository resides on a central database server and is accessed and updated by special-purpose software tools that perform specific model-building subtasks.

6.1 Introduction

Bayesian networks, based on Bayes’ theory, have been introduced in the 1980s as a formalism for representing uncertain information and reasoning under uncertainty. Bayesian networks have the ability to conveniently describe the influences and probabilistic interactions among random variables. It provides a better solution over other approaches and overcomes most of the limitations. However, this theory does not gain enough attention among researchers until people started exploring the possibilities for developing medical applications [31, 65].

There have been great advances in algorithm design in Bayesian network inference [46] and planning with Bayesian networks [9] during the last two decades. There are both commercial [64, 52] and open-source [15] software tools available for development and inference with Bayesian networks. These tools create an integrated environment for their users, allowing them to input all necessary information for constructing a network, including random variables, dependencies between them and conditional probability tables, and to conduct inference with the constructed Bayesian network. As the study of inference and planning with Bayesian nets has matured, the range of applications for which Bayesian networks are suitable has become noticeably broader. Better planning and inference engines allow for larger and more complex stochastic domains to be modeled.
and processed. The integrated approach, good for development of small and medium-size models by individual knowledge engineers, is no longer convenient when the application domains being modeled become larger and the teams become bigger as well. Instead, modeling large, dynamic domains often requires separation of tasks, rather than their integration.

To design a complete Bayesian network model for a large application, a team of knowledge engineers must determine the random variables present in the application and their domains; possible meta-information attributes that describe specific situations in the application; conditional dependencies between the domain elements (Bayesian network structure), and the conditional probability tables associated with Bayesian network nodes and specific situations. While determination of variables must precede all other tasks, in large applications, determination of the network structure and construction of the conditional probability tables often become parallel, and even asynchronous processes, that may be conducted by different team members in collaboration with different groups of domain experts. The number of domain experts needed to construct a successful model also grows drastically: large applications may span a number of fields of science, engineering, medicine and/or social science, and the knowledge of domain experts becomes specialized and applicable only to a particular fragment of the entire framework.

The process of building a Bayesian network involves defining random variables of interest in an application domain, conditional dependencies among variables, and conditional probability distributions. Here we propose a new approach to building, updating and maintaining large Bayesian network models. Our work is particularly appropriate for such large application domains. The Bayesian network development suite (BaNDeS) described here is based on our implementation of the Semistructured Probabilistic DBMS (SPDBMS), which provides robust storage and retrieval mechanisms for large-scale collections of probability distributions. On top of the SPDBMS, we build client applications designed to deal with specific subtasks within the model construction problem. The client applications we provide allow users to accomplish the latter tasks asynchronously.

The Bayesian Network Builder (BNB) \(^1\) allows the knowledge engineer to construct a dependency structure, obtain the corresponding conditional probability distributions from the SPDBMS, and update the network structure. The Probability Elicitation Tool (PET) takes dependencies from the BNB’s output, and elicits probabilities from domain experts, which are stored as SPOs in the SPDBMS. Since the BNB queries the SPDBMS server each time a particular node is called, it can access elicited probabilities immediately. This allows experts to work on specific subtasks of the model construction process independently and asynchronously while updating the model in real

---

\(^1\)The BNB was developed jointly with Erik Jessup, and I was in charge of the design and implementation of the graphical user interface (GUI); The PET was initially developed by Jiangyu Li.
What distinguishes our approach from others is the following. On one hand, we integrate our Bayesian network builder with the SPDBMS. There are many reasons why a Bayesian network builder would use a database as a backend instead of the straightforward text-based systems common to most Bayesian network software, or a simple hashing system. The success of relational databases as opposed to ad-hoc approaches to storing and retrieving information suggests that application of database techniques for the problem of storing and manipulating large quantities of probabilistic data may be (i) more convenient and (ii) more efficient. In general, efficient management and manipulation of large collections of data are successfully handled by databases. Here, we describe how the power of the database approach can be applied to the problem of storage and management of large quantities of diverse probability tables along with other associated information. The SPDBMS described here encapsulates the above mentioned problems, allowing the knowledge engineer to focus on other problems, such as deriving the specifications of the personalized model from the query and the knowledge base. On the other hand, we split the entire system into subsystems, each of which performs a specific subtask. An expert with particular expertise needs only to work with a specific subsystem. For example, experts can use BNB to dynamically construct Bayesian network structures and navigate probability distributions for each random variable in the network. Experts can input conditional probability tables for a specific Bayesian network through a user-friendly graphical interface provided by PET.

6.2 Bayesian Network Construction

Bayesian networks can be constructed through domain experts’ knowledge, learned from data, or both. Here we use Welfare to Work as an example for constructing Bayesian networks, and will focus on the former — constructing Bayesian network from expert’s knowledge. Expert knowledge is so important when building Bayesian networks for applications such as Welfare to Work because there is very little data in written format and readily available. For many other applications such as building models for a gene regulatory network, however, there are large quantities of high-throughput experimental data available. In the latter case, machine learning algorithms are usually applied to learn Bayesian networks, including probabilities and/or structures, from the massive data.

At present, the Bayesian network development suite (BaNDeS) described here consists of three components, all implemented in Java. The architecture of BaNDeS is shown in Figure 6.1. The backbone of BaNDeS is the Semistructured Probabilistic DBMS (SPDBMS) server — our implementation of the SPO framework described in Chapter 4. Two other components were developed.
The Bayesian Network Builder (BNB) allows its users to define the application domain and construct the Bayesian network structure for it. The Probability Elicitation Tool (PET) facilitates elicitation of conditional probability tables from domain experts. Both PET and BNB use SPDBMS as the source of their input and the repository of the data they generate.

6.2.1 The Role of SPDBMS

Generally speaking, a Bayesian network model has two types of information. One is the network topology, which specifies the actual structure of the network; and the other is a collection of conditional probability tables (CPTs), which provides quantitative information for the model. During the process of constructing Bayesian network, the network structure and the CPTs need to be accessed and updated frequently. The I/O access time may become a major computational bottleneck. The SPDBMS provides the right solution for storing and managing those information efficiently. It provides the following: (i) facilities for accessing the network structure file; (ii) facilities for accessing a specific set of CPTs in the network; (iii) facilities for accessing a specific part of a CPT; (iv) facilities for transforming probability distributions according to the laws of probability.

First we extend the concept of conditional probability table (CPT) to incorporate element con-
which associates contextual information with each CPT. In this chapter, we will only consider extended conditional probability tables (eCPTs), and we will use the terms eCPT and CPT interchangeably. Formally, the extended conditional probability table (eCPT) is defined as follows.

**Definition 6.1** An extended conditional probability table $T$ is a tuple $T = (v, D, C, t)$, where

- $v$ is the random variable for which $T$ is defined;
- $C = (v_1, v_2, \ldots, v_m)$ is the set of $v$’s parents in the Bayesian network;
- $D : \text{dom}(v) \times \text{dom}(C) \rightarrow [0, 1]$ is the conditional probability function, where $\text{dom}(C) = \text{dom}(v_1) \times \ldots \times \text{dom}(v_m)$;
- $t$ is a set of global context associated with the Bayesian network.$^2$

In our approach, Bayesian network structures and CPTs are stored independently in the SPDBMS. The SPDBMS server allows storage and retrieval of both types of information. Bayesian network structures are encoded and stored as XML files, with storage operation validating the XML file against the Bayesian network structure XML Schema.$^3$ To store conditional probability tables (CPTs), the CPTs need to be converted into SPOs. In the section below, we describe how a CPT can be represented as a collection of SPOs.

Consider a Bayesian network for Welfare to Work program, as shown in Figure 6.3. Given a Bayesian network node, i.e., the random variable Communication Skills in the network, its extended conditional probability table (eCPT), as shown in the top of Figure 6.2, represents the probability distribution for the level of communication skills for 25–30 year-old women with different literacy levels.$^4$ Each combination of values of the parent nodes gives rise to a single probability distribution and consequently to a single SPO object. For example, if the domain of the literacy random variable is “good”, “fair” and “poor”, the complete eCPT for the communication skills random variable consists of three SPOs, with one for each of those values in the domain. The three SPOs which represent the eCPT for the variable Communication skills are shown in the bottom of Figure 6.2. Each SPO contains a “slice” of the conditional probability table for the dependency of the communication skills random variable on the literacy random variable. The context of the

$^2$Local context is considered in the Knowledge Based Model Construction (KBMC) framework, but not included in the current implementation.

$^3$The Bayesian network structure encoding is a derivative of the XMLBif format [16], without the description for probability tables. See Appendix * for the actual XML schema.

$^4$In the Welfare to Work application, communication skills are the basic skills used to determine possible training options. Generally speaking, it depends on more than just literacy, however in the network fragment used in this paper, we simplify the dependency structure to produce a compact example.
SPOs (top portion) consists of two attributes specifying gender and age. Communication skills is the sole participating random variable, whose domain consists of four values: “excellent”, “good”, “so-so” and “poor”, and the conditional part of the SPO stores the fact that the value of the literacy random variable is either “good”, “fair” or “poor”. In this chapter, we restrict the format of the SPOs stored in the SPDBMS to such eCPT “slices”: conditional probability distributions with a single participating random variable. In general, SPO framework allows to store a wide variety of joint and conditional probability distributions.

<table>
<thead>
<tr>
<th>Gender: Female</th>
<th>Age: 20-30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literacy</td>
<td>Communication_skills</td>
</tr>
<tr>
<td>Good</td>
<td>0.523 0.315 0.131 0.031</td>
</tr>
<tr>
<td>Fair</td>
<td>0.400 0.285 0.185 0.130</td>
</tr>
<tr>
<td>Poor</td>
<td>0.334 0.215 0.210 0.241</td>
</tr>
</tbody>
</table>

Figure 6.2: A typical eCPT for the random variable Communication_skills, and its representation as a collection of SPOs.

Recall that, in order to store SPOs, SPDBMS server uses as its back end a relational database accessed via an RDBMS (currently Oracle 8i). The SPDBMS server implements the insertion and deletion operations, as well as a full array of SP-Algebra query operations [20]. The server provides a convenient Java API through which BaNDeS components (as well as other client applications in general) can connect to it, and pass information processing and retrieval instructions. To access information stored in the SP database, we has proposed a Semistructured Probabilistic query algebra (SP-Algebra). It extends the definitions of standard relational operations and includes selection (σ), projection (π), Cartesian product (×) and join (⋈) operations as well as a new operation of conditionalization (μ). Detailed information can be found in Chapter 3.
6.2.2 Network Topology Construction

The process of Bayesian network construction starts with the Bayesian Network Builder. This tool (see Figure 6.3) provides a simple GUI that allows knowledge engineers to specify random variables, their domains and dependencies between them. The interface, similar to the graph-drawing interfaces of standard Bayesian network inference tools such as Hugin [52] and JavaBayes [15], allows the users to create network nodes on the canvas, describe their domains, move them around and draw edges signifying dependencies. As work on the Bayesian network structure proceeds, BNB maintains an internal representation of the current state of the Bayesian network.

Once a preliminary network is constructed, it can be saved to the SPDBMS server. The internal representation of the network will be first translated into XML format. A fragment of a sample XML file for the W2W network topology is shown in Figure 6.4. The network description is then sent to the SPDBMS server through the SPDBMS JDBC-like API. For example, a new SP-relation will
be created by calling the CREATE operation (i.e., in this case, CREATE \texttt{W2W}), followed by an
INSERTX operation (i.e., INSERTX INTO \texttt{W2W w2w.xml})\footnote{Here we assume the SP-relation name is \texttt{W2W} and the XML file to insert is called \texttt{w2w.xml}.}, which actually inserts the XML
file into the newly created SP-relation.

\begin{verbatim}
<?xml version="1.0"?>
<UKBIF VERSION="1.0"
    xsi:noNamespaceSchemaLocation="http://www.cs.uky.edu/~wzhao0/spo/schema/bnschema.xsd">
<NThickNetwork>
    <NAME>W2W</NAME>
    <VARIABLE>
        <NAME>Communication_Skills</NAME>
        <OUTCOME>Excellent</OUTCOME>
        <OUTCOME>Good</OUTCOME>
        <OUTCOME>Fair</OUTCOME>
        <OUTCOME>Poor</OUTCOME>
        <PARENTS>Literacy</PARENTS>
    </VARIABLE>
    ... 
</NETWORK>
</UKBIF>
\end{verbatim}

Figure 6.4: A sample XML file which represents the W2W Bayesian network topology.

\subsection{6.2.3 Probability Elicitation}

From this moment, the Bayesian network building process becomes distributed and asyn-
chronous. The network description can be imported from the SPDBMS server using a GETX oper-
ation (i.e., GETX \texttt{W2W}) into the Probability Elicitation Tool. This description allows the tool to
construct the list of CPTs that need to be elicited from domain experts. The main window of PET
is shown in Figure 6.5. The domain expert (with or without the help of a knowledge engineer) can
choose the CPTs that they want to work on using a SELECT query (i.e., $\sigma_{v} = \{x_{i}\}(W2W)$, where
$X_{i}$ is the target random variable), and can construct one CPT at a time. The user interface provides
a number of different means for entering probability values: a spreadsheet-like environment at the
top of the page, the vertical slider bars at the bottom of the screen that can be used to provide both
normalized and non-normalized user probability assessments (and controlled via a variety of inter-
face widgets) and a number of common preset distributions. The slider scales use both numeric
probability values and verbal probability scales allowing the users uncomfortable with numbers to
express their probability distributions in terms like “usually”, “frequently”, “unlikely”, “probably”
and such. Non-normalized distributions entered by the user (such as the user selecting “usually” as the value of for all instances) are automatically normalized by PET.

![Figure 6.5: Screenshot of the current probability elicitation toolkit.](image)

Once a user is satisfied with a specific CPT (which may still be incomplete), (s)he can save it to the database. The CPT gets converted into a collection of SPOs as described above and submitted to the SPDBMS server for insertion into the appropriate SP-relation. In this case, the query syntax for INSERT looks like `INSERT INTO W2W communication_skills.xml`.

### 6.2.4 Browse Bayesian Network Information

The moment a CPT (in SPO format) is inserted into the SPDBMS, it becomes available to the Bayesian Network Builder as well. Each time a node $X_i$ (with parents $Y_1, \ldots, Y_n$) of the Bayesian network $B$ is selected in the Bayesian Network Builder, the SP-Algebra query

$$
\sigma_{W} = \{X_i\} \land \forall Y_1 \in \text{dom}(Y_1) \land \ldots \land \forall Y_n \in \text{dom}(Y_n) (B)
$$


135
Figure 6.6: A sample XML file communication_skills.xml which represents the CPT for variable Communication_Skills as SPOs in the W2W Bayesian network.

(meaning "Select from SP-relation B all SPOs with participating random variable X_i and conditioned by random variables Y_1,...,Y_k") is issued to the SPDBMS server. The answer set to this query is exactly the set of SPOs for the CPT of node X available in the database. The BNB receives the answer set from the SPDBMS server, parses it, and displays the current state of the requested CPT.

Figure 6.3 pictures the state of the Bayesian Network Builder after the node Communication_Skills was selected by the user. The BNB has determined that this node had a single parent (Literacy) and has issued the query selecting all SPOs from the network W2W that have Communication_Skills as the participating random variable and the Literacy random variable in their conditional parts. The query is passed to the SPDBMS server, which in turn translates it into the sequence of SQL queries requesting the data necessary to construct the answer from the underlying RDBMS. Once the data is received, the SPDBMS query processor generates the XML representation of the returned SPOs and sends them back to the BNB, where the XML is parsed and a new window showing the contents of the requested CPT is displayed on the screen.
6.3 Bayesian Network Revision

For real life applications such as Welfare to Work, the Bayesian network needs to be accessed and revised frequently because the regulations change over time, and consequently so do the interdependency of services. Probability distributions will also change over time due to newly available information. So the capability of revision is so crucial to the success for a Bayesian network development system.

Before we get into the details, we introduce the concept of completeness of a Bayesian network.

**Definition 6.2** Given an extended conditional probability table \( T = (v, D, C, t) \), where \( v \) is the random variable for which \( T \) is defined, \( C = (v_1, v_2, \ldots, v_m) \) the set of \( v \)'s parents. \( D: \text{dom}(v) \times \text{dom}(C) \rightarrow [0, 1] \) is the conditional probability function. The conditional probability table \( T \) is complete if and only if \( \sum_{x \in \text{dom}(v)} D(x|c) = 1 \) holds for every \( c \in \text{dom}(C) \).

**Definition 6.3** Given a Bayesian network \( BN = (S, P) \), where \( S \) is the network structure and \( P \) the set of eCPTs. The Bayesian network \( BN \) is complete if and only if there exists one eCPT in \( P \) for each random variable in \( S \), and every eCPT is complete.

In the sections below, we will discuss how one can revise a Bayesian network within our framework, either in its network structure or in probability distributions.

6.3.1 Update Network Structures

Network structures can be updated through the user interface in the BNB. In order to modify the network structure, the BNB first retrieves the network file from the SPDBMS. After the current network is displayed on the screen, the user can make any changes such adding a node, removing a node, adding a dependency relationship, and removing a dependency relationship.

(a) When a node (random variable) is added into an existing Bayesian network, as shown in Figure 6.7(A), the system will do the following. First the system will update the network structure to include the new node. Second a new empty CPT for that node will be created and probabilities elicited through the PET.

(b) Adding a node into a network is straightforward. However, removing a node from the network is nontrivial. When a node is removed, as shown in Figure 6.7(B), the system first updates the network structure by removing all the dependencies from and into that node, and the node itself. Then the
6.3.2 Update Probability Distributions

Probability distributions can also be updated through the graphical user interface in PET. In order to update the probability distributions (CPTs), the network description must be imported from the SPDBMS server. This description allows the PET to list the available CPTs in a specific network, from which the user can choose to update. The CPTs being updated will be retrieved from the SPDBMS with a SELECT query, and displayed in the screen. The user can modify CPTs one at a time, and save each new CPT to the SPDBMS through an UPDATE operation. This operation
overwrites the old CPTs with the new ones.

6.3.3 Bayesian Network Consistency

What is Bayesian network consistency? Here, we call a Bayesian network consistent if the dependencies (in this case, parent-child relationship) among variables implicitly specified in all CPTs in the network match exactly those present in the network structure.

Definition 6.4 Given a complete Bayesian network $BN = (S, \mathcal{P})$, where $S$ is the network structure and $\mathcal{P}$ the set of eCPTs, $BN$ is consistent if and only if all dependencies implicitly specified in all eCPTs in $\mathcal{P}$ match exactly those present in the network structure $S$.

The SPDBMS, as a central data storage for managing network structure and CPTs, can help evaluate/maintain Bayesian network consistency. Here, we propose two efficient algorithms for maintaining Bayesian network consistency, as outlined in Figure 6.9. Basically, before any change

Figure 6.8: Add or remove a dependency.
in the state of an SP-relation is made, the database checks the database consistency for the specific Bayesian network. Following are the two basic scenarios. (i) When a network structure is modified: before the network structure in SPDBMS has been updated, all SPOs (representation of CPTs) in the specific network will be checked for consistency with the new network structure. All SPOs whose implied dependency does not match any dependencies in the network will be removed. (ii) When a CPT is modified: before a set of SPOs representing a CPT in the SPDBMS is updated, these SPOs will be checked for consistency with the current network structure.

Algorithm I: When a network structure is modified:
Step 1: Read the network structure XML file for the specific network;
Step 2: Parse the network structure XML file, and store the output in a DOM tree;
Step 3: Extract a set of all dependencies in the structure by traversing the DOM tree;
Step 4: Check if all SPOs in the specified network whose dependency matches any of those in the DOM tree;
Step 5: If YES, return;
Step 6: Construct an SP-Algebra query which deletes all SPOs whose dependency is not in the set of dependencies;
Step 7: Execute the SP-Algebra query against the specific SP-relation;
Step 8: Insert the new Bayesian network structure file.

Algorithm II: When a CPT is modified:
Step 1: Extract the dependency relationship implicitly specified in the new CPT;
Step 2: Read the network structure XML file for the specific network;
Step 3: Parse the network structure XML file, and store the output in a DOM tree;
Step 4: Check if the dependency relationship for the new CPT matches any of those in the DOM tree;
Step 5: If YES, then insert the new CPT and delete the old version; otherwise, send error message.

Figure 6.9: Algorithms for maintaining Bayesian network consistency

Proposition 6.1 All above update operations preserve Bayesian network consistency.
6.4 Conclusions and Ongoing Work

The framework proposed here can help teams of researchers build large and complex Bayesian networks in distributed and asynchronous fashion. This is the first implementation (to our knowledge) to incorporate probabilistic database technology into a solution of an Artificial Intelligence problem. The BaNDeS architecture is flexible and extensible: new client applications for performing other model-building tasks, such as data extraction from databases, can be seamlessly incorporated into it.

The framework described here is still in the initial stages of its development, both in terms of functionality and efficiency. In current system, the SPDBMS server can be accessed by BNB and PET from any computer connected to the Internet. Work on adding new functionalities and on making query processing more efficient is under way. The main development thrust in BaNDeS is towards making it suitable for KBMC applications by allowing for storage and retrieval of situation-specific information and reasoning designed to build situation-specific Bayesian network models based on the stored data. The SPO data model naturally allows for storage of such situation-specific data in the context part of SPOs and its retrieval based on the SP-Algebra queries on context. Both the Bayesian Network Builder and Probability Elicitation Tool need to acquire new functionality related to specification and use of such context information. Finally, decision procedures for KBMC reasoning need to be implemented. Work on this is also currently under way.
Chapter 7

Related Work

This work builds on the work of many people in three different fields: probabilistic databases, interval probabilities, and XML storage techniques. The overlap between these three fields is small, so we address them separately. Interval probabilities and XML storage techniques are surveyed in Section 7.2 and Section 7.3, respectively.

7.1 Probabilistic Databases

While modeling and managing uncertain information has received considerable attention in the last two decades, most of that work has been in the context of probabilistic relational databases. Different probabilistic relational models have been developed. Cavallo and Pittarelli [14] were among the first to address the problem of storing and querying probabilistic information in a database. Their probabilistic relations resemble a single probability table from our SPO. Their probabilistic system was defined as a four-tuple $B = (V, \Delta, \text{dom}, P)$, where

- $V$ is a non-empty set of distinct attributes;
- $\Delta$ is a non-empty set of domains of attributes;
- $\text{dom} : V \rightarrow \Delta$ is a function that associates a domain with each attribute; and
- $P : \text{dom}(V) \rightarrow [0, 1]$ is a probability distribution function over $V$.

Their system is similar to a standard relational database system except it uses a probability distribution function instead of a characteristic function as its fourth component. Their data model requires that the probabilities for all the tuples in a relation must add up to exactly 1. As a result, separate relations are needed to represent different objects. They focused their work on information content, functional dependency and multi-valued dependency. They used point probabilities and their algebra was quite restricted. They defined only two probabilistic relational operations, namely projection and join, in this context. The project operation is similar to ours. However, maximum entropy was applied to compute the resulting joint probability distribution for join operation.

Pittarelli [71] extended the probabilistic algebra defined in [14] to include some new operators such as select, threshold, and pooling operators. Given a probability distribution $B =$
The result of selection from \( B \) on a selection condition \( t \) is defined as \( B' = (V, \Delta, \text{dom}, P') \), where

\[
P' = \sigma_S(P)(t) = \begin{cases} 
0, & \text{if } t \notin S; \\
\frac{P(t)}{\sum_{t \in S} P(t)}, & \text{otherwise}.
\end{cases}
\]

\( \sigma_S(P) \) is undefined when \( \sum_{t \in S} P(t) = 0 \).

Their select operation is similar to our conditionalization operation. Their threshold operation is similar to their select operation except that the threshold condition is defined on the probability value. Their pooling operation combines estimates from different sources into a single distribution. A common approach is to use linear pooling, which computes a weighted average of different estimates. In our work on Knowledge-based Model Construction (KBMC), we are investigating techniques of data fusion which are similar to the idea of pooling.

Barbará, Garcia-Molina and Porter [5] presented a non-1NF probabilistic data model as an extension of the standard relational model. In their model, relations have deterministic keys, and tuples with different keys represent different real world entities. All the non-key attributes describe the properties of the entities and may be deterministic or stochastic, and independent or interdependent. Probabilities are associated with the values of stochastic attributes, and the interdependent relationship indicates that the attributes involved are jointly distributed ones. All other attributes are assumed to be independent. Figure 7.1 shows an example probabilistic relation. In this example, the variables \( \text{CS315} \) and \( \text{CS505} \) form a joint probability distribution. The interpretation is that the outcomes of random variables \( \text{CS315} \) and \( \text{CS505} \) depend on the Major under consideration. For example, the first row reads like

“The probability that a student has grades ‘A’ in both \( \text{CS315} \) and \( \text{CS505} \) given that the student is in CS major is 0.3; The probability that a student has grade ‘A’ in \( \text{MA522} \) given that the student is in CS major is 0.5.”

The non-stochastic attributes in their framework are analogous to context in our framework, while stochastic attributes are represented as participating random variables. Their framework requires that the probabilities for each attribute (or each set of joint attributes) in a single tuple must add up to 1. They noted, however, this requirement could be a problem because it is difficult to know the exact probabilities for all possible domain values. To solve this problem, their model allows the inclusion of missing probabilities to account for incomplete probability distributions, as shown in the last row of the sample relation.

Their probabilistic relational algebra extends the standard relational algebra operations such as select, project, and join. They also introduced a new set of operators to illustrate the various
Figure 7.1: An example probabilistic relation.

possibilities. For example, the *conditional* operator pulls a set of stochastic attributes into the key of the resulting relation since their framework allows only conditions specified on deterministic keys when expressing conditional probability distributions. The STOCHASTIC operator takes as input a deterministic relation (one where all attributes are deterministic) and returns a probabilistic relation according to a specified probabilistic schema. A standard relation, as shown in the left of Figure 7.2, can be transformed into a probabilistic relation, as shown in the right of Figure 7.2, by computing the probabilities for each distinct attribute (set of attributes) defined in the specified probabilistic schema, i.e., in this case STOCHASTIC(Major; CS315 CS505; MA522). The DISCRETE operator goes the other way around. It takes as input a probabilistic relation and returns a deterministic expected value relation.

Dey and Sarkar [25] provided a probabilistic database framework with relations abiding by first normal form (1NF). Unlike Barbará, et al. [5], they assigned probabilities to tuples, instead of individual attributes, in terms of a joint probability distribution. They required the sum of all probabilities associated with a key value to be no more than 1. A tuple in their model is analogous to a
single row of a probability table in ours, and their probabilistic relation might contain multiple probability distributions. Both [5] and [25] used point probabilities and assumed that all events/random variables in their models were independent. They provided a closed form query algebra and first introduced the conditionalization operation in the context of a probabilistic model. Later they proposed a non-procedural probabilistic query language called PSQL [26] as an extension of the SQL language.

Based on a first-order probabilistic logic language proposed by Halpern [48], Zimányi [90] formalized a relational model to represent probabilistic information. His data model is similar to the one in [25]. Zimányi also provided a complete method for evaluating queries in probabilistic theories. In order to evaluate queries, he introduced a special type of relation called trace relation (or t-relation). T-relation is a classical relation with an additional special field, called trace, containing for every tuple a formula that keeps track of how the tuple has been obtained. The concept of trace is similar to path in our SPO data model. Evaluation of queries are then performed by manipulating t-relations.

Lakshmanan, et al. introduced ProbView [58], a probabilistic database management system. In ProbView, probability distributions were interval, and the assumption of independence of events was replaced with the introduction of probabilistic conjunctions (and disjunctions), implementing different assumptions about the relationships between the events. Figure 7.3 shows an example of probabilistic relation in Probview. As in Barbará, Garcia-Molina and Porter’s work [5], their definition of probabilistic tuples allows any combination of deterministic and stochastic domains. They associate possible worlds with each probabilistic tuple. For example, there are eight worlds associated with the first tuple (i.e., for CS major), as shown in Figure 7.4.

<table>
<thead>
<tr>
<th>Major</th>
<th>GPA</th>
<th>CS315</th>
<th>CS505</th>
<th>MA522</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>3.3</td>
<td>{A, B}</td>
<td>{A, B}</td>
<td>{A, B}</td>
</tr>
<tr>
<td>h1(CS)=[1,1]</td>
<td>h2(3.3)=[1,1]</td>
<td>h3(A)=[0.3,0.6]</td>
<td>h4(A)=[0.4,0.6]</td>
<td>h5(A)=[0.5,0.7]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h3(B)=[0.4,0.7]</td>
<td>h4(B)=[0.4,0.6]</td>
<td>h5(B)=[0.3,0.5]</td>
</tr>
<tr>
<td>Math</td>
<td>3.1</td>
<td>{A, B}</td>
<td>{A, B}</td>
<td>{A, B}</td>
</tr>
<tr>
<td>h6(Math)=[1,1]</td>
<td>h7(3.1)=[1,1]</td>
<td>h8(A)=[0.2,0.5]</td>
<td>h9(A)=[0.3,0.6]</td>
<td>h10(A)=[0.6,0.8]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h8(B)=[0.5,0.8]</td>
<td>h9(B)=[0.4,0.7]</td>
<td>h10(B)=[0.2,0.4]</td>
</tr>
</tbody>
</table>

Figure 7.3: An example probabilistic relation.

The probability intervals associated with those worlds are computed using linear programming which can be constructed from the original probabilistic relation. They claimed that the linear program LP associated with a probabilistic relation can be constructed in time polynomial in the size of the relation, and that the size of the LP is also polynomially bounded in the size of the rela-
<table>
<thead>
<tr>
<th>Major</th>
<th>GPA</th>
<th>CS315</th>
<th>CS505</th>
<th>MA522</th>
</tr>
</thead>
<tbody>
<tr>
<td>w₁</td>
<td>CS</td>
<td>3.3</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>w₂</td>
<td>CS</td>
<td>3.3</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>w₃</td>
<td>CS</td>
<td>3.3</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>w₄</td>
<td>CS</td>
<td>3.3</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>w₅</td>
<td>CS</td>
<td>3.3</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>w₆</td>
<td>CS</td>
<td>3.3</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>w₇</td>
<td>CS</td>
<td>3.3</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>w₈</td>
<td>CS</td>
<td>3.3</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

Figure 7.4: Possible worlds for CS major.

...
All these approaches discussed above are extensions to either relational databases or object databases, with all the limitations inherent in each. The probabilistic object (e.g. as described in [28]) represents a single real world entity with uncertain attribute values. In our case, an SPO represents a probability distribution of one or more random variables. Our work combines and extends the ideas contained in these approaches and applies them to a semistructured data model, which provides us with the benefit of schema flexibility. This model overcomes some of the major problems associated with probabilistic relational models. For instance, our model provides additional contextual information, specifying general information for the probability distribution and conditional information, making it possible to represent conditional probability distributions.

Two approaches to semistructured probabilistic data management are closely related to ours: the ProTDB [69] and the PIXml [53] frameworks. In ProTDB [69], Nierman and Jagadish extended the XML data model by associating a probability to each element with modification of regular non-probabilistic DTDs. Their data model is based on the standard XML data model. They provided two ways of modifying non-probabilistic DTDs: One is to introduce to every element a probability attribute $\text{Prob}$ to specify the probability of the particular element existing at the specific location of the XML document, and the other is to attach a new sub-element called $\text{Dist}$ to each element, which makes it possible to represent probability distributions. They defined two special elements $\text{Dist}$ and $\text{Val}$, as shown in Figure 7.6.

The ProTDB data model aims to represent uncertainty in the structure of XML documents, instead of the probabilistic information itself. The ProTDB framework lacks a query facility which allows one to efficiently manipulate the stored probabilistic information. Generally speaking, a
ProTDB instance is a standard XML document with some probabilistic attributes and/or elements. Any XML document, including XML representation of an SPO, could be transformed to a ProTDB instance. Figure 7.7 shows an example of transforming an SPO into a ProTDB instance. One of the drawbacks of their model is that probabilities in an ancestor-descendant chain were related probabilistically, and all other probabilities are assumed to be independent.

In contrast, our data model focuses on the representation and manipulation of the probabilistic information, namely, the probability distributions. Our approach defines probability distributions over a set of random variables along with additional context information, providing general information for the probability distribution, and conditional information, making it possible to express conditional probability distributions. The SPO data model uses a more rigid structure than ProTDB in order to represent complex, probabilistic information. It uses our knowledge of the structure of stored data: Our SPOs are compatible with some XML schema templates. We also provide a comprehensive SP-Algebra to efficiently query the semistructured probabilistic database.

Figure 7.7: An example of transforming an SPO into a ProTDB instance.
In PIXml [53], Hung, et al. proposed a probabilistic interval XML (PIXml) data model with two types of semantics for uncertain data. The global interpretation is a distribution over an entire XML document, while the local interpretation specifies an object probability function for each non-leaf object (it has no child). Given a set of objects $O$, a set of labels $L$, and a set of types $T$, their model defines a probabilistic instance as a 6-tuple $(V, lch, \tau, val, card, ipf)$, where

- $V \subseteq O$.
- For each object $o \in V$ and each label $l \in L$, $lch(o, l)$ specifies the set of objects that are children of $o$ with label $l$.
- $\tau$ associates a type in $T$ with each leaf object $o$.
- $val$ associates a value in $dom(\tau(o))$ with each leaf object $o$.
- $card$ constrains the number of children with a label $l \in L$, and it is an integer-valued function. $card(o, l) = [min, max]$, where $max \geq min \geq 0$.
- $ipf$ associates an interval probability function with each non-leaf object $o$. For each object $c$ in the set of potential children of $o$, they define $ipf(o, c) = [lb, ub]$.

As you see here, they associates probability interval with each (parent object, potential child) pair. The PIXml data model uses relatively more flexible structure than ours and could represent almost any form of structured or non-structured data, while ESPO data model is specially designed to represent the conditional probability distribution over a set of random variables. They also proposed a path expression-based query language to access stored information. This approach overcomes some drawbacks presented in Jagadish and Nierman’s approach[69].

Instances of ESPO and PIXml can be transformed with each other. Strictly speaking, we can not represent an ESPO in an PIXml instance because there is no way we can represent the association (i.e., attributes ID & IDREF) between extended context/conditional and probability table within the probabilistic interval XML framework. However, with minor changes in the definition of semistructured instance in the PIXml framework (i.e., add ID and IDREF into the set of types $\tau$ and associate $\tau$ with all objects instead of just leaf objects), we could represent an ESPO in a PIXml instance, as shown in Figure 7.8.

We could also represent a PIXml instance into a set of ESPOs, with each ESPO representing a “unit of information” in the PIXml instance. Here we assume the “unit of information” in a PIXml instance is a node along with its directly connected children. We could represent this unit information in a single ESPO, except that some redundant information for leaf nodes may be discarded or
put in <context>. Figure 7.9 shows an example of transforming “unit of information” of a PIXml instance into an ESPO.

The major difference between PIXml and our work is that PIXml [53] is concerned with representation of uncertainty in the structure of XML documents, while our ESPO is concerned more with probabilistic data itself. At the same time, the ESPO model provides a full set of ESP-Algebra for efficiently storing and manipulating probability distributions found in many vital applications.
Hung, et al. [53] use our conditionalization formulae for their computations of conditional probabilities. This makes the two approaches comparable: Our ESPO objects can be represented as their probabilistic interval XML. At the same time, we can represent their probabilistic XML documents as a set of ESPOs or simply a joint probability distributions, and thus embed them into the ESPO model. At the same time, while ESPOs are representable in XML, our definitions of the data model and ESP-Algebra do not rely on a specific representation.

The term “Probabilistic Relational Models (PRMs)” was also used in a different context [36]. PRMs represent a language for specifying statistical models among different relational domains. A PRM models uncertainty over relationships between attributes in the same object and uncertainty over relationships among different objects. It is quite different from what we discuss here.

7.2 Interval Probabilities

Imprecise probabilities have attracted the attention of researchers for quite a while now, as documented by the Imprecise Probability Project [18]. Walley’s seminal work [84] made the case for interval probabilities as the means of representing uncertainty. In his book, Walley talks about the computation of conditional probabilities of events. His semantics is quite different from ours, as Walley constructs his theory of imprecise probabilities based on gambles and betting, expressed as lower and upper previsions on the sets of events. Conditional probabilities are also specified via gambles by means of conditional previsions. A similar approach to Walley’s is found in the work of Biazzo, Gilio, et al. [7, 6] where they extend the theory of imprecise probabilities to incorporate logical inference and default reasoning.

Walley [84] calls consistency and tightness properties “avoiding sure loss”, and “coherence”, respectively. Biazzo and Gilio [6] also use the term “g-coherence” as a synonym for “avoiding sure loss”. The terminology that we have adopted originated from the work of Dekhtyar, Ross and Subrahmanian on a specialized semantics for probability distributions used in their Temporal Probabilistic Database model [22]. The possible world semantics for interval probabilities also occurs in Givan, Leach and Dean’s discussion of Bounded Parameter Markov Decision Processes [42].

De Campos, Huete and Moral [17] studied probability intervals as a tool to represent uncertain information. They gave definitions for consistency and tightness, which they called reachability. They developed a calculus for probability intervals, including combination, marginalization, and conditioning. They also explored the relationship of their formalism with other uncertain theories, such as lower and upper probabilities. When they defined their conditioning operation, however,
they switched back and applied lower and upper probabilities to uncertain information instead of probability intervals, and gave a definition of a conditioning operation for bidimensional probability intervals. Our semantics follows the one in [17], with different notation.

A more direct approach to introducing interval probabilities is found in the work of Weichselberger [85], who extended the Kolmogorov axioms of probability theory to the case of interval probabilities. Building on Kolmogorov probability theory, the interval probability semantics was defined for a $\sigma$-algebra of random events. Weichselberger defined two types of interval probability distributions over this $\sigma$-algebra: $R$-Probabilities, similar to our consistent interval probability distribution functions and $F$-Probabilities, similar to our tight interval probability distribution functions. In his semantics, an event is specified as a Boolean combination of atomic events from some set $\Omega$. Each event partitions the set of possible worlds into two sets: those in which the event has occurred and those in which it has not. A lower bound on the probability that an event has occurred is immediately an upper bound on the probability that it has not occurred. Thus, for $F$-probabilities, Weichselberger’s analogs of our tight p-interpretations, lower bounds uniquely determine upper bounds.

Weichselberger completed his theory with two definitions of conditional probability: “intuitive” and “formal”. His “intuitive” definition semantically matches our Definition 5.21. On the other hand, the “formal” definition specifies the probability interval for $P(A|B)$ as $[\frac{\min(P(AB))}{\max(P(B))}, \frac{\max(P(AB))}{\min(P(B))}]$, which is somewhat different from our Theorem 5.3. There, to determine the lower bound we minimized the numerator and try to maximize the denominator. Similarly, for the upper bound, we maximized the numerator and try to minimize the denominator.

In our semantics, atomic events have the form “random variable $X_1$ takes value $a_1$ and random variable $X_2$ takes value $a_2$ and . . . and random variable $X_m$ takes value $a_m$.” The negation of such an event is the disjunction of all other atomic events that complete the joint probability distribution of random variables $X_1, \ldots, X_m$. Our interval probability distribution functions specified only the probability intervals for such atomic events, without explicitly propagating them onto the negations. This means that even for tight interval probability distribution functions, both upper and lower bounds are necessary in all but marginal cases, as illustrated in Figure 7.10.

Interval probability distributions of discrete random variables generate a set of linear constraints on the acceptable probability values for individual instances. This set of linear constraints, however, is quite simple. It consists of constraints specifying that the probabilities of individual instances must fall between the given lower and upper bounds, and a constraint that specifies that the sum of all probabilities must be equal to 1. It is possible, however, to study more complex collections of constraints on possible worlds. Significant work in this area has been done by Cano
Figure 7.10: Lower bounds do not uniquely define upper bounds for tight interval probability distribution functions.

and Moral [13].

The conditionalization operation as a database operation was first introduced by Dey and Sarkar [25] in a probabilistic database model, and then by Dekhtyar, Goldsmith and Hawkes in a Semistructured Probabilistic database model [20]. However, conditionalization operation for probability intervals was not included in any data models until in our ESPO database framework [44, 88]. We note, however, that Jaffray [54] has shown that interval conditional probability estimates will not be perfect, and that the unfortunate consequence of this is that conditionalizing is not commutative: \( P((A|B)|C) \neq P(A|(B|C)) \) for many \( A, B, \) and \( C \). But conditionalization is still useful in many situations. To our knowledge, our algorithm computes the best possible results one can obtain for conditionalization operation. Thus, a conditionalization operation is included into ESP-Algebra with the caveat that the user must take care in the use of and interpretation of the result.

### 7.3 XML Storage Techniques

XML is becoming the \textit{de facto} standard data format for electronic data exchange through the Internet. The advantage of using XML as information media is well-known since XML is self-descriptive and platform-independent. Efficient storage and fast retrieval of XML documents has been studied extensively in the last several years [3]. Three major approaches have been proposed to store XML documents, namely, native XML databases [2, 63, 82], object-oriented databases [81] and relational databases [24, 32]. Native XML databases\(^1\) are specialized for the purpose of storing and querying XML documents. This approach, however, does not utilize the sophisticated storage and query capacity provided by existing relational database systems, and query processors for native XML data are still immature. Object-oriented database approaches suffer the same problem, and query processors of object-oriented databases are themselves still not mature yet. There are many benefits to storing XML in a relational database. As a dominant form of storage for a few decades,

\[^1\]According to the definition given in [78], a native XML database is not required to have any particular underlying physical storage model, and it can be built on a relational, object-oriented, or other proprietary storage formats.
relational databases are a widely-accepted and mature technology. Therefore, we will focus our discussion on managing XML with relational databases.

Relational databases are well suited for storage and maintenance of data-centric XML documents, but the table-based flat data model of RDBMS does not fit the hierarchical and inter-related nature of XML objects. A RDBMS needs to break XML data into inter-related tables, and a query over XML documents would result in many relational queries such as join operations. This considerably degrades the performance of relational databases. Approaches have been proposed for reducing the effects of such undesirable consequences [77, 81].

The typical procedure for storing XML in relational databases is described as follows [76]. First, based on a predefined relational schema generation technique relational tables are created for the purpose of storing XML documents. Second, XML documents are decomposed and stored as rows in the tables created in the first step. Structural relationships in XML documents are captured via values such as key/foreign key relationships and order/level encodings. Third, queries over XML documents are translated into a sequence of SQL statements over the relational tables. Finally, tuple streams returned from the relational database are postprocessed if necessary and tagged to generate the final result(s) in XML format.

The basic idea behind these approaches is that XML documents are considered as directed, ordered, and labeled graphs. So there are different ways to flatten graphs (XML documents) into tuples. Algorithms that describe the mapping of XML documents into relational tables have been proposed [35, 77, 81]. These algorithms provide different methods to map elements (attributes and values) in XML documents into rows in relational model, which are sometimes called relational schema generation techniques. However, appropriate relational schemas may depend on many factors, among which are the nature of XML data, the maximum query workload and the availability of DTD/XML schema. None of the above storage models can meet all the requirements at once with respect to query response time and/or scalability [81].

Florescu and Kossmann [34] reported a performance evaluation of different mapping schemes for storing XML data in relational databases. They outlined different mapping schemes by the rules of how to represent attributes and values in XML documents. They presented the results of comprehensive performance experiments that analyzed alternative mapping schemes with respect to database size or query performance. Their results indicated that attribute approach is the best among all alternatives to map attributes, and inlining outperforms separate value tables.

In parallel, Shanmugasundaram, et al. [77] explored techniques for storing XML documents, including simplifying DTDs, special schema conversion techniques, and inlining technique. They also proposed methods to convert semistructured queries to SQL queries. Their results showed that
it is possible to handle most queries on XML data using relational databases, except for certain types of queries with complex recursion.

Tian, et al. [81] extended the work of [35] and [77]. They studied several different strategies for storing XML documents by evaluating a predefined set of XQuery queries. They demonstrated that three forms of clustering are desirable when storing XML documents: clustering elements of the same real world object, clustering elements based on type, and clustering elements in the same order as in the original documents. Their results indicated that DTD information is crucial in order to achieve compact data representation and subsequently good query performance. The relational/DTD approach outperforms other approaches on average. However, there are situations where it is difficult to define XML data by a fixed schema or a DTD [57].

One of the drawbacks of schema generation techniques described above is that each relation schema generation technique requires its own query processor to translate queries over XML documents into a sequence of SQL statements over the generated relational schema. Shanmugasundaram, et al. [76] proposed a general technique for querying XML documents using a relational database system. Their approach makes it possible for the same query processor to be used with most relational schema generation techniques. An XML view is created, based on the stored relational tables, to reconstruct the XML documents, and queries over the original XML documents are then treated as queries over the reconstructed XML view. A drawback is that a specialized query processor may be more efficient than this general technique when translating from XML queries to SQL statements. Manolescu, et al. [62] describe a similar work in the context of data integration.

```xml
<sop path = "S1" id = "1">
  <context id = "2">
    <ele id = "5"><name>major</name> <val>CS</val></ele>
    <ele id = "6"><name>instructor</name> <val>Dekhtyar</val></ele>
  </context>
  <table id = "3">
    <variable id = "7">
      <name>CS505</name> <name>CS570</name>
    </variable>
    <row id = "8"> <val>A</val> <val>A</val> <p>0.3</p> </row>
    <row id = "9"> <val>A</val> <val>B</val> <p>0.2</p> </row>
    <row id = "10"> <val>B</val> <val>A</val> <p>0.3</p> </row>
    <row id = "11"> <val>B</val> <val>B</val> <p>0.2</p> </row>
  </table>
  <conditional id = "4">
    <ele id = "12"><name>CS315</name> <val>A</val></ele>
  </conditional>
</sop>
```

Figure 7.11: A sample SPO, with ids added for non-leaf nodes.
To summarize different techniques for storing and managing XML documents in relational databases, there are two broadly-used approaches: the edge approach [57, 35, 75, 81] and the attribute approach [35, 81]. Now suppose that we have a sample SPO, as described in Figure 7.11. When the edge approach is applied, all SPOs, including the sample SPO, can be stored in a single edge table. Each tuple in the edge table corresponds to a single edge in a directed graph (represent-
ing an XML document). The edge table contains the following fields.

- **SourceID** stores the id of the parent node of an edge;
- **Target** stores the id of the child node of an edge (or its text value if the child node has only one single text child);
- **Name** stores the name of the child node;
- **Ordinal** stores the ordinal position of the child node within the children of the parent node; and
- **Type** stores the node type of the child node, which tells whether it is an attribute or element.

The final result for the sample SPO is shown in Figure 7.12. The attribute approach uses a similar relational schema to store XML documents, but it partitions the entire table based on the element name. So the field **name** can be omitted, since every tuple in the same table will have the same value. Different elements are stored in separate tables. Figure 7.13 shows the final results for the attribute approach with the single-row tables omitted. As you see here, storing SPOs with either the edge approach or the attribute approach is perfectly fine. However, retrieving SPOs will be involving large number of **join** operations, which definitely slows down query processing.

For example, if we want to issue a query which retrieves all the SPOs regarding students in Computer Science major, then we need to find all SPOs which have **context** attribute **major** of value ‘CS’. Assume the SP-relation storing all the courses is called $S$, then the SP-Algebra will be $\sigma_{\text{con}}[\text{major} = \text{‘CS’}](S)$. Here we list the SQL statement for finding all SID. The corresponding SQL statements for the edge approach and the attribute approach are shown in Figure 7.14 and Figure 7.15, respectively. Both of them require joining four relations. In contrast, our approach does not require any **join** operations in order to find all SIDs in the database. The SQL statement for SPDBMS is the following.

```
"SELECT con.SID FROM SPO_CONS con WHERE con.elemname='major' AND con.elemvalue='CS' AND con.type=‘cont’"
```

Our approach is especially favorable when operations require manipulation of probability distributions in addition to the retrieval of the stored information (i.e., conditionalization), since our approach stores actual probability tables in their entirety.

Our approach combines and extends both the edge approach and the attribute approach. These approaches use a relational schema generation technique to store arbitrary XML documents, and do
not assume any knowledge of the structure of the stored documents. We begin with the assumption that documents stored in our system conform to a schema template (see Appendix B), which ensures the probability information is stored correctly. Our approach exploits the nature of the known DTD/XML schema and employs compact storage and query translation for our XML data. In this
SELECT e1.SID
FROM edge e1, edge e2, edge e3, edge e4
WHERE e1.SID=e2.SID AND e2.SID=e3.SID AND e3.SID=e4.SID AND
e1.Target=e2.SourceID AND
  e2.Target=e3.SourceID AND
  e3.SourceID=e4.SourceID AND
  e1.Name='context' AND
  e2.Name='elem' AND
  e3.Name='name' AND
  e4.Name='val' AND
  e3.Target='major" AND
  e4.Target='CS'

Figure 7.14: The SQL statement for finding all SPO SIDs in edge approach.

SELECT c.SID
FROM context c, elem e, name n, val v
WHERE c.SID=e.SID AND e.SID=n.SID AND n.SID=v.SID AND
c.Target=e.SourceID AND
  e.Target=n.SourceID AND
  n.SourceID=v.SourceID AND
  n.Target='major' AND
  v.Target='CS'

Figure 7.15: The SQL statement for finding all SPO SIDs in attribute approach.

sense, our approach is also similar to the relational-DTD approach described in [76]. Our approach utilizes an internal object model to represent intermediate XML objects for efficient retrieval. Our query processor distinguishes computations executed by itself or by the underlying relational database, and pushes down as much query computation as possible to the relational database engine, which is similar to the approach in [33]. More important, the approaches described in this subsection focus on storing and querying general XML data. In particular, they do not deal with storing and querying probabilistic data, while our approach handles probabilities according to the laws of probability theory.

\[2\] Although less generic, this lightweight object model provides a much more efficient way to answer queries than the DOM representation.
Chapter 8

Conclusions and Future Work

We presented a semistructured probabilistic database framework by extending the initial SPO data model proposed by Dekhtyar, Goldsmith and Hawkes in [20] for storing and managing probabilistic information. Accordingly, we extended the Semistructured Probabilistic Algebra (SP-Algebra). We added a path expression in the SPO data model to specify the origin of a particular object. We defined a new join operation according to the laws of probability theory, replacing the one in the initial framework. The SPO data model has been defined flexibly enough to represent probabilistic distributions over arbitrary sets of random variables, along with additional information associated with them. We also developed a set of equivalence relations and cost models for query optimization framework. Our query algebra allows the user to retrieve data based on probabilities, variables, or associated information, and to transform the probability distributions according to the laws of probability theory. We also reported our work on query optimization for the database framework.

Based on the new SPO framework, we designed and implemented a Semistructured Probabilistic Database Management System (SPDBMS) on top of a relational DBMS. The XML encoding of SPOs makes it easier for data interchange between diverse applications. A storage strategy and a translation mechanism from SP-algebra queries into sequences of relational SQL statements were developed. We reported a performance evaluation of the SPDBMS for each type of SP-Algebra query, which indicates that the SPDBMS performs well in terms of response time and scalability even without query optimization.

We developed an extension of the standard Semistructured Probabilistic Object (SPO) data model to improve the flexibility of the original semistructured data model. The Extended Semistructured Probabilistic Object (ESPO) data model extends the flexibility of SPOs with two new features: (i) support for interval probabilities and (ii) association of context and conditionals with individual random variables.

The Semistructured Probabilistic Database may be used as a backend to any applications that involve the management of large quantities of probabilistic information, such as building stochastic models. We designed and implemented a system, the Bayesian Network Development Suite (BaNDeS), which builds Bayesian networks with full data management support of the SPDBMS. This development tool allows us to build, update and maintain large Bayesian network models in the distributed environment.
The following are major foci of our ongoing and future work:

- We have developed the query optimization framework for the SPDBMS, including a set of rewrite rules and an initial set of cost model for all SP-Algebra queries. We are planning to implement a query optimizer and evaluate the performance of query optimization. The current version of SPDBMS does not include a query optimizer. During query processing, an SP-Algebra query is parsed into an algebra parse tree, converted into a sequence of SQL statements (without query optimization) which are executed directly by the underlying relational query engine. The returned tuple streams are then transformed into an instance of Internal Object Model (IOM). The IOM instance represents the intermediate XML objects and further operation such as post-processing and assembly may be applied on it. As multiple equivalent query trees can be generated from a single SP-algebra query, the final query tree should be carefully chosen to reflect the query optimization strategy adopted by the query engine (i.e., rewrite rules and cost model). As part of the query optimization, the query tree may be simplified by a set of heuristic rules to reduce its processing cost. With the query optimizer implemented, the database system should be able to choose the best (nearly best) query tree and subsequently improve the query execution efficiency. An evaluation of the query optimization may be performed afterwards.

- We are extending the SPDBMS to handle interval probability distributions. The SPO framework assumes for simplicity that all probabilities contained in the SPOs are point probabilities, i.e. the probability space $\mathcal{P} = [0, \infty]$. So the SPDBMS implemented so far deals with only point probability distributions. In order for the SPDBMS to correctly handle interval probability distributions according to the laws of probability theory, we need to extend the SPDBMS so that it complies with the ESPO data model and its associated ESP-algebra. The new system will apply a new data storage strategy to reflect the new data model. It also requires new implementation for each ESP-algebra operation by extending the current implementation for SPDBMS.

- The current implementation of our Bayesian Network Development Suite (BaNDeS) allows us to build Bayesian networks from expert knowledge. While this approach is good for some applications such as W2W program, other applications such as those in bioinformatics domain require to build Bayesian networks directly from massive biological data, i.e., microarray data. So we plan to incorporate machine learning algorithms into the current implementation of our BaNDeS to allow one to build Bayesian network models directly from data.
Appendix A  Conversion from SP-Algebra into SQLs

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Get the SPO ID list based on given probability value:</td>
<td>SELECT DISTINCT id FROM rid_SPO_CON WHERE elemname = 'major' AND elemvalue = 'CS'</td>
</tr>
<tr>
<td>2.</td>
<td>Get the variable list for each SPO in the ID list:</td>
<td>SELECT id, varname, position FROM rid_SPO_VAR WHERE id IN (ID list)</td>
</tr>
<tr>
<td>3.</td>
<td>Retrieve context/conditional:</td>
<td>SELECT id, elemname, elemvalue, idref FROM rid_SPO_CONS WHERE id IN (ID list)</td>
</tr>
<tr>
<td>4.</td>
<td>Retrieve probability table</td>
<td>SELECT * FROM rid_SPO_1 WHERE id IN (ID list) UNION ALL SELECT * FROM rid_SPO_i WHERE id IN (ID list) UNION ALL SELECT * FROM rid_SPO_max WHERE id IN (ID list)</td>
</tr>
</tbody>
</table>

Figure A.1: Steps for selection on context, i.e., $\sigma_{\text{cnt.major}='CS'}(S)$. 

162
step 1. Get the SPO ID list based on given probability value:
SELECT DISTINCT id
FROM rid_SPO_CON
WHERE elemname = 'AI'
    AND elemvalue = 'B'

step 2. Get the variable list for each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}

step 4. Retrieve probability table
SELECT *
FROM rid_SPO_1
WHERE id IN {ID list}
...
UNION ALL
SELECT *
FROM rid_SPO_i
WHERE id IN {ID list}
...
UNION ALL
SELECT *
FROM rid_SPO_max
WHERE id IN {ID list}

Figure A.2: Steps for selection on conditional, i.e., $\sigma_{cnd.AI='B'}(S)$. 

step 1. Get the SPO ID list based on given probability value:
SELECT DISTINCT id
FROM rid_SPO_VAR
WHERE varname = 'DB'

step 2. Get the variable list for each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}

step 4. Retrieve probability table
SELECT *
FROM rid_SPO_1
WHERE id IN {ID list}
... UNIION ALL
SELECT *
FROM rid_SPO_i
WHERE id IN {ID list}
... UNIION ALL
SELECT *
FROM rid_SPO_max
WHERE id IN {ID list}

Figure A.3: Steps for selection on participating variable, i.e., \( \sigma_{var.DB \in \mathcal{V}(S)} \).
step 1. Get the SPO ID list based on given probability value:
SELECT DISTINCT id
FROM rid_SPO_VAR
WHERE varname = 'DB'

step 2. Get the variable list for each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}

step 4. Retrieve probability table
SELECT *
FROM rid_SPO_1
WHERE id IN {ID list}
    AND var_position = 'A'
... 
UNION ALL
SELECT *
FROM rid_SPO_i
WHERE id IN {ID list}
    AND var_position = 'A'
... 
UNION ALL
SELECT *
FROM rid_SPO_max
WHERE id IN {ID list}
    AND var_position = 'A'

Figure A.4: Steps for selection on probability table, i.e., \(\sigma_{tbl\_DB='A'}(S)\).
step 1. Get the SPO ID list based on given probability value:
SELECT DISTINCT id
FROM rid_SPO_1
WHERE p > 0.1
...
UNION ALL
SELECT DISTINCT id
FROM rid_SPO_i
WHERE p > 0.1
...
UNION ALL
SELECT DISTINCT id
FROM rid_SPO_max
WHERE p > 0.1

step 2. Get the variable list for each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}

step 4. Retrieve probability table
SELECT *
FROM rid_SPO_1
WHERE id IN {ID list}
   AND p > 0.1
...
UNION ALL
SELECT *
FROM rid_SPO_i
WHERE id IN {ID list}
   AND p > 0.1
...
UNION ALL
SELECT *
FROM rid_SPO_max
WHERE id IN {ID list}
   AND p > 0.1

Figure A.5: Steps for selection on context, i.e., $\sigma_{p > 0.1}(S)$. 

166
step 1. Get the SPO ID list based on given probability value:
SELECT DISTINCT id
FROM rid_SPO_CON
WHERE elemname = 'major'

step 2. Get the variable list for each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}
   AND elemname = 'major'

step 4. Retrieve probability table
SELECT *
FROM rid_SPO_1
WHERE id IN {ID list}
...
UNION ALL
SELECT *
FROM rid_SPO_i
WHERE id IN {ID list}
...
UNION ALL
SELECT *
FROM rid_SPO_max
WHERE id IN {ID list}

Figure A.6: Steps for projection on context, i.e., $\pi_{\text{cnt, major}}(S)$. 
step 1. Get the SPO ID list based on given probability value:
SELECT DISTINCT id
FROM rid_SPO_CON
WHERE elemname = 'AI'

step 2. Get the variable list for each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}
    AND elemname = 'AI'

step 4. Retrieve probability table
SELECT *
FROM rid_SPO_1
WHERE id IN {ID list}
...
UNION ALL
SELECT *
FROM rid_SPO_i
WHERE id IN {ID list}
...
UNION ALL
SELECT *
FROM rid_SPO_max
WHERE id IN {ID list}

Figure A.7: Steps for projection on conditional, i.e., \( \pi_{end \ AI}(S) \).
step 1. Get the SPO ID list based on given probability value:
SELECT DISTINCT id
FROM rid_SPO_VAR
WHERE varname = 'DB'

step 2. Get the variable list for each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}

step 4. Retrieve probability table
SELECT var_position
FROM rid_SPO_1
WHERE id IN {ID list}
...
UNION ALL
SELECT var_position
FROM rid_SPO_i
WHERE id IN {ID list}
...
UNION ALL
SELECT var_position
FROM rid_SPO_max
WHERE id IN {ID list}

Figure A.8: Steps for projection on participating variable, i.e., $\pi_{var,DB}(S)$. 
step 1. Get the SPO ID list based on the variable name(s):
SELECT DISTINCT id
FROM rid_SPO_VAR
WHERE varname = 'DB'

step 2. Get the variable list for each SPO in the ID list:
SELECT id, varname, position
FROM rid_SPO_VAR
WHERE id IN {ID list}

step 3. Retrieve context/conditional:
SELECT id, elemname, elemvalue, idref
FROM rid_SPO_CONS
WHERE id IN {ID list}

step 4. Retrieve probability table
SELECT id, {var_i list}, P
FROM rid_SPO_i
WHERE id IN {ID list}
    AND var_position = 'A'
...
UNION ALL
SELECT id, {var_j list}, P
FROM rid_SPO_i
WHERE id IN {ID list}
    AND var_position = 'A'
...
UNION ALL
SELECT id, {var_k list}, P
FROM rid_SPO_max
WHERE id IN {ID list}
    AND var_position = 'A'

Figure A.9: Steps for selection on context, i.e., $\mu_{\text{tbl}, \text{DB}='A'}(S)$. 

170
Appendix B  XML Schema for SPOs

<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
    targetNamespace="http://dblab.csr.uky.edu/~wzhao0/schema/spo"
    elementFormDefault="qualified"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:spo="http://dblab.csr.uky.edu/~wzhao0/schema/spo">

<xsd:simpleType name="nameType">
    <xsd:restriction base = "xsd:string">
        <xsd:pattern value="[A-Z|0-9]+"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="valType">
    <xsd:restriction base = "xsd:string">
        <xsd:pattern value="[A-Z|0-9]+"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="eleType">
    <xsd:sequence>
        <xsd:element name="name" type="spo:nameType"/>
        <xsd:element name="val" maxOccurs="unbounded" type="spo:valType"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="contextType">
    <xsd:sequence>
        <xsd:element name="ele" minOccurs="0" maxOccurs="unbounded" type="spo:eleType"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="variableType">
    <xsd:sequence>
        <xsd:element name="name" maxOccurs="unbounded" type="spo:nameType"/>
    </xsd:sequence>
</xsd:complexType>

</xsd:schema>

Figure B.1: XML schema template for SPOs.
Figure B.2: XML schema template for SPOs (cont.).
References


Vita

**Date and Place of Birth:** July 2nd, 1968, Jiangsu, China

**Education:**
- Master of Science in Chemical Engineering, special subject: Polymer Processing
  - East China University of Science and Technology, 1995
- Bachelor of Science in Chemical Engineering
  - East China University of Science and Technology, 1990

**Professional Positions:**
- Assistant Professor
  - East China University of Science and Technology, Shanghai, China, 1995–1998
- Chemical Engineer
  - Shanghai Chloro-Alkali Chemical Company, Ltd., Shanghai, China, 1990–1992

**Honors:**
- Research Assistantship
  - University of Kentucky, 2001–2004
- Research Assistantship
  - University of Kentucky, 1999–2000
- Graduate Teaching Assistantship
  - University of Kentucky, 1998–1999
Professional Publications:


Wenzhong Zhao