CONTEXT-SENSITIVE NETWORK: A PROBABILISTIC CONTEXT LANGUAGE FOR ADAPTIVE REASONING

ROHIT JOSHI

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2009

Acknowledgements

"The hardest arithmetic to master is that which enables us to count the things that most deserve our gratitude" – Eric Hoffer (modified)

Though it will not be enough to express my gratitude in words, I would still like to give my many, many thanks to all those people who made my stay at NUS memorable.

First and foremost, I would like to express my sincere thanks to my thesis supervisor and mentor, Dr Leong Tze Yun, for her guidance, patience and immense support throughout my graduate career. I have learned a lot from her; without her help and trust in me I could not have finished my dissertation successfully.

A very special thanks goes out to Professor Poh Kim Leng, for being an excellent teacher and for his inputs and interest in this research; and to our groups' collaborators especially Dr Lim Tow Keang, Dr Lee Kang Hoe and Dr Heng Chew Kiat. The Pneumonia case study would not have been possible without the expert guidance of Dr Lim and Dr Lee. Dr Heng provided the valuable Heart disease data set for this research.

I am also much indebted to Professor Tham Chen Khong, my undergraduate thesis supervisor, and Professor Liyanage De Silva, my undergraduate mentor. Their advice and encouragement gave me the confidence to pursue the Ph.D. degree.

I would like to thank several of my professors for their support and encouragement over the years especially Professor Lee Wee Sun, for his technical insights to the problems discussed in the graphical models reading group; Professor David Hsu, for his guidance

in precise and effective presentation of technical ideas; Professor Winnie Hsu, for accepting me as her Teaching Assistant; Professor Leslie Kaelbling and Professor Anthony Tung for teaching me about Artificial Intelligence and Data Mining.

I have been blessed with a friendly lab environment and cheerful group of fellow students. Many thanks go to: Chen Qiongyu, my next seat lab mate, for her help, patience and support in all the occasions; Li Guoliang, for being a good friend and wonderful colleague who was always available for any technical discussions; Yin Hongli, for relieving me of system administration responsibilities; other past and present BIDE group members including Zeng Yifeng, Sreeram Ramachandaran, Xu SongSong, Dinh Thien Anh, Truong Huy, Ong Chen Hui and Zhu Ailing, for bringing enthusiasm and fun both inside and outside the lab. I would like to especially thank Dr Li Xiaoli and Dr Vellaisamy Kuralmani for their advice and encouragement; and Ms. Gwee Siew Ing, an efficient administrative officer, for her support in financial processes.

My humble gratitude to all my friends over the years at NUS who have influenced me greatly, especially Gopalan Sai Srinivas, Tushar Radke, Hari Gurung, Harshad Jahagirdhar, Ranjan Jha, Raymond Anthony Samalo, Ashwin Sanketh and Ayon Chakrabarty. I would also like to thank Dr Namrata Sethi for her editing assistance.

I must acknowledge my parents for their unbounded love; my sister and sister-in-law for their encouragement; and my wife, Prachi, for her belief in me, for standing by me in good and bad times, and for the much needed motivation especially during the last phase of my Ph.D. work.

This research has been supported by research scholarship from NUS and Research Grants No. R-252-000-111-112/303 and R-252-000-257-298 under which I was employed as a Research Assistant.

Contents

Summary

This thesis considers the problem of capturing situational variations as *contexts* to model information that holds in specific conditions under uncertainty.

We introduce a new asymmetric probabilistic graphical language, Context-Sensitive Network, that extends Bayesian network to domains where the variables (or nodes) and their relations (or edges) are functions of the contexts. CSN aims to support scalable and flexible structural adaptation with varying context atttributes and values, while exploiting the graphical properties of an asymmetric representation. A CSN is a directed bipartite graph that represents the product of Conditional Part Factors (CPFs), a new internal representation for a partition of a conditional probability table (CPT) in a specific context. By properly partitioning the CPT of a target variable in a context-dependent manner, we can exploit both local parameter decomposition and graphical structure decomposition. A CSN also forms the basis of a local context modeling scheme that facilitates knowledge acquisition.

We describe the theoretical foundations and the practical considerations of the representation, inference, and learning supported by, as well as an empirical evaluation of the proposed language. We demonstrate that multiple, generic contexts, such as those related to the 5 "W"s of a situation - who, what, where, which, and when - can be directly incorporated and integrated; the resulting context-specific graphs are much simpler and more efficient to manipulate for inference and learning. Our representation is particularly useful when there are a large number of relevant context attributes, when the context attributes may vary in different conditions, and when all the context values or evidence may not be known *a priori*.

We also evaluate the effectiveness of CSN with two case studies involving actual clinical situations and demonstrate that CSN is expressive enough to handle a wide range of problems involving context in real-life applications.

List of Tables

List of Figures

Notation as a tool of thought

K.E.Iverson, Turing awards lecture theme

1

Overview: An Executive Summary

Many software applications and systems are not situation-aware, i.e., they provide results or make decisions in general, without considering the user's personal, social, and cultural contexts. For example, a generic restaurant recommendation system is highly unlikely to consider the weather, the location, or the company of a user before returning a list of restaurants. A major difficulty is in accurate representation and maintenance of a large collection of possible contextual profiles to cater to each specific situation. A strategy to overcome this problem is to ask the user to explicitly state his/her context or profile, for example his age, gender, location or special preferences. A situation-specific model is then instantiated for answering a query that is well-tailored to the user's situation-specific requirements. Situation-specific representations usually lead to smaller models and faster inference; these in turn would improve the effectiveness and quality of service of the target applications.

Unlike computers, humans do not always need contextual information to be stated

explicitly; we can adapt to any situational variations and hence reason much more effectively and accurately. An important question is, therefore: Can situation-specific representations be extended to consider the uncertainty over any generic contexts such as the 5 "W"s of a situation - who, what, where, which, and when? If that is possible, how can we capture the situational variations succinctly? Do we need to know all the possible situational variations beforehand? This thesis addresses the problem of capturing situational variations as contexts and investigates the theoretical issues and practical challenges in representing and reasoning with scalable and adaptable context-sensitive information in Bayesian networks.

1.1 Background and Motivation

In early 1930's, Whorf [1956] did an influential work in the psychology of human thought behavior and postulated a famous hypothesis that the thoughts and behavior of humans are determined (or are at least partially influenced) by language. This hypothesis can be used to explain why the direct probabilistic approach that required an unreasonable amount of numbers for uncertainty representation was completely discarded in 70s. But when Pearl proposed the Bayesian network notation [Pearl, 1988] in the early 80's, it became a dominant strategy for representing uncertain domain models. The point is that choosing the right formalism helps to save many hours of unnecessary efforts in knowledge representation.

Building probabilistic domain models for uncertain reasoning is gaining importance for knowledge engineering. A knowledge engineer's job is to design an appropriate reasoning model based on expert knowledge by selecting the required domain attributes, modeling the correct relationships among them, and eliciting or estimating the probability parameters. Knowledge engineering a probabilistic model is partic-

ularly useful when: a) relationships among the attributes can be modeled; b) only limited amount of the training data can be obtained; and c) the number of attributes may not be known *a priori*. However, in practice, direct knowledge engineering of probabilistic models for complex domains is hard and one must design methods and notations that can simplify the representation and elicitation of the models.

In the area of probabilistic graphical network representation, the community has been slowly adding representation techniques (Figure 1.1) that can be categorized as:

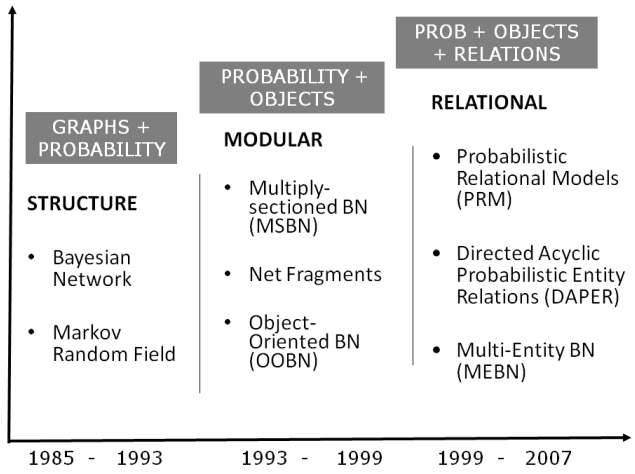


Figure 1.1: *Evolution of Probabilistic Graphical Models*

Type 1: Representation for local knowledge fragments [Laskey & Mahoney, 1997; Ngo et al., 1995; Heckerman, 1991; Poh & Fehling, 1993]

Type 2: Representation providing integrated, multi-level and multi-perspective view [Leong, 1998; Sundaresh et al., 1999; Wu, 1998]

Type 3: Representation utilizing other knowledge representation frameworks such as Logic and algebraic languages models [Goldman & Charniak, 1993; D'Ambrosio, 1994; Ngo et al., 1995]

Type 4: Representation borrowing concepts from knowledge representation and programming languages [Koller & Pfeffer, 1997; Koller, 1999]

Type 5: Representation targeted at generalization [Frey, 2003]

Recently, some efforts have proposed representations to target special domain applications and adapted concepts from particular domains such as Module Networks [Segal et al., 2004] in genetics, Dependency Networks [Heckerman et al., 2000] for collaborative filtering, and Multiply-Sectioned Influence Diagrams [Zeng, 2006] for distributed agent modeling. This thesis is based along the similar general theme and focuses on the emerging requirements of modeling "*context*" as a new dimension.

Representation languages that capture a formal notion of "context" and exploit context-sensitive modeling would be useful to effectively support various analytical tasks in many applications. For instance, in medicine, Clinical Practice Guidelines (CPGs) have emerged as an an excellent source of certified expert knowledge to reduce variations in clinical practice. Recently some efforts [Sanders, 1997; Zhu & Leong, 2000; Zhou, 2005] have suggested similarities between CPGs and probabilistic graphical networks. However, effective utilization of CPGs for engineering a probabilistic graphical network remains a challenge because CPGs are highly asymmetric in nature, i.e., some information is valid only in particular situations. Similarly, modeling situational variations to capture generic contexts is also gaining importance in other domains such as context-aware or self-aware computing [Dey, 2000; Terziyan, 2006].

Understanding the underlying conceptual models of context-dependent reasoning and the context-related requirements in various fields will contribute towards designing a general methodology for context-based reasoning under uncertainty.

The rest of the chapter includes an overview of the scope and content of the work, and a detailed guide to the rest of the thesis.

1.2 Understanding Context

The term "context" is used frequently, but its definition and usage vary across different disciplines. Even within Artificial Intelligence, the usage of this term varies with the domain. For example, context-aware and mobile computing use *context* [Dey, 2000] as information about object and its physical surroundings such as object's environment and location, while databases, ontology or rule-based formalisms use *context* to define the conditions of activation and delimit the scope, or to act as a screening filter for presenting minimal information content.

In this work, "*context*" associates situational aspects with the information content and defines the information that holds in a specific situation. For instance, in a pneumonia management model, context can be used to separate the information related to inpatient treatment from that related to outpatient treatment.

We now describe the context modeling problem using Bayesian networks.

1.2.1 Context Modeling Problem in Bayesian Networks

Bayesian network (BN) [Pearl, 1988] provides a language to represent and reason with uncertain information using the probability theory. A BN is a factored representation of the joint probability distribution over a set of random variables. It is a directed acyclic graph (dag) whose structure depicts conditional independences among the variables. Each variable or node X in a BN is associated with a set of conditional probability distributions of the form $P(X|\mathbf{Pa}(X))$, normally encoded in a conditional probability table (CPT). $\mathbf{Pa}(X)$ is the set of predecessor or parent variables on which X, the target or consequent variable is conditioned on. The nodes in the network denote the random variables and the edges between these nodes denote the conditional

probabilistic dependences among the variables.

The generalization over a class of variables or an object can be expressed using relational logic extensions of the BN framework [Friedman et al., 1999; Heckerman et al., 2004]. For instance, a probabilistic logic rule ∀z, $P(X(z)|\mathbf{Pa}(X(z)))$ expresses the fact that this conditional probability distribution applies to attribute X in all the instantiations of an object z. For inference, a relational network is usually rolled into a propositional BN network.

Context, in the BN sense, refers to an assignment of values to a subset of variables [Boutilier et al., 1996], called the context variables or context attributes. A BN is propositional in nature as it models a fixed number of variables with predetermined probabilistic relations. Hence, BNs cannot effectively represent situations where the exact number of variables may not be known *a priori*, as in the relational BNs. BNs also cannot fully exploit the structural variations that arise with changes in specific context attributes or values. For instance, if the patient is male, then all the complications related to pregnancy in a general diabetes management model become irrelevant. To represent such variations, a BN must capture all the potential context values in the CPTs. If the context values are known, irrelevant variables or values may be identified in the BN. But a BN is a symmetric representation that is unable to capture and exploit such value-level contextual independence [Boutilier et al., 1996; Zhang & Poole, 1999; Geiger & Heckerman, 1996]. In particular, the classical definition of conditional independence is too restrictive to capture these independences [Zhang & Poole, 1999].

Most previous efforts have mainly focused on the propositional level rather than the relational level. They incorporate context-sensitive representations in BN by: 1) targeting at local parameter decomposition for specific variables, e.g., structured CPTs [Boutilier et al., 1996; Poole & Zhang, 2003; D'Ambrosio, 1995]; 2) assuming the con-

text to be known *a priori* [Mahoney & Laskey, 1998; Ngo et al., 1995]; or 3) modeling different BNs for relevant contexts [Geiger & Heckerman, 1996]. Context-sensitive information may induce a systematic structure decomposition of the BN graph and not just the local parameter decomposition of a variable in the BN. This problem escalates in the relational BNs as the parents as well as the context variables cannot always be generalized and are more likely to be valid in some particular situations. Moreover, a single context variable may induce partitioning of the complete knowledge involved [Guha, 1993] and affect multiple consequent variables. Hence, inference efficiency can be improved with an effective manipulation of context-specific graph structures. Furthermore, increase in the number of context variables may lead to highly complex BNs where both the exact and approximate inferences may be intractable.

1.2.2 Context modeling under uncertainty: Challenges

We will discuss the challenges for context-based reasoning under uncertainty in detail in Chapters 2 and 3. We now briefly summarize the main challenges that we address in capturing context-sensitivity in a BN framework:

1. *Representational Challenge*: In Chapter 2, we show that the context modeling in the relational BN requires handling a special problem, which we call the problem of situational variation. Situational variations induce the following representational challenges:
    - The exact number of context variables and/or context values may not be known beforehand.
    - Association among the variables in the network may vary with specific context values.
    - Both the graph and the CPT structures may vary with the number of

context variables, their possible values, and the available context (value) observations or evidences.

2. *Inferential Challenge*: The network becomes complex with context uncertainties such that the inference over the BN framework may be too slow or intractable.
3. *Knowledge Engineering Challenge*: Direct knowledge engineering of a BN with many context variables is harder than using context-specific local BNs, e.g., small BNs created for each set of context values. This, however, induces the issue of dependencies among the contexts.
4. *Scalability*: The framework should be able to scale well with the increase in the number of context attributes.
5. *Adaptability*: Fast and repeated adaptations are needed to exploit context-specific structures and to improve the inference efficiency.

1.2.3 Impact

The above challenges directly impact the representation and inference of a BN. In the experimental results in Chapter 8, we would empirically demonstrate the effects using two examples in Chapter 2. The main areas of impact are as follows:

a) ***Representation:***

- *Larger graphs*: The network may be encoded with many irrelevant variables in a specific context.
- *Larger CPT sizes and extra parameters*: Each variable can have several parent variables leading to an exponential increase in the size of the CPT. More parameters in the network also imply extra costs in parameter elicitation or estimation.

- *Extra observation acquisition costs*: A query over the BN may require more variable instantiations than necessary in a specific context, e.g., the number of questions asked to a user before answering a query.

b) ***Inference:***

- *Higher memory requirements*: With many more contextually irrelevant variables and parameters, the inference has higher memory requirements.
- *Slower*: Larger CPTs imply higher factor size, which can make inference computationally slower.
- *Intractable*: This also results into an exponential increase in the inferential complexity such that not only the junction tree inference algorithm, a de-facto exact inference method for BNs, but also the loopy belief propagation inference, a popular approximate inference method, runs out of memory even on the problems with a small number of contexts.

1.3 Research Objectives

This work attempts to answer the following questions on major context related issues within the scope of the BN framework:

- What are the different requirements for context modeling under uncertainty?
- Can we exploit context to improve transparency of the model representation under situational variations?
- How can the model adapt to context-specific structure?
- Can we improve the inference efficiency?
- Can such a framework be extended to learn probability estimates from data?
- How can the context representation be practically useful?

1.4 The New Idea

1.4.1 Context-Sensitive Network

We propose a special graphical language, called Context-Sensitive Network (CSN), to reason with contextual information under uncertainty. CSN[1] is a graph representation that consists of three types of nodes: a) *Variable nodes*, b) *Context function nodes*, and c) *Function nodes*. In Figure 1.2, variable nodes R, N, S, L, B denote random variables. L and B are *context variables*, i.e., their value assignments indicate situational variations in the BN. Nodes 1, 2 and 3 (shown as small rectangles) are the context function nodes; each of them specifies a (partial) context-specific probability distribution among the variables connected. A context function node has a context label to indicate the context in which the function is true, e.g., the context label for Node 3 is "L0, B1". The function node Fs (shown as big rectangle) represents a collection of all the context function nodes having the same consequent variable.

CSN, like BN, combines graph theory with probability theory and graphically represents the factorization of the joint probability distribution of all the attributes in the domain. We will show that CSN presents a theoretically sound approach that scales well with contextual information, and is unaffected by the presence of uncertain information. CSN is based on the Directed Factor Graph (DFG) representation [Frey, 2003], which is a generalization of BN. The difference between the BN formalism and CSN is that CSN, like DFG, explicitly represents the quantitative function on the graph. However, unlike DFG that allows arbitrary factorization and hence needs to deal with the normalization conditions, the CSN is always normalized and expresses the factorization of the joint probability distribution based on the notion of contextual independence. This also differentiates the CSN from the BN, as the factorization

[1]Some preliminary results of this work were presented in [Joshi & Leong, 2006; Joshi et al., 2007].

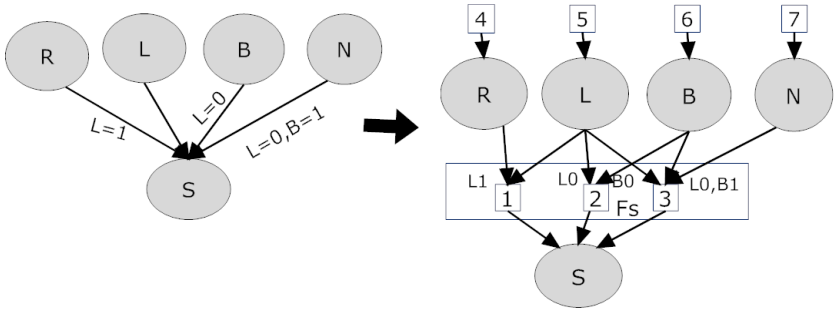


Figure 1.2: *A simple BN (left) with context-specific associations and an equivalent CSN (right). Labels on the edges of the BN indicate that associations are context-specific, i.e. if* $(L = 1)$*, then* S *is dependent on* R *but not* B *or* N*, and if* $(L = 0, B = 1)$*, then* S *is dependent on* N *but not* R*. In* $(L = 0, B = 0)$*,* S *is independent of both* R *and* N*.* L, B *are called context variables/attributes in our work and their specific values or assignments (*$L = 1$ *or* $L = 0, B = 1$*) are called context assignments. However, BN is based on conditional independence, so such context-specific associations cannot be easily exploited. BN needs a full-blown CPT while CSN can represent context-specific probability partitions*

in BN is based on the notion of conditional independence. Moreover, unlike BN and DFG, CPTs in CSN can be represented using context-specific partitions, whereas the BN or DFG traditionally utilizes full-blown CPTs. We will show that the CSN representation can exploit contextual dependence of the probability functions and address the desiderata for context-sensitive reasoning. Like BN, the CSN formalism can answer multiple queries by modeling multiple dependent variables. Furthermore, like BN, CSN can be extended to provide a methodology for estimating contextual probabilities if the data are available.

1.4.2 Local Context Representation

One advantage of CSN is that it can be used as an underlying framework for a local context modeling scheme; in other words, a representation scheme can serve as a meta-representation layer for transparent knowledge engineering. We propose Contextual

Local Views, a representation scheme, to encode the local knowledge of the related variables and their relationships in a specific context value. The representation scheme supports capturing multiple contextual scenarios within one local network and scales linearly with the number of contexts. Contextual Local Views also address the issue that the full CSN graph can become cumbersome for larger graph size.

1.4.3 Inference with probability partitions

In context modeling, the functions in a specific context may represent only the partitions of the full CPT. However, the message passing algorithms, in fact almost all algorithms, for BN have been defined to work mainly with full conditional probability distributions and not their partitions. To address this, we extend the belief propagation algorithm and propose three new operations for message passing: Context-sensitive Factor Product, Context-node Marginalization and Context Marginalization. We show that the overall computations on CSN are similar to those on BN.

1.4.4 Dynamic model adaptation

We show that the CSN representation extends the formal notion of d-separation [Pearl, 1988] to utilize contextual dependencies for determining contextual relevance. CSN also supports easy incremental model adaptation using only the graph manipulation operations. As BN is symmetric in nature, model adaption approaches based on the BN representation cannot utilize the concept for contextual dependencies. We show that model adaptation in CSN is much more efficient in exploiting both local parameter decomposition and graphical structure decomposition than the BN-based approaches, which typically require additional and expensive manipulations of the CPTs.

1.5 Contributions

The major contributions of this work are as follows:

Firstly, we propose a new general methodology for context-based reasoning under uncertainty. CSN preserves both the general and the context-specific representations while effectively supporting different possible scenarios for context-specific inference.

Secondly, the CSN representation allows local context modeling under context uncertainty. This is unlike previous local context modeling approaches that assume context as a deterministic attribute. Furthermore, we demonstrate how local context modeling can facilitate newer ways of knowledge engineering such as using guideline representation structures.

Thirdly, the CSN representation allows flexibility in model adaptation. By introducing a new paradigm of dynamic model adaptation, we break from the mold of using single graphical models for each task and advocate the design of weaving multiple models together using a context.

Fourthly, we propose a new message passing inference algorithm for reasoning with CPT partitions. Message passing is a general technique applicable to many other domains [Aji & Mceliece, 2000] such as multi-agent systems [Xiang, 2002]. However, message passing algorithms traditionally assume that the nodes are associated with full probability factors. By proposing three new operations and showing how the context-specific partition functions can be utilized for inference, we hope that other research domains can benefit from our approach.

Finally, the research provides insights into the nature of, and the difficulties in context-based reasoning in several application domains. These results can serve as guidelines for future research that addresses similar problems or improves current techniques.

1.6 Case Study and Results

We have developed a prototype implementation to empirically examine the effectiveness of the proposed methodology. The main summary of results include: a) CSN encoded much fewer total parameters and induced smaller maximum parameter widths than the corresponding BN; b) With a large number of variables and contexts, CSN outperformed BN significantly in both memory size occupied and inference time taken. For instance, in one case, CSN only took about 30 secs while the equivalent BN took 11 mins. Efficient implementation in the BNT toolbox [Murphy, 2002a] on the same case took 5 mins while the exact inference junction tree algorithm failed.

We have also informally evaluated the effectiveness of CSN with two case studies that involve actual clinical situations: the first one involves Coronary Artery Heart Disease (CAD) [Joshi et al., 2007] and the second one involves Community-acquired Pneumonia (CAP). Based on these case studies, we demonstrate that CSNs are expressive enough to handle a wide range of problems involving contexts in medicine. The case study on CAP also illuminates how a context-sensitive representation framework can utilize newer knowledge acquisition techniques and explicates the novel use of clinical guidelines for knowledge engineering probabilistic graphical networks.

1.7 Structure of Thesis

This introductory chapter has briefly described the background and the motivation of the work, summarized the challenges involved, and presented the research objectives and target contributions of this work. The remainder of the dissertation is organized as follows:

Chapter 2 introduces the context problem and the challenges involved.

Chapter 3 discusses the definition and usage of context in different domains, briefly relates the developments in the field, explains the desiderata for context modeling, introduces the current approaches for contextual reasoning, and finally reviews their advantages and limitations.

Chapter 4 is the heart of the thesis and formally introduces Context-Sensitive Network. We explain the syntax, semantics, theories and properties of CSN.

Chapter 5 defines the algebra and theory for inference, formulates the belief propagation algorithm, explains the algorithm using an example, shows the visualization of the computations, and describes the pros and cons of the inference method.

Chapter 6 presents a local contextual representation scheme, defines the interface to combine local contextual models into the underlying CSN, sketches context structural adaptation, and examines the different types of inference supported in CSN.

Chapter 7 discusses relational modeling and parameter learning in CSN.

Chapter 8 contains the experimental evaluation and examines the effectiveness of CSN based on two case studies.

Finally, Chapter 9 summarizes the achievements and limitations of this work, compares it with related work, and offers some ideas for future research.

2

The Problem of Situational Variation

In this chapter, we describe the problem of capturing situational variations. We present two examples to illustrate the problem, to describe the challenges and to motivate our language design. The problem of situational variation poses the following question: How can we compactly capture different situational variations in a single graphical representation instead of capturing them in several BNs/relational BNs? Such situational variations can occur with uncertainty over any generic contexts such as the 5 "W"s of a situation - who, what, where, which, and when. An interesting aspect of the solution to this problem is that the original BNs/relational BNs are then just a few context-specific instances of this "general" graphical model.

2.1 Examples

2.1.1 Example 1:

Consider a modified version of the "dog network" from [Charniak, 1991] as shown in Figure 2.1. Assume that you are visiting your relatives. You know that if the family is out ($familyOut$), then the light is on ($lightOn$) and the family dog would be out in the backyard ($dogOut$). You know that a dog would also be out if it has a bowel problem ($bowelProb$). You may or may not hear a dog bark ($hearBark$) if it is out in the backyard. You want to infer whether your relatives' family is out ($familyOut = out$) given that you hear a bark ($hearBark = true$).

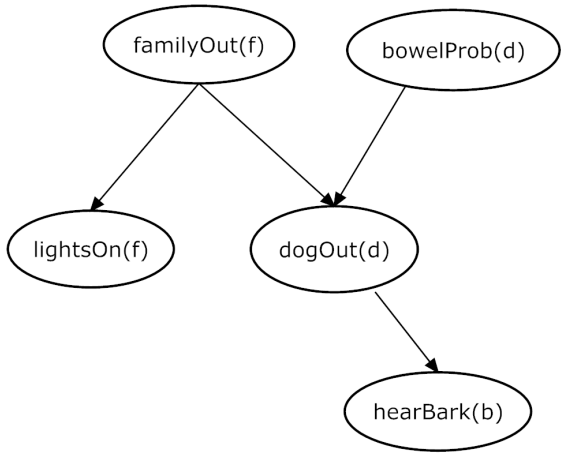
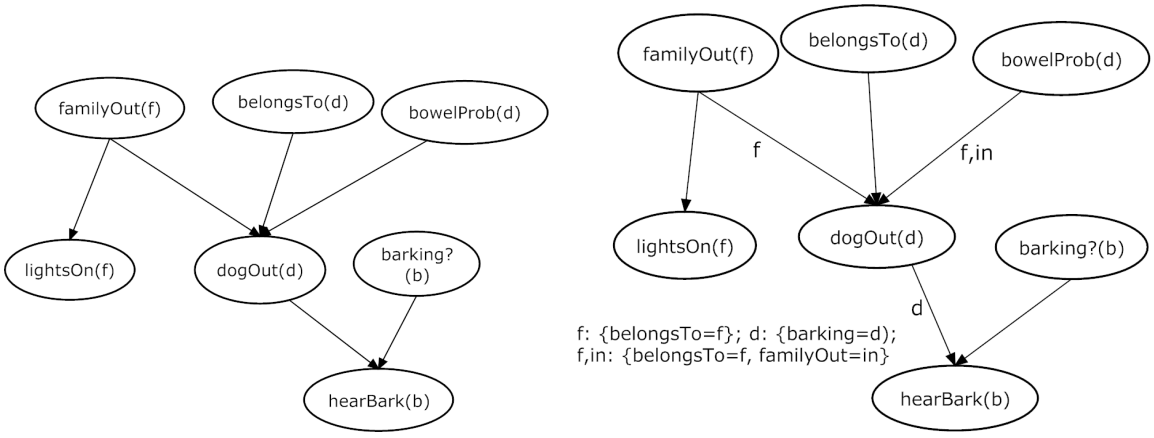


Figure 2.1: *The Dog relational BN network defines an abstraction of dependencies and their relationships for multiple dogs and families, and the relevant properties. Here f: refers to a specific family, d: a specific dog and b: a type of bark*

Figure 2.1 shows a relational Bayesian network for this example. This relational BN generalizes the original Bayesian network (BN) in [Charniak, 1991] to a class of families, dogs and barks. The underlying assumption for relational modeling in Figure 2.1 is that you know how your relative's dog looks and you can recognize its bark.

Example 1 (continued):

However, what if you do not know whether your relatives have a dog, multiple dogs and/or how their dog(s) would look like. For instance, if there are many families and many dogs in the neighborhood, you are not sure to which family each dog belongs (belongsTo). If you hear only one type of bark (barking), you are not sure which dog is barking.



(a) dog relational BN with context **(b)** Context labels on edges

Figure 2.2: *Dog relational network with context uncertainty. Here f: refers to a specific family, d: a specific dog, b: a specific bark and probabilistic rule table for node dogOut(d) (read as every dog d has an attribute dogOut) represents $\forall d, f$, P(dogOut(d)| bowelProb(d), familyOut(f), belongsTo(d)). Labels in 2.2(b) show the associated structure uncertainty with context, for example, label (f,in) represents context: {belongsTo=f, familyOut=in }. (f,in) means that the dogOut is associated with bowelProb only when a particular family 'f' to whom the dog belongs is inside(in) the house.*

In such cases, the associations among the variables hold only under specific situations. Furthermore, the relevant graph and CPT structures can vary substantially depending upon the number of context attributes and their values. Figure 2.2(a) shows the modified version of the Figure 2.1 augmented with the two additional variables indicating contextual uncertainty. In Figure 2.2(b), the labels show that associations hold

only in specific contexts, i.e., specific assignments, upon observations, of values of the context variables. Table 2.1 shows the relations and domain values of the variables involved.

Context variables, in our work, are the parents of the target variables for which they form the contexts. Unlike the ordinary random variables, context variables are also special variables that, if known, can induce significant simplification in the state space and/or model structure. This example shows a few different types of context variables: causal ($familyOut$), non-causal ($belongsTo, barking$) and relational context variables ($familyOut$, $belongsTo$). In this example, $familyOut$ is a causal context variable because if you know that when a family is out ($familyOut$), its dog is certainly kept out ($dogOut$) in the backyard, whether or not the dog is having a bowel problem ($bowelProb$), i.e., *P(dogOut|familyOut, bowelProb) = P(dogOut|familyOut,* $\neg$ *bowelProb).*

In Example 1, there are three possible cases involved: a) you know about both to whom the dog/dogs belongs/belong and which dog/dogs is/are barking; or b) you know about only one of the conditions; or c) you do not know about any of the two conditions. For case (a), there is no uncertainty in any context variable and Figure 2.1 is sufficient to support reasoning about the situation. But, cases (b) and (c) involve uncertainty over at least one context attribute and can involve multiple dogs and/or families.

The situational uncertainty over the variables $belongsTo$ and $barking$ induces uncertainty over the objects *family* and *dog* of the parent variables such as $familyOut(f)$ and $dogOut(d)$ in the BN. In this case, we are uncertain of which dog(s) and family(ies) to associate the links with, exhibiting *reference uncertainty* [Friedman et al., 1999; Laskey et al., 2001]. We also do not know what parameters to include, e.g., the probability parameters such as $P(hearBark|dogOut)$ for each dog can be different,

exhibiting *parameter uncertainty* [Terziyan, 2006].

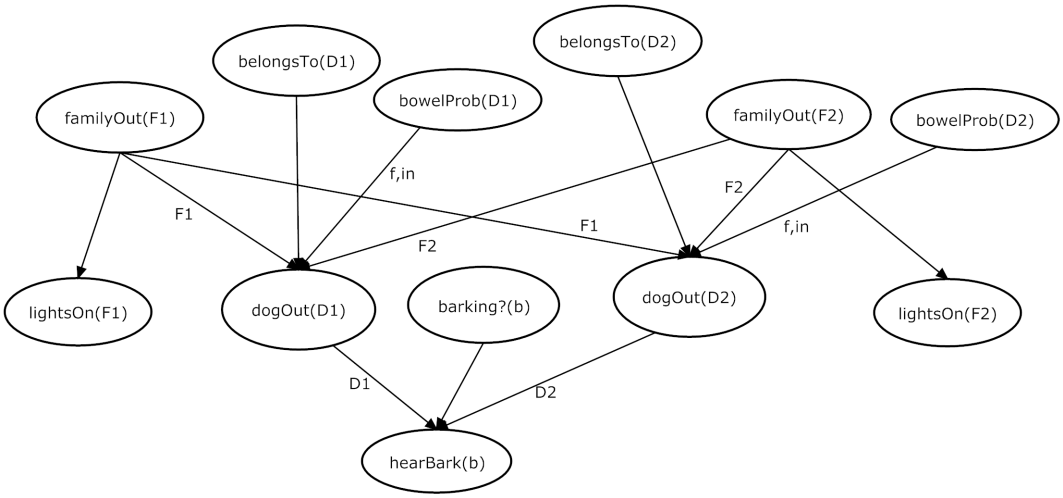


Figure 2.3: *Instantiated Relational BN of Figure 2.2(b) with 2 dogs(d)and 2 families(f). Each dog as well as each family has a set of associated properties or attributes (dogOut, belongsTo, familyOut), i.e., dogOut(D1) means the dogOut attribute of dog D1. Labels on arc refers to context values as explained in 2.2(b). 'f,in" label means dog belongs to family 'f' that is inside(in) the house. In this thesis, objects are represented by lower case letters, for example, (d, f) and the instantiations of the objects are upper case letters followed by the instantiation number, for example (D1, D2, F1, F2)*

However, in a BN that models these scenarios, all the possible families and dogs must be encoded. Case (c) involves uncertainty over some or all families and dogs in the neighborhood. Figure 2.3 models one such situation by rolling out a complete BN from the relational abstraction in Figure 2.2(b). Figure 2.3 assumes that there are two families, F1 and F2, and two dogs, D1 and D2. We have put labels on edges in Figures 2.2(b) and 2.3 to show that the dependencies are only valid, i.e., the edges are only present in specific contexts. Similarly, Case (b) involves uncertainty in some of the context attributes. Figure 2.4 models the situation when the values of two of the context variables are known or observed.

Relations	Entity Related to	Context	Domain values
familyOut	Family	Yes	out,in
bowelProb	Dog	No	yes,no
dogOut	Dog	No	true,false
lightsOn	Family	No	on,off
hearBark		No	yes,no
belongsTo	Dog	Yes	F1,F2
barking		Yes	D1,D2

Table 2.1: *Example 1 description for 2 dogs and 2 families*

What if some of these situational variables are only known to you later? Given different values of context attributes *belongsTo* and *barking*, the resulting possible worlds can vary substantially too. Figure 2.4 and 2.5 show two possible worlds given different context observations.

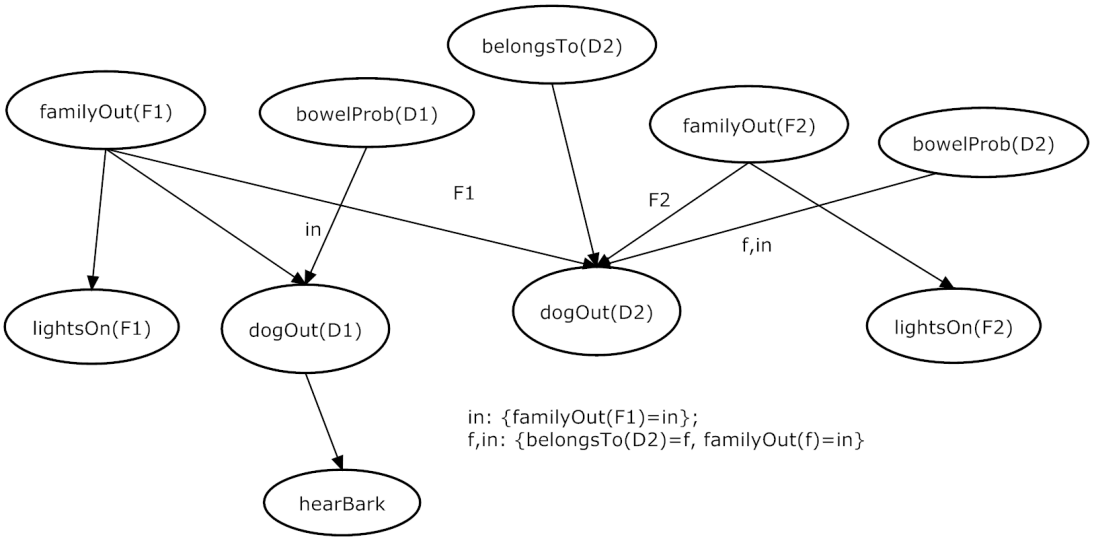


Figure 2.4: *Context-specific structure given the context involving two observed or assigned context values: belongsTo(D1)= F1, barking = D1 (observed context variable nodes are removed). This structure is different from Figure 2.2(a) and Figure 2.3 and shows that the relevant model structure can vary substantially depending upon the number of context variables, their possible values and the available context (value)observations or evidences*

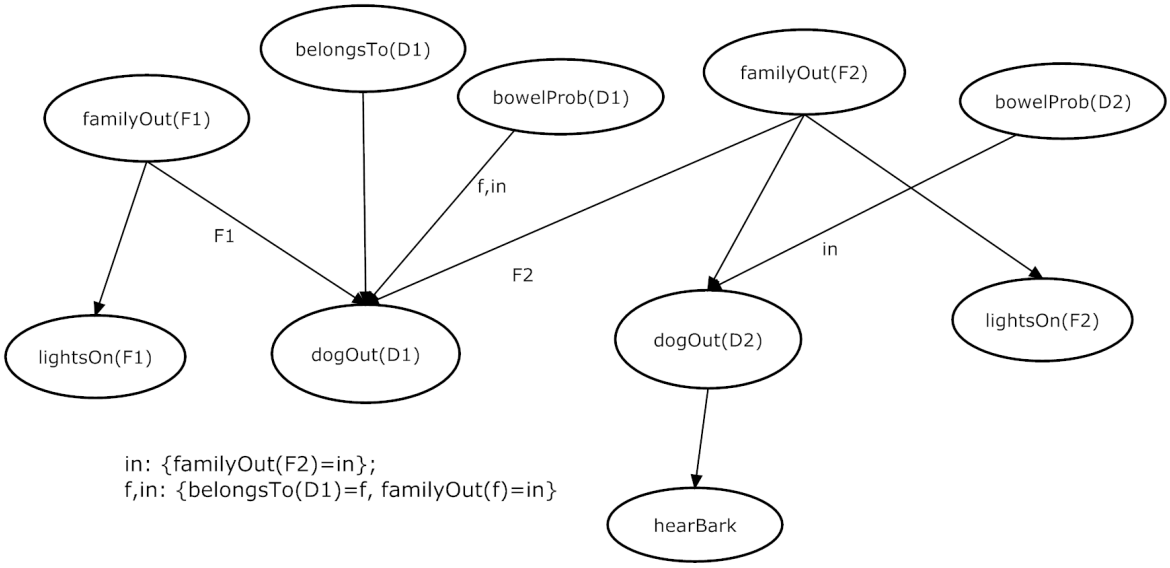


Figure 2.5: *Context-specific structure for the given context: belongsTo(D2)= F2, barking = D2 (observed context variable nodes are removed). Note the difference between this structure and that of Figure 2.4*

In general, such association uncertainties can be induced with uncertainty over any generic contexts, such as those related to the 5 "W"s of a situation. Modeling contextual dependence of the variables can facilitate dynamic adaptation to different graph structures for different scenarios.

2.1.2 Example 2:

The importance and demand of context-aware applications is continuously increasing in ubiquitous, and proactive computing domains. In the previous example, the relational context attributes were not directly dependent on each other. Consider another relational graph as shown in Figure 2.6 where one relational context attribute is dependent on another relational context attribute.

Assume that to improve customer satisfaction and increase patronage, a restaurant

chain wishes to predict the likely ratings of a specific restaurant in the light of the customer's context profile. The relational representation in Figure 2.6 captures the following facts:

a: A boy is accompanied to a restaurant by a girl {attribute: accompany(b)} and the food likings {attribute: likes(g)} of that girl influences the food order placed by the boy in the restaurant {attribute: order(b)}. The following probabilistic logic rule is defined for the attribute order: $\forall b, g$, P(order(b)|likes(g), accompany(b));

b: The boy places an order of a particular food type and the food quality of that food type {attribute: quality(ft)} influences the restaurant rating for the boy {attribute: rating(b)}. The following probabilistic logic rule is defined for the attribute rating: P(rating(b)| quality(ft), order(b));

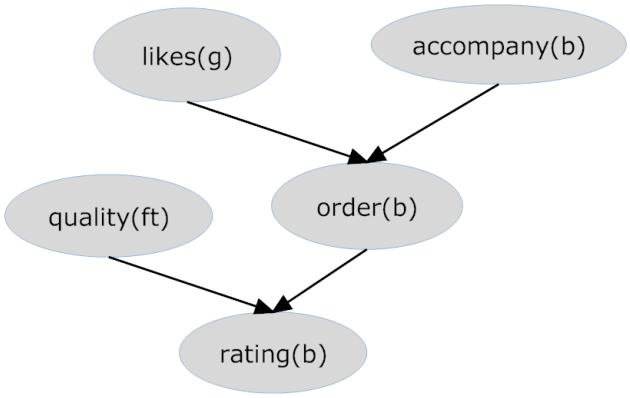


Figure 2.6: *Relational BN Representation of Example 2*

Rule-based systems have been commonly used to input user-defined rules for each specific context in problems as shown in Example 2. Figure 2.7 show two rules that would capture the above facts for context attributes *accompany* and *order* in an IF-THEN template. The instantiation of these rules can be used to infer the information content in different scenarios. Rule-based systems, however, have limited uncertainty

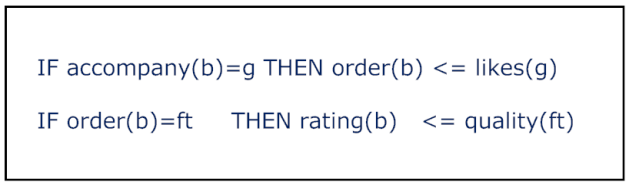


Figure 2.7: *User-defined rules for context attributes: accompany and order. Here b:boy; ft:foodTypes; g: girl*

handling capability and suffer from some well-known problems such as the inability to handle bidirectional inference. For instance, in Example 2, given the customer's rating, it is difficult to infer the values for the *likes* attribute in a rule based systems. Probabilistic representation as shown in Figure 2.6, on the other hand, can be used to infer bidirectionally and answer questions such as: a) the possible rating that a new boy is likely to give to the restaurant, b) the food quality of a particular food type given the rating, c) the food likings of a particular girl, and d) the boy's likely order given rating.

Context in this example includes customer's situational attributes such as *accompany* and *order*. The context attribute *order* is dependent on another context attribute *accompany*. The situational uncertainty over context variables *accompany* and *order* can induce uncertainty over the relational variables *likes* and *quality* in the BN. In this case, we are uncertain of which girl's *likes* and which food type's *quality* to associate the links with.

Furthermore, the number of context variables can exponentially increase depending on the number of objects such as the boys, the restaurants, the girls, and the food types. The number of domain values of a context or non-context variable can be a function of the number of instantiations of some objects, for instance, we have assumed here that *likes* and *order* have the same number of domain values as the number of food types. A customer's rating of a restaurant is likely to reflect the food

Relations	Entity	Context	Domain values	Instantiated Attributes
likes	Girl	No	food types: FT1, FT2	likes(G1) likes(G2)
accompany	Boy	Yes	girls: G1, G2	accompany(B1) accompany(B2)
order	Boy	Yes	food types: FT1, FT2	order(B1) order(B2)
rating	Boy	No	High, Low	rating(B1) rating(B2)
quality	foodType	No	Good, Bad	quality(FT1) quality(FT2)

Table 2.2: *Example 2 description*

quality of the particular dish ordered rather than the food quality of all the dishes in the restaurant. But in a BN that models the rating prediction, all the possible quality scales for all the dishes must be encoded, as BNs cannot capture such value-level contextual independence [Boutilier et al., 1996; Zhang & Poole, 1999; Geiger & Heckerman, 1996].

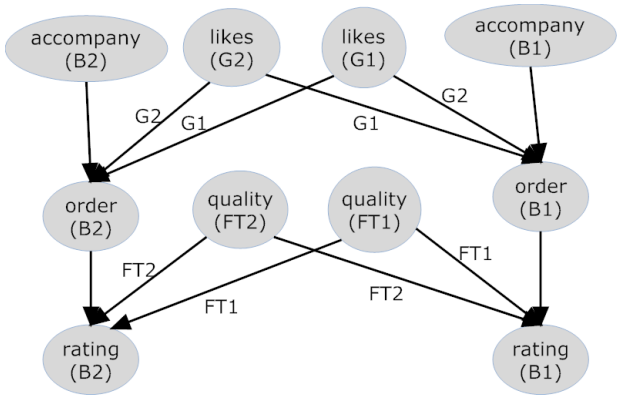
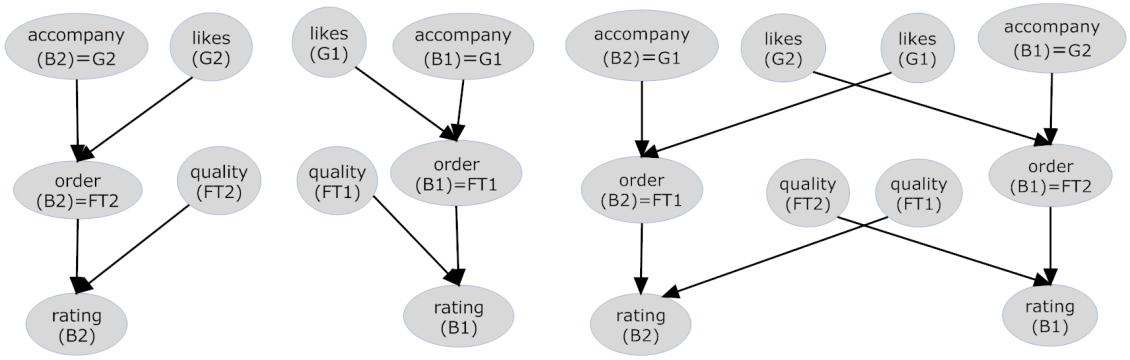


Figure 2.8: *BN Representation of Example 2 for 2 boys, 2 girls and 2 food types. Labels on the arcs denote the context(s) in which the consequent variable is dependent on the parent(s).*

Figure 2.8 shows the problem representation in the BN framework for 2 boys (B1, B2),

1 restaurant, 2 girls (G1,G2) and 2 food types (FT1, FT2). Table 2.2 describes the relations, instantiations, and domain values of the random variables involved. There are four context variables: *order(B1), order(B2), accompany(B1), accompany(B2).* Given an assignment of values to the variables such as *accompany* and *order*, the probabilistic distribution exhibits further independence. For example, if boy B1 is accompanied by girl G1, then B1's food order is independent of girl G2's food likings. Similarly, if Boy B1's ordered food type FT1 in the restaurant, then B1's rating is independent of the food quality of the food type FT2. Figure 2.9 describes a few possible worlds and the resulting context-specific model structures if some of the situational attributes are known . Situational variations can make the variables and the association among the variables to be a function of the number and the nature of uncertain contexts.



(a) possible structure 1 **(b)** possible structure 2

Figure 2.9: *Different values for four context attributes or variables {order(B1), order(B2), accompany(B1), accompany(B2)} in Figure 2.8 result in different context-specific model structures. 2.9(a) and 2.9(b) show two resulting structures in context {order(B1)=FT1, order(B2)=FT2, accompany(B1)=G1, accompany(B2)=G2} and {order(B1)=FT2, order(B2)=FT1, accompany(B1)=G2, accompany(B2)=G1} respectively.*

2.2 Summary of Challenges

Example 1 and Example 2 above demonstrate that situational variations can induce the following challenges:

1. *Representational Challenge*: Figures 2.1, 2.2, 2.3, 2.4 and 2.5 for reasoning under different situations appear to be different BNs in terms of the number of variables and the associations among the variables. However, it turns out that they are just several instances of different number of context variables, their domain values and the available context value observations or evidences. Can one single network language compactly capture all these situational variations? The representational challenges for such a language include:

    - The exact number of context variables may not be known *a priori*, e.g., the number of context attributes $belongsTo(d)$, $familyOut(f)$ and $barking(b)$ depends on the number of instantiations of the objects: dogs, families and barks.
    - The exact number of context assignments may not be known *a priori*, e.g., the domain values of context variable $belongsTo(d)$ may be equal to the number of families in the problem.
    - *Dynamic Adaptation*: Both the graph and the CPT structures can vary with the number of context variables, their domain values, and the context observations or evidences. The possible worlds in Figure 2.9 show that the language needs to dynamically adapt to the different context-specific model structures.
    - *Reference uncertainty* [Friedman et al., 1999; Laskey et al., 2001]: The relevant parents for a specific consequent or target variable may vary with the specific context values. Such association uncertainty can be induced in

both relational non-context parents as well as relational context parents. For instance, in Example 1, the $familyOut$ relational context variable may be irrelevant depending on the context variable $belongsTo$. Similarly, in Example 2, the association of context relation attribute *order* with non-context attribute *likes* is a function of values of the context relation attribute *accompany*.

- *Co-reference uncertainty* [D'Ambrosio et al., 1999]: Multiple consequent variables can be different functions of the same context variables. For instance, if you know that both dogs belong to the same family, then an evidence of that family being *out* renders the association of $bowelProb$ with $dogOut$ for both dogs irrelevant.
- *Variation in contextual independence structure*: As the relational context attributes can hold only in a particular situation, so the complete contextual independence structure in the local distribution may not be known beforehand and may vary depending upon the number of context variables, their domain values and the context observations. For instance, in Figure 2.4, $dogOut(D1)$ has 2 parents, of which 1 is a context attribute ($familyOut(F1)$), while in Figure 2.3, it has 4 parents, of which 3 are context attributes ($belongsTo(D1)$, $familyOut(F1)$, $familyOut(F2)$). The structured CPT approaches [Boutilier et al., 1996; Poole & Zhang, 2003; D'Ambrosio, 1995] proposed to incorporate context-sensitive representation in BN. However, they assume no variation in these contextual independences.

2. *Inferential Challenge*: An increase in the number of context variables may lead to highly complex BNs where the exact inference may be intractable. Furthermore, a large number of context values can induce a large set of parent variables,

exponentially increasing the size of the CPT of the consequent variable. For example, various context-specific simpler graphs in Figure 2.9 are combined together into a more complex network in Figure 2.8 due to uncertainty over the attributes *accompany* and *order*.

3. *Model decomposition:*

   - Structure decomposition: The overall probabilistic graphical model, as shown in Figure 2.8, should be able to decompose, in a systematic manner, into much simpler graphs with context conditioning, as shown in Figure 2.9, given specific context assignment. This, however, is not clear in the BN framework, as the evidence about *accompany* and *order* in Figure 2.8 does not simplify and removes all the dependencies to decompose the model into the different models as shown in Figures 2.9.
   - Local parameter decomposition: Contextual representation should also be able to exploit the sparseness in the parameters by utilizing value-level independence in the probability distribution. For instance, the BN in Figure 2.8 requires (1*6+4*8) = 38 parameters to specify the full distribution. However, the model for Example 2 can, in fact, be specified with only (1*6+4*4) = 22 parameters.

2.3 Notations

Before concluding the chapter, we introduce the notations used throughout this dissertation.

If A denotes a random variable, then we call the set of values/assignments that the random variable A can assume the *domain space* $D(A)$ of A. The domain value/as-

NOTATION	MEANING
A, B	two random variables A and B
$D(A)$	domain space of a variable A
a, b	Assignment or domain value in the domain space of a variable: $A = a, B = b$
$\mathbf{X}$	set of random variables e.g. $\mathbf{X} = \{A, B\}$
$\mathbf{x}$	set of values/assignments e.g. $\mathbf{x} = \{a, b\}$
$\mathbf{X}$=$\mathbf{x}$	a particular assignment to a set of variables e.g. $\mathbf{x} = \{a, b\}$
$\|\mathbf{x}\|$	total number of assignment states in $\mathbf{x}$ e.g. $\|D(X)\|= \|D(A)\| \ . \ \|D(B)\|$
$\lambda(A, B)$	factor with scope {A,B}, size ($\|\mathbf{a}, \mathbf{b}\|$) and probability distribution $\theta_{A,B}$
$\lambda(a, b)$	represents a factor with (a,b) as an assignment to set A,B
$\lambda(A = a)$	same as $\lambda(a)$
$\theta_{a\|b}$	conditional probability value for assignment (a,b)

Table 2.3: *Notations*

signment $A = a$ denotes the situation (event) that A takes on the value a in the domain space of A. If A, B are two random variables, then a factor λ over $\{A, B\}$, $\lambda(A, B)$, is a probability distribution table over the domain space or all assignments (a, b) of A, B. The variables set $\{A, B\}$ is called the *scope* of the factor. The total number of assignment states $|D(A)| \ . \ |D(B)|$ in A, B defines the *size* of the factor.

2.4 Summary

We discussed the problem of situational variation using two examples and described the main challenges involved. Situational variations render the variables and the associations among the variables to be functions of the number of uncertain context variables and their possible values. In the next chapter, we summarize the context modeling approaches in various fields and add a few more challenges for a general methodology for context-based reasoning under uncertainty.

3

Modeling Uncertain Context: A Survey

The use of context in many fields has been closely related to the specific tasks at hand, the research problems of present interest, and the domains of the application. This chapter briefly surveys context modeling approaches in the various fields and then describes the common features, typical context reasoning requirements and challenges across these domains. We, then, formally introduce the notion of context for probabilistic reasoning. In probabilistic modeling, we argue that a context information can cause the associations among the variables to be a function of the context variables. This introduces a finer factorization structure and asymmetry in the joint probability distribution. However, Bayesian networks are propositional and symmetric in nature. Consequently, context modeling using BN can be challenging. We review these challenges in this chapter and explain the capabilities and limitations of various possible techniques for context modeling.

3.1 Background of Context

Webster dictionary [Context, 2008] defines *context* in two different ways: a) "the parts of a discourse that surrounds a word or a passage and can throw light on its meaning"; b) "the interrelated conditions in which something exists or occurs." The Free On-line Dictionary of Computing [Foldoc, 2008] defines *context* as "that which surrounds and gives meaning to something else". The concept of context has been previously used to improve knowledge representation, to achieve computational gains, or to provide a better quality of service. Now, we briefly describe the usage of context in some areas in Artificial Intelligence.

In ubiquitous computing, Dey [2000] defines *context* as any information that can be used to characterize the situation of any entity, for example, user context (such as location, role, activity), environment context (such as surroundings, resources in neighborhood), medical and health context (such as emergency), and physical computing context (such as available bandwidth, processors).

In knowledge acquisition using *Protégé* II [Walther et al., 1992], *context* has been used to cluster, partition and organize knowledge. Context helps in reusing modules, designing concepts and modeling topological relationships among the concepts. The main claim of the approaches for knowledge acquisition is that the experts provide their knowledge in a specific context and the knowledge can only be relied upon in this context.

Context in communication and natural language has been used to understand the meaning of a word, passage or discourse based on the adjacent text. For instance, the sentence:"OS Daily: Why Linus choose Penguin?" [Song & Bruza, 2003]. In the context of "operating systems," the sentence is linked to Linux operating system: Linus Torvalds is the inventor of the Linux and Penguin refers to its logo and not the

bird. *Context* might also be used to fill the missing parameters in communication statements. Similarly, in information retrieval, *context* has been used to enhance the precision and recall quality by enhancing a query with more information than just the keywords [Jones, 2004].

Now, we briefly describe the common features and context-based reasoning requirements across some domains.

3.2 Desiderata for a Contextual Reasoning Framework

In this section, we examine the context usage and typical context reasoning requirements across some domains. Based on these common challenges, we sketch the following desiderata for a general framework for context-based reasoning under uncertainty:

1. *Uncertainty*: Uncertainty in non-context as well as context attributes
2. *Knowledge engineering*:
    - Model local context knowledge to specify context-specific clauses
    - Dependency among contexts, i.e., "nested" context
    - Modeling at the relational level
3. *Scalability*: Scale with large number of contexts
4. *Adaptability*:
    - Adapt to context-specific models given asymmetry in information
    - Allow dynamic adaptation with incremental context observations
5. *Inference*: Utilize asymmetry induced by context for efficient inference

3.2.1 Contextual reasoning in Medicine

Usage of Context: Context is an important issue in biomedicine with many applications in areas including primary care, bedside management, and preventive healthcare at home.

McCarthy [1993] described MYCIN, a program for advising physicians on treating bacterial infections of the blood and meningitis [Shortliffe, 1976], as an example of non-contextual modeling reasoning that humans always state something in a context and the context dependence of a theory needs to be explicit. He described this as: "*When MYCIN is told that the patient has Cholerae Vibrio in his intestines, it would immediately recommend two weeks of tetracycline treatment and nothing else. While this would indeed do away with the bacteria, the patient would perish long before that due to diarrhea. A contextual version of MYCIN, on the other hand, needs to add the context of a treatment and notice that any prescription must be made in the light of the fact that there is alarming dehydration.*" McCarthy's theory of context, proposition p holds (is true) in context c and he denotes this with a special predicate *ist(p,c)*. Guha [1993] extended this concept and used context to describe micro-theories, a set of axioms and vocabulary of a limited domains that are only true in a context. Guha [1993] extended this concept and used context to describe micro-theories, a set of axioms and vocabulary of a limited domains that are only true in a context.

We will see that we use similar ideas in a probabilistic framework for contextual local knowledge representation.

Example: Patient-centric models are becoming important in decision-support systems in medicine. However, the inherent uncertainty in the relations such as those among diseases and symptoms makes the patient-centric modeling even more challenging. In medicine, *context* may include several things such as patient profile,

characteristics, symptoms, risk or treatment strategies. For instance, in a general Coronary Artery Heart Disease (CAD) management model, if the patient is young, then the complications related to the family members become relevant, otherwise they may be irrelevant. For simplicity, let us assume that a physician just needs to model the following medical facts in a reasoning framework to screen for the possibility of CAD:

- The risk of CAD is the same for males and females with $age > 55$, otherwise risk for males is much higher than females
- Attributes related to family history are only important in young males with age less than 55 years

Here, Age is a context variable.

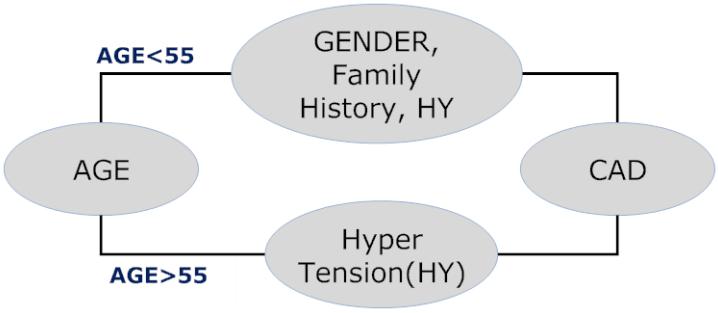


Figure 3.1: *Decision graph for partition of CAD knowledge with Age as context*

Contextual reasoning requirements: We describe the following typical requirements using this simple example:

- *Local context knowledge:* Each context attribute can partition knowledge into local micro theories where axioms and statements are specific to only a particular context, e.g., Age here is one context variable that partitions the CAD

knowledge as shown in Figure 3.1. Given that there can be several context attributes involved, local context modeling to specify context-specific clauses can simplify the knowledge representation.

- *Model adaptation and asymmetry in information:* Depending on the type of patients, we need different attributes for prediction of a disease. For instance, Gender and Family attributes are relevant for the prediction of CAD only in young patients. The attributes suitable in the case of the young patients are irrelevant for CAD prediction in an old patient.
- *Inference:* The decision diagram in Figure 3.1 shows how a context can lead to a whole piece of local knowledge becoming irrelevant for a particular patient. The inference procedure should utilize such asymmetry for more efficient inference.

3.2.2 Contextual reasoning in Systems Biology

Usage of Context: In Systems Biology, scientists model the networks of interaction among genes, proteins, molecules and diseases to achieve systems level understanding of complex life machinery. *Context*, in systems biology, may include triggering conditions such as the environment, a gene or a molecule that are required to express other genes and/or activate molecular pathways.

Example: Consider, for example, the development pathways in Arabidopsis thaliana, commonly studied plant species for understanding the genetic, and molecular biology of the flowering plants. The initiation of flowering is carried out by four different pathways: gibberellin, autonomous, vernalization and light-dependent pathways. [Blazquez et al., 2001] shows how the pathways are affected by context. *Context*, here, consists of light-conditions, temperature, time of day, and triggering molecules such as FWA. Different context values can active or deactivate different molecular pathways and the flower development may not happen if even one of these pathways

was not triggered. However, it may be difficult to ascertain the exact values of all the context attributes to infer if the flowering will happen.

Contextual reasoning requirements: Here, we describe the following typical requirements:

- *Scalability:* There can be several genes, proteins and contexts involved in a problem. Hence, context modeling representation should be scalable. One example of a large gene network is shown in [Davidson et al., 2002];
- *Model adaptation and asymmetry in information:* Associations among the attributes is a function of context leading to asymmetry in information. For instance, in the presence of the FWA molecule [Blazquez et al., 2001], an FT gene positively influences the gene expressions of the identity genes that carry out flowering; however, in its absence, the identity genes expressions' are independent of the FT gene.
- *Inference:* The inference procedure should adapt accordingly to the fact that different context conditions induce different interactions among the genes and molecules. This can substantially reduce the state space involved.

3.2.3 Contextual reasoning in Context-aware Domains

Usage of Context: Context-based reasoning in context-aware applications involves utilizing the contextual information, stating the relationships involved among contexts and objects, and inferring the attributes of interest. The researchers in context-aware computing application have focused on developing applications related to one or more of the following: acquiring information [Salber et al., 1999], identifying information [Davidyuk et al., 2004], inferring context information [Ranganathan & Campbell, 2003] and adapting application behavior based on the inferred context [Rossi et al.,

Conceptual Category	Example of Context
Who	User's Profile, mood, likings, companions
Where	Location, place, environment, weather, noise, temperature
What	Activity, request-type
When	Duration, time, date, occasion
Which	Device, thing, object, application, network and their properties

Table 3.1: *Conceptual categories of type of contextual information*

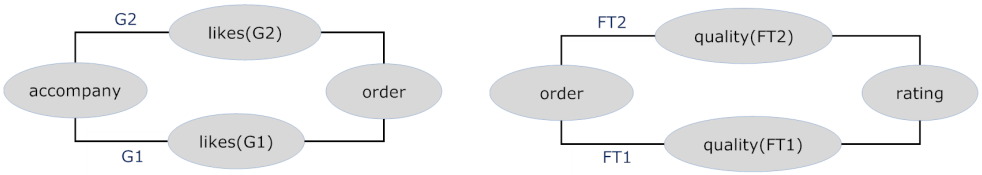
2005]. For example, mobile devices may tap on the user's age to advertise superheroes to kids and golf clubs to adults if the user is situated near a sport shop. The type of contextual information can be typically divided into five conceptual categories as shown in Table 3.1.

Recently, the semantic web community [Pessoa et al., 2007] has proposed rule-based approaches to utilize ontology concepts for context-aware applications. User-defined rules can be typically modeled in an if-then rule template written in the OWL ontology format [Bechhofer & van Harmelen, 2004]. *Context* here also defines the conditions of activations, delimits the scope and acts as a filtering layer.

Example: Consider Example 2 in Chapter 2 Section 2.1.2 (page 22). The recommendations are based on the rules defined to predict the ratings of a restaurant in light of the customer's context situation. A specialized domain ontology is defined to capture the following: a) classes such as *Customers, Restaurants*; b) sub classes such as *foodTypes*; c) Properties such as *likes, order, accompany, quality, and rating*; d) Data types such as *FT1* or *FT2* for *foodTypes* and *girlfriends* for *accompany.* Context, here, includes the customer's situational attributes such as accompany and order that act as filters to select the information. However, the values of all the situational attributes may not be known or available to an application.

Contextual reasoning requirements: Depending upon the target use and the type, context-aware applications may need to handle one or more of the following challenges:

- *Local context knowledge and contextual relevance:* With reference to Example 2 context attributes such as *order* and *accompany* partition knowledge and define scenarios that are contingent on the local context value as shown in Figure 3.2.
- *Dependency among contexts:* Each local context scenario may share both non-context attributes and context attributes with others. So, the two dependent contexts may need to be merged together for answering a query. For instance, the value of the context attribute *order* in this example is dependent on the value of another context attribute *accompany*.
- *Modeling at relational level:* Ontology modeling can help model generalizations of the rules over a class, for example, the generalized rules can be used to model relationships for all the customers and restaurants.
- *Scalability with number of contexts:* The number of situational attributes related to each customer and restaurant would linearly increase with the number of customers and restaurants.
- *Model adaptation and incremental context awareness:* All the context information may not be known *a priori* or given at the same instant. Therefore, the reasoning framework should be able to repeatedly adapt to a more context-specific scenario.
- *Inference:* Based on the context values, the inference needs only the selected attributes. For example, if the context value of *order* is FT1, then the inference should only utilize the *quality* of the FT1 food of the restaurant and should not need to know the FT2 food quality.



(a) accompany **(b)** order

Figure 3.2: *Decision graphs showing asymmetry in information in Example 2*

3.3 Context Reasoning and Rule-based Systems

We now examine the rule-based systems from the context desiderata view point laid out in Section 3.2. A formulation of Example 2 in Section 2.1.2 based on if-then logical rules is shown in the Figure 3.3. Context is modeled as the top most if-then activation condition and is local to each template. Similar to Guha's micro-theories [Guha, 1993; Akman & Surav, 1996], two rules templates for the context attributes: *accompany* and *order* can be used to describe proposition p that holds (is true) in context c.

```
If ?b accompany ?g
Then
      If  ?g likes ?f
      Then ?b order ?f
      Endif
Endif
```

(a) accompany

```
If ?b order ?f
Then
      If ?f quality LOW
      Then ?b rating LOW
      Endif
Endif
```

(b) food order

Figure 3.3: *Example 2 Rule-based templates : Here ?b:boys; ?f:foodTypes; ?g:girls*

As each context condition can be captured using a single "if-then" rule template, the rule-based approach can scale linearly with the number of contexts. Furthermore,

the local dependencies among the contexts can be modeled. For example, the context attribute *order* in Figure 3.3 is dependent on another context attribute *accompany*. The two templates need to be merged together to answer different queries. Based on the value of *accompany* and the corresponding value of *order*, the matching "if-then" rule would be instantiated. In a rule-based formalism, only the context clauses that are satisfied are considered and the irrelevant attributes in unsatisfied context clauses are removed from the model state space. So, it inherently provides the ability to dynamically adapt to a smaller model and handles global decomposition of asymmetric knowledge.

While rule-based approaches can capture some of the desirable context modeling properties, the handling of these desirable properties in a rule-based formalism is primarily based on the deterministic contexts and it is not clear if the uncertainty extension of the rule-based theories may be able to efficiently handle all the desirable properties in context-based reasoning. Furthermore, it is well known that the rule-based approaches can handle only a limited amount of uncertain information. The extension for rule-based theories to deal with uncertainty, for instance, the certainty theory in MYCIN system [Shortliffe, 1976], suffers from the inability to perform bidirectional inference and other limitations.

Many approaches for context modeling are based on the "if-then" rule formalism and uncertainty in context representation have received modest attention, mainly due to the difficulty in handling these major challenges while dealing with uncertainty. In this work, we take a probabilistic view of context and our approach differs from the rule-based approaches.

3.4 Probabilistic Context and Contextual Independence

In our work, *context*, as defined by [Boutilier et al., 1996], refers to an assignment of values to a subset of random variables of a Bayesian network [Boutilier et al., 1996], called the context variables or attributes. This example may help to visualize a probabilistic context: In the popular game show Hollywood Squares, there are 9 panels each showing a category of questions. Behind each panel is a hidden question for that category. The participant can choose any category. However, the participant is unaware if the question in a category may turn out to be hard or easy. The game host asks the participant to choose the panel category. His chance of winning depends on whether the question for the category he chooses turns out to be hard or simple. The context attribute here is the question category and the context value is the specific category the participant chooses.

Context specification may induce contextual independence which corresponds to conditional independence (CI) that exists in a specific context. As per the classical definition, conditional independence in a BN holds at the parameter level and is value-independent. If X is conditionally independent of Y given Z, then $P(X|Y,Z) = P(X|Z)$ and it means that given any value of variable Z, the probability of all the values of variable X are not affected by any values of variable Y. This classical definition of CI is, however, too restrictive in capturing independence that exist for a specific value of a variable, for instance, $P(X|Y,Z=z) = P(X|Z=z)$. The consequence of this is that a BN cannot capture these finer structures that results from the contextual dependence of the variables.

Given an assignment of values to certain variables, the probabilistic distribution can exhibit further independence, called *contextual independence*. Two major types of *contextual independence* are reported in the literature: a) Context-specific indepen-

dence (CSI)[Boutilier et al., 1996]; and b) Contextual-weak independence (CWI) [Wong & Butz, 1999]. CSI defines independence that results due to specific values of the parent variables, for instance, $P(X|Y, Z = z) = P(X|Z = z)$. CWI, on the other hand, defines the independence induced by specific values of a consequent variable, for instance, some values of a consequent variable X are not influenced by values of Y such as $P(X = 1, 2|Y, Z = z) = P(X = 1, 2|Z = z)$. In this thesis, we focus on CSI. Formally, context-specific independence (CSI) is defined as follows [Boutilier et al., 1996]:

Definition 3.4.1 *Let* $\boldsymbol{C}$ *be a set of variables. Let a context be denoted by* $\boldsymbol{C{=}c}$ *and* $\boldsymbol{c}$ *refer to set of values of variables in* $\boldsymbol{C}$*. Let* $\boldsymbol{X}$*,* $\boldsymbol{Y}$*,* $\boldsymbol{Z}$ *and* $\boldsymbol{C}$ *be four disjoint sets of variables.* $\boldsymbol{X}$ *and* $\boldsymbol{Y}$ *are independent given* $\boldsymbol{Z}$ *in Context* $\boldsymbol{C{=}c}$ *if* $P(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{Y},\boldsymbol{C{=}c}) = P(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{C{=}c})$.

3.4.1 An Example

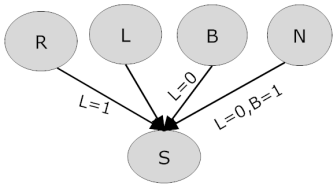


Figure 3.4: *Asymmetry in information leads to CSI*

Asymmetry in the information can lead to contextual independence in the probability distribution. In Figure 3.4, there are 2 context variables: L and B. The labels on the edges show the dependencies are context-specific and the distribution encodes contextual independence. Recall that a label on the edge means that the edge is valid only when the label assignment is true. For example, given $L = 1$, $P(S|R, L, B, N) = P(S|R, L = 1)$ as the consequent variable S is contextually independent from B and

N in context $L = 1$. Similarly, given $(L = 0, B = 0)$, $P(S|R, L, B, N) = P(S|L = 0, B = 0)$ and given $(L = 0, B = 1)$, $P(S|R, L, B, N) = P(S|N, L = 0, B = 1)$.

3.5 Context-based reasoning in Bayesian Networks

The BN framework is unable to effectively exploit context-sensitivity for compact knowledge representation; the main reasons are as follows:

- *BNs are propositional in nature*: A BN only models fixed random variables and fixed dependencies. Hence, BNs cannot effectively represent situations where the exact number of variables may not be known *a priori*, as in the relational BNs [Friedman et al., 1999; Heckerman et al., 2004]. BNs also cannot fully exploit the structural variations that arise with changes in specific context attributes or values. To represent such variations, a BN must capture all the potential context values in the CPTs and requires all the attributes in all the context values for each context variable to be modeled together in one large BN.

- *BNs are symmetric in nature*: As discussed earlier, a BN is too restrictive in representing the finer structure resulting from contextual dependence of the variables. Consequently, it has to enumerate over all assignments of all the parent variables in any context to define a CPT. This problem escalates in the relational BNs as the total number of parents can increase dramatically with an increase in the number of context variables and value assignments.

3.6 Related Work in the Bayesian Network Literature and their Limitations

Some previous approaches have been proposed to work around the inflexibility of BNs and to exploit contextual independence. As per the desiderata for context-based reasoning framework as presented in Section 3.2, related approaches can be divided based on their primary property of focus in the following categories: a) Multiple BNs; b) Local parameter decomposition; and c) Local context modeling and model adaptation.

Multiple Bayesian Networks:

Geiger & Heckerman [1996] proposed to use multiple BNs or Multinets to capture asymmetry and context-specific independences. Context is modeled as a global root node, i.e., each BN is conditioned on all values of the context variables. Each BN separately evaluates information relating to that particular context value. The advantage of the Multinets approach is that it can effectively encode the global model decomposition in a specific context. It is particularly useful in working with one context variable to compare alternative context hypotheses. However, given that the context is modeled as a global node, this approach does not scale well with the number of context attributes. For Example 2, the context set is {*accompany(B1), order(B1), accompany(B2), order(B2)*} and we need $2^4 = 16$ Multinets in total if all the context variables are assumed to be binary. Therefore, there is an exponential increase in the number of BNs required with the number of context variables. Furthermore, global context modeling is hard to maintain, i.e., contextual dependencies due to any other variables apart from the predetermined context set cannot be modeled easily and require modifications to each of the multiple BNs.

Local Parameter Decomposition:

Local parameter decomposition approaches target parameter reduction in the CPT for efficient inference. The standard tabular data structure of CPT is replaced with a tree/rule data structure in a full BN graph representation. The proposed approaches have shown how the existing inference algorithms can be modified to utilize a special data structure in CPT. For example, [Poole, 1998, 1997; Poole & Zhang, 2003] proposed an extended version of Variable Elimination algorithm to represent contexts in terms of rules and partial functions. Similarly, Boutilier et al. [1996] proposed strategies for cut-set conditioning on the tree representation for CPT. However, the gain in inferential tractability afforded by these representations of conditional probability is difficult to quantify [Jaeger, 2004]. Manipulating contexts using rules in Variable Elimination may sometimes incur substantial computational overhead due to an operation called splitting. Also, cut-set conditioning using tree CPTs has not been shown empirically to be more efficient. Another major limitation of the existing structured CPT approaches is that the junction tree algorithm, one of the most influential inference algorithms in the BN literature, is known to be invariant to the local structures in the CPT.

Boutilier et al. [1996] proposed another method, network transformation, for representing local probability decomposition on graph to utilize the standard junction tree BN algorithm. This approach, however, does not guarantee efficient inference. For many problems, the inference has the same space complexity and a more complex graph structure. For example, the transformed graph in Example 2 contains the same number of parents for *rating* as well as for *order*, but the augmented graph is more complex than the original graph.

Some non-structural approaches have also been proposed. D'Ambrosio [1995, 1994] proposed an algebraic local expression language that incorporates a costly heuristics

to perform inference. Jaeger [2004] proposed a framework of probabilistic decision graphs based on ordered binary decision diagrams. Darwiche [2002] and [Chavira & Darwiche, 2005] proposed to represent the joint distribution in a BN as a multi-linear polynomial and then converted it into an arithmetic circuit for inference. Their proposals have shown success in online computation time because the arithmetic circuits, like algebraic decision diagrams, can cache up similar calculations even over the subtrees. The algorithm, however, is exponential in space requirements.

Other related work that utilize local parameter decomposition for representation include the use of contingent BNs in BLOG [Milch et al., 2004] for defining infinite BNs, the use of reference uncertainty [Friedman et al., 1999] and identity uncertainty [Pasula et al., 2002] in probabilistic relational languages, and the dynamic situation modeling using frames [D'Ambrosio et al., 1999].

Context-sensitive information, however, may induce systematic structure decomposition of the BN graph and not just local parameter decomposition of a variable in the BN. Inference efficiency can be improved with effective manipulation of context-specific graph structures.

Local Context Modeling and Model adaptation:

Some knowledge-based model construction (KBMC) languages allow model construction based on explicit local context. For example, the Network Fragments [Laskey & Mahoney, 1997] approach is used to construct a knowledge base using independent smaller networks. Situation-specific BN [Mahoney & Laskey, 1998] builds upon the network fragments approach and models context as an observed variable. Using this approach, a context-specific BN is constructed. Ngo et al. [1995] developed a logic-based framework to capture the local knowledge using designated deterministic contexts. Given the context, they generated a subnetwork which is sufficient for the computation of a given query. Context in these approaches is not only mod-

eled as a deterministic non-random variable, but also once the BN is constructed, the frameworks are not further adaptive to exploit contextually relevant knowledge with incremental context observations. Jaeger [2001] built on [Ngo et al., 1995]'s approach to achieve such value-sensitive network reduction recursively in a Relational Bayesian network. They added auxiliary variables and used network transformation concepts from [Boutilier et al., 1996]. As explained earlier, the network transformation approach may result in a large transformed graph with increase in the number of context and their values. Also, the network transformation approach typically assumes that the exact context-specific independence structure is known *a priori* and a deterministic CPT structure can be specified beforehand. This may not be true in the problem of situational variation as discussed in Chapter 2.

Table 3.2 summarizes the discussion in this section:

Representation	Context uncertainty	Scalable	Adaptable	Inference efficiency
Multiple BNs	yes	no	partial	not shown
Parameter decomposition	yes	yes	limited to parameters	limited success
Local Context	no	yes	partial	not shown

Table 3.2: *Comparison of Related Work*

While the three main approaches discussed address different challenges in context modeling, unlike rule-based approach, there is a lack of a single framework in probabilistic reasoning that can effectively address all the earlier mentioned challenges. For instance, neither propositional BN nor relational BN approaches described above can be used directly to capture the situational variations as shown in Figures 2.1 (page 17), 2.4, and 2.5 (page 22) as one single graphical model. The majority of BN inference algorithms fall in this category as they need complete probability factors to work rather than the context-specific partitions of a probability factor. Further-

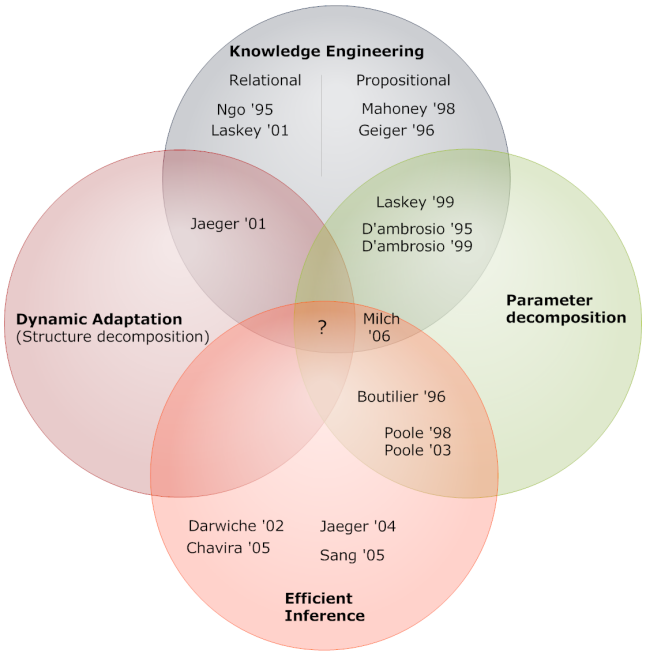


Figure 3.5: *Categorization of related work based on the primary properties of focus as per the context-based reasoning framework desiderata*

more, those that utilize parameter decomposition do not take asymmetric advantage of graph structure decomposition to exploit context-specific graph structures. The previous parameter decomposition approaches have assumed that the structure in the CPT table is fixed, i.e., the number of context, their possible values and the resulting contextual independence are known *a priori*. However, in relational context modeling, this assumption is not true. Extending these representations to dynamically change the CPT structure may not be an effective strategy as the CPT manipulations are much more expensive. An effective framework needs to combine the relational knowledge engineering advantages with structural decomposition and parameter decomposition for both representation and inference.

3.7 Summary

In this chapter, we discussed context modeling approaches in various fields, described the pros and cons of the rule-based systems approach, introduced the notion of context for probabilistic reasoning, and reviewed the capabilities and limitations of previous work in the BN literature. This chapter added a few more challenges to those as explained in Chapter 2. The complete list of challenges in context modeling, therefore, include the following:

- supporting situational variations in relational modeling
- dealing with uncertainty in context
- handling dependency among contexts
- utilizing context-induced asymmetry for efficient inference
- providing dynamic model adaptation based on context values
- handling scalability with the number of context

A unifying framework that addresses these challenges may not only simplify representing context-based knowledge but may also provide new insights for efficient inference.

4

Context-Sensitive Network

Based on the survey in Chapter 3, we formulated the following challenges: a) What will be an effective framework to reason with contextual information under uncertainty; and b) How do we exploit contextual independence that naturally exists in several domains to improve the transparency of model representation and flexibility as well as adaptability of the model. In this chapter, we present a new probabilistic graphical framework, Context-Sensitive Network (CSN), to address some of these challenges in context modeling under uncertainty. We will describe the foundations, semantics, theoretical properties and advantages of our proposed framework.

4.1 Context Definition

Overall, our definition of our notion of context is reflected in the etymology of word "context." "Context" has been derived from the latin word "contexere" which means

to weave together. In our work, *context* refers to a meaningful attribute or set of attributes that helps in weaving different situational graphs together. Context attributes/variables are the parents of the target variables for which they form the context. Context variables can influence causally, e.g., ($familyOut$) in Example 1, as well as non-causally, e.g., ($belongsTo$, $barking$) in Example 1, in ; they define the conditions to activate the situations and the information that holds in different situations. Example context variables may include any attributes that denote, for instance, profile, environment, location as well as object type and reference. Each context random variable has a domain space. The assignment of values to the context variables in their domain spaces forms a specific context assignment.[1]

Context variables can facilitate modeling in scenarios that involve the following properties:

- *Irrelevant attributes and sub-models:* e.g., the *Age* context in Section 3.2.1 in Chapter 3 renders family history attributes irrelevant. Similarly, if the patient is male, then the sub-model depicting complications related to pregnancy in a general diabetes management model becomes irrelevant.
- *Attribute hierarchy:* e.g., location and radiation of pain are contingent upon the type of chest pain.
- *Situational attributes:* e.g., 5 "W" categories of a situation as described in Table 3.1.
- *Uncertainty over object type and reference:* e.g., attributes *belongsTo*, *barking* in Example 1.
- *Selector attribute:* e.g., attribute *barking* in Example 1 is a selector for associating $hearBark$ with $dogOut(d)$ of a particular dog.

Context awareness, i.e., specific context assignments being known, sheds light on the

[1]I use context, context values, and context assignments interchangeably in my thesis

essential information that needs to be considered for the reasoning task. In Example 1, *order(B1)* and *order(B2)* are two context random variables whose awareness sheds light on what variables, for instance *quality(FT1)* or *quality(FT2)*, are relevant to the given problem.

4.2 Representation Framework

In the rest of this chapter, we present a new representation framework for reasoning using context-specific probability partitions. We propose a special graph representation, called Context-Sensitive Network (CSN), as a conceptual abstraction that may facilitate understanding and formulating a common basis related to context-based reasoning in various domains. The CSN utilizes contextual independence instead of conditional independence for compactly representing a conditional probability distribution. The capturing of contextual independence relationship can facilitate: a) Knowledge representation, b) Parameter elicitation, and c) Inference. In contrast to the traditional BN representation where the quantitative parameters are hidden and not represented on the graph, our representation explicitly represents the quantitative functions on the graph that change with context. The intuition is that the required function changes with the context and the explicit representation may facilitate understanding of the specific functions used in the current context. This representational change may help to exploit contextual dependence of the probability distributions and enhance the representational power that we seek.

Let us first convey an intuition about our representation using simple algebra. Assume that a function can be written as $f = (a^x.b^{1-x})$, where a, b are two parameters, and *if x=1, f=a else b*. Here, the selection between a or b is dependent on the value of x. We use a similar idea to represent contextual independence in the probability

distribution. The distribution $f(X|\mathbf{Pa}(X))$ exhibiting contextual independence with respect to context variable C with value c1 and c2 can be written as a product of $f(X|\mathbf{Pa}_1(X))^{C=c1} f(X|\mathbf{Pa}_2(X))^{C=c2}$, where $\mathbf{Pa}$(X)=$\mathbf{Pa}_1(X) \cup \mathbf{Pa}_2(X) \cup C$. Given this intuition, we view a full CPT of a variable as a set of disjoint partitions conditional on specific contexts. We call these partitions Conditional Part Factors (CPFs). They encode parameters representing the partitions of the conditional probability distribution of the consequent variable, given different value configurations or assignments of its context-specific parents. Each context or context assignment is encoded as an Indicator function.

We now define CPFs for capturing the contextual information. Then, we present the CSN framework for representing and reasoning with contextual information captured in terms of the CPFs.

4.3 Conditional Part Factors

Conditional Part Factors (CPFs) represent the partition of a CPT in a particular context (i.e., a particular assignment of values of the context variables). Given contextual independence, the conditional probabilistic distribution $P(X|\mathbf{Pa}(X))$ for a random variable X given its parents $\mathbf{Pa}(X)$ is equal to the product of a disjoint set of partitions, i.e., $P(X|\boldsymbol{Pa}(X)) = P(X|\mathbf{csPa}_1(X))^{C=c_1} \ . \ P(X|\mathbf{csPa}_2(X))^{C=c_2} \ . \ P(X|\mathbf{csPa}_n(X))^{C=c_n}$, where $\mathbf{Pa}(X) = \mathbf{csPa}_1(X) \cup \ldots \mathbf{csPa}_n(X)$ and $c_1 \ldots c_n$ are n domain values of context variable C. $P(X|\mathbf{csPa}_1(X))$ is said to be an active partition of the conditional probabilistic distribution or function $P(X|\mathbf{Pa}(X))$ in the context $C = c_1$.

A CPF is denoted as $\Phi(A)^{I(\mathbf{C=c})} = \lambda(A, \mathbf{c})$ where A is a random variable and $\mathbf{C}$ is a set of context variables, also called the context set of the CPF and $\lambda(A)$ represents a

factor with all the possible value assignments of random variables in A with respect to **C=c**. The *scope with context* is A ∪ **C**. The CPF $\Phi(A|\mathbf{Pa}(A))^{I(\mathbf{C}=\mathbf{c})}$ denotes conditional factor with consequent variable A and parent set **Pa**(A) in context **C=c**.

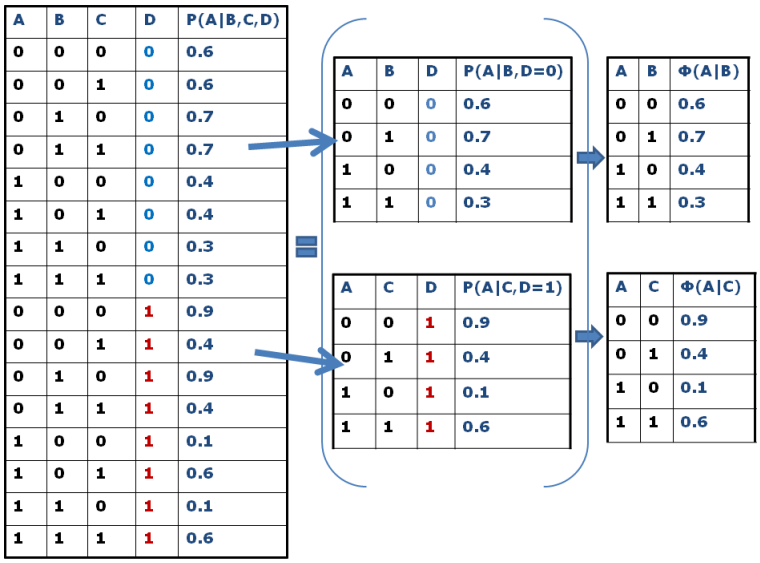


Figure 4.1: *Understanding CPFs: distribution P(A|B,C,D) exhibits contextual independence with context attribute D, i.e., given D=0, P(A|B,C,D=0)=P(A|B,D=0) and given D=1, P(A|B,C,D=1)=P(A|C,D=1). So, P(A|B,C,D) can be represented as a factor product (shown as × in figure) of two more compact partitions P(A|B,D=0) and P(A|C,D=1). P(A|B,D=0) = CPF:* $\Phi(A|B)^{I(D=0)}$ *and P(A|C,D=1) = CPF:* $\Phi(A|C)^{I(D=1)}$

Formally, we define CPF as

Definition 4.3.1 *Let X be the consequent variable, $\boldsymbol{Pa}(X)$ be the parent set of X. Let $\boldsymbol{C}(X) \subseteq \boldsymbol{Pa}(X)$ be the set of context parent variables and $\boldsymbol{Y} = \boldsymbol{Pa}(X) \setminus \boldsymbol{C}(X)$ be the set of non-context parent variables of X.* $\Phi(X|\boldsymbol{Z})^{I(\mathbf{C}(X)=\mathbf{c})}$ *is said to be a* conditional part factor *in context $\boldsymbol{C}(X)=\boldsymbol{c}$ if $\boldsymbol{Z} \subseteq \boldsymbol{Y}$ and $\boldsymbol{Z}$ represents only those variables in $\boldsymbol{Y}$ that are parents of X in context $\boldsymbol{C}(X)=\boldsymbol{c}$ such that $P(X|\boldsymbol{Y}, \boldsymbol{C}(X)=\boldsymbol{c}) = P(X|\boldsymbol{Z}, \boldsymbol{C}(X)=\boldsymbol{c})$. The scope of the CPF is the set $\{X, \boldsymbol{Z}, \boldsymbol{C}(X)\}$ and the scope*

without context of the CPF is the set $\{X, \boldsymbol{Z}\}$[2].

Figure 4.1 explains CPFs numerically. In Figure 4.2, Φ*(rating(B1)|quality(FT1))*$^{I(order(B1)=FT1)}$ represents the probability partition *P(rating(B1)|quality(FT1), order(B1)=FT1)*. Table 4.1 shows the CPFs for the variables in Example 2. For the well formed CPFs, the consequent variable must be in the scope and should be a child, for example, Φ*(quality(FT1), quality(FT2), order(B1))* is not a conditional part factor. The factors are compact as they utilize contextual independence, for example, *P(rating(B1) | quality(FT1), quality(FT2), order(B1)=FT1) = P(rating(B1) | quality(FT1), order(B1)=FT1)*. The switching between these CPFs depends on values of the context variables.

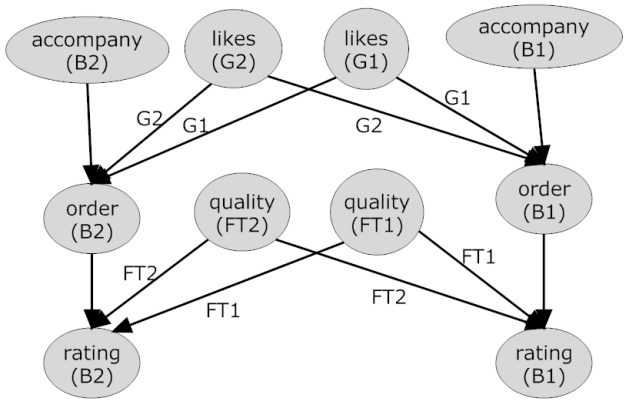


Figure 4.2: *Instantiated BN with 2 boys, girls and food types as described in Table 2.2. Labels on the edges denote the context (s) in which the consequent variable is dependent on the parent(s).*

Any full CPT exhibiting contextual independence such as *P(rating(B1) | quality(FT1), quality(FT2), order(B1))* can be divided into a factor product of CPFs, for example, *P(rating(B1) | quality(FT1))*$^{order(B1)=FT1}$. *P(rating(B1) |quality(FT2))*$^{order(B1)=FT2}$. CPFs that are valid in all possible contexts are represented with empty context infor-

[2]This definition can be generalized to set of consequent variables $\mathbf{X}$ too

Variable	CPFs
$order(B1)$	$\Phi(order(B1)\|likes(G1))^{I(accompany(B1)=G1)}$
	$\Phi(order(B1)\|likes(G2))^{I(accompany(B1)=G2))}$
$rating(B1)$	$\Phi(rating(B1)\|quality(FT1))^{I(order(B1)=FT1)}$
	$\Phi(rating(B1)\|quality(FT2))^{I(order(B1)=FT2)}$
$order(B2)$	$\Phi(order(B2)\|likes(G1))^{I(accompany(B2)=G1)}$
	$\Phi(order(B2)\|likes(G2))^{I(accompany(B2)=G2))}$
$rating(B2)$	$\Phi(rating(B2)\|quality(FT1))^{I(order(B2)=FT1)}$
	$\Phi(rating(B2)\|quality(FT2))^{I(order(B2)=FT2)}$

Table 4.1: *Example of CPFs*

mation. If all the parents of the consequent variable are valid in all contexts i.e., no contextual independence in the local probability distribution, then the CPF is same as the CPT in BN framework.

Probability distribution in CPF is represented using standard tabular data structure. Just like CPT, CPF representation can utilize other local parametric representations such as noisy-OR for capturing independence in a specific context. Furthermore, CPF can capture both types of contextual independences: CSI and CWI, although this thesis primarily focuses on CSI. Representations such as tree CPT can also be represented in a CSN. A complete tree CPT can be written as a product of CPFs each with context assignment or value as the path from the root of the tree to the leaves.

The CPFs can further be viewed as general mathematical functions, supporting useful mathematical operations. The following operations are implicitly used for adaptation and inference defined later:

- Φ^0=1 (context is false), Φ^1= Φ (context is true); Φ*1= Φ
- $\Phi^x = \Phi^{x.y}$ if y=1
- $\Phi_1^x \ . \ \Phi_2^y = \Phi_1^x$ if y=0;

$= \Phi_2^y$ if x=0

The indicators are always binary as CPFs are either relevant or irrelevant in a specific context.

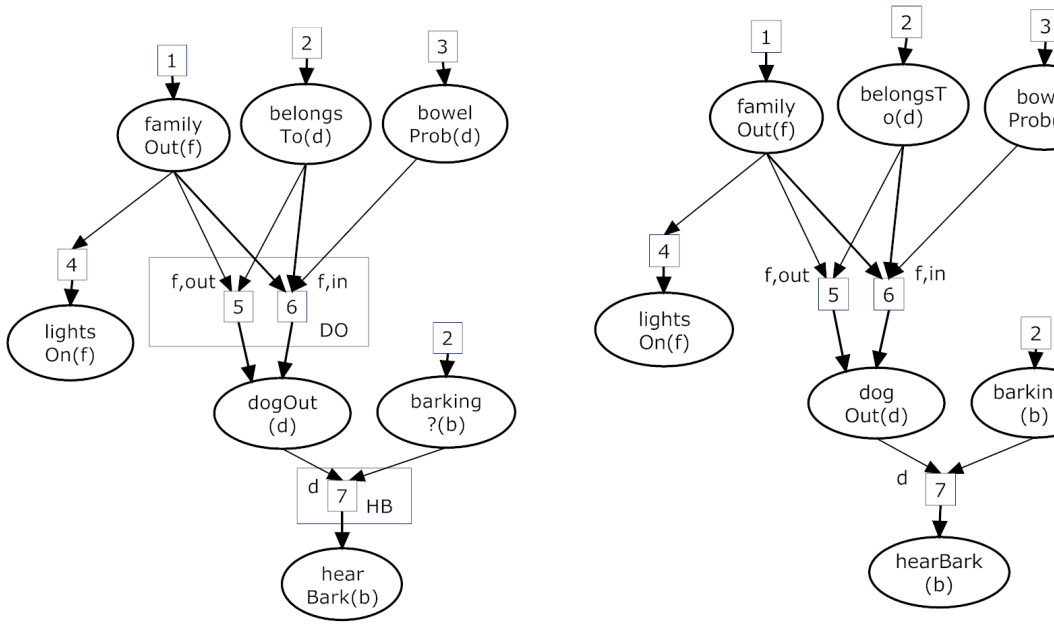
4.4 Context-Sensitive Network

A Context-Sensitive Network (CSN), an extension of the Bayesian network, consists of three types of nodes: a) *Variable nodes*, b) *Context function nodes*, and c) *Function nodes*. Each random attribute, which may or may not be a context attribute in the domain is represented as a variable node. A context function node is a qualitative abstraction of a mathematical function that holds in a particular context and is associated with a probabilistic function. Each context function node is associated with a context label to indicate the context in which the function is true. A function node represents a collection of all CPFs for a consequent variable. The edges into the context function nodes denote the contextual and/or conditional probabilistic relationships among the attributes or variables in the network.

A CSN is a directed bipartite graph that is normalized and expresses the structure of the factorization resulting from contextual independence. The complete probability distribution of the graph in a CSN can be written as

$$\prod_{i=1}^{k} CPF_i = \prod_{i=1}^{n} \prod_{j=1}^{m} P(X_i|\mathbf{csPa}(X_i))^{I(\mathbf{C}(X_i)=\mathbf{c}_j)}$$

where k is the total number of CPFs, n is the number of variable nodes, m denotes the number of CPFs associated with each function node, $\mathbf{csPa}(X_i)$ denotes the context-specific Parent set of X_i in context assignment $\mathbf{c}_j$ and $\mathbf{C}(X_i)$ is the context parent set of X_i. $\mathbf{csPa}(X_i)$ and $\mathbf{C}(X_i)$ are disjoint subsets of $\mathbf{Pa}(X_i)$, the parent set. Formally,



(a) CSN with function nodes **(b)** At contextual level

Figure 4.3: *Context-Sensitive network for Example 1 with context information. Node 5 with context (f,out) encodes the probabilistic logic rule ∀ d,f, P(dogOut(d) | belongsTo(d)=f, familyOut(f)=out). Node 6 with context (f,in) encodes ∀ d,f, P(dogOut(d) | bowelProb(d), belongsTo(d)=f, familyOut(f)=in)*

Definition 4.4.1 *A Context-Sensitive Network is a directed acyclic graph representing factorization of the global joint probability distribution of n random variables into a product of local conditional part factors. A CSN composes of:*

- *a set of random variable nodes* ***V****, each represent a relevant attribute in the problem domain;*
- *a set of context function nodes* ***CF****, each indicating a CPF;*
- *a first set of edges, each connecting a variable node V_i to a context function node CF_i if and only if V_i is in the argument of the CPF associated with CF_i*

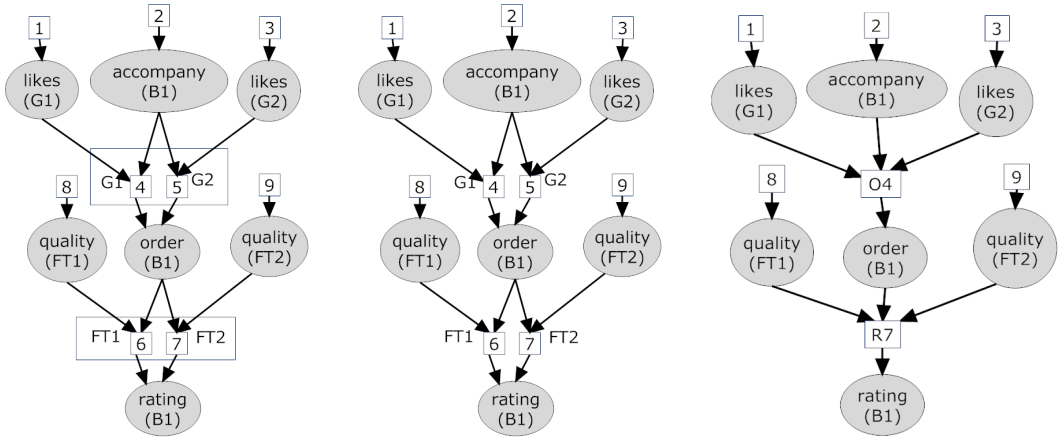
---

and is a parent;

- *a second set of edges, each connecting a context function node CF_i to a variable node V_i if V_i is the consequent variable in the CPF associated with CF_i.*
- *a set of function node, each of which is a collection of context function nodes associated with a variable. A function node F_i for each variable V_i represents a collection of all CPFs for V_i, and is equivalent to the CPT in a BN framework.*

Graphically, a CSN consists of a set of variable nodes (shown as ovals), context function nodes (small rectangles) and function nodes (big rectangles with small rectangles inside). In Figure 4.3(a), context function node 5 and 6 with context $\{belongsTo = f, familyOut(f) = out\}$, and $\{belongsTo = f, familyOut(f) = in\}$ are associated with CPFs $\Phi(dogOut(d))^{I(belongsTo=f,familyOut(f)=out)}$ and $\Phi(dogOut(d)|bowelProb)^{I(belongsTo=f,familyOut(f)=in)}$ respectively, where f is a parameter denoting a particular family. In Figure 4.3(a), nodes 5 and 6 are associated with function node DO representing the product of the CPFs with the *dogOut* consequent variable. Similarly, in Figure 4.4(a), context function nodes $4, 5$ are associated with $\Phi(order(B1)|likes(G1))^{I(accompany(B1)=G1)}$, $\Phi(order(B1)|likes(G2))^{I(accompany(B1)=G2)}$ respectively. In Figure 4.4(a), the context function node 4 has a non-context node $likes(G1)$ and context node $accompany(B1)$ as its parents. Node 4 reflects the context $accompany(B1) = G1$ and node 5 reflects the context $accompany(B1) = G2$.

Figure 4.4 shows different perspectives of a CSN: a) complete CSN, b) without function nodes (contextual level), c) with only function nodes (functional level). Function node $O4$ represents node 4 and 5 that are associated with the variable $order(B1)$. The concept of a function node facilitates understanding of the related concepts and shows the similarity to BN representation. All context function nodes pointing to the same consequent variable node are a part of one function node.



(a) With function nodes **(b)** At contextual level **(c)** At functional level

Figure 4.4: *Different perspectives of CSN (Example 2)*

4.4.1 Well-formed CSN

A CSN $(\mathbf{G}, \mathbf{\Theta}, \mathbf{C})$ is said to be well-formed if

1. $\mathbf{G} = (\mathbf{N}, \mathbf{E})$ is a bipartite DAG, where $\mathbf{N}$ is a finite set of nodes and $\mathbf{E}$ is set of edges. A node can be a variable node, context function node or function node.

2. Each variable node V is associated with a random variable having $\mathbf{d}$ ($\mathbf{d} \geq 2$) mutually exclusive domain values.

3. Each context function node CF is associated with a CPF with a probability distribution and a particular context assignment c.

4. A function node F is associated with each variable and represents all the context function nodes that are the parents of a variable node and is equivalent to the full CPT in the corresponding BN representation.

5. $\mathbf{E}_{VCF}$ is a set of directed edges that connect variable nodes in $\mathbf{V}$ to the context function nodes in $\mathbf{CF}$; an edge $e_{ij} \in \mathbf{E}_{VCF}$ if and only if a variable node V_i is in the argument of the CPF associated with the context function node CF_j and is a parent of CF_j. $\mathbf{E}_{CFV}$ is a set of directed edges that connect context function nodes in $\mathbf{CF}$ to variable nodes in $\mathbf{V}$; an edge $e_{jk} \in \mathbf{E}_{CFV}$ if and only if a variable node V_k is the child of context function node CF_j. There is only one edge from a context function node to a variable node, i.e., each context function node is connected to one consequent variable.

6. A CPF represents a valid probability distribution and $\mathbf{\Theta}$ represents the set of CPFs and the product of which defines the joint probability distribution of the domain.

7. $\mathbf{C}$ represents the set of context variables for context function nodes. The context assignments or values $\mathbf{c}$ for all the context function nodes associated with a function node F are assumed to be mutually exclusive and covering.

4.5 CSN: Properties

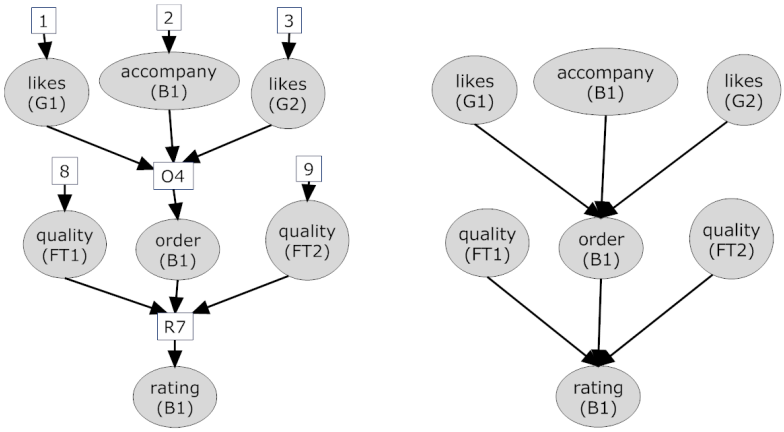
In the last section, we have introduced the CSN representation. This section introduces some of its properties.

Property 1: *The CSN at the functional level is equivalent to the corresponding BN.*

Proof: The BN represents the factorization $\prod_{i=1}^{m} P(V_i|\mathbf{Pa}(V_i))$, where m is the number of random variables in the graph. By definition, each variable in the CSN representation is associated with only one function node. Each function node associated with a variable V represents a collection of CPFs associated with that variable. Due to the contextual independence property, the product of these CPFs is equivalent

to the CPT of that consequent variable V in the BN. Henceforth, each function node is associated with the probability distribution equivalent to the CPT. The function node is equivalent to an explicit graphical representation of the CPT of the corresponding BN. Hence, the CSN at the functional level represents the factorization $\prod_{i=1}^{m} P(V_i|\mathbf{Pa}(V_i))$. ■

For example, CSN at functional level of Example 2 as shown in Figure 4.5(a) is equivalent to the corresponding BN shown in Figure 4.5(b).



(a) CSN at functional level **(b)** BN

Figure 4.5: *CSN at functional level is equivalent to the BN*

Property 2: *The complete joint probability distribution of a CSN is equal to the product of local CPFs:* $\prod_{i=1}^{n} CPF_i$ *where* n *is the total number of context function nodes in the CSN.*

Proof: Using Property 1, the graph at the functional level represents the factorization $\prod_{i=1}^{m} P(V_i|\mathbf{Pa}(V_i))$. Each function node represents the factorization resulting due to contextual independence property. The probability distribution associated with a function node is equivalent to the product of CPFs associated with that node. Assume

there are $|\mathbf{c}| = k_i$ number of mutually exclusive and covering context assignments for the context set $\mathbf{C}(V_i)$ of variable V_i with $\mathbf{csPa}_1 \ldots \mathbf{csPa}_{k_i}$ context-specific parent sets respectively, then

$$\begin{aligned}
\prod_{i=1}^{m} P(V_i|\mathbf{Pa}(V_i)) &= \prod_{i=1}^{m} P(V_i|\mathbf{csPa}_1(V_i), \mathbf{C}(V_i) = \mathbf{c}_1) \ldots . P(V_i|\mathbf{csPa}_{k_i}(V_i), \mathbf{C}(V_i) = \mathbf{c}_{k_i}) \\
&= \prod_{i=1}^{m} \Phi(V_i|\mathbf{csPa}_1(V_i))^{I(\mathbf{C}(V_i)=\mathbf{c}_1)} \ldots . \Phi(V_i|\mathbf{csPa}_{k_i}(V_i))^{I(\mathbf{C}(V_i)=\mathbf{c}_{k_i})} \\
&= \prod_{i=1}^{m} \prod_{j=1}^{k_i} \Phi(V_i|\mathbf{csPa}_j(V_i))^{I(\mathbf{C}(V_i)=\mathbf{c}_j)} \\
&= \prod_{i=1}^{m} \prod_{j=1}^{k_i} CPF_{ij} = \prod_{l=1}^{n} CPF_l
\end{aligned}$$

Assume that there are only n CPFs, then the complete joint probability distribution is equivalent to the product of local CPFs. ■

Property 3: *All BNs can be converted to CSNs and vice versa.*

Proof: Each CPT in a BN can be written as a product of its individual rows with context assignment equal to the particular assignment of the parent variables. Each row represents a CPF with only the consequent variable in scope and context attribute set equal to all the parents of a variable. Hence, each CPT can be converted to the product of CPFs, equivalent to the number of domain values of the parent variables. Lets assume that there are $|\mathbf{pa}(V_i)| = k$ total number of mutually exclusive and covering assignments of the parent variable set of a variable V_i , then

$$\begin{aligned}
P(V_i|\mathbf{Pa}(V_i)) &= \text{Product of its individual rows} \\
&= P(V_i|\mathbf{C}(V_i) = \mathbf{c}_1) \ldots . P(V_i|\mathbf{C}(V_i) = \mathbf{c}_k) \ [here\ \mathbf{Pa}(V_i) = \mathbf{C}(V_i)] \\
&= \Phi(V_i)^{I(\mathbf{C}(V_i)=\mathbf{c}_1)} \ldots . \Phi(V_i)^{I(\mathbf{C}(V_i)=\mathbf{c}_k)}
\end{aligned}$$

Hence, all BNs can be represented as a CSN. Similarly, all CPFs can be multiplied and combined to form a CPT of the BN and the CSN can in turn be converted into a BN. ■

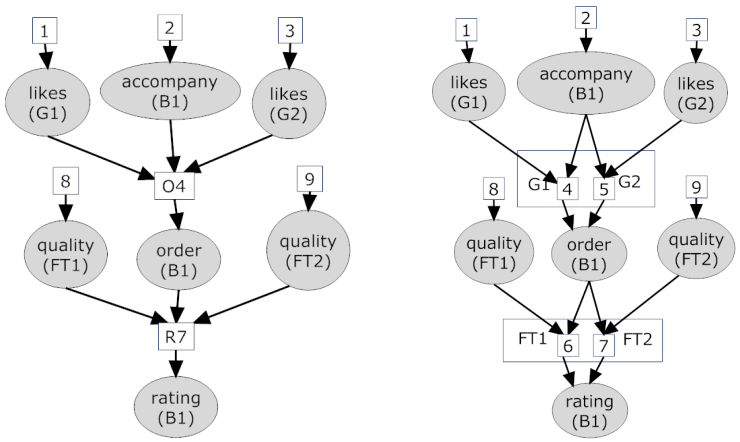
Property 4: *Local consistency in a CPF ensures global consistency in the joint probability distribution.*

Proof: Using Property 3, the local consistency in a CPF implies each row in the corresponding CPT in a BN is consistent. Hence, it ensures global consistency. ■

CSNs can be viewed as a special subclass of the recently proposed probabilistic graphical models - Directed Factor Graphs (DFGs) [Frey, 2003], which are in turn generalizations of BNs. One of the advantages of CSN is that they can admit d-separation, i.e., if X and Y are d-separated in a CSN with evidence e entered, then $P(X|Y,e) = P(X|e)$. We extend the d-separation property for DFGs [Frey, 2003] for deducing contextual independence in CSN:

Property 5: *d-separation property: A path is said to be blocked if one or more of the following conditions are satisfied: a) One of the variables in the path is observed (similar to the diverging and the linear conditions in BN); b) One of the function nodes in the path has two incoming edges that are part of the path, and neither the consequent variable associated with function node nor any of its variable descendants are observed (similar to the converging condition in BN); c) The context attribute set associated with the context function node in the path is observed and context value of the context function node is not the same as the observed value (Contextual independence property).*

The first two cases are equivalent to the d-separation conditions for conditional independence in BN [Pearl, 1988]. If path at the functional level is blocked, it is also blocked at the context function level. The third case implies that influence can only



(a) CSN at functional level **(b)** CSN

Figure 4.6: *Understanding d-separation in CSN: if the context attribute is observed but the context value of the context function node is different from the observed value, then the influence path is blocked through that node*

flow through a particular context function node if the value of the observed context variable associated with it is in accordance with the observed context values or evidence.

In Figure 4.6(a), *likes(G2)* is not d-separated from *likes(G1)* given *rating(B1)* but the two nodes are d-separated given *rating(B1)* and *accompany(B1)*. Note, this notion of d-separation in CSN is different from that of a BN. In a BN, the observation of *accompany(B1)* does not affect the dependence between *likes(G2)* and *likes(G1)* given *rating(B1)* as the BN cannot d-separate attributes based on contextual independence. In Figure 4.6(a) (CSN at functional level), there is one path from *likes(G2)* to *likes(G1)*: *1-likes(G1)-O4-likes(G2)* at the functional level and two associated paths in Figure 4.6(b):a) *1-likes(G1)-4-accompany(B1)-5-likes(G2)*; b) *1-likes(G1)-4-order(B1)-5-likes(G2)* at the context functional level. Path *1-likes(G1)-4-likes(G2)* is not blocked if only *rating(B1)* is observed as the influence can flow through both the associated paths a) and b); however it is blocked if *accompany(B1)* is observed as

both its associated path a) and b) at context functional level are blocked. Path a) is blocked because variable *accompany(B1)* is observed. Path b) is blocked because context function nodes 4 and 5 have mutually exclusive context values of *accompany(B1)* at the context functional level. So, either context function node 4 or 5 but not both is relevant given *accompany(B1)*.

Similarly, *likes(G2)* and *likes(G1)* are d-separated given *order(B1)* and *accompany(B1)*; *quality(FT2)* and *quality(FT1)* are d-separated given *rating(B1)* and *order(B1)*; *likes(G2)* is d-separated from *quality(FT2)* given *rating(B1)* and *accompany(B1)=G1*. In the later, the path from *likes(G2)* to *quality(FT2)* in Figure 4.6(b) is blocked given *rating(B1)* and *accompany(B1)=G1* as node 5 in 4.6(b) has a different context value.

4.6 Summary

To address the context modeling challenges mentioned earlier, we have proposed a new asymmetric probabilistic language, CSN, which is based on the notion of contextual independence, and explained the theoretical foundations and the language properties. In the following chapters, we would show that CSN provides the following advantages: The language a) allows uncertainty in context; b) facilitates local context modeling for knowledge engineering; c) captures asymmetric information and local decomposition of conditional probability; d) handles situation-specific inference; and e) supports dynamic model adaptation to exploit structural variations in both the parameters and the graph structure.

5

Inference

This chapter describes how to perform inference in a CSN by exploiting the partitions of the CPTs. We propose a message passing loopy belief propagation algorithm for CSN. We will extend the belief propagation algorithm and describe three new operations for message passing: Context-sensitive Factor Product, Context-node Marginalization and Context Marginalization to work with partitions of probabilistic functions. We will also show that the overall inference computations can be visualized graphically on a CSN.

5.1 Preliminary: Algebra

As discussed in Chapter 2 Section 2.3, a factor $\lambda(A, B)$ over two random variables A and B is a probability distribution table over all assignments $(\mathbf{a}, \mathbf{b})$ of A, B. In this section, we introduce some preliminary operations and concepts supporting calcula-

tions in a message passing inference algorithm for the CSN.

Factor product

Let A and B be two random variables and $\lambda_1(A,B)$ and $\lambda_2(A,B)$ be two factors, then the factor product $\lambda_1.\lambda_2$ is a factor $\lambda : \lambda(a,b) = \lambda_1(a,b) \cdot \lambda_2(a,b)$ for each assignment (a,b) in A,B. An example of factor product is shown below:

$$\begin{bmatrix} states & prob \\ a1,b1 & \theta_{a1|b1} = 0.7 \\ a1,b2 & \theta_{a1|b2} = 0.6 \\ a2,b1 & \theta_{a2|b1} = 0.3 \\ a2,b2 & \theta_{a2|b2} = 0.4 \end{bmatrix} \cdot \begin{bmatrix} states & prob \\ a1,b1 & \theta'_{a1|b1} = 0.8 \\ a1,b2 & \theta'_{a1|b2} = 0.1 \\ a2,b1 & \theta'_{a2|b1} = 0.2 \\ a2,b2 & \theta'_{a2|b2} = 0.9 \end{bmatrix} = \begin{bmatrix} states & prob \\ a1,b1 & \theta_{a1|b1}\theta'_{a1|b1} = 0.56 \\ a1,b2 & \theta_{a1|b2}\theta'_{a1|b2} = 0.06 \\ a2,b1 & \theta_{a2|b1}\theta'_{a2|b1} = 0.06 \\ a2,b2 & \theta_{a2|b2}\theta'_{a2|b2} = 0.36 \end{bmatrix}$$

Similarly, let $\lambda_1(A,B)$ and $\lambda_2(B,C)$ be two factors with different scope, then the factor product $\lambda_1 \cdot \lambda_2$ is a factor $\lambda : \lambda(a,b,c) = \lambda_1(a,b) \cdot \lambda_2(b,c)$ for each assignment (a,b,c) in A,B,C. The scope(λ) is scope(λ_1)$\cup$scope(λ_2).

Factor Marginalization

Let **A** and **B** be two disjoint sets of variables and λ(**A,B**) be a factor. Factor marginalization of **B** in λ denoted as ($\sum$) is a factor λ over **A** such that

$$\lambda(\mathbf{A}) = \sum_{\mathbf{B}} \lambda(\mathbf{A},\mathbf{B})$$

Factor Marginalization of B sums over all the assignments of **b**'s for each assignment

a in A and yields:

$$\sum_{B} \begin{bmatrix} states & prob \\ a1, b1 & \theta_{a1,b1} = 0.24 \\ a1, b2 & \theta_{a1,b2} = 0.42 \\ a2, b1 & \theta_{a2,b1} = 0.16 \\ a2, b2 & \theta_{a2,b2} = 0.18 \end{bmatrix} = \begin{bmatrix} states & prob \\ a1 & \theta_{a1} = \theta_{a1,b1} + \theta_{a1,b2} = 0.24 + 0.42 = 0.66 \\ a2 & \theta_{a2} = \theta_{a2,b1} + \theta_{a2,b2} = 0.16 + 0.18 = 0.34 \end{bmatrix}$$

5.2 Inference Operations in CSN

We have introduced the main ideas in message passing in Appendix A Section A.5 (page 174). We have chosen message passing as the inference algorithm for CSN because: a) the belief propagation (BP) algorithm, a type of message passing, is one of the most popular inference algorithms for BN [Pearl, 1988]; and b) message passing is a general technique that forms the basis of a wide range of algorithms defined in many research areas [Kschischang et al., 2001; Aji & Mceliece, 2000], e.g., multi-agent systems [Xiang, 2002] and graphical games [Ortiz & Kearns, 2002]. We hope that our approach can be extended to benefit other research areas in future.

Message passing algorithm assumes that the nodes are associated with full probability factors. However, in our representation, only context-specific probability factors are associated with nodes. Hence, we extend message passing algorithm to perform inference using the partitions of a CPT. There are two cases that we need to consider for the variable to which the message is passed: a) variable is not a part of context of context function nodes, and b) when it is a part of context or in other words it is a context variable. The message passing on the CSN uses three operations to be defined below instead of two (factor product and factor marginalization) used in general message passing.

5.2.1 Goal

The goal of inference in CSN is to make use of the CPFs for reasoning by exploiting contextual independence and supporting context structure adaptation of the graphical model. We will use Figure 5.1, a simple CSN with 5 variables, 7 context function nodes and two context variables L, B, to understand the operations and the algorithm. In Figure 5.1, the variable node S is associated with three context function nodes $1, 2, 3$ and three CPFs: $\Phi_1(S|R)$, $\Phi_2(S)$, $\Phi_3(S|N)$. Let us assume the following probabilities shown in Table 5.1.

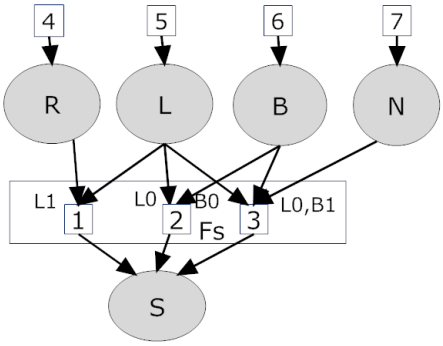


Figure 5.1: *CSN to support visualization of computations*

FACTOR	**PROBABILITIES**	
$\Phi_1(S\|R)$	[$\theta_{s0\|r0}$ $\theta_{s0\|r1}$ $\theta_{s1\|r0}$ $\theta_{s1\|r1}$]	[0.6 0.7 0.4 0.3]
$\Phi_2(S)$	[θ_{s0} θ_{s1}]	[0.8 0.2]
$\Phi_3(S\|N)$	[$\theta_{s0\|n0}$ $\theta_{s0\|n1}$ $\theta_{s1\|n0}$ $\theta_{s1\|n1}$]	[0.9 0.4 0.1 0.6]
$\lambda(L)$	[θ_{l0} θ_{l1}]	[0.8 0.2]
$\lambda(B)$	[θ_{b0} θ_{b1}]	[0.3 0.7]
$\lambda(R)$	[θ_{r0} θ_{r1}]	[0.4 0.6]
$\lambda(N)$	[θ_{n0} θ_{n1}]	[0.1 0.9]

Table 5.1: *Factors and Probabilities in Figure 5.1*

5.2.2 Context-sensitive Factor Product

Context-sensitive Factor Product facilitates factor product under factorization resulting from the finer structure due to CSI.

Let us first convey the intuition. Node 1 in Figure 5.1 has variable neighbors R, S, and L. L is a context attribute and $L = 1$ is a context assignment. R and S are non-context variables and $\Phi_1(S|R)$ is the conditional part factor associated with Node 1. $\lambda(R)$ and $\lambda(L)$ are factors associated with R and L respectively. Context-sensitive Factor Product defines how to carry out: a) the product of non-context factor $\lambda(R)$ with a conditional part factor $\Phi_1(S|R)$, and b) product of context factor $\lambda(L)$ with the conditional part factor $\Phi_1(S|R)$.

Definition 5.2.1 *Let $\lambda(\boldsymbol{A})$ and $\lambda(\boldsymbol{C})$ be two full factors where $\boldsymbol{C}$ is the set of context variables and $\boldsymbol{A}$ a set of non context variables. Let $\Phi(\boldsymbol{V})^{I(\mathbf{C}=\mathbf{c})} = \lambda(\boldsymbol{V},\boldsymbol{C}=\boldsymbol{c})$ be a CPF such that $\boldsymbol{V}$ is the scope without context of the CPF. We define context-sensitive factor product as a factor product conditioned on context $\boldsymbol{C}=\boldsymbol{c}$ i.e.*

$$\lambda(\boldsymbol{V},\boldsymbol{A},\boldsymbol{C}=\boldsymbol{c}) = \prod \Phi(\boldsymbol{V})^{I(\mathbf{C}=\mathbf{c})}\lambda(\boldsymbol{A})\lambda(\boldsymbol{C}=\boldsymbol{c})$$

In other words, context-sensitive factor product operation simply performs a factor product between any factor and conditional part factor conditioned on the context value of the conditional part factor. This means that the non-context factors are multiplied as they are but the context factors are conditioned on the context value. So, there can be three possibilities for the product of CPF with:

1) Only non-context factors $\lambda(\mathbf{A})$: $\lambda(\mathbf{V},\mathbf{A},\mathbf{C}=\mathbf{c}) = \prod \Phi(\mathbf{V})^{I(\mathbf{C}=\mathbf{c})}\lambda(\mathbf{A})$

   For example, only non-context attribute R for Node 1:

   $\lambda_{11}(S,R) = \Phi_1(S|R)\lambda(R) = [0.6\ 0.7\ 0.4\ 0.3].[0.4\ 0.6] = [0.24\ 0.42\ 0.16\ 0.18]$

2) Only context factors **C**: $\lambda(\mathbf{V},\mathbf{C}{=}\mathbf{c}) = \prod \Phi(\mathbf{V})^{I(\mathbf{C}=\mathbf{c})}\lambda(\mathbf{C}{=}\mathbf{c})$

For example, only context attribute L for Node 1 :

$\lambda_{12}(S, R, L = 1) = \Phi_1(S|R)\lambda(L = 1)$

$= [0.6\ 0.7\ 0.4\ 0.3].[0.2] = [0.12\ 0.14\ 0.08\ 0.06]$

3) Both non-context $\lambda(\mathbf{A})$ and context factor **C**:

$\lambda(\mathbf{V},\mathbf{A},\mathbf{C}{=}\mathbf{c}) = \prod \Phi(\mathbf{V})^{I(\mathbf{C}=\mathbf{c})}\lambda(\mathbf{A})\lambda(\mathbf{C}{=}\mathbf{c})$

For example, both context L and non-context R attributes for Node 1:

$\lambda_{13}(S, R, L = 1) = \Phi_1(S|R)\lambda(R)\lambda(L = 1)$

$= [0.6\ 0.7\ 0.4\ 0.3].[0.4\ 0.6].[0.2] = [0.048\ 0.084\ 0.032\ 0.036]$

5.2.3 Context Node Marginalization

This operation defines marginalization operation on a context function node. Context node marginalization performs factor marginalization over all the variables that are neither in the context set of the context function node nor in the scope of the variable node to which the message is being passed to. If the variable node to which the message is being passed to is a part of the context set, then context node marginalization marginalizes over all the variables and the factor left after marginalization is just the $\lambda(\mathbf{C})$.

Definition 5.2.2 *Let* $\boldsymbol{C}$ *be a set of context variables and* $\boldsymbol{A}$ *be the scope without context of factor* $\lambda(\boldsymbol{A},\boldsymbol{C})$ *on the context function node with context* $\boldsymbol{C}{=}\boldsymbol{c}$*. Let* V *be the variable in set* $\{\boldsymbol{A},\boldsymbol{C}\}$*to which the message is being passed to. We define context node marginalization to be a factor* λ *such that*

1. *Variable node is not a context node,* $V \notin \boldsymbol{C}$ *:* $\lambda(V, \boldsymbol{C}) = \sum_{\mathbf{A}\setminus V} \lambda(\boldsymbol{A},\boldsymbol{C})$

2. Variable node is a context node, $V \in \boldsymbol{C}$ *:* $\lambda(\boldsymbol{C}) = \sum_{\mathbf{A}} \lambda(\boldsymbol{A}, \boldsymbol{C})$

An example for Context Node Marginalization on Node 1 in Figure 5.1 would be:

Node 1 and message being passed to non-context node S :

$\lambda_{S1}(S, L = 1) = \sum_R \lambda_{13}(S, R, L = 1) = [0.132\ 0.068]$

Node 1 and message being passed to context node L :

$\lambda_{L1}(L = 1) = \sum_{SR} \lambda_{11}(S, R) = [1]$

Node 1 and message being passed to non-context node R :

$\lambda_{R1}(R, L = 1) = \sum_S \lambda_{12}(S, R, L = 1) = [0.2\ 0.2]$

5.2.4 Context Marginalization

Context marginalization sums over all the messages received to marginalize out the variables in context. The function node F performs *context marginalization* on the messages received from all the CF nodes that are a part of F. The message passed by each CF node after context node marginalization consists of the scope of variable node to which the message is being passed to and the context. There are two cases for the variable node to which the message is passed: a) the variable node is not a context node, and b)the variable node is a context node.

Definition 5.2.3 *Let* $\lambda_i(\boldsymbol{C}, V)$ *denote the message received from the* i^{th} *context function node. Let* $\boldsymbol{C}$ *be the set of variables in the contexts of all the n context function nodes that are part of the function node F, and* V *be the variable to which the message is being passed to, then we define context marginalization of* $\boldsymbol{C}$ *to be a factor* λ *over* V *such that*

1. *Variable node is not a context node, $V \notin \boldsymbol{C}$:*
   $\lambda(V) = \sum_{i=1}^{n} \lambda_i(\boldsymbol{C}, V)$

2. *Variable node is a context node, $V \in \boldsymbol{C}$, $\boldsymbol{C}' = \boldsymbol{C} \setminus V$:*
   $\lambda(V = m) = \sum_{i=1}^{n} \lambda_i(\boldsymbol{C}', V)^{I(C', V=m)}$

Each function node has a collection of messages from each context function node associated with it. Context marginalization operation adds up these messages to form a valid probability distribution. The first case implies that if the variable is not in the scope of context, the context marginalization simply reduces to the summation of all the messages. For example, let's assume if A is the variable node to which the message is being passed with a context variable set **C**, then $\lambda(A)$ = message in context **C=c1** + message in context **C=c2** = $\lambda(A, \mathbf{c1}) + \lambda(A, \mathbf{c2}) =$

$$\begin{bmatrix} a1, c1 & \theta_{a1}\theta_{c1} \\ a2, c1 & \theta_{a2}\theta_{c1} \end{bmatrix} + \begin{bmatrix} a1, c2 & \theta'_{a1}\theta_{c2} \\ a2, c2 & \theta'_{a2}\theta_{c2} \end{bmatrix} = \begin{bmatrix} a1 & \theta_{a1}\theta_{c1} + \theta'_{a1}\theta_{c2} \\ a2 & \theta_{a2}\theta_{c1} + \theta'_{a2}\theta_{c2} \end{bmatrix}$$

In the second case, for each assignment of a context variable V, the summation is done over all the messages with that specific context assignment, say c, and is stored as value for $\lambda(V = c)$. For example, for calculating message $\lambda(L)$, L being the context variable, all messages with $L = 1$ are added to specify the value for $\lambda(L = 1)$. However, there can be a case when not all the messages from context function node have the context variable V explicitly defined in their scope. For example in Figure 5.1, there are two context variables L and B and B is not in neighborhood of context function node 1. So, B is not in the context scope of the message from the context function node 1. Recall that the CPFs can be viewed as a general mathematical functions as explained in Chapter 4 and $\Phi^x = \Phi^{x.y}$ if y=1. So, the message from the context function node that does not have a context variable V in the CPF's context

is true for all values c of that context variable V and is added to calculation of all values of $\lambda(V = c)$. For example, the message to variable B from the node 1 with no variable B in context will be added to the message from node 2 in context B=1 as well as to the message from node 3 in context B=0.

An example for Context Marginalization in Figure 5.1 would be:

For non-context node S: if λ_{S1}, λ_{S2} and λ_{S3} are messages from node 1,2 and 3, then

$$\lambda(S) = \lambda_{S1}(S, L = 1) + \lambda_{S2}(S, L = 0, B = 0) + \lambda_{S3}(S, L = 0, B = 1)$$
$$= [0.132\ 0.068] + [0.192\ 0.048] + [0.252\ 0.308] = [0.576\ 0.424]$$

Here, λ_{S2} and λ_{S3} are calculated in the same way as λ_{S1} shown earlier.

Similarly, following the same notations for context node L, $\lambda(L)$:

$$\lambda(L = 1) = \lambda_{L1}(L = 1)$$
$$\lambda(L = 0) = \lambda_{L2}(L = 0, B = 0) + \lambda_{L3}(L = 0, B = 1)$$

5.3 Message Passing Algorithm

Extending the original message passing inference methodology for factor graphs [Kschischang et al., 2001], we have defined a new loopy belief propagation algorithm that works naturally and efficiently over CSNs to support different types of inference. In a nutshell, each context function node receives messages from its neighbors and performs a *context-sensitive factor product* with its local CPF and then performs *context node factor marginalization* for all the variables not in context. The function node F performs *context marginalization* on the messages received from all CF nodes that are part of F. Figure 5.3 explains the algorithm in mathematical form and Figure 5.4 outlines how the algorithm works in words.

5.3.1 An Example

There are three types of messages flowing around in our message passing algorithm:

a) Message from Variable to Context Function (CF) node (msgVtoCF{V,CF}: λ_{v2cf})

b) Message from CF node to Function (F) node (msgCFtoF{CF,F}: λ_{cf2f})

c) Message from F to Variable node (msgFtoV{F,V}: λ_{f2v})

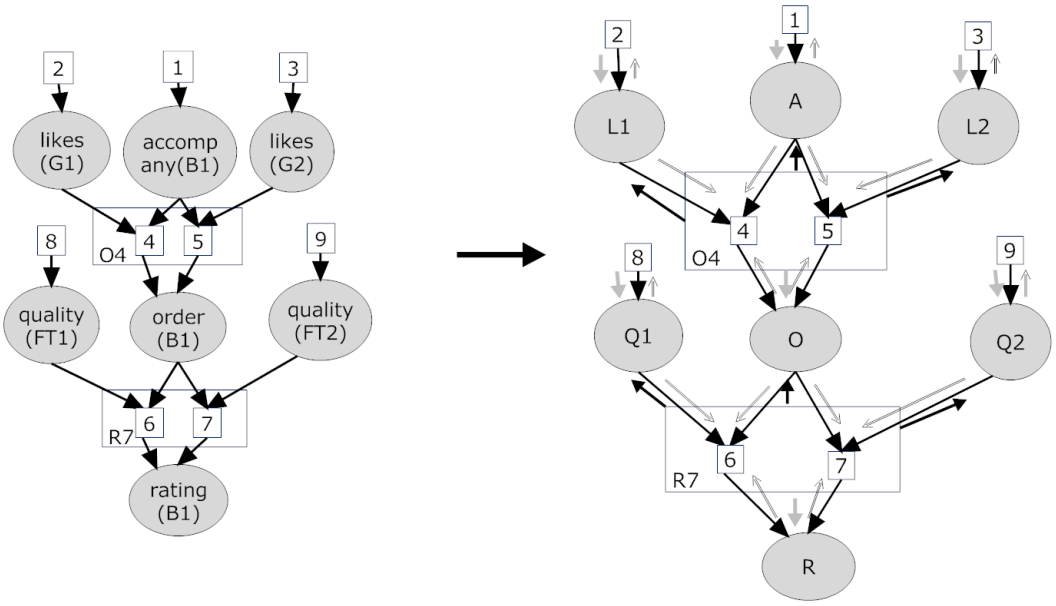


Figure 5.2: *Understanding Inference on Example 2: we have changed the labels on the figure on right for simplification. Messages flow in both forward as well as backward direction as shown in the figure. Arrows pointing towards a variable node denote λ_{f2v} and arrows pointing away/outwards from a variable node denote λ_{v2cf}*

In Figure 5.2,

Set of Variable nodes = $\{A, L1, L2, O, Q1, Q2, R\}$

Set of Context function nodes = {1,2,3,4,5,6,7,8,9}

Set of Function nodes = $\{1, 2, 3, O4, R7, 8, 9\}$. Context function nodes with no context information are also function nodes, e.g., node 1, 2, 3, 8, and 9 in this example.

$\mathbf{N}_v, \mathbf{N}_{cf}, \mathbf{N}_f$: set of neighbors of a variable v, cf and f
$\mathbf{N}_{vf}$: set of function node neighbors of a variable
$\mathbf{C}_{cf}, \mathbf{C}_f$: set of context variables of cf and f
$F(cf)$: function node f to which cf is associated with
$\lambda_{a2b}(z)$: message factor of scope z passed from node type a to type b

function SENDMESSAGEVCF(v, cf)
  for each context function cf node in $\mathbf{N}_v$
    $\lambda_{v2cf}(v) = \prod_{i \in (\mathbf{N}_{vf} \setminus F(cf))} \lambda^i_{f2v}(v)$

function COMPUTEMESSAGE(f)
  for each context function node cf in f
    for each v in $\mathbf{N}_{cf}$
      $\lambda_{cf2f}(\mathbf{C}_{cf}, v)$=
  $\sum_{\mathbf{N}_{cf} \setminus (v + \mathbf{C}_{cf})} \lambda_{cf}(\mathbf{N}_{cf}) \prod_{j \in (\mathbf{N}_{cf} \setminus v)} \lambda^j_{v2cf}(j)$

  $\prod$ here represents context-sensitive factor product,
  $\sum$ here represents context node marginalization

function SENDMESSAGEFV(f, v)
  for each variable v in $\mathbf{N}_f$
    $\lambda_{f2v}(v) = \sum_{\mathbf{C}_f \setminus v} \lambda_{cf2f}(\mathbf{C}_{cf}, v)$
  $\sum$ here represents context marginalization

Figure 5.3: *Pseudo code for each iteration of the Loopy Belief Propagation algorithm*

Message passing algorithm defined in Figure 5.3 does the following:

a) **Collect Evidence or absorb messages**

At Variable node:

do factor product of all messages received from function nodes, e.g.,

VarProd{A}= msgFtoV{O4,A}. msgFtoV{1,A}

VarProd{O} = msgFtoV{O4,O}. msgFtoV{R7,O}

Update Rule at variable node:
for each variable node V
1) Receive messages from all function nodes
2) Do factor product

Update Rule at context function node level:
for each context function node CF
1) Receive messages from each variable in neighborhood of CF
2) Do context-sensitive factor product
3) Perform context node marginalization
4) Pass message to the corresponding function node

Update Rule at function node level:
for each function node F
1) Receive messages from all CF nodes that are part of F
2) Perform context marginalization
3) Pass message to a variable in neighborhood of F

Figure 5.4: *Summarized rules for Loopy Belief Propagation on CSN. Update rules apply to message passing in both forward as well as backward direction.*

At Context Function Node:

do context-sensitive factor product of all messages from neighbor variable node, e.g.,

contextFacProd{4}= msgVtoCF{A,4}**.** msgVtoCF{L1,4}**.** msgVtoCF{O,4}

b) **Distribute Evidence or send messages**

At Context Function Node:

Divide the old message from the message receiving node, e.g.,

Let oldmsgFtoV denote the msgFtoV in previous iteration

msgVtoCF{O,4} = varProd{O}/oldmsgFtoV{O4,O}

Do context-sensitive factor product and perform context node marginalization, e.g.,

msgCFtoF{4,O4}= $\sum_{L1}$ $\Phi(O|L1)$ **.** (contextFacProd{4}/msgVtoCF{O,4})| (A=g1)

msgCFtoF{5,O4}= $\sum_{L2}$ $\Phi(O|L2)$ **.** (contextFacProd{4}/msgVtoCF{O,4})| (A=g2)

At Function Node:

Perform context marginalization and pass the message, e.g.,

msgFtoV{O4,O}= msgCFtoF{4,O4} + msgCFtoF{5,O4}

Here, we showed the calculations in the forward direction for one node O. Similar calculations are done for message passing in the backward direction from the children of O and propagation to its contextual and non-contextual parents and siblings.

5.3.2 Correctness of Message Passing

Theorem 5.3.1 *Message Passing on CSN performs the same calculations as those on the corresponding BN.*

Proof: Message Passing on BN is the same as the message passing at the functional level by Property 1 of the CSN, and also due to its correspondence to factor graphs. Message passing in factor graphs involves two types of messages: a) message from the function node to the variable node, and b) message from the variable node to the function node. We conjecture that if CSN calculates similar values for both types of messages, then the same messages are being passed between a function node and a variable node on both the factor graph and the CSN and the algorithm is correct. We show that

- Message from a function node to a variable node is same as that on the corresponding factor graph of an equivalent BN
- Message from a variable node to a context function node is same as that on the corresponding factor graph of an equivalent BN

We show calculations performed only on one function node as it is the same for the rest. Let $\mathbf{V}_{NEIGH}$ be the set of variables in neighborhood of function node F. Let V be the variable to which message is being passed. Let $\mathbf{V}_{NEIGH} \setminus V = \{\mathbf{V}_{NC}, \mathbf{V}_C\}$ where $\mathbf{V}_{NC}$ is the set of m variables not in context attribute set and $\mathbf{V}_C$ is the set of n variables in the context attribute set of F and $\lambda(V^i_{NC})$, $\lambda(V^i_C)$ are the corresponding messages of their i^{th} member.

Let us assume that there are k distinct context assignments for function node F, i.e., let there be k context function nodes. Note that $k \leq |d|^n$ (where $|d|$ is the total domain assignments of each context variable) as not all the context variables may be relevant for a particular context situation. By definition, function node F denotes the product of its CPFs. Let $F_{potential}$ refers to the CPT factor associated with function node F. Message passing on the BN or factor graph

$\lambda(V) = \sum_{\mathbf{V}_{NEIGH} \setminus V} F_{potential} \prod$(messages from its neighbors except V)
$= \sum_{\mathbf{V}_C} \sum_{\mathbf{V}_{NC}} \prod_{i=1}^{k} CPF_i \prod_{i=1}^{m} \lambda_i(V^i_{NC}) \prod_{i=1}^{n} \lambda_i(V^i_C)$
$= \sum_{\mathbf{V}_C} B. \prod_{i=1}^{n} \lambda_i(V^i_C)$ where $B = \sum_{\mathbf{V}_{NC}} \prod_{i=1}^{k} CPF_i \prod_{i=1}^{m} \lambda_i(V^i_{NC})$.

Each CF node is associated with one CPF and one context value (by definition).

Let us separate the k CPFs such that $B_j = \sum_{\mathbf{V}_{NC}} \prod_{i=1}^{m} \lambda_i(V^i_{NC}) CPF_j$. Variables in $\mathbf{V}_{NC}$ that are not in the scope of CPF_j sum to 1 after marginalization, so $B_j = \sum_{\mathbf{V}'_{NC}} \prod_{i=1}^{l} \lambda_i(V^i_{NC}) CPF_j$, where $\mathbf{V}'_{NC} \subseteq \mathbf{V}_{NC}$ are l variables in the scope of CPF_j or in other words the neighborhood of context function node j. This $\sum$ is the context node marginalization.

There are a total of n context variables in the set $\mathbf{V}_C$, each having d assignments. Let $c_{i1} \ldots c_{id}$ be d assignments to the i^{th} context variable V^i_C. Just like the variables in $\mathbf{V}_{NC}$ that sum to 1, we can further divide $\mathbf{V}_C$ into $\mathbf{V}_{CIN}$ and $\mathbf{V}_{COUT}$ to separate

the terms that sum out to 1. $\mathbf{V}_{CIN}$ and $\mathbf{V}_{COUT}$ indicate the context variables that are directly related to each CF node and those that are not.

$$\lambda(V) = \sum_{\mathbf{V}_C} B \prod_{i=1}^{n} \lambda_i(V_C^i)$$

$$\lambda(V) = (\prod_{i=1}^{n} \lambda_i(V_C^i = c_{i1})(B_1)) + \cdots + (\prod_{i=1}^{n} \lambda_i(V_C^i = c_{id})(B_{|d|^n}))$$

Taking the common part out (B's) and grouping together the rest, the terms containing $\mathbf{V}_{COUT}$ sum out to 1 as there are only k distinct values of context $= \sum_{i=1}^{k}$(messages from each context function node). This $\sum$ is what we have defined as context marginalization. For the variables in context, the second case of context marginalization can be worked out in the same manner. Therefore, a message from a function node to a variable node is the same as that on a corresponding factor graph of an equivalent BN.

This also implies that all the messages from the neighboring function nodes of a variable node are the same on both the CSN as well as the factor graph. Consequently, the messages calculated on the variable nodes are similar because variable nodes compute only the factor product. Therefore, the message passed to the context function node is also the same as the message from a variable to a function node on the corresponding factor graph. ■

5.4 Visualization

Just as in the factor graphs [Kschischang et al., 2001], the algorithmic computations of message passing can be visualized on a CSN as shown in Figure 5.1. In Figure 5.1, there are three context function nodes 1, 2, 3 contained in the function node O. To

pass a message to node S, node 1 receives messages from R, L, performs context-sensitive factor product with its local CPF $\Phi(S|R)^{I(L=1)}$ and then performs context node marginalization over R, the variable other than context variable L. The function node O receives messages from all the context function nodes that are part of F, i.e., node 1, 2 and 3 and performs context marginalization. We can also directly write the message passing equation from the CSN, for example, $\lambda(S)$ can be written as follows from Figure 5.1:

a) Message from context function node 1 with context {L=1}: $\sum_R \lambda(L=1)\Phi_1(S|R)\lambda(R)$
b) Message from context function node 2 with context {B=0, L=0}: $\lambda(L=0)\lambda(B=0)\Phi_2(S)$
c) Message from context function node 3 with context {B=1, L=0} :
$\sum_N \lambda(B=1)\lambda(L=0)\Phi_3(S|N)\lambda(N)$
$\lambda(S)$ = Sum of Messages from each context function node

Similarly, $\lambda(L=1)$ can be written as $\sum_{RS} \Phi_1(S|R)\lambda(R)\lambda'(S)$ and $\lambda(L=0)$ can be written as $\sum_{SN} \lambda(B=1)\Phi_3(S|N)\lambda(N)\lambda'(S) + \sum_S \lambda(B=0)\Phi_2(S)\lambda'(S)$ where $\lambda'(S)$ is the message of S from its children (none in this case).

5.5 Advantages, Limitations, and Complexity

Loopy belief propagation is an approximate algorithm which iterates the beliefs repeatedly over the network until they converge. It has also been shown that the loopy belief propagation usually converges to an approximately exact value. An important limitation, however, is that for a general graph, the loopy belief algorithm may not converge. Nevertheless, loopy belief propagation has recently enjoyed tremendous success in the real-world applications, even on the difficult instances of NP-Hard problems such as problems in error-correcting decoding [Kschischang & Frey, 1998].

Two important advantages of our inference algorithm on CSN is that firstly, the message passing exploits the partitions of a full factor in different contexts and utilizes CSI for inference with much fewer parameters and smaller factor size. The inference complexity decreases with factor size. Secondly, it preserves the ability to adapt the graphical structure. The structural adaptation as discussed in the next chapter allows structure decomposition into context-specific sub-graphs or models, thus further reducing the inference complexity.

Similar to the inference in a BN, the loopy belief propagation in the CSN in each iteration is linear in the parameter size. The inference is O(n) where n is the total number of parameters. The number of total parameters in a CSN is fewer than that in the corresponding BN. A CSN not only compactly represents the distribution but also performs fewer multiplications and additions in calculating a message than in the corresponding BN. In Figure 5.1, a CPT table of S requires $2^5 = 32$ entries in a BN representation. In contrast, in the CSN, the CPF for node 1 has only 4 entries, node 2 has 2 entries and node 3 has 4 entries, i.e., a total of only 10 entries. In general, if there are k context function nodes for one function node and let q denote the number of context-specific parents and c_i the number of context variable parents for each of k context function nodes and assuming all binary variables, then there are only $k.2^{q+1}$ entries in a CPF $<<$ 2^{p+1} entries in a CPT of a BN with p parents and $q << p$. This also suggests that the factor width, the scope of the factor, is much smaller in a CSN than in a BN. Furthermore, in Figure 5.1, the number of calculations required for factor $\lambda(S)$ in a CSN are 16 multiplications and 8 additions, while on a corresponding BN, 60 multiplications and 30 additions are required. The total number of multiplications in a CSN on a function node will be $\sum_{i=1}^{k}(c_i-1)+(2^{q+2}-2)$ and the total number of additions for context node marginalization will be $\sum_{i=1}^{k}(2^{q+1}-2)$. The total number of additions for context marginalization will be $2(k-1)$ if variable v is not in the context and $k-2+$ (number of messages without v in context), if variable v

is in the context. In contrast, in a BN with binary variables and p parents, it will take $(2^2+2^3+\cdots+2^{p+1}) = 2^{p+2}-4$ multiplications and $(2+2^2+\cdots+2^p) = 2^{p+1}-2$ additions. This shows that substantially fewer additions and multiplications are performed in a CSN if the distribution exhibits significant contextual independence. The size of a CSN, however, linearly increases with the number of CPFs.

5.6 Summary

In this chapter, we have extended the loopy belief propagation algorithm to utilize CSI in the probability distribution and to exploit the smaller factor size for the computational gains. We have proposed three new operations: context-sensitive factor product, context node marginalization and context marginalization. We have also shown how the algorithmic computations of message passing are carried out and can be visualized on a CSN. In the next chapter, we show how preserving asymmetry and CSI in the CSN framework facilitate exploiting contextual evidence for further inferential gain.

6

Context-based Knowledge Representation and Adaptation

Is there a difference between an ordinary random variable and a context variable? This chapter focuses on exploiting this key difference and describes context-based knowledge engineering and model adaptation that are few of the desirable features in context-based reasoning as discussed in Chapter 3. In this chapter, we propose our complete representation framework for context-based knowledge engineering consisting of the Contextual Local Views, an interface layer and the underlying CSN. We first establish Contextual Local Views, a meta-representation layer based on CSN semantics to facilitate knowledge engineering. Contextual Local Views support encoding the local contextual information for a specific context. Then, we describe an interface layer to translate contextual local views into an underlying CSN framework. The Contextual Local Views, the Interface, and the CSN form a full

context representation and reasoning framework.

Furthermore, we show how CSN provides the ability to adapt the graphical model structure in a given context. We then discuss the issue of irrelevancy and show that exploiting the context-specific structures can provide inferential advantages in many complex problems. We also explain how context structural adaptation facilitates CSN to preserve both the generality and the context-specific representations while effectively supporting different possible scenarios for context-specific inference.

6.1 Contextual Local Views: A Representation Scheme

Direct knowledge engineering of a CSN can be as difficult as that of a BN. One advantage of CSN is that it can be used as an underlying framework for a local context modeling scheme. In other words, a separate, user-friendly representation and knowledge acquisition scheme can serve as a meta-representation layer to encode the local contextual information. In this section, we propose to model context locally and show that CSN can be utilized as an underlying mechanism independent of the local representation scheme used. This also addresses the issue of the full CSN graph being cumbersome for larger graph sizes. The general idea is to model context-sensitive knowledge using context scenarios. Each context scenario encodes local context knowledge for a specific context value or assignment. Then, we use an interface language to combine all of these scenarios together to build a complete CSN. We call these context scenarios Contextual Local Views.

Local context scenario modeling is challenging because of the following reasons:

1. *Large number of scenarios:* Each context variable in a context set may have several domain values; modeling local scenarios for each possible domain value

increases the number of modeled scenarios exponentially.

2. *Probabilities encoded can be specific to the context scenario:* The CPT for a variable in each scenario represents only a partition of the full CPT function. So, a full CPT in the BN formalism needs to be formulated from several such partitions.
3. *Dependence among several local context scenarios:* Each local context scenario may share both non-context variables and context variables with others and merging of a number of different scenarios may be required for answering different queries.
4. *No preservation of contextual independence upon transformation:* Translating into the BN formalism requires formulating full CPTs; the independences that exist in a specific context are lost in the translation.
5. *Implementation issue - Insert, merge and split operations using non-tabular data structure can be costlier:* Data structures such as rules and trees can be used to capture multiple context scenarios. However, inserting new context variables, and merging and splitting different scenarios for inference could turn out to be costlier.

Contextual local views facilitate encoding local context knowledge of the related variables and their relationships in a specific context. Figure 6.1 shows the local contextual view for context $(belongsTo(d) = f, familyOut(f) = in)$, i.e., with the family (f) that the dog belongs to being inside (in) the house in Example 1. Each view contains a CSN dag with random variables and context function nodes. The context attributes are local root nodes, i.e., context nodes $belongsTo$ and $familyOut$ do not have any parent node in this local context. Each contextual view models the contextual dependencies for a particular context value or assignment. Each context function node is associated with a probability distribution. Nodes such as $lightOn$ in

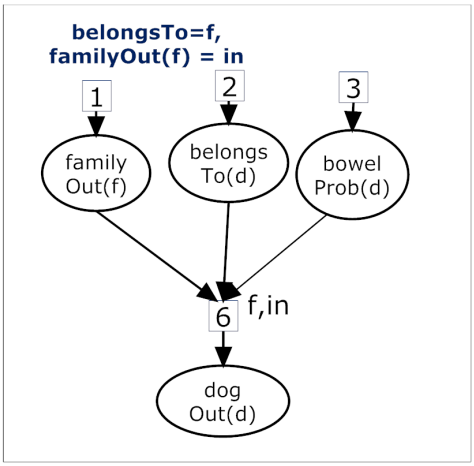


Figure 6.1: *Contextual Local View for context belongsTo, familyOut with family f being inside (in) the house*

Figure 4.3(a) (page 59) are modeled as being context-independent. The probability distribution associated with node 6 in Figure 6.1 is the CPF in Figure 4.3(b) (on page 59).

6.1.1 Well-formed Contextual Local Views

A contextual local view $(\mathbf{V}', \mathbf{CF}', \mathbf{E}'_{VCF}, \mathbf{E}'_{CFV}, \mathbf{\Theta}', \mathbf{C}')$ for a set of context variables is well-formed if a view for each context value can be combined together to encode a well-formed CSN for that specific context variable set. Specifically,

- Each contextual local view is associated either with a context variable set $\mathbf{C}'$ or with particular context value assignment $\mathbf{C}'=\mathbf{c}$.
- A contextual local view consists of a CSN *dag* at the contextual level with a set of variable nodes $\mathbf{V}' \subseteq \mathbf{V}$ and a set of context function nodes $\mathbf{CF}' \subseteq \mathbf{CF}$.
- All context-specific parents for a variable are specified in a contextual view.

- Each dependency edge $E_{VCF} \in \mathbf{E}'_{VCF} \subseteq \mathbf{E}_{VCF}$ is assumed to be context-independent unless it is labeled to hold only in a specific context $\mathbf{C}'=\mathbf{c}$ i.e., each context function node defined in a contextual local view is assumed to be associated with an empty context set unless it is specifically associated with a particular context value or assignment. $\mathbf{E}'_{CFV}$ represents the set of edges from context function nodes to variable nodes for that specific context set.
- $\mathbf{\Theta}' \subseteq \mathbf{\Theta}$ is the set of conditional part factors associated with each context function node

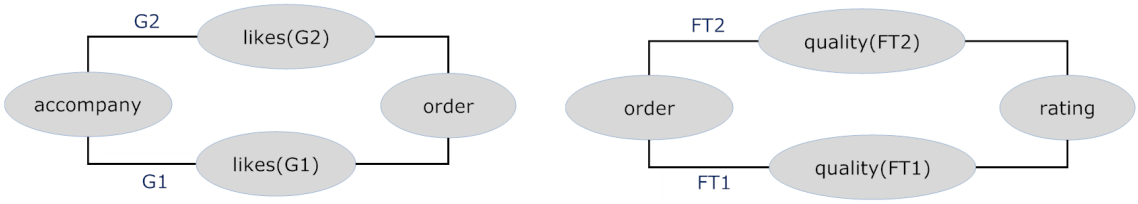
6.1.2 Property

The contextual local view inherits the following useful property by virtue of being a "mini-CSN":

Property A: *Each contextual local view can encode all the situations related to each value of the local context attribute together in one local view*

Proof: Each context attribute can have several domain values. The probability distribution associated with each context function node is the CPF. So, in a single view a number of context function nodes can be defined to capture context-specific dependencies for different context values of the local context variables. Hence, each contextual view can be used for covering all the context values of the relevant set of context attributes of contextual local view.

As per Property A, CSN dag allows modeling multiple contextual dependencies for different context values within one local contextual view as shown later in Figure 6.3(c).



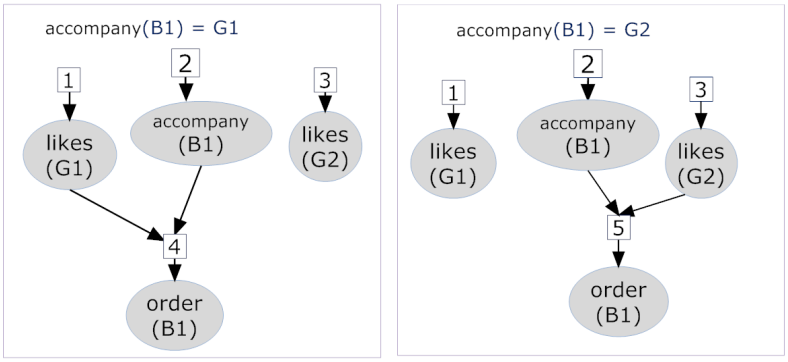
(a) accompany **(b)** order

Figure 6.2: *Asymmetry in knowledge in Example 2*

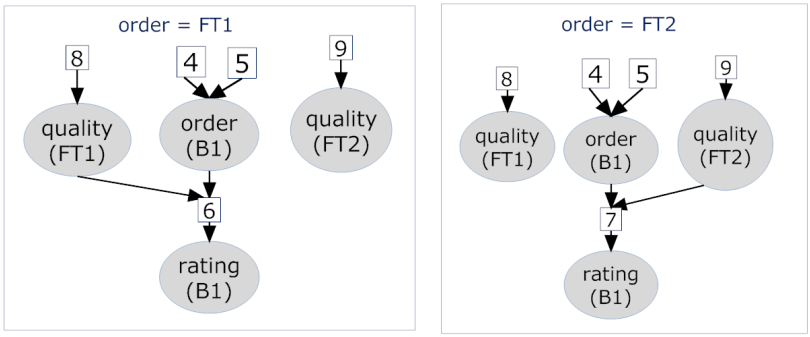
6.1.3 Situational modeling

In this section, we show how to represent different situations in the CSN framework for Example 2. The formulation of Example 2 shows how to model the following in the CSN framework: a) Multiple dependencies in a local context, b) Dependency among context attributes, c) Scalability with number of context attributes, d)Asymmetry induced due to context-dependent information.

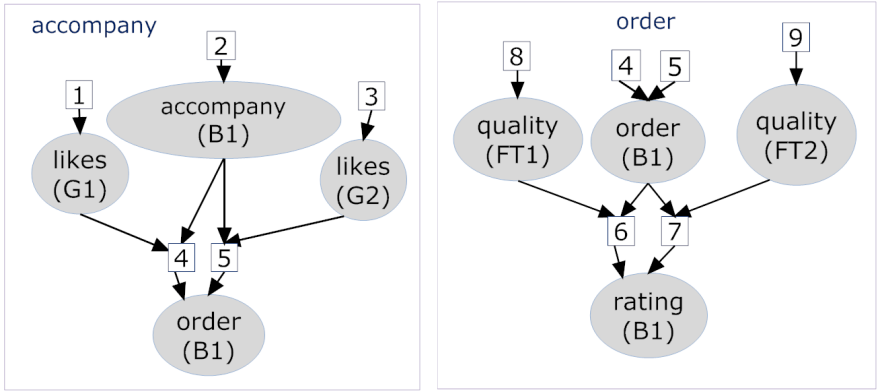
Figure 6.2 show the decision tree and asymmetry in knowledge based on the context attributes *accompany* and *order*. A contextual local view, capturing this asymmetry, can either be separately drawn for each context value or assignment (i.e., one specific pair of values for *accompany* and *order*) separately or all context values (i.e., all possible combinations of the values for *accompany* and *order*) can be combined to form one contextual local view for a set of context variables. Figures 6.3(a) and 6.3(b) show the contextual local views separately for each context value. Figure 6.3(c) encodes all the contextual dependencies together for a local context attribute set in a single view (Property A above). This is useful because: a) it limits the number of views needed to encode the complete information, b) it provides better scalability with the number of context attributes, c) it simplifies, from the maintenance point of view,



(a) Different contextual local views for context attribute: accompany



(b) Different contextual local views for context attribute: order



(c) One contextual local view for two context attributes represented in (a) and (b)

Figure 6.3: *Local contextual views: graphical scheme. Figure 6.3(c) uses 1 view for all the values of each local context variable and that reduces the number of total views required.*

the modeling of new context attributes without having to remodel all the contextual local views. In this example, the context attribute *order* is also dependent on context attribute *accompany*. Figure 6.4 show the underlying CSN for this example. The complete CSN is formed from these contextual local views using an interface layer discussed in the next section.

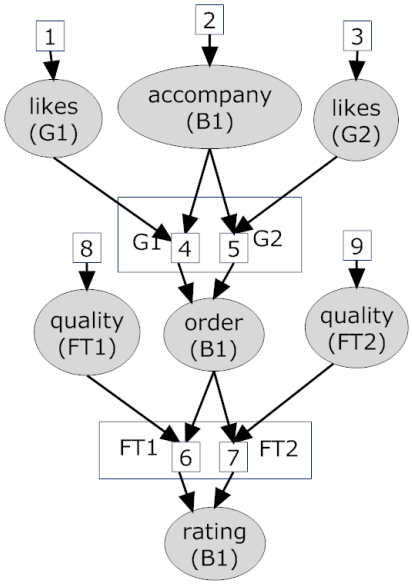


Figure 6.4: *CSN for Example 2*

6.2 Interface

The interface layer provides effective operations for transforming and encoding information in contextual views into the underlying CSN. The interface language merges all of the contextual local views together to build a complete CSN as underlying context representation formalism.

The interface layer models each contextual local view as a tuple $(\mathbf{V}', \mathbf{CF}', \mathbf{E}'_{VCF}, \mathbf{E}'_{CFV},$

function INTERFACE(CV)
// CV: collection of all contextual views each having $(V, CF, EVCF, ECFV, \theta, C)$;
G, Θ, $Context$ = empty // full CSN graph and CPFs
for cv = 1: CV
    update G with V_{cv},CF_{cv}
    for cf = 1: CF_{cv}
        add $EVCF_{cv}(cf)$, $ECFV_{cv}(cf)$ to G
        add C_{cv} to $Context(cf)$
        add $\theta(cf)$ to Θ
return G,Θ, $Context$

Figure 6.5: *Pseudo Code for translating all contextual local views into a full CSN*

$\mathbf{\Theta'}, \mathbf{C'}$) where $\mathbf{C'}$ is the context information for each CF node in the contextual view, $\mathbf{\Theta'}$ is set of distribution, and $(\mathbf{V'}, \mathbf{CF'}, \mathbf{E'}_{VCF}, \mathbf{E'}_{CFV})$ define the sub-CSN dag at the contextual level. Function nodes set is obtained using the following criteria: all context function nodes that point to the same consequent variable node are a part of one function node. The following properties need to be checked to ensure a well-formed CSN:

- The global graph formed by merging contextual local views should be acyclic
- The context for each contextual local view should be mutually exclusive and covering or exhaustive

Theorem: *Information captured in each contextual local view (*$\boldsymbol{V'}, \boldsymbol{CF'}, \boldsymbol{E'}_{VCF}, \boldsymbol{E'}_{CFV}, \boldsymbol{\Theta'}, \boldsymbol{C'}$*) can be translated into a well-formed CSN (*$\boldsymbol{G}, \boldsymbol{\Theta}, \boldsymbol{C}$*) if the contextual local views for each context variable are well-formed.*

Proof: By definition, each contextual local view is well-formed if they correctly encode a CSN for the local context variable set. All contextual local views together

encode the complete information needed to from a full CSN. We assume that the context assignment of the contextual local views are mutually exclusive, exhaustive and covering. Hence, if all the contextual local views are well-formed themselves, a full CSN is well-formed too.

Figure 6.5 shows the algorithm to translate the contextual local views into the underlying CSN. The interface definition encodes the contextual dependencies and preserves the asymmetry in information using only the graph manipulation operations without performing any CPT manipulation. Another advantage of defining the interface layer is that it also renders the underlying CSN representation independent of the language used to encode knowledge in the contextual local views. It is not essential to use the graphical modeling scheme defined in the previous section for modeling the contextual local views. For instance, a mixed scheme as shown in Figure 6.6 can be used to encode a Contextual Local View. Our prototype system can be easily extended to incorporate such a representation scheme.

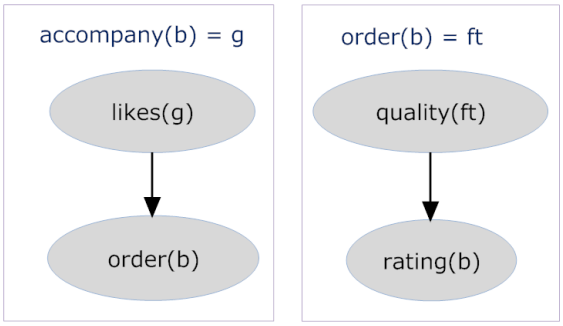


Figure 6.6: *Mixed graphical scheme*

6.3 Context Structural Adaptation

We now shift our focus to describe the context structural adaptation problem. In this section, we first discuss some background materials and then explain how CSN provides the ability to adapt the reasoning structure in a given context. We show that CSN supports dynamic context-based model adaptation to exploit structural variations in both the parameters as well as the graph structure.

6.3.1 Background

Evidence Handling

Given a query about any variable Q in a BN Graph **G** and an evidence **E=e** about a set of variables in **G**, the probability P(Q |**E=e**) is calculated using $\sum_{\mathbf{G}\setminus Q}$ P(**G**|**E=e**). Based on the strategy for evidence handling, the BN inference algorithms can be categorized into two types: static pre-compilation and dynamic compilation. The static pre-compilation approaches such as a junction tree algorithm transform the original BN graph into a tree representation beforehand in an offline manner. The tree is built without any knowledge of the evidence. The evidence is entered over this special tree representation. The extra time spent for building a tree helps static pre-compilation approaches to process a query faster. However, this also reduces their chance to utilize a more optimal graph and probability factors based on the knowledge of evidence. For example, the junction tree algorithm over a BN graph is known to be invariant to any local parameter structure, so the context evidence that induces a parameter decomposition have no effect on the transformed junction tree structure. Consequently, there is no difference between the handling of a context evidence or an ordinary variable evidence on the junction tree over a BN graph. In Example 1,

the transformed junction tree will have a clique width of about O(number of families*number of dogs). This graph can quickly become intractable. On the contrary, the dynamic compilation approaches such as the variable elimination or the cut-set conditioning algorithm take a lazy approach. They spend extra time in utilizing the evidence to reduce the original graph and associated conditional probability distributions in an online manner. This helps them to take an advantage of the type of the evidence observed and allows them to utilize much more optimal graphs for inference. As detailed in this chapter, CSN takes a dynamic compilation approach for context evidence handling.

Context-based adaptation in BN

As discussed in Section 3.6, the proposed approaches [Mahoney & Laskey, 1998; Ngo et al., 1995] for constructing a specific BN graph based on the observed context value primarily provide a first-level adaptation. Once a specific BN is constructed, no further model adaptation can be done even if the context-specific BN contains more context variables which can be observed at a later stage. Hence, their adaptation ability is limited. One reason may be that modifying a BN with large CPT sizes at the inference-time is time consuming, as noted in [Alvarado & Davis, 2005]. In BN, we need to do the following two operations for adaptation:

- Graph Manipulation: involves removing arcs or edges from the graphs depending upon the context
- CPT Manipulation: involves dynamically manipulating CPT function to contain only context-specific local probability distribution.

The graph manipulation is not a complex operation with a complexity of O(no. of edges to be removed). However, the CPT manipulation can be complex depending

upon the number of context variables and affected CPTs. Consider, for example, a standard tabular data structure for *order(B1)* in Example 2. A simple reduction algorithm for 1 context variable *accompany(B1)* would require that for each specific value of a context variable, we evaluate if each parent is rendered independent by checking whether all the probabilities for each domain value are equal. Therefore, each CPT manipulation with one context variable can take O(cpd) where c is the number of domain values of a context variable, p is the number of parents other than the context and d is the number of their domain values. This extra time to manipulate a CPT may not be useful at all for evidence handling. This can become even more expensive in the problems with more than one context variable as parents. Providing a structure to the conditional probability distribution such as tree or rule for faster manipulation can be useful. However, even with the tree/rule representation for the conditional probability distribution, we still need to evaluate each parent separately for the contextual independence.

6.3.2 Context Structural Adaptation Problem

Context variables, unlike the ordinary random variables, are special variables that, if known, can induce significant simplification in the state space and/or model structure. For example, in Example 1, if the values of the context variables $belongsTo(d)$ and $barking(d)$ are given, then the Figure 2.1 (page 17), instead of Figure 2.3 (page 20), is sufficient for a query, say about $familyOut(F1)$. However, in the BN representation, there is no special difference between an ordinary variable and a context variable, hence even if we are aware of the values of $belongsTo(d)$ and $barking(d)$, the model will not decompose into Figure 2.1. Given a context evidence, the contextual independence in the distribution can render only a part of the graph $\mathbf{G}$, say $\mathbf{G_1}$ where $\mathbf{G_1} \subseteq \mathbf{G}$, as sufficient to answer the query about a variable Q_1.

Context structural adaptation refers to the ability to adapt the graph to use a minimal number of relevant attributes required to calculate a query in a given context. The dynamic model adaptation ability supports continual adaptation of the model structure based on incremental context observations.

CSN inherently provides an easy way to adapt the model dynamically by considering the clauses that are satisfied in a specific context, and removing the irrelevant attributes in unsatisfied clauses from the model state space repeatedly. This also reduces the acquiring cost associated with some irrelevant observations, for example, the number of irrelevant questions asked to a user before the query can be answered. Formally,

Definition 6.3.1 *Let* $\mathbf{\Theta_G}$ *refers to the joint probability distribution in the graph* $\boldsymbol{G}$*. Given a context evidence* $\boldsymbol{c}$ *and a contextual independence in* $\mathbf{\Theta_G}$ *such that* $\mathbf{\Theta_{G|c}} = \prod_n \mathbf{\Theta_{G_n|c}}$ *(*$n \geq 1$*) where* $\mathbf{G_n}$ *are* n *disjoint subgraphs of* $\mathbf{G}$*, then a query* Q *about any variable in the graph* $\boldsymbol{G}$ *can be calculated from graph* $\mathbf{G_k}$ *if* $Q \in \mathbf{G_k}$ *as:*

$$\begin{aligned} P(Q|\boldsymbol{c}) &= \sum_{\mathbf{G}\backslash Q} P(\mathbf{G}|\boldsymbol{c}) = \left(\sum_{\mathbf{G_k}\backslash Q} P(\mathbf{G_k}|\boldsymbol{c})\right)\left(\prod_{j\neq k}^{n-1}\sum_{\mathbf{G_j}} P(\mathbf{G_j}|\boldsymbol{c})\right) \\ &= \sum_{\mathbf{G_k}\backslash Q} P(\mathbf{G_k}|\boldsymbol{c}) = \sum_{\mathbf{G_k}\backslash Q} \mathbf{\Theta}_{\mathbf{G_k}|c} \end{aligned}$$

6.3.3 Handling Context Evidence in CSN

A CSN provides an ability to easily adapt the reasoning structure to a given context evidence. The formal notion of d-separation is used in a CSN to exploit contextual dependencies for determining the contextual relevance.

Similar to the local contextual view translation, context structural adaptation problem in CSN also involves only the *graph manipulation* operations. This allows an easy

```
// structural adaptation and Graph Manipulation
function ADAPT(G,CF,c)
//G: CSN graph; CF:list of cfnodes in G with context C; c : observed value of context C
// context(cf): function to give context value of cf node
// neighbors(cf): all variable neighbors of cf node
L = empty; // list of affected variables due to arc removal;
G'=G
for cf=1: CF
    val = context(cf)
    ifval <> c
      add neighbors(cf) to L
      drop all incoming edges and outgoing edges from cf in G'
      remove cf from G'
return G',L
```

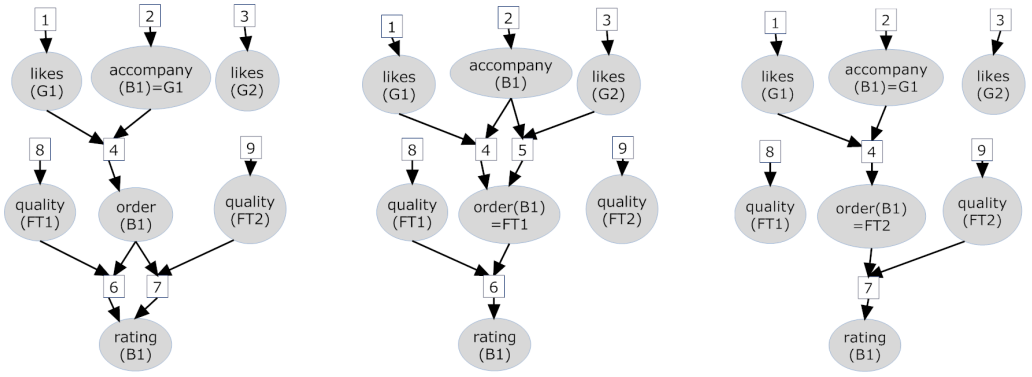
Figure 6.7: *Pseudo Code for structural adaptation*

and a faster structure decomposition without involving any CPT manipulations.

As per the d-separation property defined in Chapter 5 (page 65), a path is blocked if the context variables associated with the context function node in the path are observed and the context value of the context function node is not the same as the observed context value. In Figure 4.6(b) (page 66), if we know the value of context variable *accompany* for boy B1, say its G2, then B1's *order* is d-separated from G1's *likes*. The indicator function of context function node 4 evaluates to 0 and that of CF node 5 evaluates to 1. Thereafter, the CF node 4, being unrelated, can be removed, and all the incoming and outgoing edges can be dropped. The CPF associated with node 4 is automatically not considered as part of the inference. This switching between the CPFs depends upon the context variables. The rest of the structure remains the same. Hence, a CSN can adapt the structure to the required number of attributes for a given context using only graph manipulations.

Figure 6.8 shows the resulting CSN structures for the different context values. In

Figure 6.8(b), the attribute *quality(FT2)* is rendered irrelevant. In fact, if we assume another subgraph associated with *quality(FT2)* in Figure 6.8(b), then the whole sub-graph would have been rendered irrelevant. Similarly, in Figure 6.8(c) with more context observations, 2 sub-graphs are rendered irrelevant.



(a) accompany(B1)=G1 **(b)** order(B1)=FT1 **(c)** accompany(B1)=G1, order(B1)=FT2

Figure 6.8: *Structural adaptation with observed context values or evidence*

Even the complex problems with the more than one context variables as parents and more than 1 CPF evaluating to 1 can be easily handled using graph manipulation. Furthermore, the inference algorithm as explained in Section 5.3 remains unmodified.

6.3.4 Issue of Irrelevancy

In this section, we define algorithms that can test the decomposition of a graph into a sub-graph given a context evidence. The structural adaptation can render a part of the graph completely irrelevant for a query given a specific context. The advantages of irrelevancy and relevance structural decomposition had been examined earlier for the BN inference [Lin & Druzdzel, 1997; Druzdzel & Suermondt, 1994].

```
// determining if CSN graph is fully separable into subgraphs upon context evidence
function ISSEPARABLE(G,L)
//G: CSN graph after adaptation; L: list of affected variables due to adaptation
slist, sgraph = empty
for v = 1: |L|
    others = L \ v
    list = reachable(G,v) //list of nodes reachable from v
    disconnect = others ∩ list
    ifisempty(disconnect) // graph containing v is separable
      add v to slist // slist: list of nodes whose subgraphs can be separated
      add list to sgraph // sgraph: list of nodes reachable from v including v
      remove v from L
return slist, sgraph

// output smaller graphs
function SUBGRAPH(V,S,G)
// V:slist; S:sgraph; G:CSN graph after adaptation
newGraphs= empty
for each v in V
    G'=empty
    L= S(v) // sgraph(v): the list of nodes reachable from v including v
    G'= G[L]
    add G' to newGraphs
return newGraphs
```

Figure 6.9: *Pseudo Code for separating irrelevant sub-graphs*

The graph manipulation algorithm removes all the dependencies from the original graph that are not relevant for a specific context . Given this graph, the algorithms defined in Figure 6.9 first determine if each variable is reachable from the query variable list. If they are reachable, they are relevant, otherwise, they are not. The graph reachability problem is well researched with several available algorithms [Agrawal et al., 1989]. We adopt the depth-first search algorithm. Consider, for example, given specific values of the context variables $belongsTo(d)$ and $barking(d)$, the graph manipulation algorithm removes all the dependencies related to any other families and

dogs, rendering their related attributes as irrelevant. The algorithms will separate the graph into sub-graphs. It is also possible to disintegrate a full CSN into its constituent contextual local views given the appropriate context evidence.

6.3.5 Exploiting Dynamic Adaptation for Inference

In this section, we explain how context structural adaptation facilitates CSN to preserve both the generality and the context-specific representations while effectively supporting different possible scenarios for context-specific inference. Furthermore, we explain how context structural adaptation can be useful for solving intractable problems.

Given some evidence about attributes, three types of cases are possible:

Case 1: No context observation, i.e., all contexts are uncertain
Case 2: Mixed, i.e., some are observed while others are uncertain
Case 3: All contexts are observed, i.e., no context uncertainty

We categorize these 3 cases into two types of inference:

a) Situational uncertainty: no context observation
b) Situation-specific: some or all contexts are observed

We use a dynamic compilation strategy for the CSN framework and will now show that the CSN inference algorithm, explained in Section 5.3 can perform both these types of inference.

a) **Situational uncertainty:** For situational uncertainty (Case 1), there is uncertainty over all the context variables in the CSN, so there is no graph reduction possible, hence the input to the belief propagation inference would be a full CSN graph. The

inference algorithm then computes the belief with evidence for the ordinary random variables.

b) **Situation-specific:** In contrast to the Case 1, in Case 2 and 3 the graph manipulation algorithm removes the affected dependencies from the CSN in a given context and then the subgraph algorithm outputs the subgraphs formed. Inference algorithm then runs for each of these graphs individually. If the target nodes or query nodes are identified, then only the relevant subgraph is used for the inference. Hence, the complexity of inference can be significantly reduced.

For situation-specific inference, when some contexts are observed, while others are uncertain (Case 2), there may be some variables/graphs that may be rendered irrelevant for a given query. The reduced graph is the input to the inference algorithm. Depending upon the query, CSN can exploit inferential advantage due to smaller graph size. With more context evidence, the graph is simpler and hence the inference is simpler. For example, the inference with no context observation in Figure 1.2 (page 11) will be done using all the 7 variable nodes while the inference with context *accompany* as G1 in Figure 6.8(a) will be done using 5 variable nodes.

When all the context variables are observed (Case 3), the CSN graph contains an equal number of CF nodes and variable nodes. So, it is also equivalent to a BN graph. For example, in Figure 6.8(c), both the context variables *accompany(B1)* and *order(B1)* are observed. Only one context function node 4 is associated with the function node for *order(B1)* and only one context function node 7 is associated with the function node for *rating(B1)*. All other variables are also associated with only 1 context function node. So, the graph has no probability partitions. This also means that any BN inference algorithm can work in this case. However, we need to point out that this is only true if all the contexts are given, otherwise we need to have a special inference algorithm that works with context-specific probability partitions as

defined earlier in Chapter 5.

Theorem 6.3.2 *A CSN with evidence for all the context variable nodes is equivalent to a BN after graph reduction, hence any BN inference algorithm will work on it*

Proof: CSN, by definition, factorize the joint probability distribution into a product of CPFs. Each CPF represents a partition of the CPT that is valid in a specific context. Given a context value, the function node associated with the affected variable is only associated with that specific CPF. Given a full context assignment, i.e., an assignment with values for all the context variables, all the function nodes in CSN have only 1 CPF each. Hence, the context functional level of CSN is equivalent to the functional level of CSN. Therefore, any BN inference algorithm will work without any modification.

Another advantage of the context structural adaptation ability of CSN is that the cut-set conditioning [Pearl, 1988], an exact inference method, can be applied directly on CSN for smaller graphs. The cut-set conditioning method reduces multiply-connected graph to a tree by conditioning the graph on a variable set, a cut-set, that cuts all the loops in the graph. Th cut-set conditioning inference algorithm does reasoning by cases. As the cut-set reduces the graph to a tree, the message passing on the CSN would be exact. Though cut-set conditioning can be applied directly, we expect the run-time efficiency for exact inference to be further improved with the recent recursive conditioning algorithms [Darwiche, 2001].

In general, we believe that the conditioning approaches may be more effective for CSN framework than the junction tree algorithm. This is because the context conditioning can remove contextually irrelevant cycles dynamically, and reduce the graph size and the induced clique width significantly; this may in turn make many more graphs tractable. Moreover, efficient inference in CSN is facilitated by preserving contextual

independence during multiplication and marginalization in the inference procedure; this may be difficult to achieve in junction tree transformation.

6.4 Summary

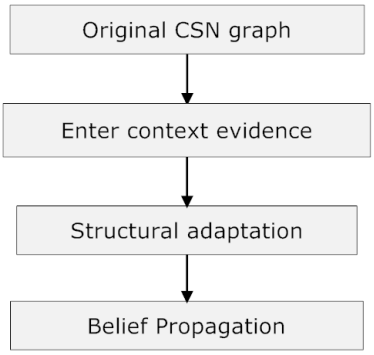


Figure 6.10: *System view of context evidence and structural adaptation*

Figure 6.10 summarizes and provides a system view of context evidence and structural adaptation handling approach. First, the original CSN is defined as detailed in Chapter 4, and then the evidence is entered. If there is a context evidence, then the context structural adaptation algorithm generates a context-specific structure, for example Figure 6.8(a). Inference is then done over this graph.

In this chapter, we have proposed a complete representation framework for context-based knowledge engineering consisting of the Contextual Local Views, an interface layer and the underlying CSN. We exploit contextual dependencies for transparency and flexibility in model representation. We have also discussed how to exploit context evidence and established dynamic model adaptation as one of the key advantages of the CSN framework. We have also explained that CSN effectively supports different possible scenarios for context-specific inference and showed how these context-specific

relevant structures can decrease the model complexity. So far, we have discussed the semantics and theoretical properties of the CSN framework, and the ability to utilize partitions for inference and model adaptation. In the next chapter, we extend the CSN language to support relational modeling and parameter learning.

7

Relational Modeling and Parameter Learning

This chapter discusses some extensions to the CSN language. We show that the modeling extensions for simplifying knowledge engineering in BNs can also be easily incorporated in the CSN framework to facilitate context modeling. In particular, we sketch two important extensions: a) how context can be included in relational probabilistic languages, and b) how to do parameter learning from data in CSN.

7.1 Relational Extension of CSN

As discussed in Appendix A.4, first-order relational modeling can help to encode generalization about objects. The relational extension of CSN extends relational BN to domains where both the exact number of ordinary variables as well as context vari-

ables may not be known *a priori*. A relational CSN encodes generalizations about the objects in a specific context and shows that the CSN language can easily incorporate recent extensions done in the Bayesian field.

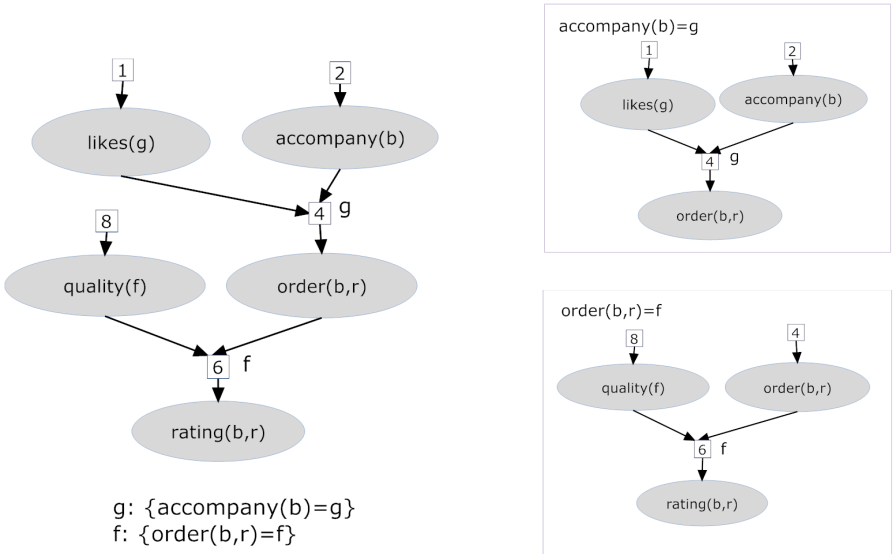
7.1.1 Background

In relational logic modeling, the world is modeled at the class level rather than at the individual level, as in the propositional logic. The world is seen to be consisting of objects (or entities) and their relationships (or specific interaction among entities). A relational logic BN is an abstraction of the dependencies among the object's properties together with their relationships with other objects. Just as a logic rule makes it explicit that this statement applies to any object instantiation in a class of objects, a probabilistic logic rule makes it explicit that this probability statement applies to any object instantiation in a class of objects, for example, $\forall$x, $P(A(x)|B(x), C(x), D(x))$ is a belief table for all instantiations of an object x. All instantiations of the object share the same local probability distribution. For inference, a relational network is usually rolled into a propositional BN network. Inference is then done over the standard propositional BN using established algorithms. It turns out that the relational BNs can roll out into large propositional networks over which exact inference is generally not tractable, and therefore approximate inference methods are generally preferred. For example, in the food network in Example 2, the junction tree exact inference method would run out of memory with only 5 girls and 8 food types in one of our experiments, to be discussed later.

We have extended the CSN to a relational level and allow relational probability statements to be encoded either in the contextual local views or directly in a complete CSN. Extending CSN from the propositional to a relational domain helps in using quantification in modeling. Quantification makes knowledge engineering easier. It

provides generalization for a class of individuals and eliminates modeling the redundant clauses required for each individual of the same class as they exhibit a similar behavior.

7.1.2 Relational knowledge representation



(a) CSN **(b)** Contextual Local Views

Figure 7.1: *Relational CSN for Example 2*

Relational knowledge representation can be encoded either using contextual local views or directly in a full CSN as shown in Figure 7.1. Each contextual local view defines a schema for a specific context as shown in Figure 7.1(b). Each probabilistic logic rule can be true in a specific context or in all contexts. The interface language allows joining these separate views together. The probabilistic models encoded at the logical level may not have large graph size, so all the probability rules can directly be encoded as full CSNs rather than using contextual local views as shown in Fig-

ure 7.1(a). The probability distribution is defined for each relation (also represented as a random variable or node here) and all the instantiations of the object have the same conditional probability distribution for the same relations, for example, the belief table for CPF *order* $P(order(b, r)|likes(g))$ remains the same for all instantiations of boys and restaurants.

The schema for the full CSN is represented as follows:

- The world consists of base objects $(O_1 \dots O_n)$
- Each object type is characterized by a set of attributes/properties/variables $\mathbf{V}(O_i)$. Each attribute/property $V_j \in \mathbf{V}(O_i)$ assumes values from a finite domain, $D(V_j)$. The domain does not need be fixed beforehand and can be a function of number of instantiations of an object O_m.
- Elements of these objects are connected through a collection of relations $(R_1 \dots R_k)$
- The context relation defines the relational associations that holds in that specific context
- Logical rules are defined for each context value
- Local probability distributions or CPFs are encoded for each logical rule

In Figure 7.1(a), boys, girls, and food types of each restaurant are entities or objects; *accompany* is a property of Boys, *likes* is property of Girls, *quality* is the property of each restaurant food types; *rating* and *order* are relations between boys and restaurant. Instantiations of the entities in 7.1(a) are shown in Table 7.1

Objects/relations	instantiations	{domain values}	CPFs distribution
Boys (b)	Set of (B1, B2)		
Girls (g)	Set of (G1, G2)		
Restaurant (r)	R1		
Food types (f)	Set of (FT1, FT2)		
accompany	Set of b	girls: {G1, G2}	$\Phi(accompany)$
likes	Set of g	food types: { FT1, FT2 }	$\Phi(likes)$
order	Set of b×r	food types:{ FT1, FT2 }	$\Phi(order\|likes)$
quality	Set of r×f	{good, bad}	$\Phi(quality)$
rating	Set of b×r	{high, low}	$\Phi(rating\|quality)$

Table 7.1: *Description of relations in Example 2*

7.1.3 Inference by converting to Propositional CSN

Just as in the relational BN language, a relational CSN is transformed into a propositional CSN for the inference. For each instantiation, a number of propositional statements are generated, and the instantiated graph then rolls out into a CSN, as shown in Figure 7.2.

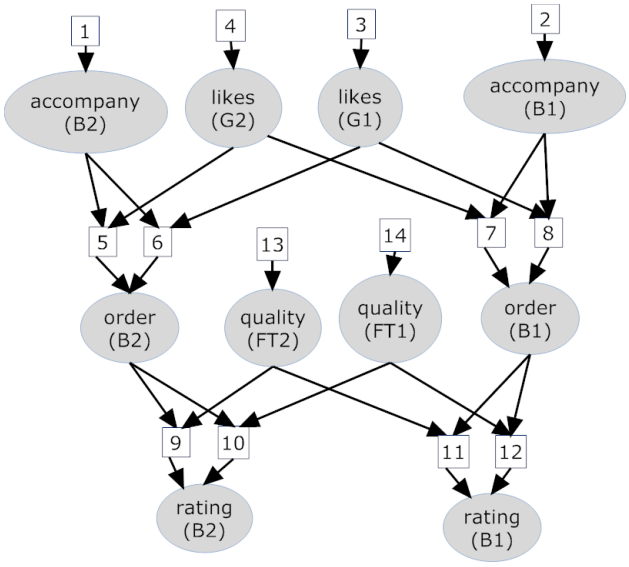


Figure 7.2: *Rolled out propositional CSN*

```
function ROLLOUT(G,R)
//G: relational graph; R: set of relations
G' = empty
for each Relation R_i
    num_i = product of no. of instantiations for all Object O_j in R_i
    for k=1: num_i
        instantiate a variable V_ik
        for each edge V_i to CF_j of R_i in G // Edge E_VCF
            if CF_jk is instantiated // context function node exists in G'
               add an edge from V_ik to CF_jk in G'
               CPF(CF_jk) = CPF(CF_j)
            else
                Instantiate CF_jk
                add an edge from V_ik to CF_jk in G'
        for each edge CF_l to V_i of R_i in G // Edge E_CFV
            add an edge from CF_lk to V_ik in G'
return G'
```

Figure 7.3: *Pseudo Code for converting Relational to Propositional CSN*

The inference algorithm defined for the propositional CSN is used to support inference over queries regarding the base entities.

7.1.4 Context Structural Adaptation

The structural adaptation algorithm as shown in Section 6.3.3 can be applied as it is. An important advantage is that quantification can be applied to context evidence too i.e., the evidence about context relational attributes can be generalized to the whole class. The structural adaptation algorithm then renders all the instantiations of the affected object and their relations as irrelevant. In other words, the structural adaptation can be done both at the class level before rolling out and at the individual instantiation level after rolling out to the propositional CSN.

7.1.5 Advantages and Limitations of Relational inference

Apart from eliminating the redundant sentences, another important advantage of extending CSN to a relational domain over the propositional domain is that by keeping the local probability parameters the same across the class, it is able to answer queries for new instantiations of the same class with unknown training data. For instance, in the restaurant modeling example, we can use a similar model for another boy to infer the probability of his ratings for the restaurant even though we may not have any of his attributes in the training data. The downside is that the relational model can roll out into large propositional networks over which the inference may be slower or intractable. Currently, our prototype implementation of relational CSN has limited expressive power as it does not incorporate the concept of Aggregates in Probabilistic Relational Models [Friedman et al., 1999] which helps in constructing a single feature to characterize a set from the set of objects, though it can be done in a straightforward manner.

7.2 Learning Parameters from Data in CSN

One of the important advantages of a Bayesian framework is that the probability parameters can be estimated from data if an adequate amount of the training data is available. This makes it an ideal representation for combining prior knowledge and data. In this section, we show how probability parameters can be estimated in the CSN framework.

7.2.1 Objective

In parameter learning, the network structure is specified and the goal of learning is to find the parameters that maximize the likelihood of the training data. The objective of the parameter learning module is to estimate the probabilities of each CPF given the CSN structure, a context **C=c** associated with all CF nodes, and a complete dataset.

7.2.2 Procedure

We use the Maximum Likelihood Estimate (MLE) criteria for estimating the probability parameters of the fully observed model. We first show that CSN follows the local decomposition property and then prove that MLE on CSN reduces to estimating the local counts.

Proof: Let us assume a training dataset D with N independent and identically distributed (i.i.d.) samples and l variables $(X_1 \dots X_l)$. We first show that the global likelihood function decomposes as per the local structure of the graph. This decomposition property implies that the independent estimation can be done for each local CPF and MLE reduces to estimating counts of the local probability distribution in CSN.

Let Θ denote the joint probability distribution. Log likelihood expression $\mathrm{L}(\Theta{:}\mathrm{D}) = log \prod_{i=1}^{N} P(X_1 \dots X_l : \Theta)$. By property 2 in Section 4.5, CSN graph factorizes the joint probability distribution into a product of CPFs. Therefore, let us assume m CPFs, each with a conditional probability distribution θ_j, then:

$$\mathrm{L}(\Theta{:}\mathrm{D}) = log \prod_{i=1}^{N} \prod_{j=1}^{m} (CPF_j : \theta_j) = \sum_{j=1}^{m} [\sum_{i=1}^{N} log(CPF_j : \theta_j)] = \sum_{j=1}^{m} [L_i(\theta_j : D_i)]$$

This shows that the global log-likelihood expression decomposes into a local log-

likelihood. Each CPF is assumed to have a multinomial distribution. The parameters of each CPF are non-overlapping, and thus we can maximize each of them individually to give the maximum likelihood estimate under the constraints that the parameters in each of the CPF vectors sum to 1. Hence, the parameter estimation just reduces to local counts in the dataset.

The sufficient statistics for the CPF estimation is

$$\Theta_{x_i|cspa(x_i)}^{I(C=c)} = N[X_i, \mathbf{csPa}(X_i), \mathbf{c}]/N[\mathbf{csPa}(X_i), \mathbf{c}]$$

, where **csPa**: context-specific Parent set, **csPa** and Context **C** are disjoint, **C=c**: the specific context, N: number of data samples, .

The expression above counts the occurrences of X_i together with the occurrences of context-specific parents of X_i within **C=c** rows in the complete dataset. Hence, the parameter learning algorithm in CSN estimate local counts based on the local family data in the specific context.

For example, in Figure 7.2, $\Phi(order(B1)|likes(G1))^{I(accompany(B1)=G1)}$
= *N[order(B1), likes(G1), accompany(B1)=G1] / N[likes(G1), accompany(B1)=G1].*

7.2.3 Advantages

CSN offers two advantages for parameter learning:

1. As CSN factorizes the joint probability distribution based on contextual independence rather than conditional independence, fewer parameters need to be estimated for a CSN than that in traditional BN. Given the same size of the dataset, the fewer the number of parameters, the better is the confidence about their estimates [Friedman & Goldszmidt, 1996].

```
function LEARN(G,N)
//G: CSN graph; N: data cases
n = number of CFs in G
Θ = empty
for cf = 1: n
    CP = set of contextual parents of cf in G;
    V = consequent variable of cf in G;
    val = context value of cf in G;
    θ_cf = N( V,CP,val)/N(CP,val)
    add θ_cf to Θ
return Θ
```

Figure 7.4: *Pseudo Code for parameter learning*

2. CPFs in different contexts can share the same probability distribution. For example, the two probability factors $P(rating|foodQualtiy(F1))$, $P(rating|foodQuality(F2))$ though valid in different contexts are likely to be similar. As probability factors are qualitatively represented on CSN, it can be easier to visualize such parameter tying on CSN. This can further reduce the number of parameters required for estimation, which can in-turn improve the overall quality of the probability estimates. The concept of parameter tying has been previously applied to learning relational BNs [Friedman et al., 1999].

7.3 Summary

In this chapter, we have discussed some extensions to the CSN language. We have extended CSN to a relational level to model associations as functions of both the instantiated objects as well as the contexts. Relational probability statements eliminate redundancies by quantification and generalization for an object class and can be either encoded in the contextual local views or directly in a complete CSN. Inference

is carried out by rolling the relational CSN structure to the propositional level and structural adaptation can be exploited both at the object's class level as well as at the individual instantiation level. We have also shown how probability parameters can be estimated in the CSN framework and discussed the advantages of estimating fewer parameters.

8

Experiments and Case Studies

This chapter examines the effectiveness and efficiency of CSN based on a set of empirical studies and two case studies. We also discuss the prototype implementation and the experimental evaluation of the proposed methodology. We informally evaluated the effectiveness of CSN with two case studies involving actual clinical situations: the first one involves Coronary Artery Heart Disease (CAD) and the second one involves Community-acquired Pneumonia (CAP). Based on these case studies, we demonstrate that CSNs are expressive enough to handle a range of problems involving context in medicine. The case study on CAP also illuminates how a context framework can utilize newer knowledge acquisition techniques and describes the novel use of clinical guidelines to support knowledge engineering in probabilistic networks.

8.1 Prototype Implementation

Figure 8.1 shows the system architecture of a prototype implementation of the CSN framework. The system is implemented in MATLAB and tested on Pentium M 1.8 GHz with 1 GB memory. The implementation follows the theory discussed in the Chapters 4, 5, 6 and 7. The system supports context model representation either as a CSN, or using contextual local views, or in a relational language. The system includes an interface translator, allows context structural adaptation and provides parameter learning. Model specification is currently strictly text based with no graphical views. Furthermore, two different ways of belief updating for message passing inference algorithm, have been implemented and tested. In the first implementation, we followed a random order scheme, a standard method for loopy belief propagation (LBP) implementation, where each node continues to compute and send its messages till they converge. In the second implementation, we utilized ordered update scheme to update in a tree-like preorder and/or postorder fashion for every iteration. The probability distribution of the CPFs can be specified in two ways: either manual specification or automatic generation using a stochastic distribution function. The probability table is specified as per the standard probability laws to define a valid distribution. The parameter learning module can also be used to estimate probabilities given the data samples.

8.2 Experimental Setup

8.2.1 Tasks

We first present an extended evaluation of our framework on the generalization of Example 1 to multiple families, dogs and barks heard. The generalized experi-

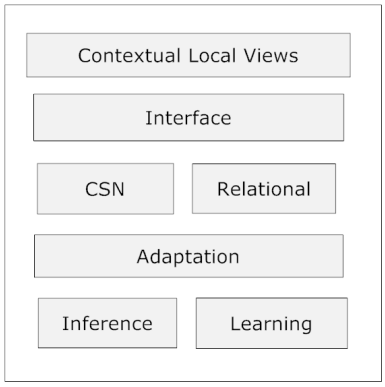


Figure 8.1: *Prototype Implementation: Systems View*

ments captured several important patterns: increase in the number of context variables ($belongsTo, familyOut, barking$), context values ($belongsTo, barking$), non-contextual parents ($bowelProb, dogOut$), affected consequent variables ($dogOut, hearBark$), and graph complexity with a large number of cycles. The model was coded using our prototype implementation. For model specification, we used the contextual local views representation scheme. A total of 3 contextual local views were defined for context values or assignments: $belongsTo = f, familyOut(f) = in$; $belongsTo = f, familyOut(f) = out$; and $barking = d$, where f, d are the parameters to represent a specific family and dog respectively, and in, out are the domain values of the specific family f's attribute $familyOut(f)$. The complete relational CSN formed is as shown in Figure 4.3(b) (page 59). In the model, the attributes $belongsTo$, $dogOut$, and $bowelProb$ are associated with object class dog; and the attributes $familyOut$ and $lightsOn$ are associated with the object class family. Depending upon the number of dogs and families, each attribute was instantiated correspondingly. The attributes $barking$ and $hearBark$ are assumed to be instantiated depending upon the number of bark heard. We utilized both the methods for probability specification, for instance, the CPF probability table for

belongsTo and *barking* were specified as a uniform distribution function over the number of families and the number of dogs respectively, while the probability table for $familyOut$ was manually specified. The relational CSN was first rolled out into a propositional CSN. The CSN was then converted to generate an equivalent BN representation for comparison. Query can be asked about any variable(s) in the induced network. The system was tested with several combinations of different context as well as non-contextual attributes. Some example queries included: *P(familyOut(F1)|hearBark1=true)*, $P(belongsTo(D1)|hearBark1 = true)$, $P(familyOut(F2)|belongsTo(D1) = F1)$, $P(belongsTo(D2)|belongsTo(D1) = F1)$, *P(bowelProb(D1) | belongsTo(D1)= F1, familyOut(F1)=out, hearBark1=true)*, *P(familyOut(F2)| belongsTo(D1)=F2, familyOut(F1)=out, barking=D1, hearBark1= true)*.

We then discuss the evaluation on generalization of Example 2 to multiple girls, food types, boys and restaurant. A total of 2 contextual views were defined to encode the relational model. The attribute *accompany* is associated with object class boys, the attribute *likes* is associated with object class girls, the attribute *quality* is associated with object class food types, and the attributes *order* and *rating* are a relation between the object class boys and restaurants. Again, the query can be asked about any variable(s) in the induced network. Some example queries include the different combinations of instantiation of the following relational attributes: $P(accompany|rating)$, $P(rating|accompany)$, $P(rating|order)$, $P(likes|rating, quality)$, *P(likes|accompany, order)*. The evaluation strategy follows that of Example 1.

8.2.2 Experiments

We report the two types of test performed: a) without any context evidence (situational uncertainty), and b) with some context evidence given (situation-specific). The tests showcase that the context may or may not be known *a priori*, and can change under different conditions. For the first test without the context evidence, the system was evaluated by running the LBP over the full CSN and no structural adaptation strategy was applied. For the second case, we used context structural adaptation strategy to utilize more context-specific graph structures for inference. Inference was carried out on all the generated sub-graphs. The results reported below in Table 8.3 are based on the largest subgraph generated.

8.3 Experimental Results

We compared the proposed LBP inference algorithm on CSNs with that on equivalent BNs on a Pentium M 1.8Ghz machine with 1 GB RAM. Both are our own implementations using similar underlying libraries in MATLAB. We also compared our results with more efficient LBP and junction tree algorithms from the BNT toolbox [Murphy, 2002a]. We note that many previous efforts in this field, including [Boutilier et al., 1996; D'Ambrosio, 1995; Mahoney & Laskey, 1998; Jaeger, 2001, 2004] have not shown an empirical evaluation of their proposed methods. Previously proposed data structures to exploit local parameter decomposition are known to work only with some special inference algorithms. Hence, it has been difficult to compare with them empirically. We have highlighted the differences in the design principles of these methods earlier in Figure 3.5. Only recently, few researchers [Darwiche, 2002; Poole & Zhang, 2003; Chavira & Darwiche, 2005] have been successful in demonstrating an empirical evaluation of their proposed methods. They have followed a strategy of

comparing their new CSI-based inference method with that of its traditional equivalent BN inference algorithm. Therefore, we adopt a similar comparison methodology as in [Darwiche, 2002; Poole & Zhang, 2003; Chavira & Darwiche, 2005] to compare LBP inference algorithm on CSN with that on equivalent BN.

We have considered the feasibility of comparing our work with the previously proposed CSI approaches. We have realized, however, that: a) no previous work has utilized the loopy belief propagation inference method to exploit local parameter decomposition; b) all but one of the previous approaches [Boutilier et al., 1996; D'Ambrosio, 1995; Mahoney & Laskey, 1998; Poole & Zhang, 2003] propose an exact inference method for propositional BNs. Exact inference methods are known to fail on relational networks as the instantiated networks can become quite complex; c) none of the proposed exact inference CSI methods for structured CPTs are guaranteed to work better than their traditional BN counterparts. [Poole & Zhang, 2003] have shown the advantage of their exact inference method over only one network. It has been really hard to understand and identify the network patterns in which a CSI-based algorithm has a high chance of outperforming any other traditional BN inference algorithms; and d) while an approximate inference algorithm has been proposed in [Milch, 2006], the author themselves point out that their implementation is extremely slow. In view of these reasons, we have explained theoretically why the other CSI data structures and their inference methods are not suitable to address the problem of situational variations in Chapter 3. We would aim at making more extensive empirical comparisons in future, perhaps for specific classes of problems that are applicable to all the different CSI frameworks, when implementations of other CSI frameworks are more readily available.

We examined the effectiveness of the proposed methodology for the following:

- Advantages of modeling local parameter decomposition

- Efficiency of inference
- Impact of dynamic model adaptation
- Effect on parameter learning

8.3.1 Representation and Inference

Table 8.1 shows the more extended evaluation of runtime behavior on worst cases with no context evidence on Example 1. The inference time is an average over 3 runs over the same network. Each different $hearBark$ node is assumed to have a different $barking$ node. Table 8.1 shows that: a) CSN encoded much fewer total parameters and induced smaller maximum parameter width than the corresponding BN; b) With fewer contexts (first two cases), CSN might occupy larger memory size and take more time for inference; this is expected due to the extra context function nodes in the graph; c) With increasing number of variables and contexts, CSN outperformed BN significantly in both memory size occupied and inference time taken. For instance, in Case 5 (Net ID 5) in Table 8.1, CSN only took about 30 secs while BN took 11 mins. BNT's LBP took 5 mins while junction tree failed. Case 6 (Net ID 6) ran on CSN but the evaluation of an equivalent BN raised an "out of memory" error as the CPT size was really large. The calculated probability marginal for all the attributes in the CSN and BN matched within 0.01 tolerance limit.

Table 8.2 shows the runtime behavior in worst cases with no context evidence on Example 2. Table 8.2 shows the results with only 2 boys, 2 restaurant, and 6 context variables with multiple girl and food types in domain. In this table, we demonstrate the impact of just increasing the domain values (or the context assignments) of each context variable while having the same number of context variables, 6 in this case, in the network. The outcome space of $likes(g)$ of each girl is also assumed to be a function of $foodTypes$ and each girl may have different probabilities for foods

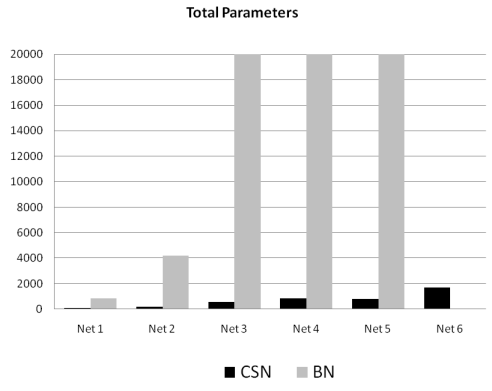
(a) CSN Vs BN (our implementation)

NET	NET FEATURES				CSN						BN			
ID	F	D	B	Ct	**V**	**CF**	Θ	Max θ width	Mem size	Inf time	Θ	Max θ width	Mem size	Inf time
1	4	5	1	10	25	64	106	12	82.36	3.15	836	160	33.78	1.06
2	6	5	2	13	31	94	166	18	126.54	5.1	4216	768	94.21	1.45
3	10	10	5	25	60	295	575	30	473.95	19.21	256175	20480	4164.7	10.12
4	10	15	5	30	75	430	850	30	745.12	31.25	2765050	491520	44332.5	194.7
5	15	10	5	30	70	405	790	45	685.15	30.37	9881840	983040	158192.1	667.62
6	20	15	10	45	105	830	1700	60	1645.97	74.35	NA	NA		

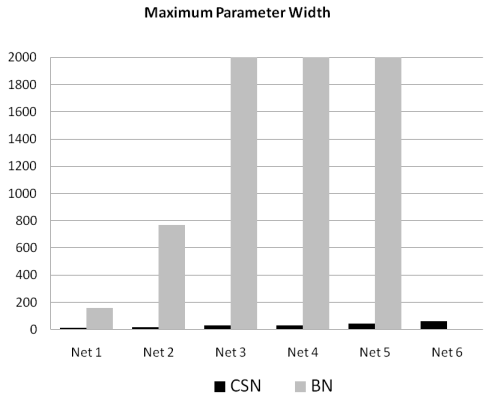
(b) BNT's result on the same networks

NET	BNT	
ID	LBP time	JT time
1	1.07	0.17
2	1.53	0.15
3	8.37	1.82
4	151.56	NA
5	304.98	NA
6	NA	

Table 8.1: *Comparison of CSN and equivalent BN with no context evidence (F: number of Families; D: number of Dogs; B: number of Barks instantiated; Ct:total Contexts; Θ: Parameters; JT: Junction tree; Time: sec; Memory: kb; NA: out of memory error)*

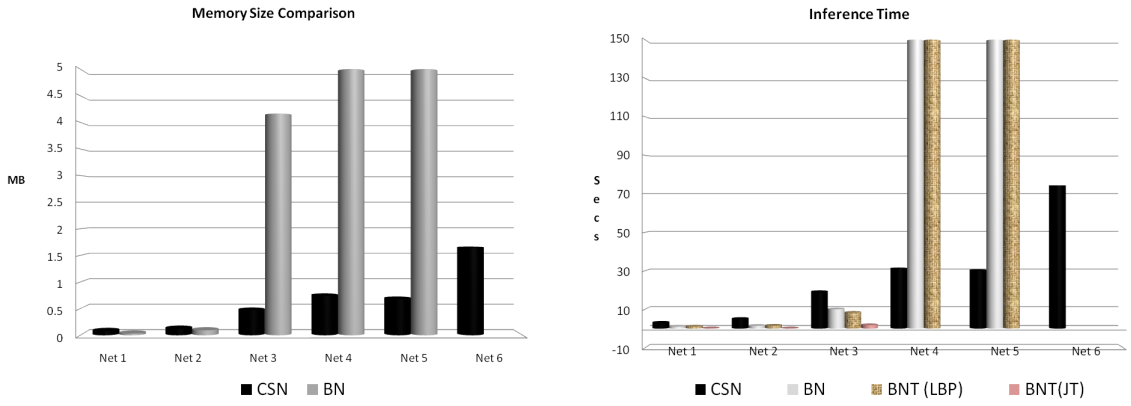


(a) Total Parameters



(b) Max Parameter Width

Figure 8.2: *Comparison of Parameters: CSN Vs. equivalent BN based on Table 8.1*



(a) Memory Size **(b)** Inference time

Figure 8.3: *Comparison of Memory Size and Inference time: CSN Vs. equivalent BN based on Table 8.1*

according to her taste and preferences, for instance in Net ID 1 in Table 8.2 with five food types, each $likes(g)$ attribute has 5 domain values.

NET			CSN						BN				BNT	
ID	G	FT	V	CF	Θ	Max θ width	Mem size	Inf time	Θ	Max θ width	Mem size	Inf time	LBP time	JT time
1	3	5	18	42	301	60	54.43	2.42	6661	1500	84.53	0.98	1.38	0.35
2	5	5	20	52	473	100	69.16	3.09	250673	62500	2526.54	4.31	3.31	1.46
3	5	8	23	67	1235	280	96.08	4.06	4595763	1146880	42093.54	66.34	34.78	NA
4	8	8	26	82	1934	448	123.30	5.22	NA				NA	NA
5	5	10	25	77	1943	450	116.04	4.59	18041023	4500000	160676.96	237.87	NA	
6	20	20	50	202	30998	7600	560.2	16.66	NA					
7	50	50	110	502	492998	122500	5009.08	49.73	NA					

Table 8.2: *Comparison of CSN and equivalent BN with no context evidence (G: number of Girls instantiated; FT:number of Food types instantiated; U:Users; R: Restaurants; Θ: Parameters; JT: Junction tree; Time: sec; Memory: kb; NA: out of memory error)*

Table 8.2 highlights that with an increase in distinct context values, the runtime of LBP over CSN is slightly increased while there is an exponential increase in runtime of LBP over BN. For instance, Net ID 3 has three more food types (5 Girls, 8 food types) than the food types in Net ID 2 (5 Girls, 5 food types), the inference time in CSN increases to about 4 secs from 3 secs while in BN, it jumps to 66 secs from 4

secs. In Case 4 with 8 Girls and 8 food types, BN runs out of memory on both our's as well as BNT's implementation. In Case 5, with only 5 Girls but 10 food types, our BN implementation computes the result in about 4 mins while BNT's implementation throws an out of memory error. In some real-world applications, the context values for a context variable may be quite large, for instance the total food types may be more than 10. CSN could still compute results after an arbitrary increase in the number of Girls and food types, as shown in Net ID 6 and 7. Generalization to multiple boys and restaurant produced similar results.

A CSN compactly represents the joint distribution with substantially fewer parameters as shown in the Figure 8.2. In general, as described earlier in Section 5.5 (page 83), if there are k context function nodes each having cp binary contextual parents, then there are only $k.2^{cp+1}$ entries in k CPFs $\ll 2^{p+1}$ entries in a CPT of BN with p binary parents. The factor width, i.e., the size of the scope of the factor, is much smaller in a CSN than in a BN. Therefore, there are fewer multiplications and additions involved in a CSN. Like on BN, the complexity of LBP on CSN is linear with respect to the parameter size but the total parameter size is much smaller than the corresponding BN. Therefore, the runtime performance of CSN becomes significantly better than that of BN with respect to increase in the number of context variables and/or values, as shown in Figure 8.3.

Table 8.3 shows the effects of context evidence (situation-specific inference) on the first 3 Net IDs from Table 8.1 where BN performed better in runtime over CSN with no context evidence. Context evidence was randomly selected, for instance, for Net ID 2 and Case 2a in Table 8.3, we assumed evidence about any two families and dogs of the total 6 families and 5 dogs are given. Table 8.3 highlights: a) With context evidence, the number of variables, parameters, memory size of CSN decreased significantly with adaptation; b) Inference time became comparable to BN with just a

NET	EVIDENCES			CSN							BN			
ID	FO (f)	BT (d)	BO	V	CF	Θ	Max θ width	Mem size	Adapt time	Inf time	Θ	Max θ width	Mem size	Inf time
1a	1	1	0	21	49	80	10	61.20	0.26	2.23	463	160	27.84	1.04
1b	2	2	1	13	28	42	8	32.79	0.28	0.86	166	32	23.16	0.95
1c	2	3	0	15	29	47	10	34.73	0.23	1.25	271	160	24.83	0.94
2a	2	2	1	22	53	90	14	66.87	0.25	2.45	870	192	40.80	1.40
2b	3	3	1	18	38	66	13	46.66	0.26	1.51	463	160	94.21	1.42
3a	3	3	1	47	195	387	25	291.66	0.48	11.21	60805	10240	1039.2	5.6
3b	5	5	2	38	135	266	21	191.42	0.421	7	36390	10240	648.75	4.78

Table 8.3: *Comparison of CSN after adaptation and equivalent BN given context evidence(s). NET ID 1 to 3 are the same networks as that in Table 8.1. ID 1a, 1b, 1c means the same network ID 1 evaluated with different context evidences. (FO(f): number of familyOut's observed; BT(d): belongsTo's observed; BO:barking's observed; Θ: Parameters; Time:sec; Memory:kb).*

few context-value observations; c) Adaptation time was usually short as compared to reduction in inference time that's achieved. But modifying a BN with large CPT sizes at inference-time is time consuming as noted in [Alvarado & Davis, 2005]. Adaptation to accommodate only contextually-relevant variables in a CSN also reduces the efforts in eliciting irrelevant observations, e.g., by minimizing the number of questions asked to a user.

We have so far only examined the inference efficiency of the LBP algorithm in CSN. LBP is an approximate algorithm which iterates the beliefs repeatedly over the network until they converge. Though cut-set conditioning [Pearl, 1988], an exact inference method, can be applied directly on CSN for smaller graphs, we expect the run-time efficiency for exact inference to be further improved with the recent recursive conditioning algorithms [Darwiche, 2001], as discussed in Chapter 6.

We had two different implementations utilizing different message updating strategies for the message passing algorithm as discussed in Section 8.1. Table 8.4 shows that runtime results are dependent on the way message passing is implemented and second implementation is about 40% faster than the first. The results in this chapter are based on the first (slower) implementation of message passing algorithm.

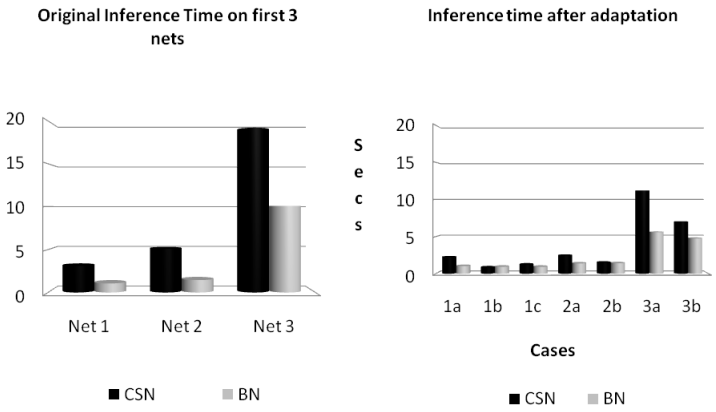


Figure 8.4: *Comparison of Inference time: Original Vs After Adaptation*

Number of Variables	Implementation 1	Implementation 2
59	5.6	3.34
150	18.28	10.7
280	37.18	21.98
500	75.78	48.17

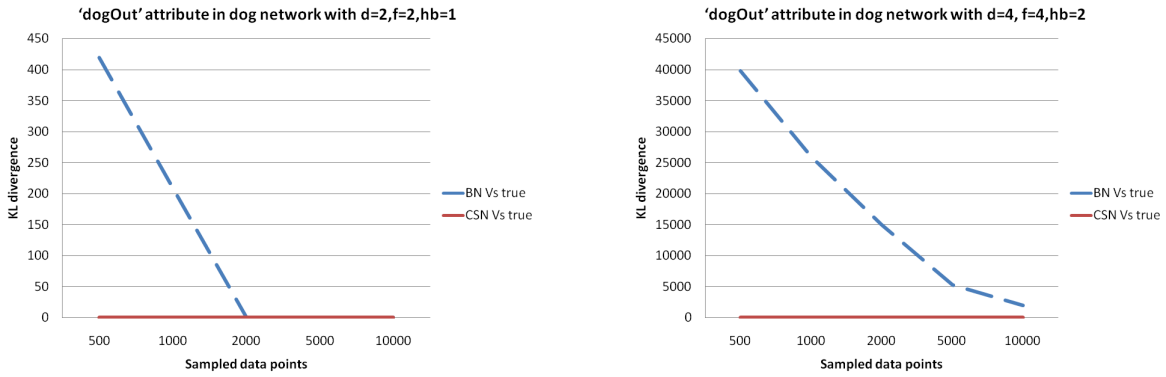
Table 8.4: *Comparison of speed (sec) in two different implementations of message passing*

8.3.2 Parameter Learning

We tested the correctness of the implemented parameter learning module. As we need context-specific independence (CSI) in the overall joint probability distribution for testing, we first converted the dog network in Figure 2.2(b) (page 18) of Example 1 to an equivalent dog BN (referred to as dog-BN here). We, then, sampled data points from this dog-BN. We sampled the following number of data points: 500, 1000, 2000, 5000, and 10000. The parameter learning module was then used to estimate the parameters for the CSN in Example 1 (dog-CSN) using these sampled data points.

The same procedure was followed for learning the parameters for Example 2 as well. For comparison, we also implemented a parameter learning module using standard maximum likelihood estimate (MLE) learning for BN [Buntine, 1996] over discrete probability distribution in MATLAB.

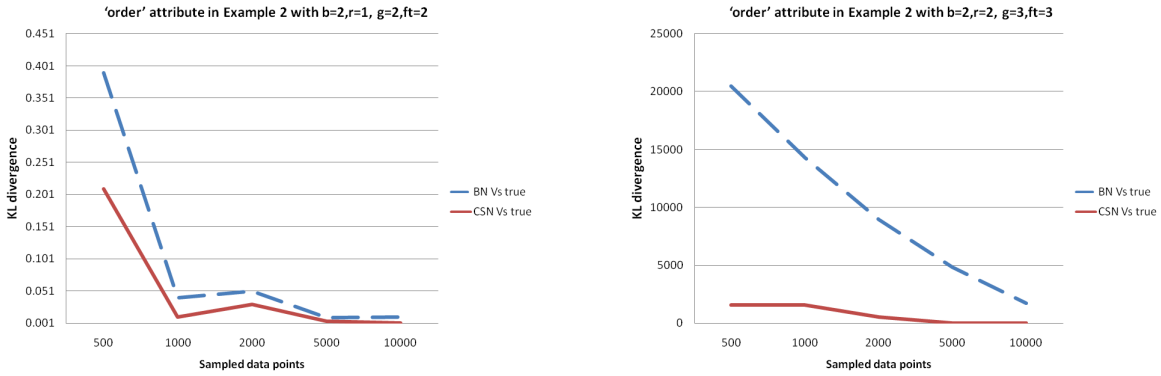
We applied the Kullback−Leibler (KL) divergence method for comparing the error bounds between the probability parameter estimated and the true probability parameters. KL divergence, represented as $D_L(P||Q) = \sum_i p_i log(p_i/q_i)$, is a commonly used asymmetric measure of the difference between two probability distributions – P, the true distribution and Q, the estimated distribution. Intuitively, KL measures the expected difference in the number of bits required to code samples from P when using a code based on P, and when using a code based on Q. The probability parameters in the original dog-BN constitute the true distribution. The probability parameters learnt constitute the estimated distribution.



(a) KL divergence **(b)** KL divergence

Figure 8.5: *KL divergence of parameters learnt for attribute dogOut using CSN and BN from the true distribution of dog-BN used for sampling data points. Learning in CSN reached a good approximation of the parameters even with 500 sample points. Note that the maximum point on the Y-axis of the two sub-figures are drawn different to properly reflect the large KL divergence in figure 8.5(b)*

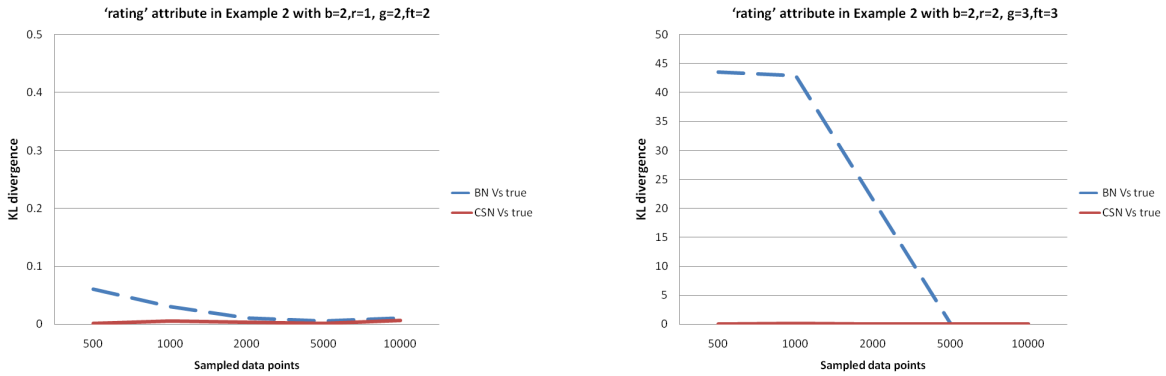
Figure 8.5 plots the KL divergence averaged over 3 runs between the true and estimated parameters for attributes *dogOut* in Example 1 for different sets of sample data points. In Figure 8.5(a), the data points were sampled from the instantiated dog-BN with 2 families, 2 dogs and 1 bark heard, equivalent to Figure 2.3. The results in this figure highlight that even with 500 sample points, parameter learning CSN achieves a good approximation, while with the same number of points, the error bounds of the parameters learnt with MLE in BN are quite high. In Figure 8.5(b), the data points were sampled from more complex 24 variables BN with 4 families, 4 dogs and 2 bark heard. The results in figure 8.5(b) show that the BN learner did not give a good estimate even with 10000 sample points, while the CSN learner achieved a good estimate with 10000 sample points.



(a) KL divergence **(b)** KL divergence

Figure 8.6: *KL divergence of parameters learnt for attribute order using CSN and BN from the true distribution of dog-BN used for sampling data points*

Similarly, Figures 8.6 and 8.7 show the KL divergence results for attributes *order* and *rating* in Example 2. In Figures 8.6(a) and 8.7(a), the data points were sampled from the instantiated BN with 2 boys, 1 restaurant, 2 girls and 2 food types as shown



(a) KL divergence **(b)** KL divergence

Figure 8.7: *KL divergence of parameters learnt for attribute rating using CSN and BN from the true distribution of dog-BN used for sampling data points*

in Figure 7.2. In this case, learning in both BN and CSN provided a good estimate for both attributes with the CSN estimates being closer to the true distribution. In Figure 8.6(b) and 8.7(b), the data points were sampled from a more complex BN with 16 variables, with 2 boys, 2 restaurant, 3 girls and 3 food types. The results in this case show that for attribute *order*, the BN learner could not achieve a good estimate even with 10000 points, while the CSN learner achieved a good approximation with 5000 sample points. Similarly, for the *rating* attribute, the CSN learner provided a good approximation with only 500 sample points, in comparison to 5000 sample points needed by the BN learner.

Both the results above show that given a distribution with CSI, the KL divergence is smaller in case of the parameters learnt in a CSN than those in an equivalent BN. So, the parameter learning module in CSN can converge faster to the true distribution with respect to the number of sampled points. The results shown here are similar to those shown earlier in [Friedman & Goldszmidt, 1996], where they assumed CSI in the underlying distribution for estimating better parameters in BN structure learning.

8.4 Case Study 1: Modeling Coronary Artery Disease

8.4.1 Purpose of case study

To demonstrate how CSNs can handle problems involving context in a real-life application domain – medicine, we formulated a CSN with a small case study on an actual clinical situation of Coronary artery disease and informally evaluated its effectiveness. We have earlier presented and evaluated a preliminary version of the CSN framework based on the limited scenarios from this case study in [Joshi & Leong, 2006; Joshi et al., 2007][1].

8.4.2 Background and Motivation

Coronary Artery Disease (CAD) is the most common cause of sudden death in many developed countries. It is a complex multifactorial process that involves lipid deposition on arteries of the heart, macrophages, blood pressure, rheology of blood flow, smooth muscle proliferation, thrombogenesis, platelet aggregation, insulin resistance and other factors. In short, it occurs when arteries supplying blood to the heart vessels (coronary arteries) becomes hardened and narrowed. With the growing affluence of Singapore, the disease patterns and health needs of Singaporeans have changed widely. Chronic degenerative diseases such as cancer and cardiovascular disease have emerged as the major causes of death.

Chest pain is the chief complaint in CAD. However, CAD is often poorly diagnosed as the chest pain spans a wide clinical spectrum of diseases from the trivial to a life threatening. So, the patients with unstable angina (chest pain) can inadvertently be discharged from an emergency department and adverse outcomes in these patients

[1] Publishers' approvals have been obtained for re-using the published material in this thesis

represent a significant cause of death [Pope et al., 2000]. Hence, early identification of CAD is believed to be of prime importance. Recently, there has been a growing interest in having proactive applications that can help in screening and alerting clinicians on the risk of CAD, both in the emergency departments as well as at homes.

BNs have been shown to be useful as screening and alerting tools in the clinical applications [Aronsky & Haug, 1999]. They can both be learnt from data as well as extracted from experts. Such probabilistic networks can be extremely useful for providing timely alerts for CAD. However, given the inability to collect an adequate size of health data at home, the expert-based knowledge engineered probabilistic networks [Intille, 2004] may be more useful for such type of applications.

8.4.3 Model Formulation and Construction

The risk for CAD depends upon each patient's characteristics, and in each patient it can occur due to the interplay of several medical reasons. Among the more established risk factors for CAD are: family history (genetic factors), plasma lipid, lipoprotein, plasma lipoprotein, diet, gender, race, elevated blood pressure, physical inactivity, etc.

Model Preparation

In this case study, we formulate a simplified problem for representing context-sensitive medical knowledge in CSN. We based our model development on a study performed by Chen et al. [2005]. The study combined the use of patient medical profile, family history and microarray-based genotypic information consisting of gene biomarkers for the CAD prediction. They worked closely with biomedical experts to construct a BN that can be used for the prediction of CAD. We used a CAD dataset from a local Singapore hospital. The dataset contained 43 variables of which we selected 38

Risk Factors	Number of States	Domain values
CAD	2	Healthy or diseased
Age (A)	2	Above 55 or below 55
Gender (G)	2	Male or female
Race (R)	3	Indian, Chinese, Malay
Diabetes (DM)	2	Healthy or diseased
Hypertension (HY)	2	Healthy or diseased
Smoker (S)	3	Smoker, non-smoker or ex-smoker
Family History of CAD (FCAD)	2	Yes or No
Family History of HY (FHY)	2	Yes or No
Family History of DM (FDM)	2	Yes or No
BMI	2	High or Low
27 Genetic biomarkers	2	High or Low

Table 8.5: *Domain attributes in Case study 1: CAD*

variables that were correlated with CAD. Table 8.5 shows the 38 data attributes, and their domain values. Out of 38 attributes, 11 described patient's profile and about 27 were gene biomarkers. The attribute *Race* and *Smoker* had 3 states while the others were binary attributes.

Representation in Contextual Local Views

For screening under limited resources, the modeling framework should exploit context-sensitive medical knowledge to narrow down the state space for each patient. To formulate a context-sensitive model, we made some simplified assumptions. We assumed two context variables *Age* and *Race*, both of which are directly correlated with CAD. Attribute *Age* was discretized into two binary states. In Chen et al. [2005]'s study and also in other medical studies, several gene-markers were found to be closely associated with each other and with Race. We assumed contextual independence among the groups of biomarkers and divided them based on their associations with the Race of the person for our case study. Based on expert suggestions, we assumed that the following medical facts needed to be modeled:

- The risk of CAD is the same for males and females with age>55 (assuming menopause at age 55 for simplicity), otherwise the risk for males is much higher than for females
- The attributes related to Family history are only important in young males with age less than 55 years
- Some recent research has shown that the race of a person is correlated with the Single Nucleotide Polymorphism (SNP) biomarkers and can help in determining the SNPs relevant for CAD. For simplicity, we assumed different groups of biomarkers are relevant for different races of individuals, for example, the SNP1 group of biomarkers is relevant for the Indians while the SNP2 group is relevant for the Chinese.

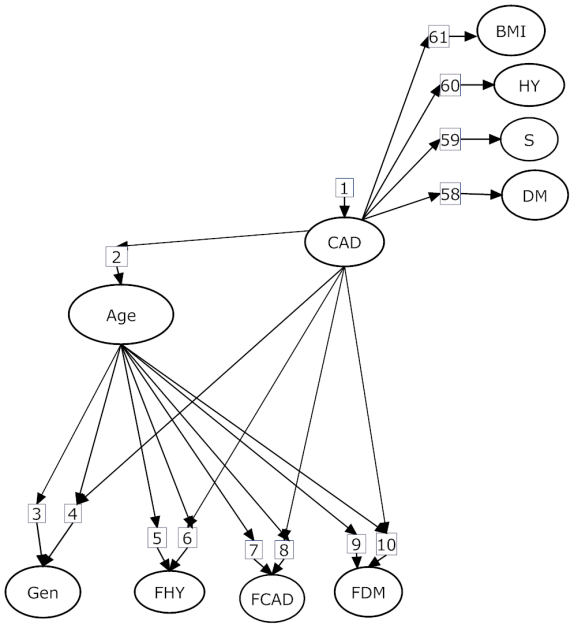


Figure 8.8: *Contextual local views for context Age in CAD model. Figure captures all the contextual dependencies for all domain values of context variable Age together as one view. Context values for the CF nodes are not shown for clarity.*

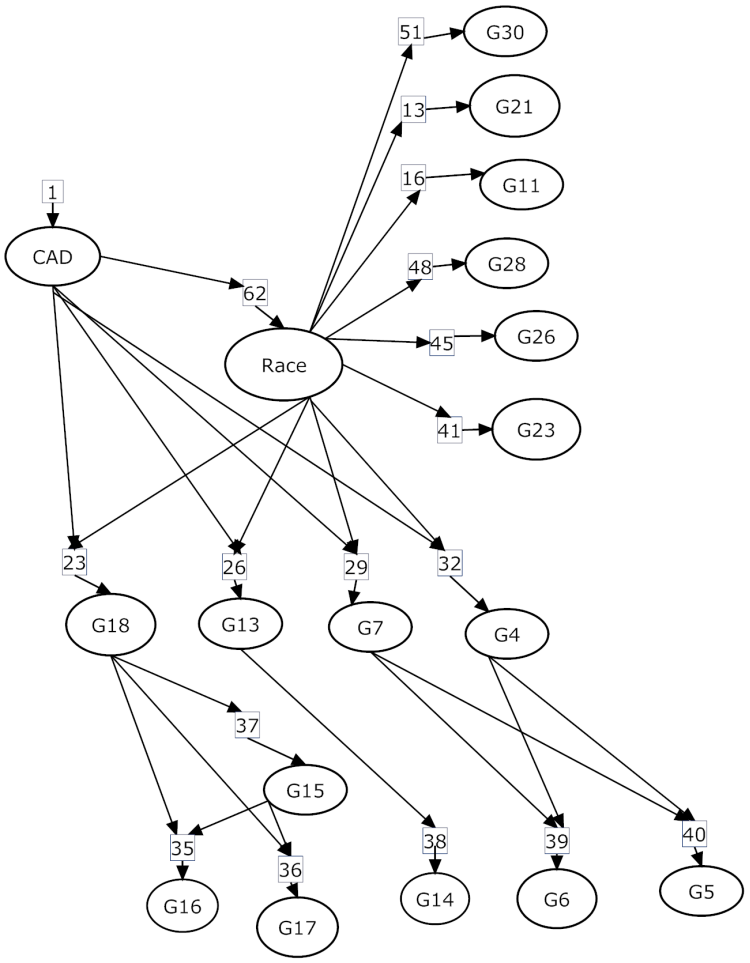


Figure 8.9: *Contextual local view for CAD model using context: Race = 'c'. Race value renders CAD independent of attributes G21,G11,G23,G26,G28 and G30 in this case.*

There are 6 relevant context scenarios: $(Age > 55, Race = c)$, $(Age > 55, Race = i)$,$(Age > 55, Race = m)$, $(Age < 55, Race = c)$, $(Age < 55, Race = i)$, $(Age < 55, Race = m)$. Figures 8.8 and 8.9 show the two ways of representation: 1) capturing all the context-specific dependencies for all the context values of the local context in a single view, and 2) capturing them separately in multiple views. We captured the information in 4 contextual views: 1 for context attribute *Age*, shown in Figure 8.8 and 3 for context attribute *Race*, shown in Figures 8.9, 8.10, and 8.11. These

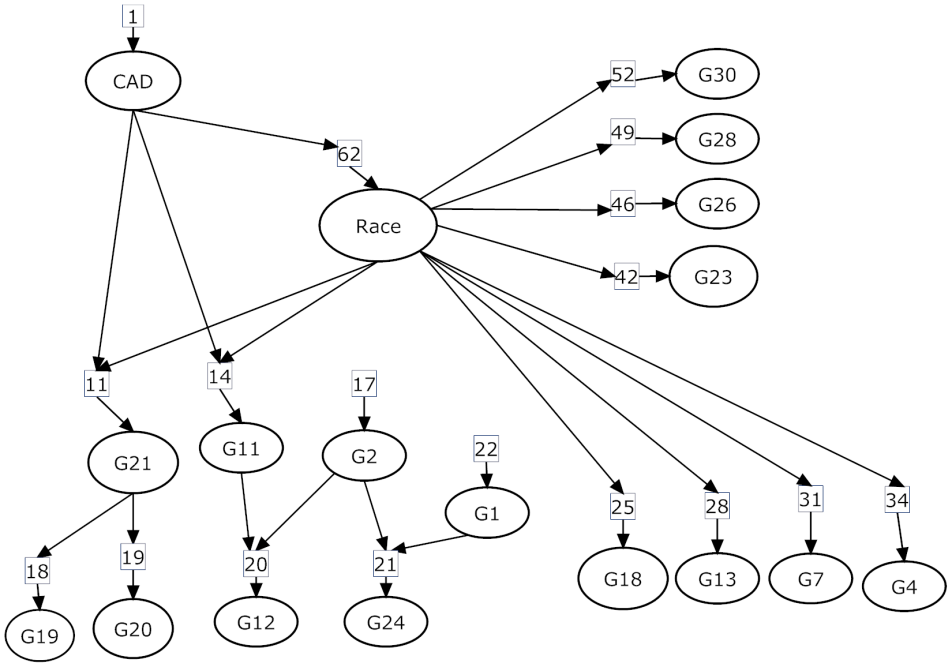


Figure 8.10: *Contextual local view for CAD model using context: Race = 'i'. Race value renders CAD independent of attributes G23,G26,G28,G30,G18, G13,G7 and G4 in this case*

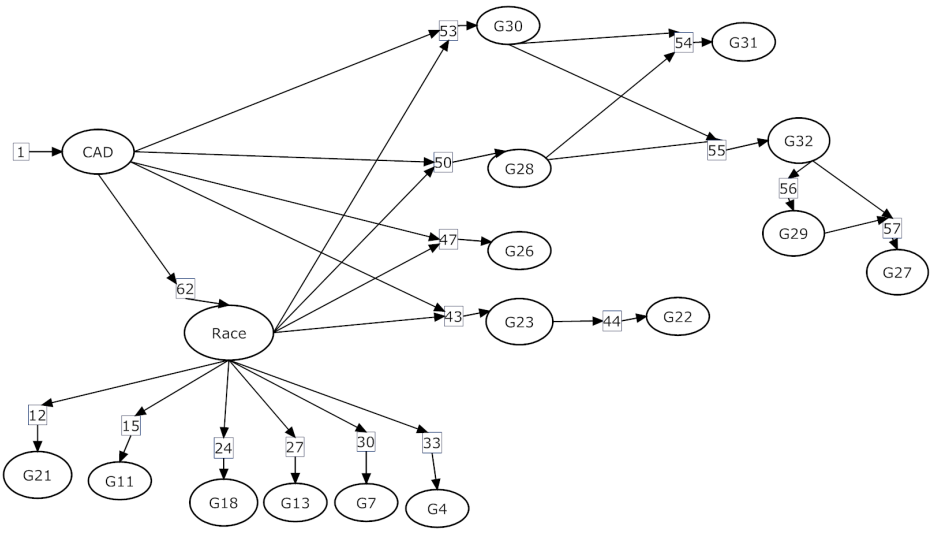


Figure 8.11: *Contextual local view for CAD model using context: Race = 'm'. Race value renders CAD independent of attributes G21,G11,G18, G13,G7 and G4 in this case.*

contextual local views were rolled out into a complete CSN for the inference. The main query attribute was CAD.

Probability Assessment: The parameters were based on Chen et al. [2005] study.

8.4.4 Model Evaluation

Experimental Setup: Overall, the reasoning network we selected had 38 variables. We formulated cases based on the different values that the attributes can take. The inference can be either situation-specific given an evidence for at least one context attribute, or under situational uncertainty with no context information. CSN effectively supports different possible scenarios for patient-specific inference, while preserving both the generality and the context-specific representations. CSN allows answering multiple queries about the patient that are either situation-specific or involve situational uncertainty. We performed the evaluation on different patient cases that can be constructed by assigning different values to each observed attribute. We first show the comparison between a CSN and an equivalent BN with no context evidence. We then present two of the patient cases for situational-specific inference where the evidence about both context attributes *Age* and *Race* are given.

Results

Situational uncertainty with no context information: Chen et al. [2005] carried out an extensive study together with the domain experts to built two networks for predicting CAD, one a smaller network with 30 variables and the other with 38 variables. To test and compare the runtime with no context evidence, we choose these 2 complex network structures. The attribute CAD is the query variable. Figure 8.13 shows the runtime comparison in the 2 networks. Results show that inferences in CSN and BN have a comparable runtime over both the networks. The calculated probability

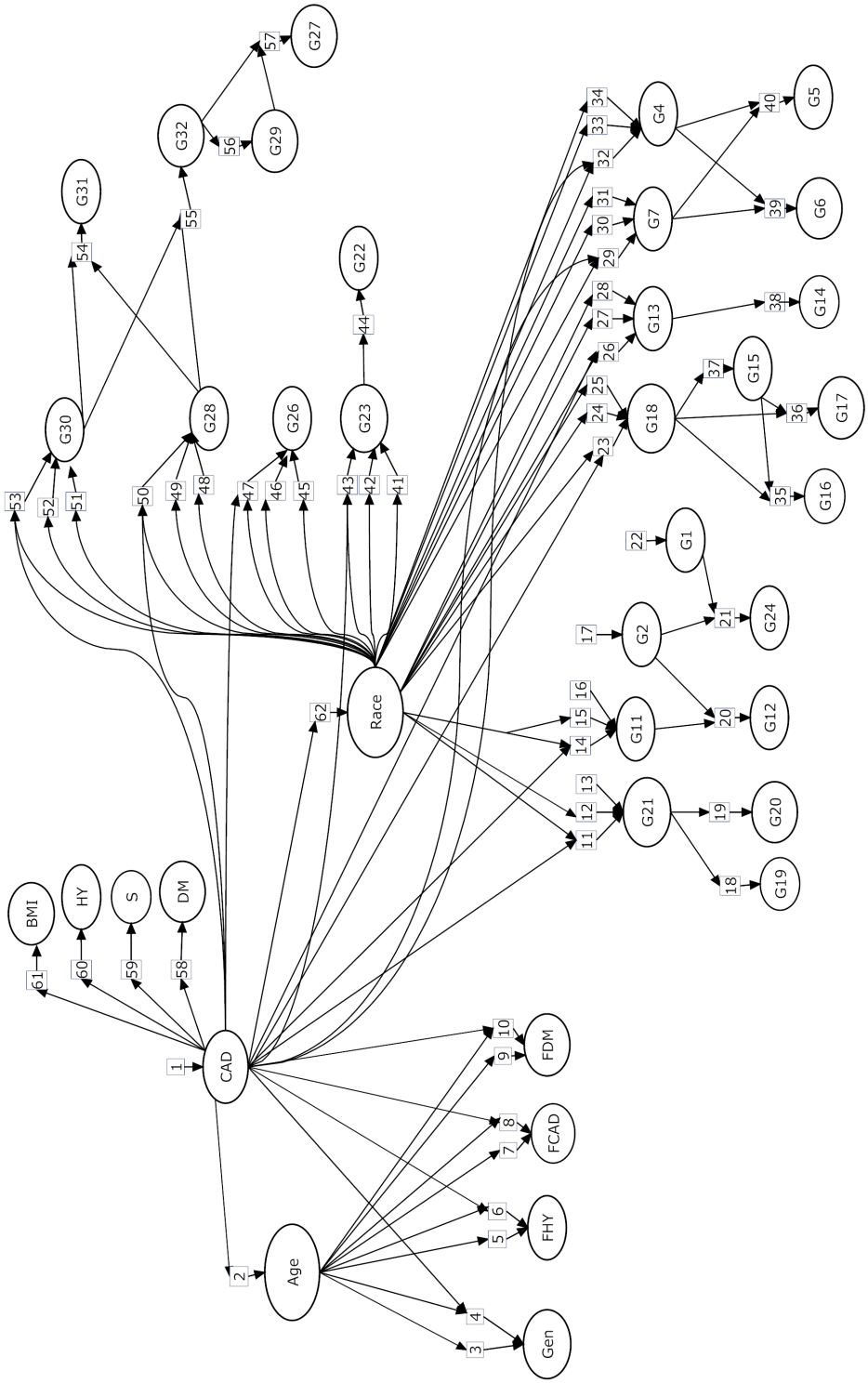


Figure 8.12: *Complete underlying CSN build from contextual local views*

marginal for each attribute was the same within the 0.01 tolerance limit.

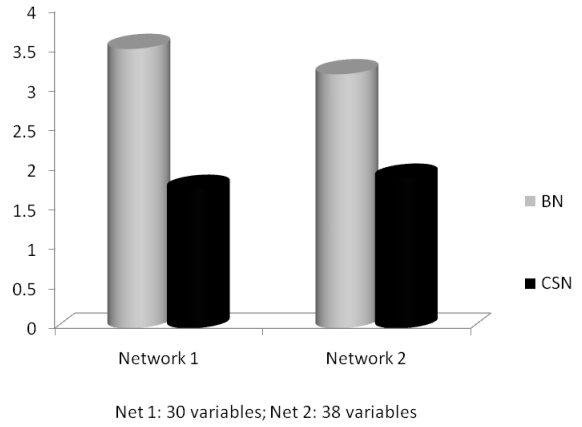


Figure 8.13: *Comparison of Inference time over 2 networks*

Situation-specific: Table 8.6 shows the results for following six different cases:

- *Patient Case 1: A young Chinese patient of age < 55 years*
- *Patient Case 2: An old Chinese patient of age > 55 years*
- *Patient Case 3: An young Indian patient of age < 55 years*
- *Patient Case 4: An old Indian patient of age > 55 years*
- *Patient Case 5: A young Malay patient of age < 55 years*
- *Patient Case 6: An old Malay patient of age > 55 years*

Given the context values for Race and Age in this case, structural adaptation was performed on the CSN and situation-specific inference was carried out . The rest of the attributes were assumed not to be observed. In the patient Case 1 in Table 8.6, the

Property	Original	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Total Variables	38	21	17	19	15	20	16
Total Parameters	115	47	39	37	29	39	31
CSN memory size (kb)	135.1	25.0	19.2	21.9	16.1	23.6	17.8
Inference Time (secs)	2.92	0.60	0.59	0.45	0.34	0.71	0.56

Table 8.6: *Comparison of CSN performance on situation-specific inference for different cases with that on the original full CSN graph*

total number of variables required was 21 as Gender and the attributes related to the Family History were relevant in this case. Inference time was substantially reduced as compared to that in the original CSN. In patient Case 2, the model required only 17 variables instead of 21. Gender, and all the attributes related to the family history were also rendered irrelevant in this case. The model in patient Case 3 required only 19 variables instead of 21 in case 1 as some of the genes biomarkers were also rendered irrelevant for this case.

Table 8.6 shows that the context structural adaptation strategy can significantly decrease the number of variables, parameters, memory size and inference time. Given any random selection of patient profile, structure decomposition may help exploit more patient-specific structures for faster inference even with limited memory resources.

In this case study, we used an example of CAD to illustrate reasoning in medical domain under context. We showed how local context modeling can help partition the knowledge into several scenarios and how these can be transformed into a CSN which can be effective in answering several patient-specific queries.

8.5 Case Study 2: Model Formulation using Guidelines

8.5.1 Purpose of case study

This second case study further demonstrates that CSN framework is not only expressive enough to handle a range of problems involving context in medicine, but also illuminates how a context-sensitive reasoning framework can support newer knowledge acquisition techniques. This preliminary study describes how clinical guidelines can be useful as a source for knowledge engineering in a contextual model. We sketch the formulation of CSN using clinical guidelines for Community-acquired Pneumonia (CAP) and then evaluate the effectiveness of the model formed.

8.5.2 Background and Motivation

Expert modeling of the BN structure in medicine is known to be time consuming because of the following reasons: a) Technical issues such as the complexity of medical knowledge and the difficulty in probability elicitation; b) Social issues such as the communication gap in multidisciplinary teams, and c) Administrative issues such as the unavailability of experts at all times. For example, in modeling a BN for a liver disease [Onisko et al., 2000], authors took several months to construct the BN. On the contrary, automated structure learning approaches are faster. However, due to the administrative issues, even obtaining an appropriate clinical dataset may be hard. For instance, most of the pneumonia patient records are still paper-based in Singapore, a country with one of the most advanced IT systems in Healthcare.

Clinical practice guidelines (CPGs) are an excellent source of expert knowledge. Recently, some efforts [Sanders, 1997; Zhu & Leong, 2000; Zhou, 2005] have indicated similarities between CPGs and Influence Diagrams (IDs), which are close cousins of

BNs. These efforts have argued that both the guidelines and the probabilistic decision models can be built from each other interchangeably. Therefore, we test the feasibility of using CPGs for authoring a probabilistic model. CPGs are inherently asymmetric in nature. Hence, a contextual framework can be useful for modeling using CPGs. We show the feasibility of the task by using part of some existing guidelines for CAP.

Clinical Practice Guidelines

Clinical practice guidelines (CPGs) are clinical algorithms for patient management, treatments, and screenings based on the scientific evidence; they are an excellent source of expert knowledge. The development of guidelines is significant for the following reasons: a) Reducing inappropriate variations in practice; b) Improving health care quality; and c) Helping to control costs. Although health care organizations develop a lot of paper-based guidelines, for instance, Wyatt [2000] calculated that each doctor receives about 15kg of clinical guidelines per annum, yet the adherence to them in routine use in the clinical settings to improve the patient outcomes has not been shown. The problem is that each patient can have a different line of management and treatment strategy based on the patient's medical records, past history and characteristics. Therefore, it is important to save physician's time by predicting relevant information proactively at the right stage of decision making. Guideline-based clinical decision support systems have been proposed for this purpose and is an important research area [Peleg et al., 2003].

Background: Community-acquired Pneumonia

Pneumonia is the most common cause of death from infectious disease in Singapore [Lim, 2006]. Pneumonia is an illness of the lungs and respiratory system in which the part of the lungs, alveoli, responsible for absorbing oxygen from the atmosphere become inflamed and flooded with fluid. A patient suffering from pneumonia may have symptoms such as cough with greenish or yellow sputum, high fever, shaking

chills, shortness of breath, chest pain, and headaches. CAP is one of the most common forms of acquiring Pneumonia.

If a patient presents with symptoms suggesting CAP, a doctor obtains a chest x-ray to confirm the diagnosis of pneumonia. Physician, then, stratifies the severity of pneumonia based on the lab reports, signs, symptoms, physical examination and medical history of the patient. The correct identification and appropriate outpatient treatment is a key aspect in the management of CAP because: a) the hospital management of CAP is much more expensive than the outpatient care; and b) low risk patients, about 75% of them with CAP, can be safely managed as outpatients.

CPGs for CAP: Singapore's clinical guideline for CAP as detailed in Lim [2006] aims to provide: a) better adherence to guidelines, b) more uniform process of care, c) appropriate site of care, and d) cost containment. We briefly describe the key points:

1. Complete guideline flow of CAP is described by Lim [2006] in Singapore as well as in [ICSI, 2006], the CPGs advocated in US.

2. Risk stratification is a key step in the management of CAP. The need for referral to hospital/emergency department and intensive care is based on the risk categories. Pneumonia severity index developed by Fine et al. [1997] is useful in risk stratification.

3. Out-Patient Management: Algorithm detailed in [ICSI, 2006] shows the outpatient management flow.

4. Wards Management: Lim [2006] details the algorithm for wards admissions and the treatment strategy.

5. ICU Management: Patients with physical and laboratory signs of severe illness should be admitted to ICU.

8.5.3 Model Formulation and Construction

We formulate the CSN model by constructing the Contextual Local Views for each context value and then translating them into a complete CSN model. Each contextual local view encodes the random variables related to the specific context value. The following steps were taken for model construction using CAP clinical practice guidelines:

Model Preparation

Firstly, CAP Guideline was divided into several stages based on the pneumonia workflow: a) Calculation of Pneumonia Severity Index (PSI); b) Outpatient Management; c) Wards management; d) ICU management.

Secondly, for each stage, we identified the random variables and context variables. For the ease of modeling, we divided them into categories, for example, Table 8.7 shows the categories and the corresponding variables for pneumonia severity and outpatient management. We use relational modeling to fix the direction of arcs for the entire class of the category and thus providing the flexibility while adding the extra attributes. The model constructed, however, may ignore some of the extra dependencies in the distribution.

Thirdly, for the ease of understanding and eliciting the relations among the variables in different contexts as per the CPG, we used decision diagrams for each context variable.

Representation in Contextual Local Views

Each contextual local view encodes the knowledge related to every context variable. For each contextual local view, we further grouped the above categories into two broad ones: hypothesis and observed to simplify selecting the direction of arcs. Hy-

Categories	Attributes
Demographics	Age, Gender
Social History	Nursing Home
Vital Signs	Temp, Pulse, BP, Respiratory rate
Lab Values	Urea, pH
Physical Appearance	Mental confusion
Medical History	Co-morbid Conditions
Characteristics	Oral Intake, Aspiration, Recent Influenza, Anti-microbial treatment
Treatment Categories	Outpatient, Wards, ICU
Context Attributes	Site of Care, PSI, Patient Characteristics

Table 8.7: *Domain attributes used in CAP case study*

pothesis variables are unobserved variables that may need to be inferred. Observed variables are those for which the evidence values can be obtained from the patient's medical record, for example, patient's profile, medical history, lab values and physical appearance. The actual model encoded in our prototype implementation is shown in Appendix B.5.

Probability Assessment

Probabilities were extracted based on the published literature as well as subjective assessments . CPG guidelines advocate the use of either the CURB score or the PORT score [Fine et al., 1997] (as explained in APPENDIX C) for the PSI prediction. For the PSI modeling, we used the published literature for the PORT score from Fine et al. [1997] to derive the required probabilities. For subjective assessments, we used the verbal-numeric scale published in van der Gaag et al. [2002]. This scale has previously been used to elicit probabilities from physicians in previous studies on Oesophageal Cancer. Subjective assessments were initially done based on the guidelines. The assessments are currently being fine-tuned along with an expert physician.

8.5.4 Model Evaluation

The model constructed is an approximate model in the sense that it leaves out some of the conditional dependences that may exist in the true distribution. The objective of this evaluation is to understand how the model built using a CPG performs on an actual clinical condition. We performed a preliminary evaluation and evaluated the prediction quality of the overall model built using CPGs based on actual patient cases.

Experimental Setup

The cases depict various patient scenarios in CAP, especially for the outpatient treatment; they have been taken from [Lim, 2006]. Different domain values of the attributes in Table 8.7 generate different permutations of patient profiles. As explained earlier, the correct identification and screening of outpatient treatment is more important in CAP, therefore we selected cases from [Lim, 2006] where an outpatient treatment was recommended. Table 8.8 show the patient cases used in the evaluation. Patient Case 1 depicts high severity of CAP infection while other cases depict scenarios of low severity CAP. Moreover, even though the CAP severity may be lower, the site of care may change depending upon some special patient characteristics as shown in patient Cases 3 and 4.

We tested different types of inference:

- Situational Uncertainty: When all context attributes are uncertain, the query was to predict: pneumonia severity index (PSI), Site-of-Care (SoC).
- Situation-specific: a) if PSI known: predict SoC ; b) if Site-of-care is known, predict possible treatment strategy

The gain in posterior probability of each domain state for the query attributes was

Patient Case	Patient Characteristics
Case 1	Age>50, LowBP, High RR, High urea, Confusion
Case 2	Age<50, BP, RR, Urea acceptable, No confusion
Case 3	Case 2 + Oral Intake ability
Case 4	Case 2 + Cannot take Orally
Case 5	Case 2 + Healthy, no comorbid, No Antimicrobial
Case 6	Case 2 + Healthy, no comorbid, Antimicrobial
Case 7	Case 2 + Comorbid, Antimicrobial
Case 8	Case 2 + Nursing Home Resident

Table 8.8: *Patient cases, BP: blood pressure, RR: respiratory rate*

calculated based on the difference between the prior and the posterior probability. Our prototype system recommended the domain state with the highest positive gain. The results were then matched with those recommended in the CAP guideline in [Lim, 2006].

Results

As shown in Table 8.9, the model performed quite well on the various outpatient scenarios. We also used the model to predict possible treatment strategies for each case. In each of the first 7 cases, the predicted treatment strategy was similar to that advocated in the guideline. For the last case, we needed to fine tune the model subjective probabilities to get the correct prediction.

This preliminary case study serves as a starting point for a more extensive evaluation of the probabilistic model built using CPGs. Real patient data are needed to better estimate the accuracy, specificity and sensitivity of the built model. Still, the results demonstrate that not only the guidelines can make authoring of approximate qualitative models easier and faster, but also the prediction quality can be acceptable depending upon the target applications. The model also shows that there is a low impact of approximate probabilities and approximate reasoning structure in this case

Patient Cases	PSI	Care Site	Guideline Recommendation
1	5	ICU	ICU
2	1	Outpatient, Wards	Outpatient, Wards
3	1	Outpatient	Outpatient
4	1	Wards	Wards
5	1	Outpatient	Outpatient
6	1	Outpatient	Outpatient
7	1	Outpatient	Outpatient
8a	2	Wards	Outpatient
8b (fine-tuned)	1	Outpatient	Outpatient

Table 8.9: *Comparison of Predicted PSI and Site-of-Care Vs Recommended*

study. In the literature, however, researchers have a mixed stand about how sensitive a model is to the accuracy of the probability parameters. In some cases, the models can be quite sensitive to the probability parameters and it may be important to elicit the correct probabilities [Onisko et al., 2000]. We would also like to point out that in this case study we only modeled the CAP domain partially and the study suffers from the following limitations: a) We concentrated only on a subset of all possible domain variables, b) We also considered only a subset of domain values, and c) We have assumed some simpler relationships among the variables.

Nevertheless, this case study illustrates how contextual modeling can utilize newer knowledge acquisition techniques and facilitate in building novel applications. For instance, guideline-based probabilistic decision-support system can not only improve adherence to CPGs but also be useful in applications such as proactive content adaptation and context-aware data entry systems.

8.6 Summary

In this chapter, we have discussed the prototype implementation, and the effectiveness of the CSN methodology based on a set of experimental evaluation and two case studies. The experimental evaluation showed the following: a) CSN effectively utilizes local parameter decomposition for eliciting much fewer parameters; b) CSN exploits context-specific relevant structures for efficient inference; and c) parameter learning in CSN converges faster to the true distribution with respect to the number of samples. Furthermore, in the first case study, we formulated the CSN model for context modeling of CAD and demonstrated the effectiveness of CSN for patient-specific inference. In the second case study on CAP, we described the novel use of clinical guidelines for knowledge engineering in a probabilistic network. Based on these case studies, we have demonstrated that CSNs are expressive enough to handle a wide range of problems involving context in medicine.

9 Conclusion

This chapter concludes this thesis by summarizing the achievements and limitations of this work and by reflecting on the lessons learnt in this research. We also compare our approach with some related work. Finally, we point out some directions for future research.

9.1 Summary

We set out to understand the effective representation and reasoning for modeling situational variations and contribute towards designing a general methodology for context-based reasoning under uncertainty. We identified common desiderata among the different usages of context. Building on the common basis, this thesis introduced a new framework for probabilistic representation and reasoning with context-sensitive information.

Context Desiderata	Related CSN framework topics
Local Context Modeling	Contextual Local Views
Dependencies among local contexts	Interface Layer allows Contextual Local Views to be dependent on each other
Information Asymmetry	CPFs, Contextual Local views, and Interface layer that preserves this asymmetry during translation
Scalability	Relational support of CSN and also Contextual Local views that support multiple context-specific dependencies in one view
Adaptability	Context Structural Adaptation

Table 9.1: *Summary of major issues in context representation and the related CSN framework topics*

The centerpiece of this thesis is Context-Sensitive Network (CSN), a representation language that captures a formal notion of probabilistic "*context*". Unlike standard BN, the framework is based on the notion of contextual independence. It exploits these independences for representational and inferential advantages. We demonstrated, as summarized in Table 9.1, that the CSN framework is designed to handle several major issues in context representation such as scalability, adaptability, information asymmetry, local context modeling and dependencies among local contexts. Furthermore, the two examples in Chapter 2 through 6 use simple relational BN model that can roll out into quite general and complex underlying BN structure as shown and used in Chapter 7 and 8. It is in this sense, that the framework contributes towards designing a general methodology for context-based reasoning under uncertainty. We implemented a prototype system and conducted experiments and case studies to evaluate the feasibility and effectiveness of the new representation in medicine field.

9.1.1 Modeling Situational Variations

CSN further enhances the benefits of combining graph theory with probability. By explicitly modeling the variables and their relations as functions of the contexts and by capturing contextual independence, our framework has shown promising potentials to improve the transparency of the model representation under situational variations.

We showed that CSN supports easier knowledge engineering using local context modeling in contextual local views, in conjunction to directly encoding a complete graph. Contextual local views scale well with the number of contexts as they support modeling multiple context-specific dependencies in one view. The interface layer preserves the contextual independence and asymmetry in knowledge while translating the contextual local views into an underlying CSN. The transformation to an underlying CSN also allows contextual local views to be dependent on each other.

We showed that CSN supports relational modeling and provides generalization on a class of individuals. Contextual relational modeling is helpful in domains where the variables as well as the context variables are not known *a priori*.

In the case study on CAP, we also described how a CSN can use knowledge acquisition techniques such as clinical guidelines for knowledge engineering in probabilistic networks.

Our representation is particularly useful when a modeling problem involves any of the following properties: a) uncertain context attributes to model one of possible context scenario as described in Section 4.1 (page 52); b) when the uncertain context attributes may vary in different situations; c) when all the context values or evidence may not be known *a priori*; d) there can possibly be a large number of relevant context attributes.

9.1.2 Model Adaptation to Context-specific Structures

The qualitative representation of the probability functions provides the flexibility to dynamically adapt to the relevant reasoning structure. We demonstrated that CSN can handle model adaptation with incremental context evidence by only performing the graph manipulations and without requiring the expensive CPT manipulations. We further illustrated that the CSN language can be extended to combine the advantages of model adaptation with relational modeling.

9.1.3 Inference Efficiency

By properly partitioning the conditional probability table of a target variable in a context-dependent manner, CSN takes advantage of both the local parameter decomposition and structure decomposition properties. We explained that local parameter decomposition can significantly reduce the number of parameters required to define the complete probability distribution. We also discussed that structure decomposition can significantly reduce the number of variables and their associations while making the graph simpler for inference. We defined a message passing algorithm for CSN to utilize Conditional Part Factors, partitions of CPT in a specific context, and showed that the algorithm takes advantage of both these properties for efficient inference. We also showed that the CSN inference algorithm performed tractable inference on graphs which were intractable using the BN framework.

9.1.4 Learning

We sketched how the probability parameters can be estimated in the CSN framework. Parameter learning in CSN utilized the contextual independence in the distribution

to converge to the estimated values faster than the standard BN with respect to the number of sampled points. Parameter learning allows CSN modeling to fully exploit both prior knowledge and available data.

9.1.5 The Prototype System

As illustrated in the case studies, the prototype system handled a range of problems involving context supporting different requirements for context-based reasoning. Our prototype system supports multiple and generic contexts, such as who, what, where, which, and when of a situation. However, more support tools such as graph-based instead of text-based model specification interfaces are needed to improve its routine usage.

9.1.6 Applications

This work attempts to appeal to the interest of several communities. Context modeling has several practical applications ranging from managing chronic diseases, developing user-specific mobile applications, and building self-aware troubleshooting systems. Our representation is potentially useful when there are a large number of relevant context attributes, when the context attributes may vary in different conditions, and when all the context values or evidence may not be known *a priori*. The case study exercise demonstrated a practical usage of the proposed framework, though most of the insights and results are still experimental.

Working on the case studies also illuminated some important aspects of practical probabilistic modeling in actual clinical domains, and of interdisciplinary collaborative research in general. The lessons learnt from this experience are summarized as follows:

- Clarifying the meaning of context is important, especially due to differences in understanding of term 'context' among the different parties. Moreover, the difficulty in establishing effective communication due to difference in technical background and proficiency of the parties involved can make the modeling process time consuming.
- Context modeling may provide a useful way to construct models based on an established knowledge representation framework in a specific field. This can simplify and speed up authoring of probabilistic models. However, it may be difficult to get the required training data for learning.
- Building large models is time consuming and labor intensive, so eliminating the modeling redundancies by generalization and by exploiting relevancy would facilitate the modeling process. Convenient interfaces to support modeling and automatic consistency checking could reduce model encoding time.

9.1.7 Limitations

The CSN framework, as currently defined, has some inherent limitations. The main inference algorithm for CSN is loopy belief propagation, which is an approximate algorithm whose convergence is not guaranteed. This limits the applicability of our framework. Exploiting CSI for exact inference over CSN is possible. We utilized cutset conditioning algorithm to address some of the issues and discussed the advantages of conditioning-based methods for exact inference over CSN. This remains as one of the directions for our future work.

9.2 Related Work

This work investigated the challenges in representing and reasoning with scalable and adaptable context-sensitive information in BNs. It showed that the problem of situational variation results in Bayesian graph structure and parameters to be a function of number of context variables and their values. This results in an exponential increase of the inferential complexity such that, not only the junction tree exact inference algorithm, but also the approximate loopy belief propagation inference runs out of memory in the BN framework even on the problems with a small number of possible contexts. Unlike the majority of BN inference algorithms that fall in this trap as they need complete probability factors to work, this thesis addressed this problem by proposing an inference algorithm that exploits the context-specific partitions of probability factors.

Most previous efforts in knowledge engineering using context have targeted the problem at the propositional level. Previously, Knowledge-Based Model Construction (KBMC) approaches [Ngo et al., 1995; Laskey & Mahoney, 1997] have utilized local contextual modeling for knowledge engineering but they assume prior knowledge of context. Moreover, extra translation costs may also increase the overall inference costs especially in online inference domains. Our work builds on these approaches and models uncertain contexts. In addition to this, context-sensitive adaptation of CSN involves only graph manipulations for translation, thereby reducing translation costs. Tabular [Mahoney & Laskey, 1998] or tree or rule CPT [Boutilier et al., 1996; Poole & Zhang, 2003] approaches, on the other hand, also need to perform the more expensive CPT manipulations in addition to the graph operations.

Geiger & Heckerman [1996] proposed the Multiple Bayesian networks approach for handling uncertain contexts. They model the context attributes as global root nodes,

Boy(B)	Restaurant(R)	No. of Context	Global Modeling	Local Modeling
B=1	R=1	2	9	2
B=2	R=2	6	729	6
B=2	R=3	10	59049	10

Table 9.2: *Comparison of the number of views required using Global Vs Local context modeling approaches for multiple boys and restaurants in Example 2. Modeling a context attribute as a global root node exponentially increases the number of views required.*

i.e., context variables do not have any parents at all. Our approach is different from theirs in the sense that we model context attributes in Contextual Local Views as local root nodes, i.e., a context variable is a parent only for capturing context-specific dependencies for its children. A context attribute can be dependent on another attribute or another context attribute in a well-formed CSN. Moreover, in contrast to the Multiple BNs approach, Contextual Local Views can capture multiple context-specific dependencies in a single view. The Multinets approach [Geiger & Heckerman, 1996] requires separate BNs for each value of context variables. The results in Table 9.2 on Example 2 show that the local context modeling in CSN require only $|C|$ local networks. So, the contextual local views can scale linearly with an increase in the number of possible contexts (or context assignments).

Structured CPT approaches such as a tree or rule CPT representation [Boutilier et al., 1996; Poole, 1997; Poole & Zhang, 2003] assume that the tree/rule structure is fixed, i.e., the number of contextual parents of a variable and their values are fixed beforehand. However, in contextual relational modeling, the number of context attributes and their values can vary. Our work proposes an alternative way, Conditional Part Factors, for representing structure in the CPT. We graphically represent them as context function nodes. This allows CSN to capture the variations in contextual independence due to the change in the number of context attributes and values. Furthermore, the CSN framework also supports dynamic adaption of

the model structure to exploit more context-specific structures. In CSN, the context addition/removal during structure translation and adaptation is fast as compared to the structured CPT because it only impacts the context function nodes in the graph.

For inference efficiency, virtually all the other relevant work [Boutilier et al., 1996; Poole, 1997; Poole & Zhang, 2003] captures structural variations in terms of local probability parameters; this limits their ability to capture structural variations that also impact the graph structure. CSN, on the other hand, extends graphical representation advantages in a novel way. It exploits context-specific structural variation advantages in both the probability parameters and the graph structure.

Our work is also related to Bayesian Metanetworks [Terziyan, 2006]. Bayesian Metanetworks model external non-causal context attributes to allow prediction over non-context attributes from the observed attributes. As opposed to Bayesian Metanetworks [Terziyan, 2006], CSN supports modeling of causal as well as non-causal context variables and also allows inferring of context attributes from the observed attributes.

The relational extension facilitates modeling in domains where the number of context variables may not be known *a priori*. In contrast to the previous efforts in handling reference uncertainty in non-context parents in BN [Friedman et al., 1999; Laskey et al., 2001; D'Ambrosio et al., 1999], CSN can handle association uncertainty (reference as well as co-reference) in both non-context as well as context parents. Figure 9.1 shows the difference between our representation and previous work.

CSN is a special sub-class of Directed Factor graphs (DFGs). By allowing context-specific nodes and edges, CSN modifies the DFG representation to capture contextual independence rather than conditional independence, and specializes in domains where the variables and their relations are functions of the contexts. This helps CSN to effectively handle the different forms of association uncertainties. Unlike DFG that allow arbitrary factorization and hence need to deal with the normalization conditions,

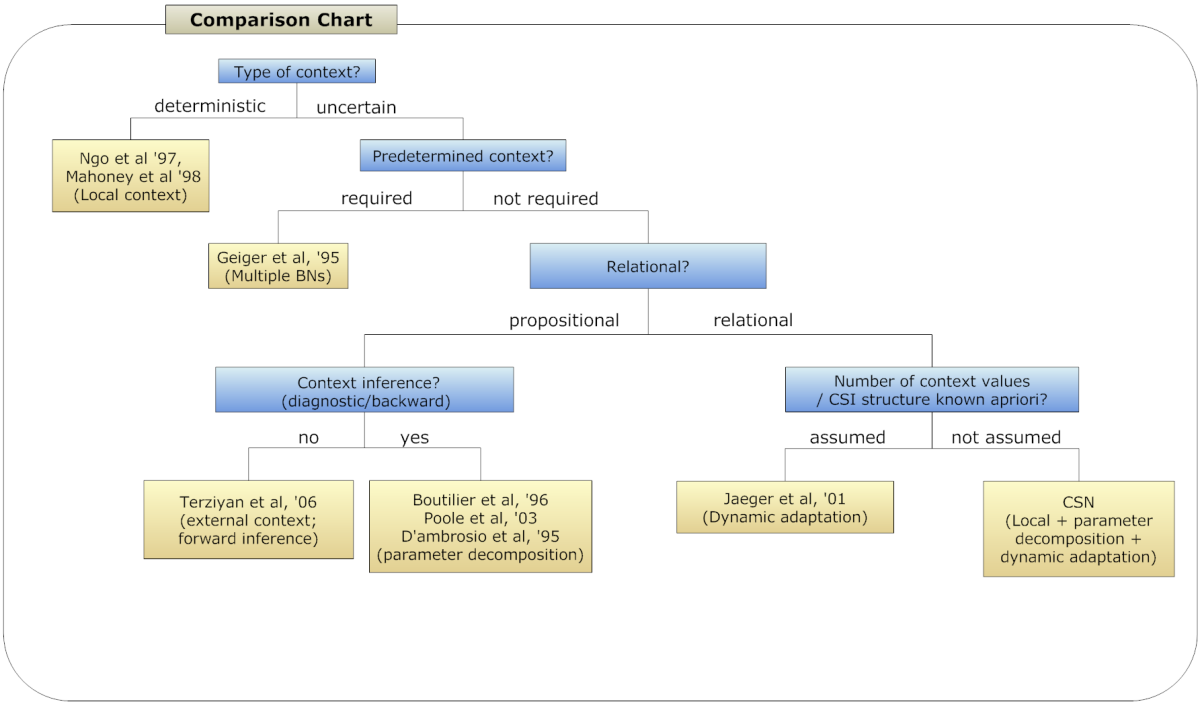


Figure 9.1: *Comparison Chart*

the CSN framework is always normalized as it expresses the factorization of the joint probability distribution based on the notion of contextual independence. By removing the arbitrary factorization and normalization conditions, a DFG is in fact similar to a BN except that they graphically represent the CPT function. Therefore, all the conceptual differences between a CSN and a BN shown earlier as well as the empirical results shown in Chapter 8 apply to the DFG representation too. In addition to this, the CSN framework also shows the advantages of a quantitative function representation for knowledge engineering.

9.3 Future work

This thesis opens up a number of interesting avenues for future research:

9.3.1 Language Extension

We have so far assumed the number of types of objects and contexts such as dogs and families are given. An interesting extension to our language is to address problems involving uncertainty in the number of context variables and values. Such number uncertainty may induce infinite BNs, as recently shown in Milch [2006]. Milch [2006] have recently used contingent BNs for doing inference over infinite BNs with unknown objects. Another shortcoming of this work is that we did not consider and exploit temporal dependencies for context modeling. Incorporating ideas from dynamic Bayesian network [Murphy, 2002b] is an important direction for future work.

Furthermore, our work is also relevant to decision-theoretic approaches to decision making under uncertainty. Asymmetry induced by following alternative strategies as well as exploiting relevant graph structures is an important research area in decision theory [Poh & Horvitz, 1996]. Methodology developed in this thesis can be extended to benefit problems in decision theory.

Ease in authoring probabilistic models remains an open question. We would like to explore strategies that may support faster and easier modeling of probabilistic systems. The Interface layer between the contextual local views and the underlying CSN admits alternate representation schemes. This allows modeling by domain users such as doctors to be fully transparent - the users may not need to fully understand the syntax and semantics of CSN. Recently, there have been some proposals to use semantic web modeling for encoding Bayesian networks [Costa, 2005]. Incorporating

context modeling can play an important role in such applications.

9.3.2 Evaluation on Large-scale applications

Future work can improve the system by applying it on larger-scale problems in specific domains. More comprehensive and larger scale evaluations in various domains can help to further assess the capabilities of CSN and to estimate the necessary language extensions needed for specific domains.

9.3.3 Inference

Recently, there have been a series of work proposing exact inference algorithms to exploit determinism and context-specific independence such as arithmetic circuits [Darwiche, 2002], weighted model counting [Sang et al., 2005] and value-elimination [Bacchus et al., 2003]. We would like to investigate the extension of these techniques for faster exact inference on CSN. Moreover, an extension to any-space recursive-conditioning algorithm [Darwiche, 2001] is also promising as this can utilize context-based graph decomposition effectively. Enhancing Junction-tree inference for exploiting context-specific cliques is much more challenging. We conducted some initial investigations by applying transformations similar to those proposed for factor graphs [Kschischang et al., 2001] for removing cycles and by extending context marginalization operation. Although the maximum clique size was reduced in some cases, the worst case complexity could still be exponential in certain cases. A hybrid combination strategy of conditioning and junction tree to exploit contextual independence for both space and time advantage is an interesting research area. Adding stochastic sampling-based algorithms on CSN may also be a good and easy to materialize idea.

9.3.4 Context-based Learning

In our thesis, we considered parameter learning algorithm with known context information. Relaxing this assumption to discover CPFs can be a useful add-on to the CSN framework. We believe an extension to a strategy similar to decision tree learning as applied to tree-CPDs can be incorporated in a straightforward manner. Friedman & Goldszmidt [1996] and Chickering et al. [1997] have proposed utilizing of local parameter decomposition for more effective global structure learning of the BNs. In the future, we wish to further investigate learning of the context variables and their local context-specific structures.

A

Preliminaries

This appendix mainly introduces some common concepts and computational methods related to the thesis and serves as a basis for a more detailed technical discussion done in the thesis chapters. In this appendix, we briefly introduce the basics of Bayesian networks, directed factor graphs, relational languages for uncertainty representation, inference using belief propagation, and learning probability parameters and structure.

A.1 Historical Background of Bayesian Network

Bayesian network (BN) is one of the most popular formalism in building practical systems for reasoning with uncertain information. The uncertain knowledge is captured using the probabilities and a graph structure. Though Naive Bayes model has appeared in early 70's, the research interest in a Bayesian framework started in the early 80's with seminal publications from [Kim & Pearl, 1983]; then in the late 80's, the

first book [Pearl, 1988] on BN was published. In the late 80's, several practical systems were built [Barnett et al., 1987; Heckerman et al., 1992; Middleton et al., 1991]. Since then, based on several key features and characteristics of the BN framework, the research work has diversified into knowledge representation and engineering, inference, and learning, and has been applied to several fields including printing process [Zhong & Li, 2000], computer troubleshooting [Locke, 1999], forest ecology management [Rieman et al., 2001], sketch recognition [Alvarado & Davis, 2005], and systems biology [Friedman et al., 2000].

A.2 Bayesian Network Theory

Bayesian network [Pearl, 1988] is the most popular framework for the uncertainty representation and is based on the probability theory. BN combines the graph theory with the probability theory and graphically represents the factorization of the joint probability distribution of all attributes in the domain. A BN is a directed acyclic graph where each node represents a random variable, and the edges represent the direct correlation among the variables. A BN factorizes a joint probability distribution (JPD) of all the variables in a domain into a set of local probability distribution by exploiting the conditional independence properties existing in the JPD. The BN framework presents a theoretically sound approach that scales well with the information, is unaffected by some of the information being uncertain or conflicting, and can handle noisy and missing values in the data.

The success of BN representation lies in its property of effectively encoding the conditional independence relations that exist among the random variables. This fundamental property is behind the ease in modeling and representing the domain knowledge, the compactness in eliciting the probability parameters and the inferential gains

achieved in evaluating the network. One objection to the use of probability theory had been that the complete specification of a probability distribution requires absurdly many numbers [Charniak, 1991], for example, if there are m binary variables, the complete joint distribution is specified in $2^m - 1$ probabilities. By providing a clean separation of the quantitative and the qualitative structure, the BN utilizes the structure in the probability distribution to sparsely represent a model. In Figure A.1 with 7 variables, the joint probability distribution requires 127 probabilities, yet we only need to specify 21 probabilities in the graph.

A fundamental property of a probability distribution is that the two variables X and Y are said to be independent if their joint $P(X, Y)$ distribution is simply equal to the product of their individual distribution i.e. $P(X)P(Y)$. Similarly, a variable X is said to be conditionally independent of a variable Y given a variable Z, if $P(X|Y, Z) = P(X|Z)$. This statement can also be read as: given any value of a variable Z, the probability of all values of the variable X are not affected by any values of the variable Y. A BN encodes these conditional independence statements in the probability distribution. The BN network topology also provides a formal way of deducing these statements by means of a graph theoretic property called as *d-separation* [Pearl, 1988].

Formally, the BN language consists of a directed acyclic graph G with a set of random variables and edges $(\mathbf{V}, \mathbf{E})$ and a probability distribution $\boldsymbol{\Theta_G}$. Each random variable V is represented with a node, has domain values $\mathbf{d}$ ($\mathbf{d} \geq 2$) and is associated with a local probability distribution. The local probability distribution θ_v is a stochastic function of the variable conditioned on its parent variables, denoted as $\theta_v = P(V|\mathbf{Pa}(V))$, where the parents ($\mathbf{Pa}$) set represent a set of direct predecessor variables, and the stochastic function encodes the probability values over the enumeration of all the domain values in set $\{V, \mathbf{Pa}\}$. Edges represent the relationships among

the random variables. θ_v is also called as the Conditional Probability Distribution (CPD), and is typically specified in a tabular data structure, termed as the Conditional Probability Table (CPT). Each CPT describes the conditional distribution given the assignments of values for its parent variables. To specify the probability distribution, one must give the prior probabilities of all the root nodes i.e. the nodes with no predecessors, and the conditional probability of all the non-root nodes given all the possible combinations of their direct predecessors.

Figure A.1 shows an example of a BN. {*likes(G1), likes(G2), accompany(B1), order(B1), rating(B1), quality(FT1), quality(FT2)*}, the variable set, is represented as nodes in the figure. Edges between these nodes denote direct influences or dependencies. The BN in Figure A.1 encodes several statements of independences in the probability distribution, e.g., the nodes $accompany(B1)$, $likes(G1)$ and $likes(G2)$ are independent, but given the node $order(B1)$ they become dependent. Similarly, given $order(B1)$, $rating(B1)$ is independent of $accompany(B1)$.

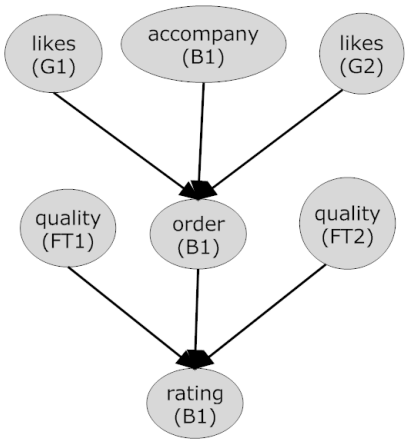
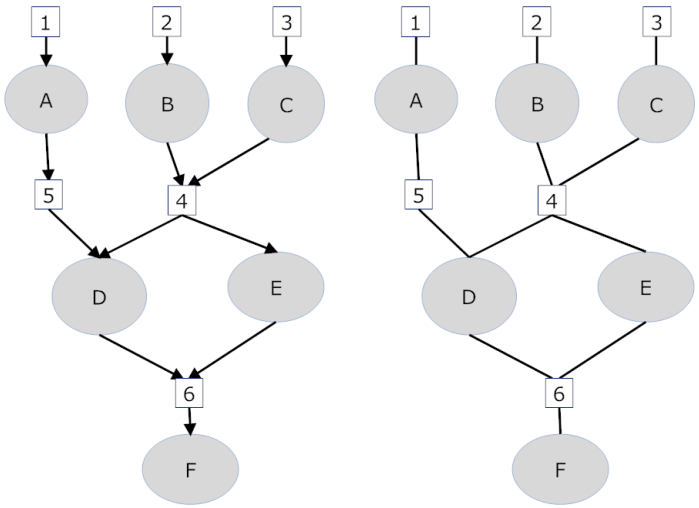


Figure A.1: *BN for example 2 in Section 2.1.2*

A.3 Directed Factor Graphs

Recently, Frey [2003] proposed Directed Factor Graphs (DFG) as an extension of the Factor graphs (described below) where the directed edges indicate conditional distribution. DFGs are bipartite directed graphs consisting of function nodes and variable nodes. DFGs generalize BNs and can express conditional independences in a model that cannot be completely expressed in BNs.



(a) DFG **(b)** FG

Figure A.2: *Example of a Directed Factor Graph and a Factor Graph*

An advantage of a DFG over a BN is that it can represent an arbitrary factorization of the joint probability distribution, for example, it can represent $P(E, D, B, C)$ as $P(E, D|B, C).P(B).P(C)$, a distribution having more than one consequent variables E, D. Arbitrary factorization representation, however, introduces the need for normalization of the probability distribution. A BN, on the other hand, is always normalized. Figure A.2(a) shows one such DFG. The DFG representation allows a function node to have multiple variable nodes as its child and a variable node to

have multiple function nodes as its parents. Frey [2003] has established two rules, instead of three in the BN, to ascertain conditional independence and determine the d-separation. As per Frey [2003], "*A path is said to be blocked if one or more of the following conditions are satisfied:*

1. *One of the variables in the path is observed* (similar to diverging and linear conditions in BN)

2. O*ne of the variables or function nodes in the path has two incoming edges that are part of the path, and neither the variable or function nor any of its descendants are observed*" (similar to converging condition in BN)

As DFGs are based on factor graphs, we now briefly explain the factor graph representation. Readers are directed to [Kschischang et al., 2001] for a more detailed discussion of a factor graph. A factor graph is a bipartite graph that expresses the factorization of a global function $g(X_1 \dots X_n)$ of n variables into a product of m local functions f_i [Kschischang et al., 2001], each containing some subset of $X_1 \dots X_n$ as arguments i.e. $g(X_1 \dots X_n) = \prod_{j=1}^{m} f_j(\mathbf{X}_j)$

A factor graph has a variable node x_i, a function node for each function f_j, and an edge-connecting a variable node x_i to the function node f_j if and only if x_i is an argument of f_j, for example, Figure A.2(b) represents the following factorization: *P(A,B,C,D,E,F) = P(A)P(B)P(C)P(B,C,D,E)P(A,D)P(E,D,F).*

The factor graph, as shown in Figure A.2(b), contains 6 variable nodes and 6 function node and represents the joint product of 6 local functions. The advantage of a factor graph is that it provides arbitrary factorization structure to be easily represented on the graph. Due to this, factor graphs have been applied as an abstraction for solving problems in various domains [Aji & Mceliece, 2000], for instance, the factor graphs have been applied in electrical engineering to solve electric circuits, in signal

processing to formulate Fourier transforms and in AI to understand message passing in logical as well as uncertain reasoning. Another advantage is that the message passing algorithms works much more intuitively on a factor graph than that on a BN, where the local message passing algorithms [Pearl, 1988] are notionally cumbersome. Frey [2003] points out that the explicit use of the function nodes enable the message passing algorithm in factor graphs to be more efficient in some cases than the belief propagation over BN.

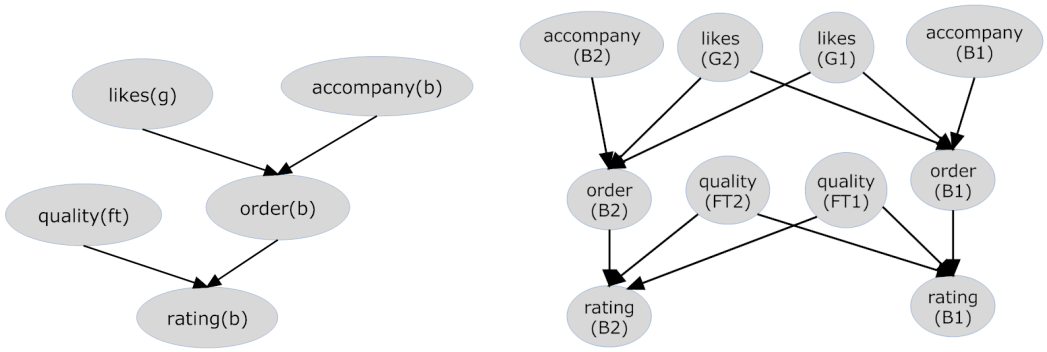
A.4 Relational Extensions to Bayesian Networks

In Chapter 2 Example 2 (page 22), we utilized a relational ontology to describe user-defined rules at the class level. In this section, we explain the relational approaches proposed in the BN literature.

The usefulness of a relational extension lies in utilizing the combined modeling advantages of both the relational logic and the probabilistic framework. In propositional logic, we can express the particular facts but not generalizations, for example, we can say "if A orders food type I in a restaurant and the food quality of the food type I is bad, then $A's$ rating of the restaurant would be low." But we cannot say that "any person who orders a food type in a restaurant and gets a bad food quality will give a low rating to the restaurant." First-order logic can express such generalization about objects. However, the first-order logic modeling lacks the ability to deal with uncertainty. Relational logic extensions of probabilistic framework can provide benefits of both the domains.

Recently, several researchers have proposed languages [Friedman et al., 1999; Heckerman et al., 2004; Laskey, 2007; Jaeger, 2001] to extend the Bayesian network formalism to model the relational data by borrowing concepts from the fields of knowledge

representation and programming languages. In relational data modeling, the world is seen to be consisting of objects (or entities), object's properties (or attributes) and their relationships (or specific interaction among entities). A relational BN model [Friedman et al., 1999] is a skeleton or an abstraction of the dependencies among the object's properties and the relationships of an object with other objects. A probabilistic logic rule makes it explicit that a probability statement applies to any object instantiation in a class of objects, for example, ∀x, $P(A(x)|B(x), C(x), D(x))$ is a belief table for all instantiations of an object x. Figure A.3(a) shows a relational BN for Example 2 in Section 2.1.2 .



(a) relational graph **(b)** propositional graph

Figure A.3: *Relational and rolled out BN for Example 2 in Section 2.1.2. As described in Table 2.2, the objects include: b:Boys, g: girl, ft: food types, B1 and B2 are b's instantiations, G1 and G2 are g's instantiations, and FT1 and FT2 are f's instantiations.*

For inference, a relational network is usually rolled into a propositional BN network. Relational network in Figure A.3(a) presents an abstraction of dependencies and relationships for the following objects: boys, girls, food types and restaurant. The propositional network is formed by instantiating attributes for each object class and capturing relationship with the corresponding object properties. Given 2 boys, 2 girls and 2 food types, the Figure A.3(a) rolls out into a propositional network shown

in Figure A.3(b). Each boy has its own attributes for *accompany*, *order* and *rating*. Both boys can be accompanied by any of the two girls and can order any of the two food types available. Each girl has an attribute: *likes*. Each food type has an attribute *quality*. The advantage of the relational representation is that the local probability distribution remains the same for all the class-level dependencies, for example in this case $P(rating(B1)|order(B1), quality(FT1), quality(FT2)) = P(rating(B2)|order(B2), quality(FT1), quality(FT2))$. However, the downside is that a relational BN can roll out into a large propositional network over which the exact inference is generally not tractable. In the BN literature, the approximate inference methods have been preferred for the relational domain.

A.5 Probabilistic Inference: Message passing

Pearl [1988] has defined a belief propagation algorithm for the BN framework. This section presents the background information of the belief propagation (BP) algorithm, one of the most popular inference algorithm for BNs. BP also forms the basis of our proposed inference algorithm discussed earlier.

Belief propagation algorithm is an instance of the local message-passing algorithms [Kschischang et al., 2001]. We first explain the intuition behing the message passing algorithms using a line of students' analogy. Let us assume that students are arranged in a straight line and a teacher wants to know how many students there are in the line. One way to do is that each student asks his neighbor about how many students there are behind him, including himself. Starting from the first student, this step recursively proceeds till the end of the line where the last student announces 1. Then, each student adds himself and passes the count to the student ahead of him recursively back to the first student. However, the problem with this naive algorithm is that every

time a new student adds or leaves a line, the teacher will not know till she queries again. A better algorithm will be that the neighboring student updates his message and passes it to the student ahead of him if there is any new student addition or deletion to the line. Every student then waits till he receives all the update messages from his neighbors.

If we assume that each student is a point and there is an edge between two neighboring points, then the topology that we get after connecting the points is a straight line. In general, message passing works correctly as long as there is only one path between any two points in the topological structure. This property is true for tree-like structures with no loops.

Why Message Passing may fail on graphs with loops?. Consider, for example, one student E has two immediate neighbors B and C ahead of him and one student D behind him as shown in Figure A.4. The overall network topology in Figure A.4 has one loop A-C-E-B-A. The student E will pass the message to both B and C that there are 2 students. Both B and C will calculate 3 students in the line and pass this message to A. A will wrongly calculate that there are 3+3=6 students behind him, while in fact there are only four students behind him. Therefore, message passing with loops can be error-prone. One solution to correct update is to remove a link, say between E and B. If each of the points in the graph divides the whole topological graph structure into two disjoint subset graphs, i.e., the graph is tree-like, then the message passing will be exact. Earlier, it was believed that the message passing has limited advantages for the BN as it works only for the tree-like structures. However, new empirical results have surprisingly shown that loopy belief propagation is extremely effective over networks with loops [Murphy et al., 1999]. Loopy belief propagation is an approximate algorithm which iterates the belief repeatedly over the network till they converge. However, the convergence is not guaranteed. Nevertheless, LBP

has recently enjoyed tremendous success even on the difficult instances of NP-Hard problems such as the problems in error-correcting decoding [Kschischang & Frey, 1998], and is quickly becoming a gold standard approximate algorithm.

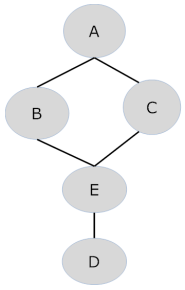


Figure A.4: *Message passing on graphs with loops*

The belief propagation can be explained using a general message passing sum-product algorithm on the factor graphs [Kschischang et al., 2001]. The sum-product algorithm over the factor graphs involves two operations: factor product and factor marginalization. Factor Product of two factors $f(X,Y)$ and $f(Y,Z)$ is defined as $f(X,Y,Z) = f(X,Y).f(Y,Z)$ where $(.)$ means that for each instantiation of X,Y,Z, the corresponding values in the different functions are multiplied together. Factor marginalization of factor $f(X,Y)$ over Y is defined as $f(X) = \sum_Y f(X,Y)$ where $\sum_Y$ means the summation of all values along the dimensions Y in factor $f(X,Y)$ for each values of X.

The following rules summarize a general message passing algorithm [Kschischang et al., 2001]:

- Each variable node receives messages from all its neighbor function nodes and each function node receives messages from all its neighbor variable nodes.
- The message out of variable node v along the edge f is the factor product of all messages towards v along all edges except f.

- The message out of function node f along the edge v is the factor product of factor associated with the function node f and all messages towards f along all edges except v, and then marginalized over all the variables in its neighborhood except v.

Yedidia et al. [2005] have argued that the mathematical equations for the above summarized rules correspond to the stationarity conditions for a functional of the beliefs called the Bethe free energy. This Bethe's free energy minimization has been successfully applied to many problems in physics. This partly explains why belief propagation works so well for general graphical models with cycles.

A.6 Learning Parameters from Data

Several researchers have argued that constructing a BN structure for application is often the easy part as humans find it easy to say what causes what, but it is hard to put exact numbers on the links. This has been aptly described in "Where do the numbers come from?" [Druzdzel et al., 1995]. Numbers or probability parameters in the BN formalism can be either extracted from the experts, or from the published literature or can be learnt from the data. In this section, we discuss the basics of learning probability parameters from the data.

In parameter learning, the network structure is specified and the goal of learning is to find the parameters of each CPD that maximizes the likelihood of the training data. We assume that all the data cases are independent and identically distributed (i.i.d) without any missing values. Given the training dataset D with N cases and m variables, the likelihood function decomposes as per the structure of the graph. This decomposition property implies independent estimation can be done for each local

CPD. Normalized log likelihood can be written as

$$\begin{aligned} L(\Theta : D) &= log \prod_{i=1}^{N} P(X_1 \ldots X_m : \Theta) = log \prod_{i=1}^{N} \prod_{j=1}^{m} P(X_j | \mathbf{Pa}(X_j) : \theta_j) \\ &= \sum_{j=1}^{m} [\sum_{i=1}^{N} log(P(X_j | \mathbf{Pa}(X_j) : \theta_j)] = \sum_{j=1}^{m} [L_i(\theta_j : D_i) \end{aligned}$$

For a multinominal distribution, the maximum likelihood estimate reduces to counts of a local distribution of the variable and its parents in the data cases set $N[X_i, \mathbf{Pa}(X_i)]$. The estimation relies on this sufficient statistics:

$$\theta_{x_i | pa(x_i)} = N[X_i, \mathbf{Pa}(X_i)] / N[\mathbf{Pa}(X_i)]$$

By following the similar arithmetic operations as above, previous works have also shown that just as the tabular CPDs, the maximum likelihood estimate using a local structure in the CPDs such as a tree structure also decomposes into the counts of a local distribution [Friedman & Goldszmidt, 1996; Chickering et al., 1997]).

A.7 Learning Structure from Data

In addition to the parameter learning, another advantage of the Bayesian formalism is that the reasoning structure can also be learnt from the data. Such structure learning algorithms use only a dataset to uncover both the relationships among the variable and the probability parameters. Three main types of structure-learning approaches have emerged: score and search-based, constraint-based and averaging-based. The score and search-based methods introduce a scoring function such as Bayesian Information Criteria (BIC) that evaluates each network according to a metric with respect

to the training data, and a search method such as greedy hill climbing for utilizing the highest scoring network. The common scoring metrics include Bayesian score [Cooper & Herskovits, 1992; Heckerman et al., 1995], BIC [Heckerman, 1995] and minimum description length (MDL) [Lam & Bacchus, 1994]. Instead of using a score function, the constraint-based approaches [Buntine, 1996] use conditional independence or mutual information statistical tests to validate a structure. The averaging approaches, on the other hand [Friedman & Koller, 2003], average out the collection of high scoring structures instead of selecting the highest scoring structure. The averaging approach has been applied in problems with a limited amount of the training data, for instance, the problem of learning a gene-regulatory network [Friedman & Koller, 2003]. In this work, however, we do not address the problem of structure learning.

A.8 Summary

We discussed the basics of probabilistic graphical languages for modeling uncertainty and computational methods for inference and learning in this appendix.

B

Prototype Implementation

This appendix details the prototype implementation of the CSN framework. The system architecture is shown in Figure B.1. In this chapter, we describe three ways of context model representation supported in our prototype implementation: a) modeling as a full CSN, b) modeling using contextual local views, and c) modeling using a relational language. The representation is encoded in MATLAB. We also show how to invoke the context structural adaptation, inference and parameter learning modules.

B.1 Complete CSN representation

In its most simplest form, CSN can be directly encoded as follows :

1. Define the number of variables and context function nodes (or CPFs)
2. Encode the CSN graph (DAG) at the contextual level

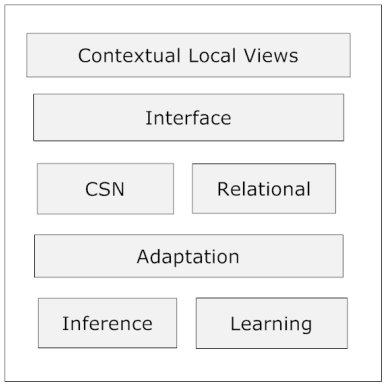


Figure B.1: *Prototype Implementation: Systems View*

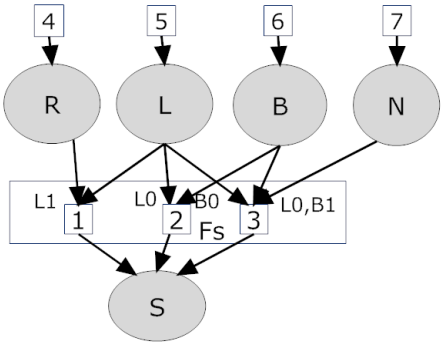


Figure B.2: *A simple CSN*

3. Describe the domain size of each variable node
4. Specify probability and context of each CPF
5. Associate a function node with the group of context function nodes having same consequent variable
6. Call *csnMake* procedure
7. Enter evidence and run inference algorithm

Listing B.1 encodes the CSN as shown in Figure B.2.

Listing B.1: *CSN code for Figure B.2*

```
% defining number of vars and functions in the CSN
var=5;
cfunc=7;
% STEP1: give each var and CPF a number
L=1;R=2;B=3;N=4;S=5;
srlCPF=1;sblCPF=2;sbnlCPF=3;rCPF=4;lCPF=5;bCPF=6;nCPF=7;
% STEP2: define edges of a CSN: '1' for var to cfunc ; '2' for cfunc to var;
dag=zeros(var,cfunc);
dag([R L],srlCPF)=1;
dag([B L],sblCPF)=1;
dag([B N L],sbnlCPF)=1;
dag(S,[srlCPF sbnlCPF sblCPF])=2;
dag(R,rCPF)=2;
dag(L,lCPF)=2;
dag(B,bCPF)=2;
dag(N,nCPF)=2;
%STEP 3: define domain size of each variable
% nodeDomainSizes = 2*ones(1,var)
nodeDomainSizes=[2 2 2 2 2];
%STEP 4: Input CPF : Factors and Context
F2.id=sblCPF;
F2.probvalues=[0.8 0.2]; % P(S|L=2,B=1) = phi(S)
F2.context={[L B],[2 1]};
F3.id=sbnlCPF;
F3.probvalues=[0.9 0.4 0.1 0.6];% phi(S|N) format (s|n s|n0 s0|n s0|n0)
F3.context={[L B],[2 2]};
F1.id=srlCPF;
F1.probvalues=[0.6 0.7 0.4 0.3]; %phi(S|R) for L=1
F1.context={[L],[1]};
F4.id= rCPF;
F4.probvalues= [0.3 0.7];
F4.context={};
F5.id= lCPF;
F5.probvalues= [0.2 0.8];
F5.context={};
F6.id= bCPF;
F6.probvalues= [0.4 0.6];
F6.context={};
F7.id= nCPF;
F7.probvalues= [0.1 0.9];
F7.context={};
% STEP 6: define which CPF are under one functional node and a set of all
% functional nodes
Func={[F1.id,F2.id,F3.id],[F4.id],[F5.id],[F6.id],[F7.id]}; % set of all functional nodes
CPF={F1,F2,F3,F4,F5,F6,F7};
% make CSN
% Input : nodeDomainSize,Dag,F,CPF
csn=csnMake(dag,nodeDomainSizes,CPF,Func);
evidence=cell(1,nVar);
marg= belpropv3tree(csn,evidence)
```

B.2 Contextual Local Views

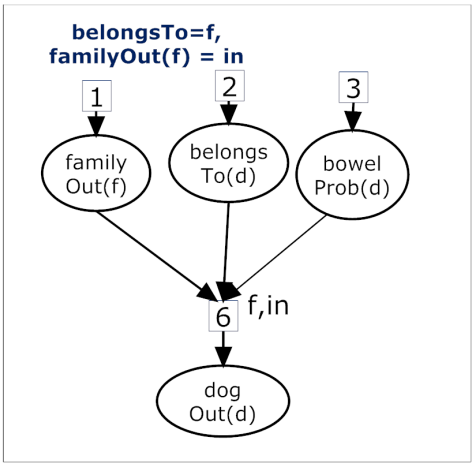


Figure B.3: *A Contextual Local View*

In our implementation, each contextual local view is encoded as follows:

1. Define the context variables and/or its assignments for each view
2. Encode the DAG at the contextual level
3. Label the context function node
4. Specify the probabilities

Listing B.2 encodes the contextual local view as shown in Figure B.3.

Listing B.2: *Contextual Local View for Figure B.3*

```
% context graph with Context age: BT1=1,F1=2
count=count+1;
% Define the context scope and value of contextual view
context{count}=[BT1,F1];
value{count}=[1,2];
% Define GRAPH View
dg_bt11f12=2*eye(nVar,nFactor); %initialize the graph, use all variables
```

```
8  dg_bt11f12([BT1 F1 BP1], [D1])=1; % context variable will also have an arc
9  context_nodes{count}=[D1]; %  Context node Labels in each graph
10 probvalues{D1}{count}=[0.7 0.2 0.3 0.8];
11 dg_bt11f12;
12 dg_Con{count}=dg_bt11f12;
```

B.3 Relational CSN representation

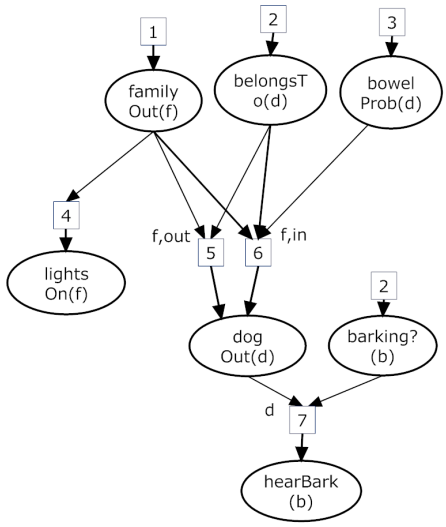


Figure B.4: *Context-sensitive network for Example 1*

Our implementation supports a relational CSN to be encoded directly or using contextual local views. Listings B.3, B.4, and B.5 encode the relational CSN shown in Figure B.4 as follows:

1. Initialize (as shown in Listing B.3)
    - Define the number of objects and their instantiation

- Define the attributes for each object and their domain size
- Initialize the probability values

2. Encode relational contextual local view in a similar way as propositional contextual local view (as shown in Listing B.4)
   - Define the context attributes and/or values
   - Define the DAG at the contextual level
   - Encode the probabilitites
3. Convert to propositional CSN (as shown in Listing B.5)
   - Procedure *meta2csn* converts a relational CSN to a propositional CSN
   - Enter evidence at the propositional level. Implementation can easily be extended for entering evidence at the relational level.
4. Infer (as shown in Listing B.5)
   - Perform context structural adaptation
   - Run Inference procedure

Listing B.3: *Dog Network: Initialization*

```
% number of object instantiations
numFam=4; numDog=5;numBark=1;
% assigning unique identifier
BT = [1:numDog]; F = BT(end)+[1:numFam];
D= F(end)+[1:numDog]; LO = D(end)+[1:numFam];
BP= LO(end)+[1:numDog]; BO= BP(end)+[1:numBark];
HB= BO(end)+[1:numBark];

% calculating total variables
nVar= HB(end)
nFactor=nVar; % nCPF=nVAR in BN representation.

% assigning domain size
nodeDomainSizes = 2*ones(1,nVar);
nodeDomainSizes(BT(:))=numFam;
nodeDomainSizes(BO(:))=numDog;

% CSN representation nCPF > nVAR
```

```
19 context={}; % store global context variables of the scenario
20 value={}; % store value of each of the variables
21 context_nodes={};global_nodes={};
22 dg_Con={}; % collection of context fragment graphs
23 count=0; % counter for number of the contextual view graph
24
25 % initialize probability values
26 probvalues=cell(1,nVar);
27 for i=1:length(D)
28     probvalues{D(i)}={};
29 end
30 for i=1:length(HB)
31     probvalues{HB(i)}={};
32 end
33 for i=1:length(BT)
34     probvalues{BT(i)}=(1/numFam)*ones(1,numFam);
35 end
36 for i=1:length(F)
37     probvalues{F(i)}=[0.3  0.7];
38 end
39 for i=1:length(LO)
40     probvalues{LO(i)}=[0.7  0.2  0.3  0.8];
41 end
42 for i=1:length(BP)
43     probvalues{BP(i)}=[0.1  0.9];
44 end
45 for i=1:length(BO)
46     probvalues{BO(i)}=(1/numDog)*ones(1,numDog);
47 end
```

Listing B.4: *Dog Network: Relational Contextual Local Views*

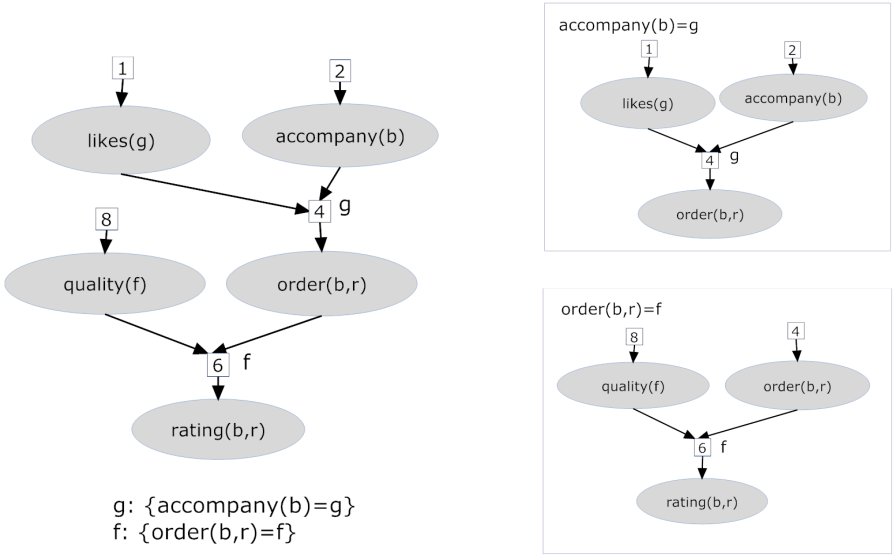
```
1  % context graph with Context age: BT=1,F1=1
2  for i=1:length(BT)
3      for j=1:length(F)
4          dg=2*eye(nVar,nFactor);
5          count=count+1;
6      % Define the context scope and value of contextual view
7          context{count}= [BT(i),F(j)];
8          value{count}=[j,1];
9          % Define GRAPH View
10         dg([BT(i) F(j)], [D(i)])=1;
11         dg([F(j)],[LO(j)])=1;
12         context_nodes{count}=[D(i)]; %  Context node Labels in each graph
13         probvalues{D(i)}{count}=[0.7  0.3];
14         dg_Con{count}=dg;
15     end
16 end
17 % context graph with Context age: BT=1,F1=2
18 for i=1:length(BT)
19     for j=1:length(F)
20         dg=2*eye(nVar,nFactor);
```

```
21          count=count+1;
22          context{count}= [BT(i),F(j)]; value{count}= [j,2];
23          dg([BT(i) F(j) BP(i)], [D(i)])=1;
24          context_nodes{count}=[D(i)];
25          probvalues{D(i)}{count}=[0.7 0.2 0.3 0.8];
26          dg_Con{count}=dg;
27      end
28  end
29  % context graph with Context age: BO1=1
30  for i =1:length(BO) % num (BO) is same as num(HB)
31      for j=1:length(D) % num(D) = domain values of BO
32          dg=2*eye(nVar,nFactor);
33          count=count+1;
34          context{count}=[BO(i)]; value{count}= [j];
35          dg([BO(i) D(j)], [HB(i)])=1;
36          context_nodes{count}=[HB(i)];
37          probvalues{HB(i)}{count}=[0.7 0.2 0.3 0.8];
38          dg_Con{count}=dg;
39      end
40  end
```

Listing B.5: *Dog Network: convert to CSN and infer*

```
1  disp(['==========CONVERTING TO CSN==========='])
2  meta2csn
3  disp(['==========ORIGINAL GRAPH STATISTICS==========='])
4  statistics(csn)
5  fstatistics(csn,evidence)
6  totalcontext=length(BT)+length(F)+length(BO)
7  evidence=cell(1,csn.nVars);
8  % enter evidence
9  evidence{BT(1)}=2;
10 evidence{F(2)}=1;
11 evidence{F(1)}=1;
12 evidence{BT(3)}=2;
13 evidence{BO(1)}=2;
14 disp(['==========STRUCTURAL ADAPTATION==========='])
15 observed=observedCSN(csn,evidence);
16 newcsn=observed.original;
17 queryGraphs=observed.query;
18 disp(['==========INFERENCE==========='])
19 marg= belpropv4tree(csn2);
```

Listings B.6 and B.7 encode the relational CSN for the food network shown in Figure B.5.



(a) CSN **(b)** Contextual Local Views

Figure B.5: *Relational CSN for Example 2*

Listing B.6: *Food Network: Initialization*

```
numUser=2; numRes=2; numResUser=numUser*numRes;
numGirl=5;numFood=10;
mat=ones(numUser,numRes);
Who = [1:numUser];
Girl = Who(end)+[1:numGirl];
Order= Girl(end)+[1:numResUser];
Food = Order(end)+[1:numFood];
Rating= Food(end)+[1:numResUser];

nVar= Rating(end);
nFactor=nVar;
nodeDomainSizes = 2*ones(1,nVar);
nodeDomainSizes(Who(:))=numGirl;
nodeDomainSizes(Order(:))=numFood;
nodeDomainSizes(Girl(:))=numFood;

% CSN representation nCPF > nVAR
context={}; % store global context variables of the scenario
value={}; % store value of each of the variables
context_nodes={};global_nodes={};
dg_Con={}; % collection of context fragement graphs
count=0; % counter for number of the contextual view graph

probvalues=cell(1,nVar);
```

```
for i=1:length(Who)
    %probvalues{Who(i)}=(1/numGirl)*ones(1,numGirl) ; % Uniformdist(numFam)
    fam_sz=nodeDomainSizes([Who(i)]);
    probvalues{Who(i)}= mk_stochastic(myrand(fam_sz));
end
for i=1:length(Order)
    probvalues{Order(i)}={};
end
for i=1:length(Rating)
    probvalues{Rating(i)}={};
end
for i=1:length(Girl)
   % probvalues{Girl(i)}=[0.3 0.7];
     fam_sz=nodeDomainSizes([Girl(i)]);
    probvalues{Girl(i)}= mk_stochastic(myrand(fam_sz));
end
for i=1:length(Food)
    % probvalues{Food(i)}=[0.8 0.2];
    fam_sz=nodeDomainSizes([Food(i)]);
    probvalues{Food(i)}= mk_stochastic(myrand(fam_sz));
end
```

Listing B.7: *Food Network: Relational Contextual Local Views*

```
%Context Who=g

for i=1:numGirl % total context domain values =g, the total instantiation of Girl
    for j=1:numUser %total instantiations of object Who(u)
        for k=1: numRes % total possible instantiations of Order(r,u)
            ru=sub2ind(size(mat),j,k);
            dg=2*eye(nVar,nFactor);
        % context graph with Context Who
            count=count+1;
        % Define the context scope and value of contextual view
            context{count}= [Who(j)];
            value{count}=[i];
        % Define GRAPH View
            dg([Who(j) Girl(i)], [Order(ru)])=1;
            context_nodes{count}=[Order(ru)];
            %probvalues{Order(ru)}{count}=[0.7 0.2 0.3 0.8];
            fam_sz=nodeDomainSizes([Girl(i) Order(ru)]);
            probvalues{Order(ru)}{count}= mk_stochastic(myrand(fam_sz));
            dg_Con{count}=dg;
        end
    end
end

%=============================================================
%Context Order=f %, total food types or instantiation of foods

for i=1:numFood % total context domain values =numFood
    for j=1:numUser %total instantiations of object Order(r,u), Rating(r,u)
```

```
29          for k=1: numRes % total possible instantiations of Order(r,u), Rating(r,u)
30              ru=sub2ind(size(mat),j,k);
31              dg=2*eye(nVar,nFactor);
32              % context graph with Context Order
33              count=count+1;
34              % Define the context scope and value of contextual view
35              context{count}= [Order(ru)];
36              value{count}=[i];
37              % Define GRAPH View
38              dg([Order(ru) Food(i)], [Rating(ru)])=1;
39              context_nodes{count}=[Rating(ru)];
40             % probvalues{Rating(ru)}{count}=[0.7 0.2 0.3 0.8];
41             fam_sz=nodeDomainSizes([Food(i) Rating(ru)]);
42             probvalues{Rating(ru)}{count}= mk_stochastic(myrand(fam_sz));
43             dg_Con{count}=dg;
44          end
45      end
46  end
47
48  %=========================================================
49  %Food Network: convert to CSN and enter evidence
50  meta2csn
51  evidence=cell(1,csn.nVars);
52  evidence{Who(1)}=2;
53  evidence{Order(1)}=2;
```

B.4 Parameter learning

Parameter learning module assumes the CSN network has been encoded. It currently supports two modes for parameter estimation:

1. Testing mode: Generates a number of samples based on the defined CSN and then estimate parameters based on generated sampled data-set
2. Estimation mode: Estimate parameter given a full data-set

Listing B.8: *Parameter Learning*

```
1  % First define a network as shown for the Dog or the food networks
2  % Estimation mode (Data available): load data
3  %data1=data
4  csn2=csn;
```

```
% Testing mode: define number of samples
num_samples=10000;
%call sampler
data1=sampler(csn2,num_samples);
%call learner module
csn_new=learn_params(csn2,data1);
```

B.5 CSN Context model for the Case Study

This section illustrates one of the model defined for the pneumonia case study using the clinical guidelines. This case study encodes a propositional CSN using contextual local views. In this model, we use a separate contextual local view for each value of context variables. Each contextual view captures the information and associated conditional probability distribution as explained earlier in Chapter 8 Section 8.5.3. Listing B.9 shows different contextual views for 3 context variables: pneumonia severity index (PSI), site-of-care (SoC) and patient characteristics (PC).

Listing B.9: *Pneumonia Case Study*

```
PSI=1;SoC=2;Oral=3;Age=4; LowBP=5; RR=6; Urea=7; Conf=8;H=9;
OT=10;PC=11;Anti=12;Healthy=13;Comorbid=14; Asp=15; NHR=16; Inf=17;IT=18;
nodeDomainSizes=[5 3 2 2 2 2 2 2 2 3 5 2 2 2 2 2 2 3];
nVar=18;
nFactor=nVar;

% CSN representation nCPF > nVAR
context={}; % store global context variables of the scenario
value={}; % store value of each of the variables
context_nodes={};global_nodes={};
dg_Con={}; % collection of context fragement graphs
count=0; % counter for number of the contextual view graph

% CPFs
probvalues=cell(1,nVar);
probvalues{SoC}={};
probvalues{OT}={};
probvalues{IT}={};

% context graph with Context PSI: PSI=1
count=count+1;
```

```
% Define the context scope and value of contextual view
context{count}=[PSI];
value{count}=[1];
% Define GRAPH View
dg_PSI1=2*eye(nVar,nFactor); %initialize the graph
dg_PSI1([PSI],[H])=1;
dg_PSI1([H],[Age LowBP RR Urea Conf])=1;
dg_PSI1([PSI Oral], SoC)=1;
context_nodes{count}=[SoC];
probvalues{SoC}{count}=[0.7 0.2 0.2 0.7 0.1 0.1];
probvalues{PSI}=[0.1 0.15 0.2 0.35 0.2] ;
probvalues{H}=[0.25 0.5 0.75 0.9 0.95 0.75 0.5 0.25 0.1 0.05] ;
probvalues{Oral}=[0.5 0.5];
probvalues{Age}=[0.25 0.65 0.75 0.35];
probvalues{LowBP}=[0.1 0.01 0.9 0.99];
probvalues{RR}=[0.2 0.01 0.8 0.9];
probvalues{Urea}=[0.2 0.01 0.8 0.99];
probvalues{Conf}=[0.2 0.01 0.8 0.99];
dg_PSI1;
dg_Con{count}=dg_PSI1;

% context graph with Context PSI: PSI=2
count=count+1;
% Define the context scope and value of contextual view
context{count}=[PSI];
value{count}=[2];
% Define GRAPH View
dg_PSI2=2*eye(nVar,nFactor);
dg_PSI2([PSI Oral], SoC)=1;
context_nodes{count}=[SoC];
probvalues{SoC}{count}=[0.7 0.15 0.05 0.8 0.25 0.05];
dg_PSI2;
dg_Con{count}=dg_PSI2;

% context graph with Context PSI: PSI=3
count=count+1;
% Define the context scope and value of contextual view
context{count}=[PSI];
value{count}=[3];
% Define GRAPH View
dg_PSI3=2*eye(nVar,nFactor);
dg_PSI3([PSI], SoC)=1;
context_nodes{count}=[SoC];
probvalues{SoC}{count}=[0.15 0.75 0.1];
dg_PSI3;
dg_Con{count}=dg_PSI3;

% context graph with Context PSI: PSI=4
count=count+1;
% Define the context scope and value of contextual view
context{count}=[PSI];
value{count}=[4];
% Define GRAPH View
dg_PSI4=2*eye(nVar,nFactor);
dg_PSI4([PSI], SoC)=1;
```

```
77  context_nodes{count}=[SoC];
78  probvalues{SoC}{count}=[0.1  0.5  0.4];
79  dg_PSI4;
80  dg_Con{count}=dg_PSI4;
81
82  % context graph with Context PSI: PSI=5
83  count=count+1;
84  % Define the context scope and value of contextual view
85  context{count}=[PSI];
86  value{count}=[5];
87  % Define GRAPH View
88  dg_PSI5=2*eye(nVar,nFactor);
89  dg_PSI5([PSI], SoC)=1;
90  context_nodes{count}=[SoC];
91  probvalues{SoC}{count}=[0.05  0.25  0.7];
92  dg_PSI5;
93  dg_Con{count}=dg_PSI5;
94
95
96  % context graph with Context SoC,PC: SoC=1,PC=1
97  count=count+1;
98  context{count}=[SoC PC]; % Define the context scope and value of contextual view
99  value{count}=[1  1];
100 dg_SoCPC1=2*eye(nVar,nFactor);
101 dg_SoCPC1([PC],[Healthy Comorbid Asp NHR Inf])=1; % Define GRAPH View
102 dg_SoCPC1([SoC PC Anti], OT)=1;
103 context_nodes{count}=[OT];
104 probvalues{OT}{count}=[0.25  0.7  0.7  0.25  0.05  0.05];
105 probvalues{PC}=[0.2  0.2  0.2  0.2  0.2];
106 probvalues{Anti}=[0.5  0.5];
107 probvalues{Healthy}=[0.95  0.05  0.15  0.15  0.15  0.05  0.95  0.85  0.85  0.85];
108 probvalues{Comorbid}=[0.05  0.95  0.15  0.15  0.15  0.95  0.05  0.85  0.85  0.85];
109 probvalues{Asp}=[0.05  0.15  0.95  0.15  0.15  0.95  0.85  0.05  0.85  0.85];
110 probvalues{NHR}=[0.05  0.15  0.15  0.999  0.15  0.95  0.85  0.85  0.05  0.85];
111 probvalues{Inf}=[0.05  0.15  0.15  0.15  0.95  0.95  0.85  0.85  0.85  0.05];
112 dg_SoCPC1;
113 dg_Con{count}=dg_SoCPC1;
114
115 % context graph with Context SoC,PC: SoC=1,PC=2
116 count=count+1;
117 context{count}=[SoC PC]; % Define the context scope and value of contextual view
118 value{count}=[1  2];
119 dg_SoCPC2=2*eye(nVar,nFactor);
120 dg_SoCPC2([SoC PC Anti], OT)=1;
121 context_nodes{count}=[OT];
122 probvalues{OT}{count}=[0.15  0.45  0.8  0.45  0.05  0.1];
123 dg_SoCPC2;
124 dg_Con{count}=dg_SoCPC2;
125
126 % context graph with Context SoC,PC: SoC=1,PC=3
127 count=count+1;
128 context{count}=[SoC PC]; % Define the context scope and value of contextual view
129 value{count}=[1  3];
130 dg_SoCPC3=2*eye(nVar,nFactor);
131 dg_SoCPC3([SoC PC], OT)=1;
```

```
132 context_nodes{count}=[OT];
133 probvalues{OT}{count}=[0.25 0.7 0.05];
134 dg_SoCPC3;
135 dg_Con{count}=dg_SoCPC3;
136
137 % context graph with Context SoC,PC: SoC=1,PC=4
138 count=count+1;
139 context{count}=[SoC PC]; % Define the context scope and value of contextual view
140 value{count}=[1 4];
141 dg_SoCPC4=2*eye(nVar,nFactor);
142 dg_SoCPC4([SoC PC], OT)=1;
143 context_nodes{count}=[OT];
144 probvalues{OT}{count}=[0.15 0.8 0.05];
145 dg_SoCPC4;
146 dg_Con{count}=dg_SoCPC4;
147
148 % context graph with Context SoC,PC: SoC=1,PC=5
149 count=count+1;
150 context{count}=[SoC PC]; % Define the context scope and value of contextual view
151 value{count}=[1 5];
152 dg_SoCPC5=2*eye(nVar,nFactor);
153 dg_SoCPC5([SoC PC], OT)=1;
154 context_nodes{count}=[OT];
155 probvalues{OT}{count}=[0.15 0.8 0.05];
156 dg_SoCPC5;
157 dg_Con{count}=dg_SoCPC5;
158
159 % context graph with Context SoC,PC: SoC=2
160 count=count+1;
161 context{count}=[SoC]; % Define the context scope and value of contextual view
162 value{count}=[2];
163 dg_SoC2=2*eye(nVar,nFactor);
164 dg_SoC2([SoC], OT)=1;
165 context_nodes{count}=[OT];
166 probvalues{OT}{count}=[0.05 0.05 0.9];
167 dg_SoC2;
168 dg_Con{count}=dg_SoC2;
169
170 % context graph with Context SoC,PC: SoC=3
171 count=count+1;
172 context{count}=[SoC]; % Define the context scope and value of contextual view
173 value{count}=[3];
174 dg_SoC3=2*eye(nVar,nFactor);
175 dg_SoC3([SoC], OT)=1;
176 context_nodes{count}=[OT];
177 probvalues{OT}{count}=[0.05 0.05 0.9];
178 dg_SoC3;
179 dg_Con{count}=dg_SoC3;
180
181 probvalues{WT}={};
182 % context graph with Context SoC,PC: SoC=1
183 count=count+1;
184 context{count}=[SoC]; % Define the context scope and value of contextual view
185 value{count}=[1];
186 dg_SoCWT1=2*eye(nVar,nFactor);
```

```
187 dg_SoCWT1([SoC], WT)=1;
188 context_nodes{count}=[WT];
189 probvalues{WT}{count}=[0.05 0.05 0.9];
190 dg_SoCWT1;
191 dg_Con{count}=dg_SoCWT1;
192
193 % context graph with Context SoC,PC: SoC=3
194 count=count+1;
195 context{count}=[SoC]; % Define the context scope and value of contextual view
196 value{count}=[3];
197 dg_SoCWT3=2*eye(nVar,nFactor);
198 dg_SoCWT3([SoC], WT)=1;
199 context_nodes{count}=[WT];
200 probvalues{WT}{count}=[0.05 0.05 0.9];
201 dg_SoCWT3;
202 dg_Con{count}=dg_SoCWT3;
203
204 % context graph with Context SoC,Suspect, PA: SoC=2,Suspect=1; PA=1
205 count=count+1;
206 context{count}=[SoC Suspect PA];
207 value{count}=[1 1 1];
208 dg_SoCSuspectPA1=2*eye(nVar,nFactor);
209 % Define GRAPH View
210 dg_SoCSuspectPA1([SoC Suspect PA], WT)=1;
211 context_nodes{count}=[WT];
212 probvalues{WT}{count}=[0.7 0.25 0.05];
213 probvalues{Suspect}=[0.7 0.3];
214 probvalues{PA}=[0.7 0.3];
215 dg_SoCSuspectPA1;
216 dg_Con{count}=dg_SoCSuspectPA1;
217
218 % context graph with Context SoC,Suspect, PA: SoC=2,Suspect=1; PA=2
219 count=count+1;
220 context{count}=[SoC Suspect PA];
221 value{count}=[1 1 2];
222 dg_SoCSuspectPA2=2*eye(nVar,nFactor);
223 % Define GRAPH View
224 dg_SoCSuspectPA2([SoC Suspect PA PH], WT)=1;
225 context_nodes{count}=[WT];
226 probvalues{WT}{count}=[0.15 0.2 0.7 0.75 0.15 0.05];
227 probvalues{PH}=[0.6 0.4];
228 dg_SoCSuspectPA2;
229 dg_Con{count}=dg_SoCSuspectPA2;
230
231
232
233 % context graph with Context SoC,Suspect, PA: SoC=2,Suspect=2; TBRisk=1
234 count=count+1;
235 context{count}=[SoC Suspect TBRisk];
236 value{count}=[1 2 1];
237 dg_SoCSuspectTBRisk1=2*eye(nVar,nFactor);
238 % Define GRAPH View
239 dg_SoCSuspectTBRisk1([SoC Suspect TBRisk], WT)=1;
240 context_nodes{count}=[WT];
241 probvalues{WT}{count}=[0.7 0.25 0.05];
```

```
242 probvalues{TBRisk}=[0.7 0.3];
243 probvalues{AFB}=[0.7 0.3];
244 probvalues{MDT}=[0.7 0.3];
245 dg_SoCSuspectTBRisk1;
246 dg_Con{count}=dg_SoCSuspectTBRisk1;
247
248 % context graph with Context SoC,Suspect, PA: SoC=2,Suspect=2; TBRisk=2, AFB=1
249 count=count+1;
250 context{count}=[SoC Suspect TBRisk AFB];
251 value{count}=[1 2 2 1];
252 dg_SoCSuspectTBRiskAFB1=2*eye(nVar,nFactor);
253 % Define GRAPH View
254 dg_SoCSuspectTBRiskAFB1([SoC Suspect TBRisk AFB], WT)=1;
255 context_nodes{count}=[WT];
256 probvalues{WT}{count}=[0.7 0.25 0.05];
257 probvalues{AFB}=[0.7 0.3];
258 dg_SoCSuspectTBRiskAFB1;
259 dg_Con{count}=dg_SoCSuspectTBRiskAFB1;
260
261 % context graph with Context SoC,Suspect, PA: SoC=2,Suspect=2; TBRisk=2, AFB=2
262 count=count+1;
263 context{count}=[SoC Suspect TBRisk AFB];
264 value{count}=[1 2 2 2];
265 dg_SoCSuspectTBRiskAFB2=2*eye(nVar,nFactor);
266 % Define GRAPH View
267 dg_SoCSuspectTBRiskAFB2([SoC Suspect TBRisk AFB MDT], WT)=1;
268 context_nodes{count}=[WT]; % Context node Labels in each graph
269 probvalues{WT}{count}=[0.15 0.2 0.7 0.75 0.15 0.05];
270 probvalues{MDT}=[0.7 0.3];
271 dg_SoCSuspectTBRiskAFB2;
272 dg_Con{count}=dg_SoCSuspectTBRiskAFB2;
273
274 % context graph with Context SoC,Suspect, PA: SoC=3
275 count=count+1;
276 context{count}=[SoC];
277 value{count}=[3];
278 dg_SoC3=2*eye(nVar,nFactor);
279 % Define GRAPH View
280 dg_SoC3([SoC], IT)=1;
281 context_nodes{count}=[IT];
282 probvalues{IT}{count}=[0.7 0.25 0.05];
283 dg_SoC3;
284 dg_Con{count}=dg_SoC3;
285
286
287 % context graph with Context SoC,PC: SoC=1
288 count=count+1;
289 context{count}=[SoC];
290 value{count}=[1];
291 dg_SoCIT1=2*eye(nVar,nFactor);
292 dg_SoCIT1([SoC], IT)=1;
293 context_nodes{count}=[IT];
294 probvalues{IT}{count}=[0.05 0.05 0.9];
295 dg_SoCIT1;
296 dg_Con{count}=dg_SoCIT1;
```

```
297
298 % context graph with Context SoC,PC: SoC=2
299 count=count+1;
300 context{count}=[SoC];
301 value{count}=[2];
302 dg_SoCIT2=2*eye(nVar,nFactor);
303 dg_SoCIT2([SoC], IT)=1;
304 context_nodes{count}=[IT];
305 probvalues{IT}{count}=[0.05 0.05 0.9];
306 dg_SoCIT2;
307 dg_Con{count}=dg_SoCIT2;
308
309 meta2csn
310
311 evidence=cell(1,csn.nVars);
312
313 disp(['==========ORIGINAL GRAPH STATISTICS==========='])
314 statistics(csn)
315
316 evidence{Age}=1;
317 evidence{LowBP}=2;
318 evidence{RR}=2;
319 evidence{Urea}=2;
320 evidence{Conf}=2;
321
322 evidence{Oral}=1;
323 evidence{Anti}=1;
324 evidence{Healthy}=2;
325 evidence{Comorbid}=1;
326 evidence{Asp}=2;
327 evidence{NHR}=1;
328 evidence{Inf}=2;
329
330 observed=observedCSN(csn,evidence);
331 newcsn=observed.original;
332 queryGraphs=observed.query;
333
334 %%=========================================================================%%
335 % Inference over all seperable graphs
336 %%=========================================================================%%
337 for i=1:length(queryGraphs)
338     csn2=(queryGraphs{i}{1}.csn);
339     varindex=queryGraphs{i}{1}.varindex;
340     cpfindex=queryGraphs{i}{1}.cpfindex;
341     evidence2=queryGraphs{i}{2};
342     if isempty(match(newcsn.onodes,varindex))
343         disp(['==========OBSERVED GRAPH STATISTICS==========='])
344         disp(['Graph Number:' num2str(i)])
345         disp(['Variables in the Graph=' mat2str(varindex)])
346         disp(['Factors in the Graph=' mat2str(cpfindex)])
347         statistics(csn2)
348      %   marg= belpropv4tree(csn2);
349     else
350         disp(['==========OBSERVED GRAPH STATISTICS==========='])
351         disp(['Graph Number:' num2str(i)])
```

```
352           disp(['Variables in the Graph=' mat2str(varindex)])
353           disp(['Factors in the Graph=' mat2str(cpfindex)])
354           statistics(csn2)
355            %marg= belpropv4tree(csn2)
356       end
357   end
```

Glossary

This appendix contains a glossary of simplified medical concepts from [MedlinePlus, 2008; Stedman, 2005].

Aspiration :

The accidental inhaling in of foreign particles or fluids into the lungs.

Antimicrobial Drug ability of Resistance :

The ability of microorganisms, especially bacteria, to resist or to be tolerant to antibiotics or other chemotherapeutic or antimicrobial agents.

Body-Mass Index :

A number calculated from a persons weight and height.

Community-acquired Pneumonia :

An infection of the lungs (pneumonia) in individuals who have not recently been hospitalized.

Coronary Artery Disease :

A condition in which cholesterol and other substances build up inside the coronary arteries that supply the heart muscle with oxygen-rich blood.

Comorbidity :

The presence of co-existent diseases.

Empyema :

The presence of pus in a hollow organ or body cavity.

Oral Macrolides :

A class of antibiotics (such as erythromycin or clarithromycin) that are used to treat infections in the lower respiratory tract.

Pleural Aspiration (Thoracocentesis) :

The aspiration of fluid or air from the pleural space.

Pneumonia Severity Index (PSI) :

A severity score useful in assessing the probability of morbidity and mortality among patients with community acquired pneumonia. Two scores are often used: CURB and PORT score. CURB-65 is an acronym for Confusion, Urea (greater than 7 mmol/L), Respiratory rate (30/min or greater), low Blood pressure, and an age of 65 or older. PORT (Pneumonia Outcome Research Team) score is a more elaborated calculation of the severity score based on over 20 clinical patient attributes.

Quinolones and Fluoroquinolones :

Family of broad-spectrum antibiotics effective in the treatment of selected community acquired and nosocomial infections. Fluoroquinolones, a type of quinolone containing a fluorine atom, possess characteristics that make them more effective as anti-infectious agents.

References

Agrawal, R., Borgida, A., & Jagadish, H. (1989). Efficient management of transitive relationships in large data and knowledge bases. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*. 102

Aji, S. M., & Mceliece, R. J. (2000). The generalized distributive law. *IEEE Transactions on Information Theory*, *46*(2), 325–343. 13, 70, 171

Akman, V., & Surav, M. (1996). Steps towards formalizing context. *AI Magazine*, *17*(3), 55 72. 40

Alvarado, C., & Davis, R. (2005). Dynamically constructed bayes nets for multi-domain sketch understanding. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*. 97, 129, 167

Aronsky, D., & Haug, P. J. (1999). Diagnosing community-acquired pneumonia with a bayesian network. In *American Medical Informatics Association (AMIA) Annual Symposium Proceedings*. 135

Bacchus, F., Dalmao, S., & Pitassi, T. (2003). Value elimination: Bayesian inference via backtracking search. In *Proceedings of the 19th Uncertainty in Artificial Intelligence (UAI-03)*. 164

Barnett, G., Cimino, J., Hupp, J., & Hoffer, E. (1987). DXplain - an evolving diagnostic decision-support system. *Journal of the American Medical Association (JAMA)*, *258*(270), 67–74. 167

Bechhofer, S., & van Harmelen, F. (2004). Owl web ontology language reference:. Tech. rep., W3C Recommendation.
URL www.w3.org/TR/2004/REC-owl-ref-20040210/ 38

Blazquez, M., Koornneef, M., & Putteril, J. (2001). Flowering on time: genes that regulate the floral transition. Tech. Rep. 2, EMBO reports. 36, 37

Boutilier, C., Friedman, N., Goldszmidt, M., & Koller, D. (1996). Context-specific independence in bayesian networks. In *Proceedings of the 12th Uncertainity in Artificial Intelligence (UAI-96)*. 6, 25, 28, 42, 43, 46, 48, 123, 124, 159, 160, 161

Buntine, W. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, *8*, 195–210. 131, 179

Charniak, E. (1991). Bayesian networks without tears. *AI Magazine*, *12*(4), 50–63. 17, 168

Chavira, M., & Darwiche, A. (2005). Compiling bayesian networks with local structure. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*. 47, 123, 124

Chen, Q., Li, G., Han, B., Heng, C., & Leong, T. (2005). Coronary artery disease prediction with bayesian networks and constraint elicitation. Tech. rep., School of Computing, National University of Singapore. 135, 136, 140

Chickering, D. M., Heckerman, D., & Meek, C. (1997). A bayesian approach to learning bayesian networks with local structure. In *Proceedings of the 13th Conf. on Uncertainty in Artificial Intelligence (UAI-97)*. 165, 178

Context (2008). Context definition: Merriam-webster online dictionary. URL http://www.m-w.com/dictionary/context 32

Cooper, G., & Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*, 309–347. 179

Costa, P. C. G. (2005). *Bayesian Semantics for the Semantic Web*. Ph.D. thesis, School of Information Technology and Engineering, George Mason University. 163

D'Ambrosio, B. (1994). SPI in large BN2O networks. In *Proceedings of the 10th Conf. on Uncertainty in Artificial Intelligence (UAI-94)*. 3, 46

D'Ambrosio, B. (1995). Local expression languages for probabilistic dependence. *International Journal of Approximate Reasoning*, *13*(1), 61–81. 6, 28, 46, 123, 124

D'Ambrosio, B., Takikawa, M., & Upper, D. (1999). Representation for dynamic situation modeling. Tech. rep., Information Extraction and Transport,. 28, 47, 161

Darwiche, A. (2001). Recursive conditioning: Any space conditioning algorithm with treewidth bounded complexity. *Articial Intelligence*, *125*(1-2), 5–41. 105, 129, 164

Darwiche, A. (2002). A logical approach to factoring belief networks. In *Proceedings of Principles of Knowledge Representation and Reasoning (KR)*. 47, 123, 124, 164

Davidson, E., Rast, J., Oliveri, P., & et. al. (2002). A genomic regulatory network for development. *Science*, *2-5*, 1669–1678. 37

Davidyuk, O., Riekki, J., Rautio, V., & J., S. (2004). Context-aware middleware for mobile multimedia applications. In *3rd International Conference on Mobile and Ubiquitous Multimedia*. 37

Dey, A. (2000). *Providing Architectural Support for Building Context-Aware Applications*. Ph.D. thesis, Georgia Institute of Technology. 4, 5, 32

Druzdzel, M., & Suermondt, H. (1994). Relevance in probabilistic models: backyards in a small world. In *Working notes of the AAAI-1994 Fall Symposium Series: Relevance*. 101

Druzdzel, M., van der Gaag, L., Henrion, M., & Jensen, F. (1995). Building probabilistic networks: where do the numbers come from. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95) Workshop*. 177

Fine, M., Singer, D., Yealy, D., Coley, C., Hanusa, B., Kapoor, W., Marrie, T., & Weissfeld, L. (1997). A prediction rule to identify low-risk patients with community-acquired pneumonia. *New England Journal of Medicine (NEJM)*, *336*(4), 243–250. 146, 148

Foldoc (2008). Context definition: Free on-line dictionary of computing. URL http://foldoc.doc.ic.ac.uk/foldoc/index.html 32

Frey, B. (2003). Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the 19th Conf. on Uncertainty in Artificial Intelligence (UAI-03)*. 4, 10, 65, 170, 171, 172

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*. 6, 19, 27, 44, 47, 114, 117, 161, 172, 173

Friedman, N., & Goldszmidt, M. (1996). Learning Bayesian networks with local structure. In *Proceedings of the 12th Conf. on Uncertainty in Artificial Intelligence (UAI-96)*. 116, 133, 165, 178

Friedman, N., & Koller, D. (2003). Being Bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine Learning*, *50*(31), 95–125. 179

Friedman, N., Linial, M., Nachman, I., & Pe'er, D. (2000). Using Bayesian networks to analyse expression data. *Journal of Computational Biology*, *7*, 601–620. 167

Geiger, D., & Heckerman, D. (1996). Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, *82*, 45–74. 6, 7, 25, 45, 159, 160

Goldman, R., & Charniak, E. (1993). A language for construction of belief networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *5*(3), 196–208. 3

Guha, R. (1993). Context dependence of representations in CYC. Tech. rep., Microelectronics and Computer Technology Corporation. 7, 34, 40

Heckerman, D. (1991). *Probabilistic Similarity Networks*. The MIT Press. 3

Heckerman, D. (1995). A tutorial on learning with Bayesian networks. Tech. Rep. MSR-TR-95-06, Microsoft Research. 179

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., & Kadie, C. (2000). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, *1*, 49–75. 4

Heckerman, D., Geiger, D., & Chickering, D. (1995). Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learining*, *20*(3), 197–243. 179

Heckerman, D., Horvitz, E., & Nathwani, B. (1992). Toward normative expert systems: Part I. The Pathfinder project. *Methods of Information in Medicine*, *31*, 90–105. 167

Heckerman, D., Meek, C., & Koller, D. (2004). Probabilistic models for relational data. Tech. rep., Microsoft Research. 6, 44, 172

ICSI (2006). Community-acquired pneumonia in adults. Institute for Clinical Systems Improvement (ICSI).
URL http://www.guideline.gov/algorithm/5034/NGC-5034_2.html 146

Intille, S. S. (2004). A new research challenge: persuasive technology to motivate healthy aging. *Transactions on Information Technology in Biomedicine*, *8*(3), 235–237. 135

Jaeger, M. (2001). Complex probabilistic modeling with recursive relational bayesian networks. *Annals of Mathematics and Artificial Intelligence*, *32*, 179–220. 48, 123, 172

Jaeger, M. (2004). Probabilistic decision graphs-combining verification and AI techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge Base Systems*, *12*, 19–42. 46, 47, 123

Jones, G. (2004). The role of context in information retrieval. In *SIGIR Workshop on Information Retrieval in Context*. 33

Joshi, R., & Leong, T. (2006). Patient-specific inference and situation-dependent classification using context-sensitive networks. *American Medical Informatics Association (AMIA) Annual Symposium Proceedings*, *2006*, 404–408. 10, 134

Joshi, R., Li, G., & Leong, T. (2007). Context-aware probabilistic reasoning for proactive healthcare. In *Proceedings of the International Joint Conf on Artificial Intelligence (IJCAI) Workshop on Ambient Intelligence (AITAmI07)*. 10, 14, 134

Kim, J. H., & Pearl, J. (1983). A computation model for causal and diagnostic reasoning in inference systems. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI)*. 166

Koller, D. (1999). Probabilistic relational models. In *Proceedings of 9th International Workshop on Inductive Logic Programming*. 4

Koller, D., & Pfeffer, A. (1997). Object-oriented bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*. 4

Kschischang, F., Frey, B., & Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*, 498–519. 70, 76, 82, 164, 171, 174, 176

Kschischang, F. R., & Frey, B. J. (1998). Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Communications*, *16*(2), 219–230. 83, 176

Lam, W., & Bacchus, F. (1994). Learning bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, *10*(4). 179

Laskey, K. (2007). MEBN: A language for first-order bayesian knowledge bases. *Artificial Intelligence*, *172*, 2–3. 172

Laskey, K., & Mahoney, S. (1997). Network fragments: Representing knowledge for constructing probabilistic models. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*. 3, 47, 159

Laskey, K., Mahoney, S., & Wright, E. (2001). Hypothesis management in situation-specific network construction. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI-01)*. 19, 27, 161

Leong, T. (1998). Multiple perspective reasoning. *Artificial Intelligence*, *105*(1-2), 209–261. 3

Lim, T. (2006). Use of antibiotics in community acquired pneumonia. Singapore MOH Clinical Practice Guidelines: Use of Antibiotics In Adults. 145, 146, 149, 150

Lin, Y., & Druzdzel, M. (1997). Computational advantages of relevance reasoning in bayesian belief networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*. 101

Locke, J. (1999). Microsoft bayesian networks: Basics of knowledge engineering. Tech. rep., Microsoft Support Technology. 167

Mahoney, S., & Laskey, K. (1998). Constructing situation specific networks. In *Proceedings of the 14th Conference of Uncertainty in Artificial Intelligence (UAI-98)*. 7, 47, 97, 123, 124, 159

McCarthy, J. (1993). Notes on formalizing context. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*. 34

MedlinePlus (2008). MedlinePlus: Medical dictionary. URL http://www.nlm.nih.gov/medlineplus/mplusdictionary.html 199

Middleton, B., Shwe, M., Heckerman, D., Henrion, M., Horvitz, E., Lehmann, H., & Cooper, G. (1991). Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: Part II. Evaluation of diagnostic performance. *SIAM Journal on Computing*, *30*, 256–267. 167

Milch, B. (2006). *Probabilistic models with unknown objects*. Ph.D. thesis, UC Berkley. 124, 163

Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D., & Kolobov, A. (2004). Blog: First-order probabilistic models with unknown objects. Tech. rep., UC Berkeley. 47

Murphy, K. (2002a). Bayes Net Toolbox (BNT). (accessed: 2004). URL http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html 14, 123

Murphy, K. (2002b). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, UC Berkeley, Computer Science Division. 163

Murphy, K., Weiss, Y., & Jordan, M. (1999). Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*. 175

Ngo, L., Haddawy, P., & Helwig, J. (1995). A theoretical framework for context-sensitive temporal probability model construction with application to plan projection. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI-95)*. 3, 7, 47, 48, 97, 159

Onisko, A., Druzdzel, M., & Wasyluk, H. (2000). Extension of the hepar II model to multiple-disorder diagnosis. In *Advances in Soft Computing*. 144, 151

Ortiz, L., & Kearns, M. (2002). Nash propagation for loopy graphical games. In *Advances in Neural Information Processing (NIPS)*. 70

Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2002). Identity uncertainty and citation matching. In *Advances in Neural Information Processing (NIPS)*. 47

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann. 2, 5, 12, 65, 70, 105, 129, 167, 168, 172, 174

Peleg, M., Tu, S., Bury, J., Ciccarese, P., Fox, J., Greenes, R., Hall, R., Johnson, P., Jones, N., Kumar, A., Miksch, S., Quaglini, S., Seyfang, A., Shortliffe, E., & Stefanelli, M. (2003). Comparing computer-interpretable guideline models: A case-study approach. *Journal of the American Medical Informatics Association*, *10*, 52–68. 145

Pessoa, R., Calvi, C., Filho, J., de Farias, C., & Neisse, R. (2007). Semantic context reasoning using ontology based models. *Lecture Notes in Computer Science: Dependable and Adaptable Networks and Services*, *4606*, 44–51. 38

Poh, K., & Fehling, M. (1993). Probabilistic conceptual network: a belief representation for utility-based categorization. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence (UAI-93)*. 3

Poh, K., & Horvitz, E. (1996). A graph-theortic analysis of information value. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI-96)*. 163

Poole, D. (1997). Probabilistic partial evaluation: Exploiting rule structure in probabilistic inference. In *Proceedings of the 15th International Joint Conf. on Artificial Intelligence (IJCAI-97)*. 46, 160, 161

Poole, D. (1998). Context-specific approximation in probabilistic inference. In *Proceedings of the 14th Conf. on Uncertainty in Artificial Intelligence*. 46

Poole, D., & Zhang, N. (2003). Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence Research*, *18*, 263–313. 6, 28, 46, 123, 124, 159, 160, 161

Pope, J., Aufderheide, T., Ruthazer, R., & et. al. (2000). Missed diagnosis of acute cardiac ischemia in the emergency department. *New England Journal of Medicine*, *342*, 1163–1170. 135

Ranganathan, A., & Campbell, R. (2003). An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, *7*(6), 353 364. 37

Rieman, B., Peterson, J., Clayton, J., Howell, P., Thurow, R., Thompson, W., & Lee, D. (2001). Evaluation of potential effects of federal land management alternatives on trends of salmonids and their habitats in the interior columbia river basin. *Forest ecology and management*, *153*(1), 43–62. 167

Rossi, G., Gordillo, S., & Lyardet, F. (2005). Design patterns for context-aware adaptation. In *International Workshop on Context-aware Adaptation and Personalization for the Mobile Internet*. 37

Salber, D., Dey, A., & Abowd, G. (1999). The context toolkit: Aiding the development of context-enabled applications. In *Proceedings of Conference on Human Factors in Computer Systems (CHI-99)*. 37

Sanders, G. (1997). *Automated creation of clinical-practice guidelines from decision models*. Ph.D. thesis, Stanford Medical Informatics. 4, 144

Sang, T., Beame, P., & Kautz, H. (2005). Heuristics for fast exact model counting. In *7th International Conference on Theory and Applications of Satisfiability Testing (SAT)*. 164

Segal, E., Peer, D., Regev, A., Koller, D., & Friedman, N. (2004). Learning module networks. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-04)*. 4

Shortliffe, E. (1976). *Computer-Based Medical Consultations: MYC*. American Elsevier. 34, 41

Song, D., & Bruza, P. (2003). Towards context sensitive information inference. *Journal of the American Society for Information Science and Technology*, *54*(4), 321–334. 32

Stedman (2005). *Stedman's Medical Dictionary*. Lippincott Williams and Wilkins. 199

Sundaresh, S., Leong, T., & Haddawy, P. (1999). Supporting multi-level multiperspective dynamic decision making in medicine. In *American Medical Informatics Association (AMIA) Annual Symposium Proceedings*, (pp. 161–165). 3

Terziyan, V. (2006). Bayesian metanetwork for context-sensitive feature relevance. *Lecture Notes in Computer Science*, *3955*, 356–366. 4, 20, 161

van der Gaag, L., Renooij, S., Witteman, C., Aleman, B., & Taal, B. (2002). Probabilities for a probabilistic network: a case study in oesophageal cancer. *Artificial Intelligence in Medicine*, *25*, 123–148. 148

Walther, E., Eriksson, H., & Musen, M. (1992). Plug-and play: Construction of task-specific expert-system shells using sharable context ontologies. In *Proceedings of the AAAI Workshop on Knowledge Representation Aspects of Knowledge Acquisition*. 32

Whorf, B. (1956). *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf*. MIT Press. 2

Wong, S. K. M., & Butz, C. (1999). Contextual weak independence in bayesian networks. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*. 43

Wu, X. (1998). *Decision Model Construction with Multilevel Influence Diagrams*. Master's thesis, Department of Industrial & Systems Engineering, NUS. 3

Wyatt, J. (2000). Clinical knowledge and practice in the information age. *Journal of the Royal Society of Medicine*, *93*, 530–534. 145

Xiang, Y. (2002). *Probabilistic reasoning in multi-agent systems : a graphical models approach*. Cambridge University Press. 13, 70

Yedidia, J., Freeman, W., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on In Information Theory*, *51*(7), 2282–2312. 177

Zeng, Y. (2006). *Probabilistic modeling and reasoning in multiagent decision systems*. Ph.D. thesis, Naitonal University of Singapore. 4

Zhang, N. L., & Poole, D. (1999). On the role of context-specific independence in probabilistic reasoning. In *Proceedings of the 16th International Joint Conf. on Artificial Intelligence (IJCAI-99)*. 6, 25

Zhong, C., & Li, P. (2000). Bayesian belief network modeling and diagnosis of xerographic systems. In *Proceedings of the ASME Symposium on Controls and Imaging (IMECE)*. 167

Zhou, R. (2005). *Automated Guideline Development*. Master's thesis, National University of Singapore. 4, 144

Zhu, A., & Leong, T. (2000). Automating dynamic decision model construction to support clinical practical guideline development. In *EWGLP 2000 Workshop on Computer-based Support for Clinical Guidelines and Protocols*. 4, 144