Compiling Possibilistic Graphical Models : From Inference to Decision

THÈSE

(en cotutelle)

pour l'obtention du

Doctorat de l'Université d'Artois & de l'Université de Tunis

Spécialité Informatique

par

Raouia Ayachi

Composition du jury

Directeurs de thèse :	Salem Benferhat, Professeur des Universités (Université d'Artois, France)
	Nahla Ben Amor, Professeur de l'enseignement supérieur (Université de Tunis, Tunisie)
Rapportrices :	Hélène Fargier, Directrice de recherche CNRS (Université Paul Sabatier, France)
	Faiza Haned Khellaf, Maître de Conférences (Université des Sciences de la Technologie Houari Boumediene, Algérie)
Examinateurs :	Philippe Leray, Professeur des Universités (Université de Nantes, France)
	Bertrand Mazure, Professeur des Universités (Université d'Artois, France)
	Zied Elouedi, Professeur de l'enseignement supérieur (Université de Tunis, Tunisie)
	Karim Tabia, Maître de Conférences (Université d'Artois, France)

Centre de Recherche en Informatique de Lens – CNRS UMR 8188
Université d'Artois, rue Jean Souvraz, S.P. 18 F-62307, Lens Cedex France
Secrétariat : Tél.: +33 (0)3 21 79 17 23 – Fax : +33 (0)3 21 79 17 70
http://www.cril.fr

LAboratoire de Recherche Opérationnelle, de Décision et de Contrôle de processus
Institut Supérieur de Gestion, 41, avenue de la liberté, Le Bardo 2000, Tunis, Tunisie
Secrétariat : Tél.: +216 71 56 03 13 – Fax : +216 71 56 87 67
http://www.larodec.com

Abstract

This thesis addresses two important issues in reasoning and decision making under uncertainty. At first, we have developed compilation-based inference methods dedicated to possibilistic networks. In fact, we have adapted the standard approach initially proposed for Bayesian networks into a possibilistic framework and we have refined it using *local structure*. We have also proposed a new encoding strategy, called *possibilistic local structure*, exclusively useful in a qualitative framework. Moreover, we have implemented a purely possibilistic approach based on transforming possibilistic networks into possibilistic knowledge bases. Our second contribution consists in extending our inference approaches to possibilistic causal networks in order to efficiently compute the impact of both observations and interventions. We have confronted, in particular, mutilated-based approaches and augmented-based ones. Finally, we have explored the decision-making aspect under compilation by extending our results on compiling possibilistic networks to efficiently evaluate possibilistic influence diagrams. An experimental study evaluating the different approaches studied in this thesis is also presented.

Résumé

Cette thèse traite deux problèmes importants dans le domaine du raisonnement et de la décision dans l'incertain. En premier lieu, nous développons des méthodes d'inférence basées sur la compilation pour les réseaux possibilistes. En effet, nous commençons par adapter au cadre possibiliste l'approche de base proposée, initialement, pour les réseaux Bayésiens et nous la raffinons, ensuite en utilisant la notion de *structure locale*. Nous proposons aussi une nouvelle stratégie de codage appelée *structure locale possibiliste* appropriée dans le cadre qualitatif. Nous implémentons, par ailleurs, une méthode purement possibiliste basée sur la transformation des réseaux possibilistes en bases de connaissances possibilistes. Notre deuxième contribution consiste à étendre nos approches d'inférence dans le cadre des réseaux causaux afin de calculer l'effet des observations et des interventions d'une manière efficace. Nous confrontons, en particulier, des approches basées sur la mutilation et celles basées sur l'augmentation. Finalement, nous étudions l'aspect décisionnel sous compilation en étendant nos résultats portant sur la compilation des réseaux possibilistes afin d'évaluer les diagrammes d'influence possibilistes. Une étude expérimentale évaluant les différentes approches étudiées dans cette thèse est également présentée.

Acknowledgements

I have benefited a great deal form a large number of individuals who provided help at several levels.

I thank first Pr. Nahla Ben Amor, the supervisor of this dissertation. She inspired my scientific curiosity through many worthwhile and helpful discussions. Her permanent guidance led to a wonderful atmosphere resulting in the work at hand.

I am very grateful to my supervisor Pr. Salem Benferhat for his support and advices in the fulfillment of my PhD. I am deeply indebted to you for your encouragements and for your trust.

It is my greatest pleasure to express my gratitude to all members of my dissertation committee who accepted to assess this work. Special thanks to Mrs Hélène Fargier for her useful remarks during her visits in Tunisia.

Many thanks go to members of my laboratory *LARODEC* at the Higher Institute of Management of Tunis and those of my laboratory *CRIL* at University of Artois.

On a more private note, I thank my lovely parents and sisters for providing a wonderful environment in which I could make progress. I also thank my friends for their help. My last cordial thanks are addressed to my dear Skander for his absolute support and belief in my abilities.

Contents

General introduction **1**

I State of the art **4**

1 Logical and graphical representations of possibilistic knowledge **5**

1.1 Introduction 5

1.2 Notations and definitions 6

1.3 Possibility theory 8

1.3.1 Possibility distribution 8

1.3.2 Possibility and necessity measures 9

1.3.3 Quantitative and qualitative settings 11

1.3.4 Possibilistic conditioning 12

1.4 Possibilistic knowledge bases 14

1.5 Possibilistic networks 17

1.5.1 Recall on Bayesian networks 18

1.5.2 Representation of possibilistic networks 20

1.5.3 Inference in possibilistic networks 22

1.6 Conclusion 27

2 Knowledge compilation **28**

2.1 Introduction 28

2.2 Propositional logic . . . 28
2.2.1 Syntax . . . 29
2.2.2 Semantic . . . 30
2.3 Principle of knowledge compilation . . . 31
2.4 Knowledge compilation map . . . 33
2.4.1 Target compilation languages . . . 34
2.4.2 Succinctness . . . 36
2.4.3 Logical queries and transformations . . . 37
2.5 DNNF language . . . 40
2.5.1 DNNF's operations . . . 40
2.5.2 From CNF to DNNF . . . 43
2.6 Conclusion . . . 47

II Compiling possibilistic graphical models 48

3 Compilation-based inference in min-based possibilistic networks 49

3.1 Introduction . . . 49
3.2 Compilation of Bayesian networks . . . 50
3.3 Possibilistic adaptation of standard probabilistic inference approach . . . 53
3.3.1 Encoding phase . . . 54
3.3.2 Compilation phase . . . 59
3.3.3 Inference phase . . . 59
3.4 Possibilistic approach using possibilistic knowledge bases . . . 65
3.4.1 From a graphical to a logic-based representation . . . 65
3.4.2 Compilation-based inference using possibilistic knowledges bases . . . 66
3.5 Conclusion . . . 68

4 Refined CNF encodings of min-based possibilistic networks 69

4.1 Introduction . . . 69

4.2 Local structure . . . 70

4.2.1 CNF encoding . . . 70

4.2.2 Compiled base . . . 75

4.3 Possibilistic local structure . . . 76

4.3.1 CNF encoding . . . 76

4.3.2 Compiled base . . . 83

4.4 Illustrative example of encoding strategies . . . 84

4.5 Particular case of binary networks . . . 86

4.5.1 Binary CNF encoding . . . 87

4.5.2 Binary compiled base . . . 92

4.6 Illustrative example of binary approaches . . . 93

4.7 Conclusion . . . 96

5 Handling interventions under compilation 97

5.1 Introduction . . . 97

5.2 Refresher on possibilistic causal networks . . . 98

5.2.1 Possibilistic causal networks . . . 99

5.3 Mutilated-based approaches . . . 102

5.3.1 Mutilated Π-DNNF . . . 103

5.3.2 Mutilated compiled possibilistic knowledge bases . . . 110

5.4 Augmented-based approaches . . . 114

5.4.1 Augmented Π-DNNF . . . 114

5.4.2 Augmented compiled possibilistic knowledge bases . . . 123

5.5 Conclusion . . . 128

6 Beyond compiling min-based possibilistic networks 129

6.1 Introduction . . . 129

6.2 Compilation-based inference in product-based possibilistic networks . . . 130

6.2.1 Encoding and compilation phase . . . 130

6.2.2 Inference phase . . . 132

6.3 Evaluation of possibilistic influence diagrams . . . 135

6.3.1 Possibilistic influence diagrams . . . 135

6.3.2 Compilation-based evaluation of possibilistic IDs . . . 137

6.4 Conclusion . . . 141

7 Implementation and Experimentations 143

7.1 Introduction . . . 143

7.2 Experimental protocol . . . 144

7.2.1 DAG structure . . . 144

7.2.2 DAG parameters . . . 146

7.3 Comparing compilation-based inference approaches . . . 148

7.3.1 n-ary approaches . . . 148

7.3.2 Binary approaches . . . 158

7.4 Comparing interventions-based methods under compilation . 161

7.5 Bayesian networks vs product-based possibilistic networks under compilation . . . 164

7.6 Conclusion . . . 166

Conclusion 167

Bibliography 169

List of Figures

1.1 A Bayesian network 19

1.2 A small possibilistic network 21

1.3 A possibilistic network 23

1.4 The junction tree associated to the DAG of Figure 1.3 23

2.1 Knowledge compilation map [34] 34

2.2 An NNF formula 35

2.3 Succinctness relation between target compilation languages [34] (An edge $\mathfrak{L}_1 \rightarrow \mathfrak{L}_2$ indicates that $\mathfrak{L}_1$ is strictly more succinct than $\mathfrak{L}_2$, while $\mathfrak{L}_1 = \mathfrak{L}_2$ indicates that $\mathfrak{L}_1$ and $\mathfrak{L}_2$ are equally succinct. Dotted arrows indicate unknown relationships) 36

2.4 A DNNF conditioned on $a \wedge b \wedge c \wedge d$ 42

2.5 A DNNF after forgetting $P = \{a, b\}$ 43

2.6 A counting graph 44

2.7 A decomposition tree 46

3.1 Principle of possibilistic compilation-based inference approaches (Lines annotated by 3.3 (resp. 3.4) are relative to the method Π-DNNF (resp. DNNF-PKB), while non annotated lines are shared by both methods) 50

3.2 Computing $P(b_1, a_2)$ 54

3.3 Computing $P(a_2)$ 55

3.4 The compiled base CB of ΠG_{min} 62

3.5 The min-max circuit CB_{minmax} 62

4.1 A possibilistic network of 4 nodes 72

4.2 CB^{LS}_{minmax}, $CB^{PLS}_{minmax_l}$ and CB^{PLS}_{minmax} 86
4.3 CB_{bin}, CB^{LS}_{bin} and CB^{PLS}_{bin} 94

5.1 An Example of a causal network 99
5.2 A possibilistic causal network ΠG_{min} 100
5.3 The mutilated network ΠG_{mut} 101
5.4 The augmented network ΠG_{aug} 102
5.5 Summary of proposed methods using mutilation and augmentation 103
5.6 Principle of mutilated-based approaches (Lines labeled with (a) (resp. (b)) are relative to Mut-Π-DNNF (resp. Mut-DNNF-PKB) and unlabeled lines are relative to to both methods) 104
5.7 The mutilated compiled base CB^{mut} 107
5.8 $CB^{mut}|a_2, b_1$, ΠCB^{mut} and $\Pi_c(a_2, do(b_1))$ 108
5.9 Principle of augmented-based approaches ((a): Augmented-based approaches of the possibilistic adaptation), (b): Augmented-based approach of the possibilistic logic counterpart)) 115
5.10 The augmented compiled base C^{aug}_{DNNF} 121
5.11 Computing $\Pi_c(a_2, do(b_1))$ 122
5.12 The augmented compiled base K^{aug}_c 126

6.1 The prod-max circuit CB^{LS}_{*max} 134
6.2 A possibilistic influence diagram 136
6.3 Principle of compilation-based evaluation 138
6.4 The min-max circuit of ΠG^{ID}_{min} 141

7.1 Summary of studied methods 144
7.2 Three-nodes DAG structures 145
7.3 Three different locations of a redundant degree 147
7.4 Number of variables 151
7.5 Number of clauses 152
7.6 Number of edges 154

7.7 (a): Edges of Π-$DNNF$ and Π-$DNNF_{LS}$, (b) Edges of Π-$DNNF^{l}_{PLS}$, Π-$DNNF_{PLS}$ and DNNF-PKB 155

7.8 Breaking points between methods 155

7.9 (a): Inference time of Π-$DNNF$ and Π-$DNNF_{LS}$, (b) Inference time of Π-$DNNF^{l}_{PLS}$, Π-$DNNF_{PLS}$ and DNNF-PKB . 158

7.10 Mut-Π-DNNF vs Aug-Π-DNNF$_{LS}$ 164

List of Tables

1.1 Possibility measure Π (case of normalized possibility distributions) . . . 10

1.2 Necessity measure N (case of normalized possibility distributions) . . . 11

1.3 Joint possibility distribution after conditioning . . . 14

1.4 Joint possibility distribution of Example 1.8 . . . 16

1.5 Joint distribution of BN . . . 19

1.6 Joint distributions of ΠG_* and ΠG_{min} . . . 21

1.7 Local possibility distributions of clusters . . . 26

2.1 Interpretations of $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \Rightarrow \beta$ and $\alpha \Leftrightarrow \beta$ . . . 30

2.2 Notations for queries and transformations . . . 37

2.3 Subsets of the NNF language and their corresponding polytime queries ($\surd$ means 'satisfies' $\circ$ means 'does not satisfy unless P=NP' and ? means 'unknown') [34] . . . 38

2.4 Subsets of the NNF language and their corresponding polytime transformations ($\surd$ means 'satisfies', $\bullet$ means 'does not satisfy', $\circ$ means 'does not satisfy unless P=NP' and ? means 'unknown') [34] . . . 39

3.1 Instance indicators used in f_{MLF} . . . 51

3.2 Network parameters used in f_{MLF} . . . 52

3.3 The CNF encoding of BN . . . 53

3.4 Instance indicators used in C_{min} . . . 55

3.5 Parameter variables used in C_{min} . . . 56

3.6 The CNF encoding C_{min} of ΠG_{min} . . . 57
3.7 The CNF encoding K_Σ of Σ_{min} . . . 67

4.1 Parameter variables using local structure . . . 73
4.2 Parameter variables using possibilistic local structure . . . 77
4.3 Instance indicators used in C^{LS}_{min}, $C^{PLS}_{min_l}$ and C^{PLS}_{min} . . . 84
4.4 Parameter variables used in C^{LS}_{min}, $C^{PLS}_{min_l}$ and C^{PLS}_{min} . . . 85
4.5 The CNF encodings C^{LS}_{min}, $C^{PLS}_{min_l}$ and C^{PLS}_{min} . . . 85
4.6 Instance indicators used in C_{bin}, C^{LS}_{bin} and C^{PLS}_{bin} . . . 93
4.7 Parameter variables used in C_{bin}, C^{LS}_{bin} and C^{PLS}_{bin} . . . 93
4.8 The CNF encodings C_{bin}, C^{LS}_{bin} and C^{PLS}_{bin} . . . 94

5.1 The CNF encoding K_Σ of Σ_{min} of Figure 5.2 . . . 112
5.2 Instance indicators used in C_{aug} . . . 117
5.3 Network parameters used in C_{aug} . . . 118
5.4 The CNF encoding C_{aug} of ΠG_{aug} of Figure 5.4 . . . 119
5.5 Propositional variables used in $K_{\Sigma_{aug}}$ . . . 125
5.6 The CNF encoding $K_{\Sigma_{aug}}$ of Σ_{aug} of ΠG_{aug} of Figure 5.4 . . . 125

6.1 The possibility distribution of the utility node . . . 139
6.2 The CNF encoding C^{PLS}_{min} of ΠG^{ID}_{min} . . . 140

7.1 CNF and compiled bases parameters of the network of Figure 7.2 . . . 146
7.2 CNF and compiled bases parameters of the network of Figure 7.3 . . . 148
7.3 Π-DNNF vs Π-DNNF$_{LS}$ vs Π-DNNF$^l_{PLS}$ vs Π-DNNF$_{PLS}$ vs DNNF-PKB (better values are in bold) . . . 149
7.4 Variables ratio . . . 151
7.5 Clauses ratio . . . 152
7.6 Edges ratio . . . 154
7.7 Inference ratio . . . 157

7.8 Inference time of junction tree approach 158
7.9 Bin-Π-DNNF vs Bin-Π-DNNF$_{LS}$ vs Bin-Π-DNNF$_{PLS}$ 159
7.10 Bin-Π-DNNF vs Bin-Π-DNNF$_{LS}$ 159
7.11 Bin-Π-DNNF$_{LS}$ vs Bin-Π-DNNF$_{PLS}$ 160
7.12 Π-DNNF$_{LS}$ vs Bin-Π-DNNF$_{LS}$ 160
7.13 Π-DNNF$_{PLS}$ vs Bin-Π-DNNF$_{PLS}$ 161
7.14 Mut-Π-DNNF vs Mut-DNNF-PKB vs Aug-Π-DNNF$_{LS}$ (better values are in bold 162
7.15 Prod-Π-DNNF vs Prob-d-DNNF (better values are in bold) . 165
7.16 Average of Prod-Π-DNNF and Prob-d-DNNF parameters . . 166

General introduction

Graphical models are powerful models for representing and analyzing uncertain information. They are characterized by their clarity and their efficient management and storage of the information. *Bayesian networks* [32, 67] are studied under the broader class of probabilistic graphical models. Within the most prominent topics of Bayesian networks, we are, in particular, interested in *inference* which determines how the realization of specific values of some variables affects remaining variables, known to be NP-complete except for singly connected networks [25]. Recently, inference has been studied using new techniques, namely *knowledge compilation* [23, 24, 30, 82] which consists in preprocessing a propositional theory only once in an off-line phase, in order to make frequent on-line queries efficient [21].

Besides standard probabilistic networks, several non classical graphical models were studied to handle some particular situations which may compromise the application of probabilistic models, for instance when all numerical data are not available. We are, in particular, interested in the *possibility theory framework* [39, 42] which offers a natural and simple model to handle uncertain information numerically or qualitatively. This leads to two different ways to define the possibilistic counterpart of Bayesian networks, namely *product-based possibilistic networks* and *min-based possibilistic networks*.

Few works have addressed the inference problem in possibilistic networks [1, 11, 20, 46, 48] and most of them are based on a message passing mechanism and considered as a direct adaptation of probabilistic inference algorithms [56, 58, 67]. Despite its interest in the probabilistic case, the idea of compilation has not been explored to possibilistic inference. The aim of this thesis is to study possibilistic inference under compilation in both ordinal and numerical settings by handling *min-based possibilistic networks* and *product-based possibilistic networks*. Computing the impact of external events coming from outside the system, known as *interventions*, is also studied for possibilistic causal networks under a compilation framework. The decisional aspect using compilation will be also explored via *possibilistic influence diagrams*.

In the first part of this thesis, Chapter 1 gives essential background on possibility theory. It is also devoted to the presentation of two powerful models for representing and analyzing uncertain information in the possibility theory framework, namely graphical and logical-based representations. Chapter 2 introduces knowledge compilation and target compilation languages, in particular, the one used in our proposals, namely DNNF.

The second part is dedicated to our compilation-based inference algorithms. Our focus is, at first, on min-based possibilistic networks since the minimum operator has specific properties (e.g. *idempotency*) which are different from the product operator used in both Bayesian and product-based possibilistic networks. More precisely, we develop in Chapter 3 compilation-based inference methods dedicated for min-based possibilistic networks. The basic idea of such inference consists in encoding the initial network into a propositional base, usually a *conjunctive normal form (CNF)* and compiling it into a target compilation language that guarantees a polynomial inference. In fact, we propose a possibilistic adaptation of the standard probabilistic inference approach of [30] and a purely possibilistic inference method based on the transformation of possibilistic networks into possibilistic knowledge bases [12]. The possibilistic adaptation does not take into account any numerical value in the encoding phase. In other terms, it associates a propositional variable per parameter, regardless of its value.

Our idea in Chapter 4 consists in refining such encoding by dealing with specific values of parameters. In fact, two types of encoding strategies are explored. The first one, named *local structure* and used in both probabilistic and possibilistic networks, consists in assigning one propositional variable per equal parameters per possibility table. This encoding strategy does not take into account specific features of possibility theory such as the ordinal nature of uncertainty scale, which motivates us to propose a new encoding strategy, named *possibilistic local structure* and dealing with equal parameters from a global point of view. This latter is exclusively useful for min-based possibilistic networks since it exploits the idempotency property of the min operator. Finally, the specificity of binary variables is taken into consideration to refine the n-ary encoding (when $n = 2$) and explore compilation-based inference in min-based binary possibilistic networks.

In Chapter 5, we study handling interventions under compilation. Indeed, the concept of *intervention* is of major importance to have a complete and coherent causal analysis in a dynamic framework. It is an external event that forces some variables to have some particular values [69]. There are two ways to handle interventions using possibilistic causal networks [17]. The most intuitive one, called *mutilation*, consists in ignoring relations between the intervened variable and its direct causes. The rest of the network remains intact. Hence, causal inference resides in applying the inference algorithm to

the mutilated possibilistic network. A different but equivalent approach to represent intervention in possibilistic causal networks, called *augmentation*, is to consider it as an additional variable into the system. Our contribution in this Chapter is to extend inference approaches proposed in the previous chapters and propose mutilated-based approaches and augmented-based approaches aiming to compute the effect of both observations and interventions in an efficient manner in possibilistic causal networks.

Chapter 6 goes beyond min-based possibilistic networks. In fact, it concerns compiling product-based possibilistic networks while emphasizing on similarities and differences between product-based possibilistic networks, min-based possibilistic networks and Bayesian networks. It also explores the decisional aspect under compilation. We deal, in particular, with the possibilistic counterpart of standard influence diagrams [54], namely *possibilistic influence diagrams* [47] which model decision makers preferences on a sequence of decisions in a compact way. Our idea is to extend our results on compiling possibilistic networks to efficiently evaluate possibilistic influence diagrams and generate optimal strategies.

Finally, Chapter 7 provides the experimental study aiming to compare results of our approaches.

Part I

State of the art

Chapter 1

Logical and graphical representations of possibilistic knowledge

1.1 Introduction

Uncertain information should be modeled in most real-world problems using the appropriate tools. For instance, the standard probability theory has proved its efficiency when all numerical data are available. Nevertheless, such theory, as good as it is, is not suitable when dealing with the case of *total ignorance* [43]. Moreover, experts are generally unable to provide precise numerical values representing imperfect information.

Several non-classical theories of uncertainty have come into challenge to deal with uncertain and imprecise data such as Evidence theory [75, 76, 77], Spohn's ordinal conditional functions [78, 79] and Possibility theory [39, 38, 85] issued from fuzzy sets theory [57, 84]. We are, in particular, interested in the *Possibility theory* which is a convenient uncertainty framework offering a natural and a simple model to handle uncertain information. It is considered as an appropriate framework for experts to express their opinions about uncertainty either *numerically* using possibility degrees or *qualitatively* using a total pre-order on the universe of discourse.

Several graphical and logical-based methods have been proposed to reason with uncertain information. In this chapter, our focus will be on both logical-based representations by means of possibilistic knowledge bases and graphical-based representations, in particular, possibilistic networks [20] characterized by their clarity and their efficient management and storage of uncertain information. These latters are viewed as counterparts of probabilis-

tic Bayesian networks [32, 67] which are in their turn studied under the broader class of probabilistic graphical models. Possibilistic networks can be considered very practical when experts cannot provide numerical values or information is qualitative. Existing works on possibilistic networks focus on inference which remains an NP-complete problem as in the probabilistic framework [25].

This chapter is organized as follows: Section 1.2 introduces some notations and definitions used in this chapter. Section 1.3 is devoted to the presentation of prominent concepts relative to possibility theory. Section 1.4 and 1.5 present the logical and graphical representations of possibilistic knowledge, namely possibilistic knowledge bases and possibilistic networks.

1.2 Notations and definitions

We first give some notations and definitions.

- $V = \{X_1, X_2, ..., X_N\}$ is a set of variables,
- $D_{X_i} = \{x_{i1}, ..., x_{in}\}$ denotes the finite domain associated with the variable X_i.
- x_{ij} denotes any instance of X_i. When there is no ambiguity, we use x_i instead of x_{ij}.
- $X, Y, ..., Z$ denote subsets of variables from V,
- $D_X = \underset{X_i \in X}{\times} D_{X_i}$ denotes the Cartesian product of domains of variables in X,
- x denotes any instance of X, if $X = \{X_1, ..., X_n\}$ then $x = (x_1, ..., x_n)$,
- $\Omega = \underset{X_i \in V}{\times} D_{X_i}$ denotes the universe of discourse, which is the Cartesian product of all variable domains in V,
- Each element $\omega \in \Omega$ is called an *interpretation*, a possible *world* or a *state* of Ω. Depending on the context, we use one of the following notations:
  - either tuples: $\omega = (x_1, ..., x_N)$
  - or conjunctions: $\omega = x_1 \wedge ... \wedge x_N$, then $\omega[X_i] = x_i$.
- ϕ, ψ, φ denote the subclasses of Ω (called events) and $\neg\phi$ denotes the complementary set of ϕ i.e., $\neg\phi = \Omega - \phi$,

- $\phi \wedge \psi$ (resp. $\phi \vee \psi$) denotes the intersection (resp. the union) of ϕ and ψ.

The following notations concern networks. Let $V = \{X_1, X_2, ..., X_N\}$ be a finite set of variables. Let E be a part of $V \times V$. Then,

- A *network* is a couple $\mathcal{G}= (V, E)$ where V is the set of nodes composing $\mathcal{G}$ and E corresponds to the set of *edges* connecting some pairs of nodes in V,
- An oriented edge is called arc. An arc from X_i to X_j is denoted by $X_i \rightarrow X_j$,
- A *direct network* is a network in which all edges in E are oriented,
- In an arc $X_i \rightarrow X_j$, the node X_i is the *parent* of X_j and the node X_j is the *child* of X_i,
- The set of parents of a node X_i is denoted by $U_i = \{U_1, U_2, ..., U_m\}$ where m is the number of parents of per X_i,
- u_i, u_{ij} denote, respectively, possible instances of U_i and U_{ij},
- A *root* is a node with no parents,
- A *leaf* is a node with no children,
- Two nodes linked by an edge (resp. arc) are said to be *adjacent*,
- A *path* in a directed network is a sequence of nodes from one node to another using the arcs,
- A *cycle* is a path visiting each node once such that the first and the last node are the same,
- A *loop* is an *undirected* cycle,
- A *DAG* Directed Acyclic Graph is an oriented network without cycles,
- A *singly connected DAG or polytree* is a DAG which contains no loops,
- A *multiply connected DAG* is a DAG which can contain loops.

1.3 Possibility theory

The probability theory, which dates from the 17th century, is a classical theory that enables representing and quantifying uncertain information. It has been involved in several real world areas, for instance management, economy, industry, etc. However, such theory, that does not consider the situation of *total ignorance*, can be only used when the expert provides precise numerical values. This situation is not always feasible, which has motivated the development of alternative uncertainty frameworks.

Several non classical theories of uncertainty have been proposed in order to deal with uncertain and imprecise data such as fuzzy sets theory [84], spohn's ordinal conditional functions [78, 79], possibility theory [38], etc. We are, in particular, interested in possibility theory introduced at first by Zadeh [85] and then developed by Dubois and Prade [38]. It offers a flexible tool for representing uncertain information. In what follows, we present possibility theory concepts.

1.3.1 Possibility distribution

The basic building block in the possibility theory is the concept of *possibility distribution* π, which is a mapping from the universe of discourse Ω to the unit interval $[0, 1]$. This latter, called *possibilistic scale*, encodes our knowledge on the real world. Contrary to the standard probability theory, the possibilistic scale could be interpreted in twofold: a *numerical* interpretation when values have a real sense and an *ordinal* one when values only reflect a total pre-order between the different states of the world.

The degree $\pi(\omega)$ represents the compatibility of ω with available pieces of information. By convention, $\pi(\omega) = 1$ means that ω is totally possible, and $\pi(\omega) = 0$ means that ω is an impossible state. If $\pi(\omega) > \pi(\omega')$, this means that ω is preferred to ω'.

Possibility theory is driven by the principle of *minimal specificity*. It states that a possibility distribution π is more *specific* than π' if and only if $\forall \omega \in \Omega$, $\pi(\omega) \leq \pi'(\omega)$ [83]. In other terms, π is more informative than π'. In the possibility theory framework, there are two extreme cases, namely:

- *Complete knowledge*: $\exists \omega_0$, $\pi(\omega_0) = 1$ and $\pi(\omega) = 0$ $\forall \omega \neq \omega_0$.
- *Total ignorance*: $\forall \omega \in \Omega, \pi(\omega) = 1$.

A possibility distribution π is said to be *normalized* if there exists at least one totally possible state. Formally:

$$\exists\ \omega \in \Omega, \pi(\omega) = 1 \tag{1.1}$$

Example 1.1. *Let us consider that we submit a scientific paper to an international conference. After reviewing, the paper can be rejected or accepted for either a conference or a poster. Then, the universe of discourse related to the submitted paper after reviewing can be defined as follows:* $\Omega = \{accepted\_conference, accepted\_poster, rejected\}$.

Assuming that a reviewer gives its judgment denoted by Jd *in the form of a possibility distribution defined as follows:*
$\pi(Jd = accepted\_conference) = 1$,
$\pi(Jd = accepted\_poster) = 0.5$,
$\pi(Jd = rejected) = 0.1$.

The possibility distribution given by the reviewer is normalized since $max(1, 0.5, 0.1) = 1$.

Given a joint possibility distribution π on Ω, we can derive marginal distributions relative to subsets of variables using the *maximum* operator i.e., $\forall X \subseteq V, \forall x \in D_X$, we have:

$$\pi(x) = \max_{\omega \in \Omega} \{\pi(\omega) : \omega[X] = x\} \tag{1.2}$$

Given n joint possibility distributions π_1,...,π_n, on Ω_1,...,Ω_n, then the joint possibility distribution on $\Omega_1 \times ... \times \Omega_n$ can be derived by combining them. In [13, 40], the combination of possibility distributions has been defined in several ways. We are, in particular, interested in two forms of combination depending on the interpretation of the possibilistic scale, i.e., in an ordinal setting, we use the **minimum** operator to combine different distributions. However, in a numerical setting, the **product** operator should be used.

1.3.2 Possibility and necessity measures

The probability theory uses only the probability measure P. The probability of the complement $\neg\phi$ of an event ϕ is fully determined from the probability of ϕ, i.e., $P(\neg\phi) = 1 - P(\phi)$. This means that if ϕ *is not probable*, then $\neg\phi$ is *necessarily probable*. This is not the case in possibility theory since if we know that *"it is not possible that* ϕ *is true"*, this does not imply that *"*$\neg\phi$ *is possible"* but it leads to a stronger conclusion i.e., *"it is necessary that* $\neg\phi$*"*. Conversely, *"it is possible that* ϕ *is true"* does not entail anything about the possibility nor the impossibility of $\neg\phi$. Thus, possibility theory discriminates between the concepts of *possibility* (plausibility) and *necessity* (certainty) of an event. Let us now define these two dual measures:

Possibility measure

Given a possibility distribution π, we can define a mapping grading the *possibility measure* of any subset $\phi \subseteq \Omega$ by:

$$\Pi(\phi) = \max_{\omega \in \phi} \ \pi(\omega). \tag{1.3}$$

$\Pi(\phi)$ is called the possibility degree of ϕ. It evaluates at which level ϕ is *consistent* with our knowledge represented by π. For instance, $\Pi(\phi) = 1$ makes this event possible but does not exclude $\neg\phi$. However, we conclude that only the event ϕ can be realized if $\Pi(\neg\phi) = 0$.

Table 1.1 gives main properties of possibility measures.

$\Pi(\phi) = 1$ and $\Pi(\neg\phi) = 0$	ϕ is certainly true
$\Pi(\phi) = 1$ and $\Pi(\neg\phi) \in]0, 1[$	ϕ is somewhat certain
$\Pi(\phi) = 1$ and $\Pi(\neg\phi) = 1$	total ignorance (ϕ is unknown)
$\Pi(\phi) > \Pi(\psi)$	ϕ is a priori more plausible than ψ
$max(\Pi(\phi), \Pi(\neg\phi)) = 1$	ϕ and $\neg\phi$ cannot be both impossible
$\Pi(\phi \vee \psi) = max(\Pi(\phi), \Pi(\psi))$	decomposability axiom (disjunction axiom)
$\Pi(\phi \wedge \psi) \leq min(\Pi(\phi), \Pi(\psi))$	conjunction axiom

Table 1.1: Possibility measure Π (case of normalized possibility distributions)

Example 1.2. *Let us consider Example 1.1 such that the review process of the submitted paper is double blind. Then, if we want to know the possibility degree to have an accepted paper. We can say that it is fully possible, i.e., its possibility degree is equal to 1 since we have not yet received reviews and we have no information that contradicts paper's acceptance.*

Necessity measure

The dual of the possibility measure is the **necessity measure** defined by $\forall \phi \subseteq \Omega$:

$$N(\phi) = 1 - \Pi(\neg\phi) = \min_{\omega \notin \phi}(1 - \pi(\omega)) \tag{1.4}$$

$N(\phi)$ is called the necessity degree of ϕ. It corresponds to the certainty degree associated with ϕ. In other terms, N evaluates at which level ϕ is **certainly** implied by our knowledge represented by π. Let us illustrate the relation between N and Π. If $N(\phi) > 0$, this implies that $\Pi(\phi) = 1$. This means that ϕ is completely possible before being somewhat certain. This property ensures the following inequality $N(\phi) \leq \Pi(\phi)$.

Main properties of necessity measures are summarized in Table 1.2.

$N(\phi) = 1$ and $N(\neg\phi) = 0$	ϕ is certain
$N(\phi) \in]0, 1[$ and $N(\neg\phi) = 0$	ϕ is somewhat certain
$N(\phi) = 0$ and $N(\neg\phi) = 0$	total ignorance (ϕ is unknown)
$min(N(\phi), N(\neg\phi)) = 0$	the unique relation existing between $N(\phi)$ and $N(\neg\phi)$
$N(\phi \wedge \psi) = min(N(\phi), N(\psi))$	conjunction axiom

Table 1.2: Necessity measure N (case of normalized possibility distributions)

Example 1.3. *Assuming that we received all reviews such that the overall rate is equal to 9 of 10. Then, if my friend asks me: Do you think the paper will be rejected? I will say No since it is impossible to reject a paper having a very high rate. This is equivalent to say that it is necessary (certain) that the paper is accepted.*

1.3.3 Quantitative and qualitative settings

There are two settings in the possibility theory framework, namely a *quantitative* setting when we deal with a numerical interpretation of the possibilistic scale and a *qualitative* setting when the ordinal interpretation is considered.

In the quantitative setting of possibility theory, possibility degrees should take significant and precise values in the unit interval $[0, 1]$. Interestingly enough, the value of each possibility degree should be escorted by a justification. However, in most situations, experts cannot provide exact numerical values of possibility degrees. Indeed, it is generally easier and natural for experts to unveil that one situation is more plausible than another instead of associating a particular numerical possibility degree for each situation as probabilities, possibilities in the quantitative setting, kappa functions etc. The possibility theory framework offers the flexibility for experts by modeling uncertainty in a qualitative way.

The basic idea of the qualitative representation is to equip the referential Ω with a *total pre-order* denoted by $\geq_\pi$, instead of the unit interval $[0, 1]$. $\geq_\pi$ corresponds to a *plausibility relation* on Ω which enables us to express that some situations are more plausible than others. Three cases can be identified:

- $\omega =_\pi \omega'$: ω is as plausible as ω',
- $\omega >_\pi \omega'$: ω is more plausible than ω',
- $\omega <_\pi \omega'$: ω is less plausible than ω'.

Example 1.4. *The plausibility relation relative to the possibility distribution given by the reviewer of Example 1.1 is as follows:*
$Jd = accepted\_presentation >_\pi Jd = accepted\_poster >_\pi Jd = rejected.$

We give in what follows some definitions of plausibility relations:

- *Most plausible states*: Given $\varphi = \{\omega_1, .., \omega_n\} \subseteq \Omega$, the most plausible state(s) in the set φ is defined by $max(\varphi)$ s.t.

$$max(\varphi) = \{\omega_i : \omega_i \in \varphi, \nexists\, \omega_j \in \varphi \text{ s.t. } \omega_j >_\pi \omega_i\} \tag{1.5}$$

- *Least plausible states*: Given $\varphi = \{\omega_1, .., \omega_n\} \subseteq \Omega$, the least plausible state(s) in the set φ is defined by $min(\varphi)$ s.t.

$$min(\varphi) = \{\omega_i : \omega_i \in \varphi, \nexists\, \omega_j \in \varphi \text{ s.t. } \omega_i >_\pi \omega_j\}. \tag{1.6}$$

In the qualitative setting, the qualitative possibility distribution, denoted by π_Q, is equipped by a finite and totally ordered scale denoted by $\mathfrak{L} = \{a_0 = 1, a_1, ..., a_n, a_{n+1} = 0\}$ such that $a_0 > a_1 > ... > a_{n+1}$. Such distribution is called qualitative possibility distribution. It is a function that associates for each interpretation $\omega \in \Omega$ an element from $\mathfrak{L}$. Therefore, some states are considered more plausible than others without mentioning any numerical value. It is important to point out that even in a qualitative setting, the possibilistic scale can be numerical. For instance, in the following scale $\mathfrak{L} = \{0, 0.1, 0.2, 0.3, ..., 1\}$, only the order relation between values is significant, and not their real values.

Example 1.5. *Let us continue with Example 1.1. The possibility distribution can be represented qualitatively by the reviewer as follows:*
$\pi_Q(Jd = accepted\_presentation) = a_0,$
$\pi_Q(Jd = accepted\_poster) = a_5,$
$\pi_Q(Jd = rejected) = a_9.$

From π_Q, we can deduce that $Jd = accepted\_presentation$ is more plausible than $Jd = accepted\_poster$, which in its turn more plausible than $Jd = rejected$ since $a_0 = 1 > a_5 = 0.5 > a_9 = 0.1$.

1.3.4 Possibilistic conditioning

Conditioning consists in revising our initial knowledge, encoded by a possibility distribution π, by the arrival of a new certain piece of information $\phi \subseteq \Omega$. In a possibilistic framework, conditioning differs depending on whether the available information is qualitative or quantitative. A panoply of definitions of possibilistic conditioning was proposed [36, 39, 53]. In what follows, we focus on Hisdal's [53], Dubois and Prade's [39] and Dempster's [75] definitions.

Product-based conditioning

The product-based conditioning, also known as Dempster's rule of conditioning [75] is defined by:

$$\Pi(\phi \wedge \psi) = \Pi(\phi|\psi).\Pi(\psi) \tag{1.7}$$

Thus, the impact of an event ψ on our knowledge associated to ϕ is computed as follows:

$$\Pi(\phi \mid_p \psi) = \frac{\Pi(\phi \wedge \psi)}{\Pi(\psi)} \tag{1.8}$$

The impact of ψ on each interpretation $\omega \in \Omega$ is defined by:

$$\Pi(\omega \mid_p \psi) = \begin{cases} \frac{\pi(\omega)}{\Pi(\psi)} & \text{if} \quad \omega \models \psi \\ 0 & \text{otherwise} \end{cases} \tag{1.9}$$

Conditioning using the necessity measure is given by: $N(\phi \mid \psi) = 1 - \Pi(\neg\phi \mid \psi)$.

Min-based conditioning

In a qualitative framework, min-based conditioning is defined by the qualitative counterpart of the Bayesian rule defined by [39, 53]:

$$\Pi(\phi \wedge \psi) = min(\Pi(\phi|\psi), \Pi(\psi)) \tag{1.10}$$

The well known definition of *min-based conditioning* using the minimum specificity principle is expressed by:

$$\Pi(\phi \mid_m \psi) = \begin{cases} \Pi(\phi \wedge \psi) & \text{if} \quad \Pi(\phi \wedge \psi) < \Pi(\psi) \\ 1 & \text{if} \quad \Pi(\phi \wedge \psi) = \Pi(\psi) \end{cases} \tag{1.11}$$

When $\phi \wedge \psi$ is inconsistent, then $\Pi(\phi \wedge \psi) = 0$. If we want to compute the impact of ψ on each $\omega \in \Omega$, then min-based conditioning is defined by:

$$\Pi(\omega \mid_m \psi) = \begin{cases} \pi(\omega) & \text{if} \quad \pi(\omega) < \Pi(\psi) \text{ and } \omega \models \psi \\ 1 & \text{if} \quad \pi(\omega) = \Pi(\psi) \text{ and } \omega \models \psi \\ 0 & \text{otherwise} \end{cases} \tag{1.12}$$

Conditioning using the necessity measure is given by: $N(\phi \mid \psi) = 1 - \Pi(\neg\phi \mid \psi)$.

rate	**quality**	$\pi(\mathbf{rate}, \mathbf{quality})$	$\pi(\mathbf{rate}, \mathbf{quality}\|_{\mathbf{m}}\psi)$	$\pi(\mathbf{rate}, \mathbf{quality}\|_{\mathbf{p}}\psi)$
high	good	0.9	1	1
high	bad	0.2	0.2	$\frac{0.2}{0.9} = 0.22$
low	good	0.6	0	0
low	bad	0.7	0	0

Table 1.3: Joint possibility distribution after conditioning

Example 1.6. *Let us consider two variables relative to the quality of the paper and its overall rate such that* $quality = \{good, bad\}$ *and* $rate = \{high, low\}$. *The joint distribution is given in Column 3 of Table 1.3.*

Considering now that we receive a fully certain piece of information indicating that the overall rate is high. Then, $\psi = \{high \wedge good, high \wedge bad\}$ *and* $\Pi(\psi) = max(0.9, 0.2) = 0.9$ *using Equation (1.2). Using Equation (1.9) (resp. (1.12)), the qualitative (resp. quantitative) possibility distribution of Table 1.3 will be transformed into a new distribution presented in Column 4 (resp. 5) of Table 1.3.*

1.4 Possibilistic knowledge bases

The possibility theory framework offers different formats for representing knowledge either in a logical manner in terms of a possibilistic knowledge base or in a graphical manner using a possibilistic directed acyclic graph (DAG). This section details the logical-based representation.

Possibilistic logic [37, 59] can be viewed as an extension of classical logic where we assign to each formula a weight that reflects the degree of certainty with respect to other formulas. It provides a simple format that turns to be useful for handling qualitative uncertainty. The concept of a *possibilistic knowledge base*, which is a set of weighted formulas, is a well-used and developed compact logical-based representation of a possibility distribution. Formally:

$$\Sigma = \{(\alpha_i, a_i), i = 1, \ldots, k, a_i \neq 0\} \tag{1.13}$$

where k denotes the number of formulas in Σ, α_i is a propositional formula, and a_i its weight belonging to $]0, 1]$. Each possibilistic logic formula (α_i, a_i) expresses that α_i is certain to at least the level a_i, or more formally by $N(\alpha_i) \geq a_i$, where N is the necessity measure associated to α_i. The higher is the degree associated with a formula the more certain it is. Formulas with a necessity degree equal to 0 are not explicitly represented in the possibilistic

knowledge base. In other terms, only beliefs which are somewhat accepted are explicitly represented.

Definition 1.1. *Let Σ be a possibilistic knowledge base and $\alpha \in [0,1]$. We call the α-cut (resp. strict α-cut) of Σ, denoted by $\Sigma_{\geq\alpha}$ (resp. $\Sigma_{>\alpha}$) the set of classical formulas in Σ having a certainty degree at least equal to (resp. strictly greater than) α.*

The inconsistency degree of a possibilistic knowledge base Σ, denoted by $Inc(\Sigma)$ is defined by:

$$Inc(\Sigma) = max\{a_i : \Sigma_{\geq a_i} \models \bot\} \tag{1.14}$$

where $\Sigma_{\geq a_i}$ corresponds to the set of possibilistic formulas having a weight greater or equal than a_i. $Inc(\Sigma) = 0$ means that Σ is consistent. Lang [59] proposed an algorithm to compute the inconsistency degree of Σ with a complexity equal to $log_2\ n\ SAT$ where n is the number of different degrees involved in Σ and SAT is the propositional satisfiability test.

The inconsistency degree can be also computed using the necessity measure. In fact, $Inc(\Sigma)$ corresponds to the smaller necessity degree of the contradiction $\bot$ computed from the possibilistic knowledge base Σ.

Example 1.7. *Let us consider the following possibilistic knowledge base $\Sigma = \{(a, 0.5), (b, 0.3), (\neg a \vee \neg b, 0.2)\}$. The inconsistency degree of this base is $Inc(\Sigma) = 0.2$.*

From a possibilistic knowledge base, a syntactic possibilistic entailment has been defined as follows:

Definition 1.2. *Let Σ be a consistent possibilistic knowledge base and (α_i, a_i) a possibilistic formula, α_i is entailed from Σ to degree a_i denoted by $\Sigma \models (\alpha_i, a_i)$, iff $\Sigma_{\geq a_i} \cup \{\neg\alpha_i\}$ is inconsistent.*

When Σ is inconsistent, then (α_i, a_i) is a non-trivial consequence of Σ if and only if:
(i) $Inc(\Sigma \cup \{(\neg\alpha_i, 1)\}) > Inc(\Sigma)$,
(ii) $Inc(\Sigma \cup \{(\neg\alpha_i, 1)\}) \geq a_i$.

Possibilistic knowledge bases can be considered as compact representations of possibility distributions. Indeed, each possibilistic knowledge base induces a unique possibility distribution [37] such that $\forall\ \omega \in \Omega$,

$$\pi_\Sigma(\omega) = \begin{cases} 1 & \text{if } \forall (\alpha_i, a_i) \in \Sigma,\ \omega \models \alpha_i \\ 1 - max\,\{a_i : (\alpha_i, a_i) \in \Sigma \text{ and } \omega \not\models \alpha_i\} & \text{otherwise} \end{cases} \tag{1.15}$$

This means that the possibility degree associated to each interpretation ω corresponds to computing the maximum certainty degree of the formulas that are not satisfied by ω.

The possibility distribution π_Σ associated with a possibilistic knowledge base Σ can be found by combining local possibility distributions associated to each possibilistic formula using the minimum operator. In fact, the possibility distribution associated with a possibilistic formula (α_i, a_i) denoted by $\pi_{(\alpha_i,a_i)}$ is defined as follows: $\forall\omega \in \Omega$:

$$\pi_{(\alpha_i,a_i)}(\omega) = \begin{cases} 1 - a_i & \text{if } \omega \not\models \alpha_i \\ 1 & \text{otherwise} \end{cases} \tag{1.16}$$

Thus, the possibility distribution π_Σ can be viewed as the result of the combination of $\pi_{(\alpha_i,a_i)}$, $\forall(\alpha_i, a_i) \in \Sigma$. Formally, $\forall\omega \in \Omega$,

$$\pi_\Sigma(\omega) = \min_{i=1,\ldots,k} \pi_{(\alpha_i,a_i)}(\omega) = min\{1 - a_i : \omega \not\models \alpha_i\} \tag{1.17}$$

Example 1.8. *Let us consider the same variables used in Example 1.6 and the following possibilistic knowledge base represented by:*
$\Sigma = \{(low \vee good, 0.9), (high \vee bad, 0.5), (high \vee good, 0.2)\}$. *Then, the joint possibility distribution π_Σ associated with Σ is given by Table 1.4.*

rate	**quality**	π_Σ**(rate, quality)**
high	good	1
high	bad	0.1
low	good	0.5
low	bad	0.8

Table 1.4: Joint possibility distribution of Example 1.8

Two knowledge bases Σ and Σ' are said to be equivalent if and only if they have the same associated possibility distributions [12]. Formally, $\forall\omega \in \Omega$,

$$\pi_\Sigma(\omega) = \pi_{\Sigma'}(\omega) \tag{1.18}$$

Subsumption simplifies possibilistic knowledge bases by removing (or adding) some formulas without lose of information as follows:

Definition 1.3. *Let (α_i, a_i) be a possibilistic formula in Σ. (α_i, a_i) is said to be subsumed by Σ if Σ and $\Sigma\backslash(\alpha_i, a_i)$ are two equivalent possibilistic knowledge bases.*

Subsumed formulas are redundant formulas that can be deleted or added to a possibilistic knowledge base without changing its possibility distribution.

Example 1.9. *Let us consider the following possibilistic knowledge base* $\Sigma = \{(a, 0.5), (\neg a \vee b, 0.4), (\neg a \vee \neg b, 0.3), (a \vee \neg b, 0.4)\}$. *The possibilistic formula* $(a \vee \neg b, 0.4)$ *is subsumed by* Σ *since* Σ *and* $\Sigma \backslash \{(a \vee \neg b, 0.4)\}$ *are equivalent and yield to the same possibility distributions.*

There are two kinds of possibilistic inference:

- *Deduction of formulas*: A formula α is a logical consequence of a possibilistic knowledge base Σ, denoted by $\Sigma \models \alpha$ if and only if $N(\alpha) > 0$ where N is the necessity degree of α computed from π_Σ.
- *Deduction of formulas with certainty degrees*: A possibilistic formula (α_i, a_i) is a logical consequence of a possibilistic knowledge base Σ, denoted by $\Sigma \models (\alpha_i, a_i)$ if and only if $N(\alpha_i) \geq a_i > 0$.

1.5 Possibilistic networks

Bayesian networks are probabilistic graphical models that are used for modelling domains incorporating uncertainty [67]. They were successfully applied in several areas [80, 81]. The power of these graphical models is not only based on their efficient reasoning with several variables, but also their ability to help humans in understanding better the domain. This is mainly due to their comprehensible representation by using dependencies between variables.

Bayesian networks can be only used when the expert is able to provide precise numerical values, which is not always possible in practice. For this reason, in the case of non-numerical data, one can resort to possibilistic networks, which are the possibilistic counterpart of Bayesian networks in the possibility theory framework.

In the possibility theory framework, the two interpretations of the possibilistic scale offer two definitions of conditioning, namely a min-based conditioning using the minimum operator and a product-based conditioning using the product operator. Due to the existence of two definitions of possibilistic conditioning, there are two ways to define the counterpart of Bayesian networks: *product-based possibilistic networks* which are very close to the probabilistic ones and *min-based possibilistic networks* characterized by a different behavior when comparing them to probabilistic ones [46]. There are also possibilistic causal networks in which edges encode not only dependencies between variables but also direct causal relationships [16].

Most of possibilistic works focus on the inference topic while assuming that the model exists. In this section, we present possibilistic networks. Before, we give a recall on Bayesian networks. Possibilistic causal networks will be detailed in Chapter 5.

1.5.1 Recall on Bayesian networks

A Bayesian network [67], denoted by BN, over a set of variables V has two components, namely:
- A *graphical component* composed of a directed acyclic graph (DAG) $\mathcal{G}=(V, E)$ where V denotes a set of *nodes* representing variables and E a set of *edges* reflecting the independence relations relative to the application area.
- A *numerical component* consisting in a quantification of different links in the DAG by a **conditional** probability distribution of each node X_i in the context of its parents (U_i). Such conditional probabilities should respect the following normalization constraints for each variable X_i:

- if $U_i = \emptyset$ (i.e., X_i is a root), then the a priori probability relative to X_i should satisfy:

$$\sum_{x_i} P(x_i) = 1$$

- if $U_i \neq \emptyset$, then the relative conditional probability relative to X_i in the context of its parents U_i should satisfy:

$$\sum_{x_i} P(x_i \mid U_i) = 1$$

Given a Bayesian network, the global joint probability distribution over the set $V = \{X_1, ..., X_N\}$ can be expressed as a product of the N initial conditional probabilities via the following probabilistic *chain rule*:

$$p(X_1, ..., X_N) = \prod_{i=1..N} P(X_i \mid U_i) \tag{1.19}$$

Example 1.10. *Let us consider the Bayesian network BN, depicted by Figure 1.1, having two binary variables A and B. The joint probability distribution of BN is given by Table 1.5.*

Inference in Bayesian networks is considered the primary task of graphical models. It determines how the realization of specific values of some variables $E \subseteq V$, called *evidence* and denoted by e, affects remaining variables. More

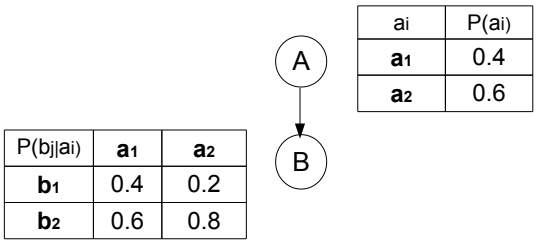


Figure 1.1: A Bayesian network

A	B	$\mathbf{P(A)}$	$\mathbf{P(B\|A)}$	$\mathbf{p}$
a_1	b_1	0.4	0.4	0.16
a_1	b_2	0.4	0.6	0.24
a_2	b_1	0.6	0.2	0.12
a_2	b_2	0.6	0.8	0.48

Table 1.5: Joint distribution of BN

explicitly, we compute for any variable of interest $X_i \in V$ the probability $P(x_i \mid e), \forall x_i \in D_{X_i}$.

Considering that a joint probability distribution is available, then inference can be performed using marginalization. Unfortunately, this is not realistic even if the Bayesian network contains a small number of variables. A key solution that avoids computing the whole joint distribution relative to a Bayesian network is to perform local computations using *probabilistic inference* algorithms. The inference task in Bayesian network is known to be **NP-complete** [25] except for singly connected networks.

The fundamental exact probabilistic inference algorithm was proposed by Kim and Pearl [58] and Pearl [66, 67] for singly connected networks. It consists in combining information deriving from parents and children of the variable of interest via a message passing mechanism. When the focus is on multiply connected networks, this algorithm does not give the correct beliefs. For multiply connected networks, the junction tree method introduced by Lauritzen and Spiegelhalter [60] and refined by [56] is considered as the standard probabilistic inference approach. Its main idea consists in transforming the initial Bayesian network into a junction tree which is a tree of variables clusters. Messages are transmitted between clusters allowing the computation of marginal distributions in two passes. More details on this algorithm will be given in its possibilistic adaptation (see subsection 1.5.3).

1.5.2 Representation of possibilistic networks

As in Bayesian networks, a possibilistic network over a set of variables V is characterized by:
- A *graphical component* composed of a Directed Acyclic Graph (DAG) $\mathcal{G}$.
- A *numerical component* consisting in a quantification of each node in the context of its parents using a conditional possibility distribution (denoted by $C\Pi T_i$). The set of all $C\Pi T_i$ is denoted by $C\Pi T$. Such conditional distributions should respect the following normalization constraints for each variable X_i using the marginalization operator, namely: the maximum operator.

- if $U_i = \emptyset$ (i.e., X_i is a root), then the a priori possibility relative to X_i should satisfy:
$$\max_{x_i} \Pi(x_i) = 1, \forall x_i \in D_{X_i}$$
- if $U_i \neq \emptyset$, then the conditional distribution of X_i in the context of its parents should satisfy:
$$\max_{x_i} \Pi(x_i \mid U_i) = 1, \forall x_i \in D_{X_i}, U_i \in D_{U_i}$$

If X_i is a binary variable, then $max(\Pi(x_i \mid U_i), \Pi(\neg x_i \mid U_i)) = 1$.

Product-based possibilistic networks

A *product-based possibilistic network* over a set of variables V, denoted by ΠG_*, is a possibilistic network appropriate for a numerical interpretation of the possibilistic scale $[0, 1]$. In such networks, conditionals are defined using product-based conditioning expressed by Equation (1.9).

The joint distribution relative to a product-based possibilistic network, denoted be π_*, can be computed in the same manner than Bayesian networks via the following **product-based** chain rule:

Definition 1.4. *(**Product-based chain rule**) Given a product-based possibilistic network ΠG_*, the global joint possibility distribution over the variable set $V = \{X_1, X_2, ..., X_N\}$ can be expressed as the product of the N initial a priori and conditional possibilities via the following product-based chain rule:*

$$\pi_*(X_1, ..., X_N) = \prod_{i=1..N} \Pi(X_i \mid U_i) \tag{1.20}$$

where $\prod$ is the product operator.

Min-based possibilistic networks

A *min-based possibilistic network* over a set of variables V, denoted by ΠG_{min}, is a possibilistic network appropriate for an ordinal interpretation of the possibilistic scale. Conditionals are defined using min-based conditioning of Equation (1.11).

The joint distribution relative to a min-based possibilistic network, denoted by π_{min}, can be computed via the following **min-based** chain rule:

Definition 1.5. *(**min-based chain rule**) Given a min-based possibilistic network ΠG_{min}, the global joint possibility distribution over the variable set $V = \{X_1, X_2, ..., X_N\}$ can be expressed as the minimum of the N initial a priori and conditional possibilities via the following min-based chain rule:*

$$\pi_{min}(X_1, .., X_N) = \min_{i=1..N} \Pi(X_i \mid U_i) \tag{1.21}$$

In [12], authors have provided a transition of min-based possibilistic networks into standard possibilistic logic bases.

Example 1.11. *Let us consider the possibilistic network, depicted by Figure 1.2, containing two binary variables A and B. The joint possibility distributions of ΠG_* and ΠG_{min} are represented in Table 1.6.*

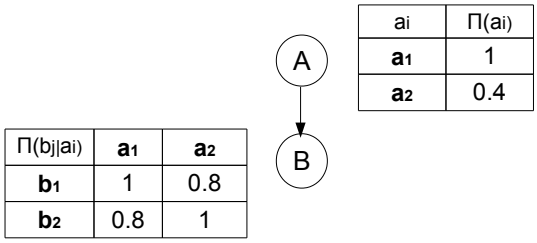


Figure 1.2: A small possibilistic network

A	B	$\mathbf{\Pi(A)}$	$\mathbf{\Pi(B\|A)}$	π_*	$\pi_{\mathbf{min}}$
a_1	b_1	1	1	1	1
a_1	b_2	1	0.8	0.8	0.8
a_2	b_1	0.4	0.8	0.32	0.4
a_2	b_2	0.4	1	0.4	0.4

Table 1.6: Joint distributions of ΠG_* and ΠG_{min}

1.5.3 Inference in possibilistic networks

One of the most interesting treatments that can be applied for possibilistic networks is to evaluate the impact of a certain event on the remaining variables. Such process can be achieved using inference algorithms consisting on computing a-posteriori possibility distributions of each variable X_i given an evidence e. Possibilistic inference algorithms proposed in literature are mainly a direct adaptation of exact methods [20, 46]. In fact, the proposed algorithms for product-based possibilistic networks are very similar to probabilistic algorithms since they share the same operator. This is not the case when using the minimum operator. Indeed, this operator has special properties, such as *idempotence*, which can be used to avoid direct adaptations. In what follows, we present the standard inference algorithm in junction trees associated to min-based possibilistic networks, that will be used in the experimental study.

The principle of this inference method is similar to the probabilistic method in junction trees presented in [56]. Indeed, it is based on the transformation of the initial DAG into a *junction tree*, denoted by JT, which will be used during the inference process. Once the building of the junction tree is achieved, the inference process can start through three stages. The first step of *initialization* allows to quantify the junction tree using initial distributions and incorporate evidence into variables. The second step of *global inference* allows to ensure global coherence of the junction tree through a collect phase and and a distribute phase of messages between clusters. In other words, this step provides exact marginals. Finally, the last step of *response to queries* provides marginals relative to on different variables.

Step S_1: Building the junction tree

Principal steps to construct a junction tree given a DAG can be summarized as follows [55]:

– *Moralization of the initial network*: allows to create a non oriented network by dropping the direction of existing edges and adding undirected edges between the parents of each variable. The moral network is denoted by MG.

– *Triangulation of the moral network*: This step aims to identify sets of variables that can be grouped into clusters, denoted by C_i. Note that it is important to find an optimal triangulation that minimizes the size of clusters to allow local computations. This problem is known as NP-complete [25].

– *Building the junction tree*: The triangulated network is transformed into a junction tree where each node represent a cluster of variables and

each edge is labeled with a separator, denoted by S_{ij}, corresponding to the intersection of its adjacent clusters.

Example 1.12. *Let us consider the possibilistic network of Figure 1.3. Figure 1.4 depicts the junction tree associated to the DAG of Figure 1.3 composed of two clusters:* $C_1 = \{ABC\}$ *and* $C_2 = \{BCD\}$ *and their separator* $S_{12} = \{BC\}$.

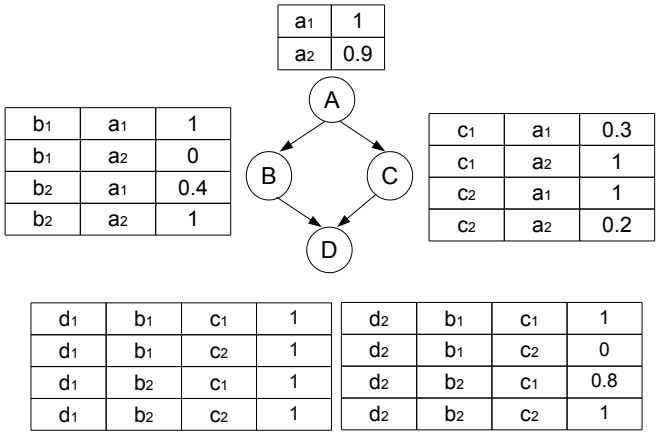


Figure 1.3: A possibilistic network

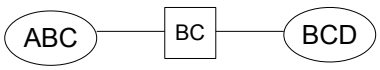


Figure 1.4: The junction tree associated to the DAG of Figure 1.3

Step S_2: Initialization

Once the junction tree is constructed, this step transforms the initial conditional possibility distributions into local joint distributions attached to clusters and separators. More specifically, for each cluster C_i (resp. separator S_{ij}) of MG, we assign a *potential* $\pi^t_{C_i}$ (resp. $\pi^t_{S_{ij}}$) where t is relative to the inference step. In particular, $t = I$ (resp. $t = C$) corresponds to the initialization step (resp. global coherence).

These potentials allow to associate a joint possibility distribution to the junction tree, denoted by π^t_{JT} and defined by:

$$\pi^t_{JT}(X_1, \ldots, X_N) = min_{\ i=1\ldots m}\ \ \pi^t_{C_i} \tag{1.22}$$

where m is the number of clusters in JT.

The initialization step can be summarized as follows:

– For each cluster C_i, assign a uniform distribution: $\pi^I_{C_i} \leftarrow 1$.

– For each separator S_{ij}, assign a uniform distribution: $\pi^I_{S_{ij}} \leftarrow 1$.

– For each variable X_i, choose a cluster containing $X_i \cup U_i$ and update its local possibility distributions: $\pi^I_{C_i} \leftarrow min(\pi^I_{C_i}, \Pi(X_i|U_i))$.

Example 1.13. *Let us re-consider the possibilistic network of Example 1.12. After the initialization step of the junction tree of Figure 1.4, local possibility distributions of clusters and the separator are given by:*
- $\pi_{C_1}(ABC) = \Pi(A).\Pi(B|A).\Pi(C|A)$,
- $\pi_{C_2}(BCD) = \Pi(D|BC)$,
- $\pi_{S_{12}} = 1$.

Step S_3: Global inference

Before presenting the global inference step, we introduce the concept of global coherence in junction trees.

Definition 1.6. *Let C_i and C_j two adjacent clusters in a junction tree $\mathcal{JT}$ and S_{ij} its separator. The edge between C_i and C_j is said to be stable or coherent if:*

$$\max_{C_i \setminus S_{ij}}\ \pi^t_{C_i} = \pi^t_{S_{ij}} = \max_{C_j \setminus S_{ij}}\ \pi^t_{C_j} \tag{1.23}$$

where $\max_{C_i \setminus S_{ij}}\ \pi^t_{C_i}$ is the marginal distribution of S_{ij} defined from $\pi^t_{C_i}$. If all edges are coherent, then $\mathcal{JT}$ is said to globally coherent.

The global inference is performed by a *message passing* mechanism between clusters until reaching the global consistency of the junction tree. Given a junction tree with m clusters, the global inference algorithm begins by choosing an arbitrary cluster to be the *pivot node* and then performing $2 * (m - 1)$ messages passes, divided into two phases:

- A *collect-evidence* phase in which each cluster passes a message to its adjacent cluster in the pivot direction, beginning with the clusters farthest from the pivot (which correspond to leaves).
- A *distribute-evidence* phase in which each cluster passes messages to its adjacent clusters away from the pivot direction, beginning with the pivot itself. In this phase messages circulate from the pivot until the leaves are reached.

If a cluster C_i sends a message to its adjacent cluster C_j, then the potentials of C_i, C_j and their separator S_{ij} are updated as follows:

1. Save the same potential for C_i
$$\pi_{C_i}^{t+1} \leftarrow \pi_{C_i}^{t}. \tag{1.24}$$
2. Assign a new potential to S_{ij}
$$\pi_{S_{ij}}^{t+1} \leftarrow \max_{C_i \setminus S_{ij}} \pi_{C_i}^{t}. \tag{1.25}$$
3. Assign a new potential to C_j:
$$\pi_{C_j}^{t+1} \leftarrow min(\pi_{C_j}^{t}, \pi_{S_{ij}}^{t+1}). \tag{1.26}$$

When the global coherence of the junction tree is guaranteed, the potential of each cluster corresponds to its local distribution.

Step S_4: Response to queries

The junction tree resulting from the previous step is globally coherent, which means that the potential of each cluster encodes the same local distribution. Thus, computing the marginal relative to each variable $X_i \in V$ can be established by marginalizing the potential of any cluster as follows:

$$\Pi(X_i) = \max_{C_i \setminus X_i} \ \pi_{C_i}^{C} \tag{1.27}$$

Example 1.14. *Considering the possibilistic network of Figure 1.3. Local possibility distributions of clusters (after global inference) are given in Table 1.7. The possibility distribution associated to the junction tree is equivalent to the one of the initial network. For instance,* $\pi_{AJ}(a_1, b_2, c_1, d_2) = 0.3$. *Computing the marginal distribution of D, we should choose a cluster containing this variable (*C_2 *in this case):* $\Pi(D) = max_{BC} \ \pi_{C_2}(DBC)$. *We obtain* $\Pi(d_1) = 1$ *and* $\Pi(d_2) = 0.8$.

A	B	C	$\pi_{C_1}(ABC)$	D	B	C	$\pi_{C_2}(DBC)$
a_1	b_1	c_1	0.3	d_1	b_1	c_1	0.3
a_1	b_1	c_2	1	d_1	b_1	c_2	1
a_1	b_2	c_1	0.3	d_1	b_2	c_1	0.9
a_1	b_2	c_2	0.4	d_1	b_2	c_2	0.4
a_2	b_1	c_1	0	d_2	b_1	c_1	0.3
a_2	b_1	c_2	0	d_2	b_1	c_2	0
a_2	b_2	c_1	0.9	d_2	b_2	c_1	0.8
a_2	b_2	c_2	0.2	d_2	b_2	c_2	0.4

Table 1.7: Local possibility distributions of clusters

Handling the evidence

The inference algorithm in junction trees can be easily extended to take into consideration an evidence e corresponding to a set of instantiated variables, i.e., for each variable $X_i \in V$, it allows to compute $\Pi(X_i \wedge e)$ instead of $\Pi(X_i)$. For this reason, the evidence related to each variable X_i should be encoded using the following distribution Λ_{X_i}:

$$\Lambda_{X_i}(x_i) = \begin{cases} 1 & \text{if } X_i \text{ is not instantiated} \\ 1 & \text{if } X_i \text{ is instantiated for } x_i \\ 0 & \text{if } X_i \text{ is instantiated but not for } x_i \end{cases} \tag{1.28}$$

To handle the evidence e, we should extend the inference procedure by transforming the initialization step so that to incorporate any certain information. Indeed, we should encode the evidence e as a likelihood (using (1.28)), then, we incorporate it into the junction tree by adding these two steps to the initialization procedure:
- For any instantiated variable X_i, encode the observation $X_i = x_i$ as a likelihood Λ_{X_i} using Equation (1.28).
- Identify a cluster C_i containing X_i: $\pi^I_{C_i} \leftarrow min(\pi^I_{C_i}, \Lambda_{X_i})$.

Through global inference and under the assumption that we have an evidence e, the potential of any cluster C_i encodes $\Pi(C_i \wedge e)$. Then, when we marginalize any cluster potential $\pi^C_{C_i}$ into a variable X_i (s.t $X_i \subseteq C_i$) using Equation (1.27), we obtain the possibility measure of X_i and e:

$$\Pi(X_i \wedge e) = \max_{C_i \setminus X_i} \pi^C_{C_i} \tag{1.29}$$

However, our goal is to compute $\Pi(X_i \mid e)$, this value can be easily obtained from $\Pi(X_i \wedge e)$ by applying the definition of min-based conditioning

as follows:

$$\Pi(X_i \mid e) = \begin{cases} \Pi(X_i \wedge e) & \text{if} \Pi(X_i \wedge e) < \Pi(e) = \max\limits_{X_i} \Pi(X_i \wedge e) \\ 1 & \text{otherwise} \end{cases} \quad (1.30)$$

1.6 Conclusion

In this chapter, we have presented two representations of uncertain information in the possibility theory framework, namely the logical-based representation using possibilistic knowledge bases and the graphical-based one using possibilistic networks. These latters, which are the possibilistic counterpart of standard Bayesian networks, can be namely product-based ones in a numerical setting and min-based in an ordinal setting. We have, especially, emphasized on the well known inference problem. Next chapter will focus on an important research topic, namely *knowledge compilation*.

Chapter 2

Knowledge compilation

2.1 Introduction

Knowledge representation consists in storing what an agent knows in a knowledge base, typically using a logical formalism. Such base is then used, through specific querying algorithms, to extract information implicitly stored in it. Problems in logic are known among computer scientists for their high computational complexity, for instance deduction in a logical formalism. *Knowledge compilation* is one of the techniques that has been proposed for addressing such computational difficulties [21].

Knowledge compilation has been acknowledged for a few years as an important research topic [27]. According to this approach, the reasoning process is split into two phases: an off-line compilation phase in which the propositional theory is compiled into some target language and an on-line query-answering phase where the compiled base is used to efficiently answer a set of queries. One of the key aspects of any compilation approach is the target compilation language into which the propositional theory is compiled.

This Chapter is organized as follows: Section 2.2 briefly outlines the basic elements of propositional logic. Section 2.3 introduces knowledge compilation. Section 2.4 is dedicated to the knowledge compilation map of [34]. Section 2.4 is devoted to the most succinct target compilation language, namely *Decomposable Negation Normal Form*.

2.2 Propositional logic

Propositional logic is a knowledge representation and reasoning formalism. It is within the simplest logics but expressive enough for many applications.

It is used to express statements to which we assign the so-called truth values: *true* or *false* and no other possible value. Such truth is vocalized in light of a possible world. In this section, we succinctly present syntactic and semantic aspects that are necessary for understanding knowledge Compilation.

2.2.1 Syntax

From a *syntactic* point of view, the *propositional variable* represents the central element in propositional logic. Also called proposition or atom, a propositional variable is a boolean variable subject to take two truth values, namely True ($\top$) or False ($\bot$). A *literal* l is either a propositional variable, called a *positive literal*, or its negation, called a *negative literal*.

Let P be a finite set of propositional variables, denoted by tiny letters $a, b, \cdots$. The language of a propositional logic, denoted by $\mathfrak{L}$, is constructed over the set of propositional variables P, boolean constants True ($\top$) and False ($\bot$), logical connectors (negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$), equivalence ($\leftrightarrow$) [1]) and parentheses. Elements of $\mathfrak{L}$ are propositional formulas, also called well-formed formulas. They are formed using a set of propositional variables and can be defined in the following way:

1. Each propositional variable as well as $\top$ and $\bot$ are propositional formulas,
2. If α and β are propositional formulas, then so are $(\neg\alpha)$, $(\alpha \vee \beta)$, $(\alpha \wedge \beta)$, $(\alpha \rightarrow \beta)$, $(\alpha \leftrightarrow \beta)$.
3. Propositional formulas are only obtained using (1) and (2).

In the remaining, we denote by $vars(\alpha)$ the propositional variables relative to the formula α.

Propositional formulas can be represented graphically using Directed Acyclic Graphs (DAG) where every leaf node is labeled by $\top$, $\bot$ or a literal and every internal node is labeled by a connector. Such DAG, called *boolean circuit*, is considered as a compact representation since it allows to reuse redundant sub-formulas rather than re-writing them twice.

Relying on literals, a *clause* is a finite disjunction of literals (in particular the constant $\bot$, when the set of literals is empty) and a *term* is a finite conjunction of literals (in particular the constant $\top$, when the set of literals is empty).

[1] Connectors $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$ have the same priority, while the negation connector (i.e., $\neg$) has the higher priority over all connectors.

A propositional formula α is said to be in a *Conjunctive Normal Form (CNF)* iff α is a *conjunction* of clauses. α is said to be in a *Disjunctive Normal Form (CNF)* iff α is a *disjunction* of terms. Let us illustrate basic definitions of propositional logic.

Example 2.1. *Let* $a, b, c, d \in P$*:*
– $\alpha = (a \rightarrow b) \wedge (c \wedge d)$ *is a propositional formula belonging to* $\mathfrak{L}$*,*
– $vars(\alpha) = \{a, b, c, d\}$*,*
– $a \vee b$ *is a clause and* $\neg a \wedge \neg b$ *is a term.*
– $(a \vee b) \wedge (\neg c \vee d)$ *is a CNF formula,*
– $(a \wedge b) \vee (\neg c \wedge d)$ *is a DNF formula.*

2.2.2 Semantic

From a *semantical* point of view, the *interpretation* represents the central element in propositional logic. Formally, an interpretation is a mapping assigning a truth value to each propositional variable of a formula pertaining to a language $\mathfrak{L}$. Let us consider two propositional formulas α and β, an interpretation I should verify the following:
– $I(\top) = True$,
– $I(\bot) = False$,
– $I(\neg\alpha) = True$ iff $I(\alpha) = False$,
– $I(\alpha \wedge \beta) = True$ iff $I(\alpha) = I(\beta) = True$,
– $I(\alpha \vee \beta) = False$ iff $I(\alpha) = I(\beta) = False$,
– $I(\alpha \Rightarrow \beta) = False$ iff $I(\alpha) = True$ and $I(\beta) = False$,
– $I(\alpha \Leftrightarrow \beta) = True$ iff $I(\alpha) = I(\beta)$.

Table 2.1, called *truth table*, synthesizes truth values of $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \Rightarrow \beta$ and $\alpha \Leftrightarrow \beta$ depending on those of α and β.

$I(\alpha)$	$I(\alpha)$	$I(\neg\alpha)$	$I(\alpha \wedge \beta)$	$I(\alpha \vee \beta)$	$\alpha \Rightarrow \beta$	$\alpha \Leftrightarrow \beta$
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

Table 2.1: Interpretations of $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \Rightarrow \beta$ and $\alpha \Leftrightarrow \beta$

Let α be a propositional formula, then an interpretation I is a model of (or satisfies) α iff $I(\alpha) = True$. and I is a counter-model of (or falsifies) α iff $I(\alpha) = False$. Concepts of validity, satisfiability and unsatisfiability are defined in what follows:

Definition 2.1.
– A propositional formula α is satisfiable iff it admits at least one model.
– A propositional formula α is unsatisfiable iff it does not admit any model.
– A propositional formula α is valid (i.e., tautology) iff it does not admit any counter-model.
– A propositional formula α is invalid iff it admits a counter-model.

Example 2.2. *Let a and b be two propositional variables, then:*
– The formula $a \vee \neg a$ is valid,
– The formula $a \vee b$ is invalid and satisfiable,
– The formula $a \wedge b \wedge \neg b$ is unsatisfiable.

Previously, we focused on the knowledge representation aspect in propositional logic. We switch now to the reasoning aspect which consists on making new formulas from initial formulas. The central question is: *From a certain amount of knowledge, what can we deduce?* and *How can we deduce?* Other important questions follow: *The reasoning principle is it correct? What I have deduced, is it really real?* and *is it Complete? Are we able to deduce everything?*

The principle of logical reasoning is the relation of *logical consequence* between statements, which sets out that a statement follows logically from another statement. Formally, logical deduction in propositional logic can be defined as follows:

Definition 2.2.
Let α and β be two propositional formulas. Then:
– β is a logical consequence of α, denoted by $\alpha \models \beta$ iff each model of α is a model of β.
– α and β are logically equivalent, denoted by $\alpha \equiv \beta$, iff $\alpha \models \beta$ and $\beta \models \alpha$ (i.e., α and β have exactly the same models).

The truth table is a certain way to check the validity of a logical deduction. However, its main drawback resides in the fact that it is not effective in practice since we can enumerate 2^{pv} interpretations to find a model, where pv is the number of propositional variables present in the considered set of formulas. Logical deduction can actually be established without knowing truth values associated to manipulated propositional variables. The fundamental result in automatic proving, knows as *refutation* theorem, states that α is a logical consequence of β (i.e., $\alpha \models \beta$) iff $\alpha \wedge \neg\beta$ is unsatisfiable.

2.3 Principle of knowledge compilation

Knowledge compilation is a common technique for propositional logic knowledge bases. It is a mapping from a given knowledge base (typically in a rep-

resentation language) into a special form of propositional bases, from which queries can be answered efficiently. Such mapping consists in splitting query answering of a particular problem into two phases [21]:

- *Off-line reasoning*: In the first phase, knowledge bases are preprocessed using the so-called *target compilation languages* in order to obtain the data structures the most appropriate for a given application. Such a phase is very expensive but it has to be performed only once.
- *On-line reasoning*: In the second phase, queries are answered efficiently using the output of the first phase.

The key motivation behind knowledge compilation is to shift the extra load of work of the on-line phase to the off-line phase in order to alleviate the treatments needed to get responses to queries. Given an input knowledge base not very often subject to change, it is compiled once in the off-line phase. Accordingly, the computational overhead of compilation is amortized over all on-line queries, as if a student studying for an exam have not slept for a week in order to master the subject it revises and therefore be able to respond to questions, regardless of their difficulties.

The following example illustrates the compilation principle [61]:

Example 2.3. *Let us consider the table of logarithms introduced during the* 17^{th} *century storing pairs* $< x, log_{10}(x) >$ *where* x *is a real number. Such table can be considered as a compiled form since it improves many computations. For instance, assuming that we want to compute the value of* $x = \sqrt[5]{1234}$. *Since* $\sqrt[5]{1234} = (1,234 * 10^3)^{\frac{1}{5}}$, *we get that* $log(\sqrt[5]{1234}) = log((1,234 * 10^3)^{\frac{1}{5}}) = \frac{log_{10}(1,234)+3}{5}$. *To get the value of* $log_{10}(1,234)$, *it is enough to look for the row of the real number* $1,234$, *i.e.,* $< 1,234, 0.09131516 >$ *and compute* $\sqrt[5]{1234} = 0.618263032$. *Finally, the value of* x *is obtained from the logarithm table by looking for* $\sqrt[5]{1234}$. *Hence,* x *is equal to* 4.152054371 *since we find* $< 4.152054371, 0.618263032 >$ *in the table. We can remark that the generation of logarithm table is a tedious task but once obtained, the computational effort spent in the off-line phase is amortized over all on-line computations.*

A problem is said to be *compilable* as defined in [21] if:

1. The data structure generated in the off-line phase should fill a polynomial space with respect to the size of the input knowledge base,
2. The algorithm of the on-line phase should be sound (each query that can be answered using the initial base should be also answered using the compiled base) and complete (only queries that can be answered in the initial base can be answered using the compiled base),

3. The algorithm answering the query should run in a polynomial time in its inputs.

It is prominent now to define a *representation* language and a *target compilation* language. In fact, a representation language is qualified as natural since humans can use it with some ease either by reading or writing. For instance, the *Conjunctive Normal Form (CNF)* language is a popular representation language. While a target compilation language does not need to be suitable for human specification and interpretation, but should be tractable enough to permit a polytime reasoning. Formally, a language is said to be a *target compilation language* if it permits a polytime clausal entailment test.

The standard knowledge compilation methods have focused mostly on target compilation languages which are variations on DNF and CNF formulas, such as *Prime implicates (PI)*. This language has been quite influential in computer science and artificial intelligence. Formally, PI is the subset of CNF in which each clause entailed by the formula is entailed by a clause that appears in the formula; and no clause in the formula is entailed by another. A dual of PI, *Prime Implicants (IP)*, can also be defined as a subset of DNF in which each term entailing the formula entails some term which appears in the formula; and no term in the formula is entailed by another term.

Other subsets of CNF are also interesting for knowledge compilation audience, for instance the set *HORN-CNF* of *Horn CNF formulas*, i.e., conjunctions of clauses containing at most one positive literal. In these standard knowledge compilation methods, it has been considered mostly *clausal entailment* queries, i.e., checking if a clause is a logical consequence of a knowledge base or not. In [34], Darwiche and Marquis have studied a relatively large number of target compilation languages and evaluated them across the following two important dimensions:

- *Degree of tractability*: the class of logical operations (*transformations* and *queries*) they support in a polynomial time,
- *Succinctness* also called *space efficiency*: the size of the smallest formula needed to represent a propositional theory.

2.4 Knowledge compilation map

Darwiche and Marquis [34] proposed a knowledge compilation map to choose the most suitable target compilation language that supports the required set of queries and transformations for a particular application. In this section, we present the target compilation language used for this thesis (i.e., DNNF), its succinctness and the set of polytime operations it supports.

2.4.1 Target compilation languages

Target compilation languages explored in [34] are derived from the *Negation Normal Form (NNF)* language, which is formally defined as the set of propositional formulas constructed from literals, $\top$ and $\bot$ using only conjunctions $\wedge$ and disjunctions $\vee$. Figure 2.1 depicts the inclusion relation between languages where the root is the NNF language and an edge $\mathfrak{L}_1 \rightarrow \mathfrak{L}_2$ denotes that $\mathfrak{L}_1$ is a subset of $\mathfrak{L}_2$.

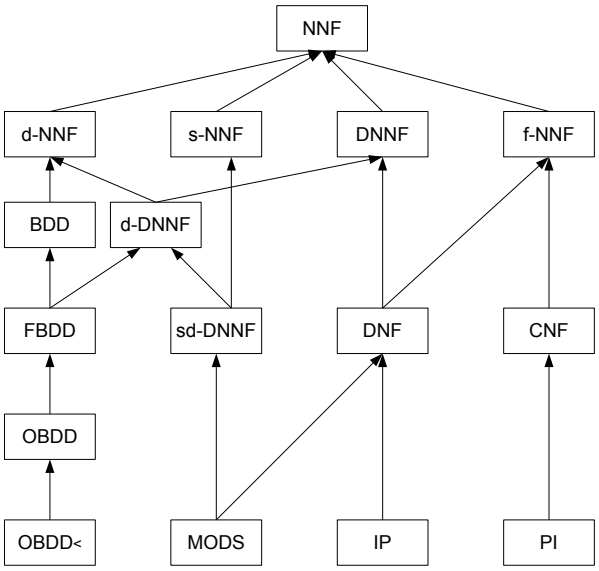


Figure 2.1: Knowledge compilation map [34]

The NNF language is not qualified as a target compilation language (unless P=NP) [65] but many of its subsets are. We are, in particular, interested in a number of these subsets where each subset is obtained by imposing further conditions on NNF, e.g. *decomposability*, *determinism* and *smoothness* defined as follows [27]:

- **Decomposability**: for each conjunction $\alpha \wedge \beta$, α and β do not share variables, i.e., $vars(\alpha) \cap vars(\beta) = \emptyset$.
- **Determinism**: for each disjunction $\alpha \vee \beta$, α is incoherent with β, i.e., $\alpha \wedge \beta \models \bot$.
- **Smoothness**: for each disjunction $\alpha \vee \beta$, α and β share the same variables, i.e., $vars(\alpha) = vars(\beta)$.

The properties of *decomposability*, *determinism* and *smoothness* lead to a number of interesting subsets of NNF defined as follows:

- The DNNF language is the subset of NNFs satisfying decomposability.
- The d-NNF language is the subset of NNFs satisfying determinism.
- The s-NNF language is the subset of NNF satisfying smoothness.
- The language d-DNNF is the subset of NNFs satisfying decomposability and determinism.
- The language sd-DNNF is the subset of NNFs satisfying decomposability, determinism and smoothness.

The following example illustrates NNF and its subsets by imposing decomposability, determinism and smoothness.

Example 2.4. *Figure 2.2 represents an NNF formula. Note that we do not orient edges since the children of each node are below it. This NNF formula is decomposable (i.e., a DNNF) since its and-nodes satisfy the decomposability property. For instance, the marked and is decomposable since its right* $((c \wedge d) \vee (\neg c \wedge \neg d))$ *and left* $(\neg a \wedge b) \vee (\neg b \wedge a))$ *conjuncts do not share variables.*

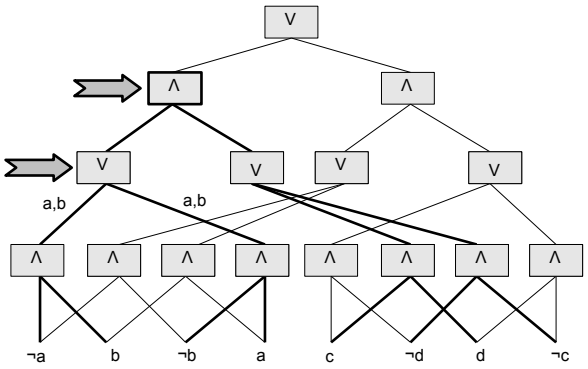


Figure 2.2: An NNF formula

Consider the or-node marked in Figure 2.2, having two children which correspond to the following sub-formulae: $\neg a \wedge b$ *and* $a \wedge \neg b$*. The conjunction of these two sub-formulas is logically contradictory, therefore the or-node is deterministic and so are the other or-nodes in Figure 2.2. In this case, the DNNF is deterministic (i.e., a d-DNNF).*

The same or-node marked in Figure 2.2 has two children, each of which mentions variables a and b, hence this or-node is smooth and so are the other or-nodes in Figure 2.2. In this case, the d-DNNF is smooth (i.e., a sd-DNNF).

2.4.2 Succinctness

Succinctness, also called *space efficiency*, is considered as an important aspect that needs to be considered when choosing a target compilation language for a task. It tells us the compactness of formulas in different languages. Formally [50]:

Definition 2.3. *A language* $\mathfrak{L}_1$ *is said to be at least as succinct as* $\mathfrak{L}_2$, *denoted by* $\mathfrak{L}_1 \leq \mathfrak{L}_2$ *iff there exists a polynomial* p *such that for every formula* α *in* $\mathfrak{L}_2$, *there exists an equivalent formula* β *in* $\mathfrak{L}_1$ *where* $|\beta| \leq p \ \ |\alpha|$ *and* $|\alpha|$, $|\beta|$ *denote the sizes of* α *and* β, *respectively.*

The relation $\leq$ is a pre-ordering. The relation strictly more succinct $\prec$ can be defined as follows: $\mathfrak{L}_1 \prec \mathfrak{L}_2$ iff $\mathfrak{L}_1 \leq \mathfrak{L}_2$ and $\mathfrak{L}_2 \not\leq \mathfrak{L}_1$.

Darwiche and Marquis have studied the succinctness criteria of the different languages studied in [34] and summarized results in Figure 2.3.

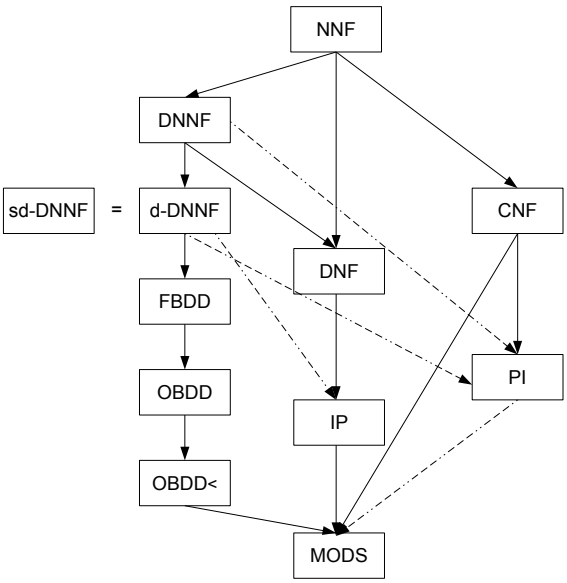


Figure 2.3: Succinctness relation between target compilation languages [34] (An edge $\mathfrak{L}_1 \rightarrow \mathfrak{L}_2$ indicates that $\mathfrak{L}_1$ is strictly more succinct than $\mathfrak{L}_2$, while $\mathfrak{L}_1 = \mathfrak{L}_2$ indicates that $\mathfrak{L}_1$ and $\mathfrak{L}_2$ are equally succinct. Dotted arrows indicate unknown relationships)

With the exception of NNF and CNF, all other languages depicted in Figure 2.3 qualify as target compilation languages. NNF and CNF are represented given their importance. Moreover, with the exception of PI language,

DNNF is the most succinct among all target compilation languages studied in [34]. It is well known that PI is not more succinct than DNNF, but we do not know whether DNNF is more succinct than PI.

Between DNNF and MODS, there is a succinctness relation: $DNNF \prec d\text{-}DNNF \prec FBDD \prec OBDD \prec OBDD_{\prec} \prec MODS$. By imposing *decomposability* on NNF, we have DNNF. By adding *determinism* on DNNF, we obtain d-DNNF. FBDD is obtained by imposing *decision* property. When we add the *ordering* property we have OBDD and $OBDD_{\prec}$ (any total ordering for OBDD and a specific one for $OBDD_{\prec}$). Adding each of these properties reduces language succinctness. However, adding *smoothness* to d-DNNF does not alter its succinctness since d-DNNF and sd-DNNF are equally succinct.

2.4.3 Logical queries and transformations

In evaluating the suitability of a target compilation language to a particular application, the succinctness of the language must be balanced against the set of queries and transformations that it supports in polytime. We consider in this part a number of queries, each of which returns valuable information about a propositional theory, and then identify target compilation languages which provide polytime algorithms for answering such queries.

A query is an operation that returns information about a theory without changing it. A transformation, on the other hand, is an operation that returns a modified theory, which is then operated on using queries. Many applications require a combination of transformations and queries. Table 2.3 (resp. 2.4) contains the set of queries (resp. transformations) of each target compilation of the knowledge map of [34].

Notation	Query	Notation	Transformation
CO	consistency	**CD**	conditioning
VA	validity	**FO**	forgetting
CE	clausal entailment	**SFO**	singleton forgetting
IM	implicant check	$\wedge$ **C**	conjunction
EQ	equivalence check	$\wedge$ **BC**	bounded conjunction
SE	sentential entailment	$\vee$ **C**	disjunction
CT	model counting	$\vee$ **BC**	bounded disjunction
MT	model enumeration	$\neg$ **C**	negation

Table 2.2: Notations for queries and transformations

$\mathfrak{L}$	CO	VA	CE	IM	EQ	SE	CT	ME
NNF	○	○	○	○	○	○	○	○
DNNF	√	○	√	○	○	○	○	√
d-NNF	○	○	○	○	○	○	○	○
s-NNF	○	○	○	○	○	○	○	○
f-NNF	○	○	○	○	○	○	○	○
d-DNNF	√	√	√	√	?	○	√	√
sd-DNNF	√	√	√	√	?	○	○	○
BDD	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	○	○	○
$\text{OBDD}_<$	√	√	√	√	√	√	√	√
DNF	√	○	√	○	○	○	○	√
CNF	○	√	○	√	○	○	○	○
PI	√	√	√	√	√	√	○	√
IP	√	√	√	√	√	√	○	√
MODS	√	√	√	√	√	√	√	√

Table 2.3: Subsets of the NNF language and their corresponding polytime queries (√ means 'satisfies' ○ means 'does not satisfy unless P=NP' and ? means 'unknown') [34]

$\mathfrak{L}$	CD	FO	SFO	$\wedge$C	$\wedge$BC	$\vee$C	$\vee$BC	$\neg$C
NNF	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
DNNF	$\surd$	$\surd$	$\surd$	$\circ$	$\circ$	$\surd$	$\surd$	$\circ$
d-NNF	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
s-NNF	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
f-NNF	$\surd$	$\circ$	$\surd$	$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\surd$
d-DNNF	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	?
sd-DNNF	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	?
BDD	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
FBDD	$\surd$	$\bullet$	$\circ$	$\bullet$	$\circ$	$\bullet$	$\circ$	$\surd$
OBDD	$\surd$	$\bullet$	$\surd$	$\bullet$	$\circ$	$\bullet$	$\circ$	$\surd$
OBDD$_<$	$\surd$	$\bullet$	$\surd$	$\bullet$	$\surd$	$\bullet$	$\surd$	$\surd$
DNF	$\surd$	$\surd$	$\surd$	$\bullet$	$\surd$	$\surd$	$\surd$	$\bullet$
CNF	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\bullet$	$\surd$	$\bullet$
PI	$\surd$	$\surd$	$\surd$	$\bullet$	$\bullet$	$\bullet$	$\surd$	$\bullet$
IP	$\surd$	$\bullet$	$\bullet$	$\bullet$	$\surd$	$\bullet$	$\bullet$	$\bullet$
MODS	$\surd$	$\surd$	$\surd$	$\bullet$	$\surd$	$\bullet$	$\bullet$	$\bullet$

Table 2.4: Subsets of the NNF language and their corresponding polytime transformations ($\surd$ means 'satisfies', $\bullet$ means 'does not satisfy', $\circ$ means 'does not satisfy unless P=NP' and ? means 'unknown') [34]

2.5 DNNF language

In this section, we focus on the DNNF language, the most succinct target compilation language among all target compilation languages studied in [34], with the exception of language PI. The DNNF language is characterized by the *decomposability* property on the *Negation Normal Form (NNF)* language. Establishing decomposability lies at the heart of many tractable languages in propositional logic. Given decomposability, one can conceive polytime algorithms for many queries and transformations that are known to be generally intractable.

2.5.1 DNNF's operations

In this subsection, we are, in particular, interested in conditioning, literal conjoin, satisfiability, entailment, forgetting and model counting.

Conditioning and literal conjoin

Let α be a propositional formula. Let ρ be a consistent term. Then, *conditioning* α on ρ denoted by $\alpha|\rho$ generates a new formula where each literal l of α is replaced by $\top$ if l is consistent with ρ, and by $\bot$ otherwise.

Example 2.5. *Let us consider the DNNF* $\alpha = (\neg a \wedge \neg b) \vee (b \wedge c)$. *Then, conditioning* α *on* b *simplifies to* c *as follows:* $\alpha|b = (\neg a \wedge \bot) \vee (\top \wedge c) = c$.

Conditioning corresponds to *restriction* in the literature of Boolean functions. The main application of conditioning is due to a theorem, which says that $\alpha \wedge \rho$ is satisfiable iff $\alpha|\rho$ is satisfiable [26, 27]. Conditioning also plays a key role in building compilers that enforce decomposability.

The operation of *literal conjoin* takes a DNNF α, a consistent term ρ and returns a DNNF which is equivalent to $\alpha \wedge \rho$. Even if α and ρ may be two DNNFs, their conjunction $\alpha \wedge \rho$ may not be a DNNF since α and ρ may share variables. Due to conditioning, DNNF supports literal conjoin. In fact, by conditioning a formula α on ρ, a new formula is generated that does not reference any literal from ρ. The conjunction of the conditioned formula $\alpha|\rho$ and the term ρ is equivalent to $\alpha \wedge \rho$. Formally, $Conjoin(\alpha, \rho) = (\alpha|\rho) \wedge \rho$ is a DNNF equivalent to $\alpha \wedge \rho$.

Example 2.6. *Let us re-consider the DNNF* $\alpha = (\neg a \wedge \neg b) \vee (b \wedge c)$. *Let* $\rho = b$ *be a literal. Then,* $(\alpha|b) \wedge b = c \wedge b$ *is equivalent to* $\alpha \wedge b$.

The operations of conditioning and literal conjoin can be done in linear time in the size of the DNNF. They are considered as the most fundamental to DNNF applications [27].

Satisfiability and entailment

A decomposable NNF formula $\wedge_i \alpha_i$ is satisfiable iff every conjunct α_i is *satisfiable*, while $\vee_i \alpha_i$ is always satisfiable iff some disjunct α_i is. The satisfiability of a DNNF can thus be tested in linear time by means of a single bottom-up pass over its DAG while visiting children before parents. Formally, it can defined as follows [27]:

1. $\text{SAT}\ (\alpha) = \begin{cases} \top & \text{if } \alpha \text{ is a literal } l \text{ or } \top \\ \bot & \text{if } \alpha \text{ is } \bot \end{cases}$
2. $\text{SAT}(\alpha = \bigwedge_i \alpha_i) = \top$ iff $\text{SAT}(\alpha_i)$ is $\top$ for every i,
3. $\text{SAT}(\alpha = \bigvee_i \alpha_i) = \top$ iff $\text{SAT}(\alpha_i)$ is $\top$ for some i.

If we have the satisfiability test, we can also define an *entailment* test. Specifically, to test whether a DNNF α entails a clause c (i.e., $\alpha \models c$), we need to test whether $\alpha \wedge \neg c$ is satisfiable. In other terms, if $\alpha \wedge \neg c$ is unsatisfiable then $\alpha \models c$. However, even if α and $\neg c$ are two DNNFs, their conjunction $\alpha \wedge \neg c$ is not guaranteed to be a DNNF. In this case, the conditioning transformation suffices for this purpose since $\alpha|\neg c$ is satisfiable iff $\alpha \wedge \neg c$ is satisfiable. Hence, to test if $\alpha \models c$, we should check the satisfiability of $\alpha|\neg c$ which is guaranteed to be a DNNF. Thanks to the conditioning transformation that ensures a linear entailment test for DNNFs [27].

Example 2.7. *Let us consider the DNNF α depicted by Figure 2.2 and the clause $c = \neg a \vee \neg b \vee \neg c \vee \neg d$. We want to test whether $\alpha \models c$. First, we should condition α on $c' = a \wedge b \wedge c \wedge d$, then we should test the satisfiability of $\alpha|c'$ as shown in Figure 2.4. We can deduce that $\alpha|c'$ is unsatisfiable, which means that $\alpha \wedge \neg c$ is unsatisfiable, i.e., $\alpha \models c$.*

Forgetting

The key operation that is intractable is *forgetting*. It is also referred to *marginalization*, or *elimination of middle terms*. Formally, let α be a propositional formula, let P be a finite set of propositional variables P_i, then forgetting P from α, denoted by $\exists P.\alpha$ (or $\text{Forget}(\alpha, P)$) is a formula that does not mention any variable P_i from P. For each formula β that does not mention any variable (i.e., $vars(\beta) \cap P = \emptyset$), we have $\alpha \models \beta$ iff $\exists P.\alpha \models \beta$. It can be defined as follows:

$$\exists P_i.\alpha = \alpha|P_i \vee \alpha|\neg P_i \tag{2.1}$$

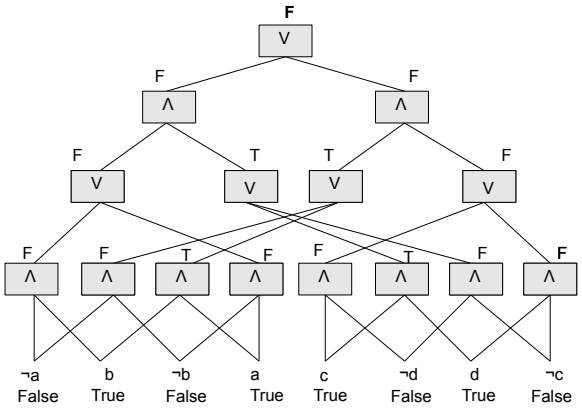


Figure 2.4: A DNNF conditioned on $a \wedge b \wedge c \wedge d$

where $\alpha|P_i$ (resp. $\alpha|\neg P_i$) is the result of conditioning of α on P_i (resp. $\neg P_i$). It is well known that forgetting can be performed in linear time using DNNF by means of a single bottom-up pass as defined in what follows [27]:

1. Forget $(\alpha, P) = \begin{cases} l & \text{if } \alpha \text{ is a literal } l \text{ and } vars(\alpha) \notin P \\ \top & \text{if } \alpha \text{ is a literal } l \text{ and } vars(\alpha) \in P \text{ or } \alpha \text{ is } \top \\ \bot & \text{if } \alpha \text{ is } \bot \end{cases}$
2. Forget $(\alpha = \bigwedge_i \alpha_i, P) = \wedge_i$ Forget (α_i, P),
3. Forget $(\alpha = \bigvee_i \alpha_i, P) = \vee_i$ Forget (α_i, P).

Example 2.8. *Let us re-consider the DNNF α of Figure 2.2. Forgetting the set of variables $P = \{a, b\}$ from α gives the DNNF of Figure 2.5 which can be simplified to $((c \wedge d) \vee (\neg c \wedge \neg d)) \vee ((c \wedge \neg d) \vee (\neg c \wedge d))$. It is obvious that the resulting DNNF does not mention variables a and b.*

Model counting

Model counting is a very prominent query. It consists in computing the number of models of a propositional formula. The most succinct target compilation language that supports such query is d-DNNF. In fact, both of decomposability and determinism properties are required to ensure model counting in linear time. By decomposability, two sets of partial models are multiplied by $\times$ only if they share no atoms. By determinism, two sets of partial models are unioned only if they contain no duplicates.

Darwiche in [28] computes the number of models using a secondary structure called *counting graph* defined as a rooted DAG containing a node labeled

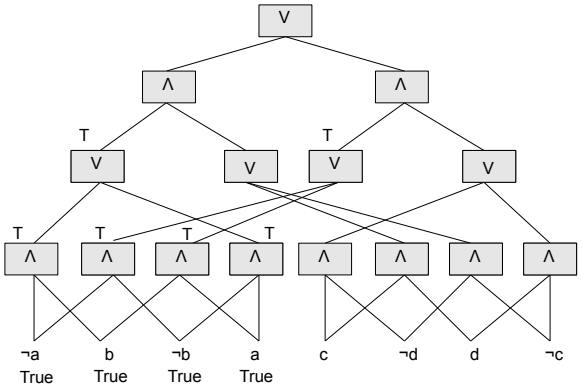


Figure 2.5: A DNNF after forgetting $P = \{a, b\}$

with l for each literal l, a node labeled with $+$ for each or-node and a node labeled with $\times$ for each and-node in the d-DNNF. The following definition shows how to perform counting operations.

Definition 2.4. *Let Nd be a node in a counting graph and S be a consistent set of literals. Then, the value of Nd in the counting graph is defined as follows:*

- $VAL(Nd) = 0$ *if N is labeled with l and $\neg l \in S$,*
- $VAL(Nd) = 1$ *if N is labeled with l and $\neg l \notin S$,*
- $VAL(Nd) = \prod_i VAL(Nd_i)$ *if Nd_i is labeled with $\times$, where Nd_i are the children of Nd,*
- $VAL(Nd) = \sum_i VAL(Nd_i)$ *if Nd_i is labeled with $+$, where Nd_i are the children of Nd.*

The value of the counting graph under literals S is the value of its root under S.

Example 2.9. *Let us consider the d-DNNF of Figure 2.2. Figure 2.6 depicts its counting graph evaluated under $S = \{a, \neg b\}$. The evaluation indicates that there are two models under this set of literals as shown in the root.*

2.5.2 From CNF to DNNF

The CNF language has few typical reasoning tasks which can be efficiently carried out on CNF representations. For this reason, one involves a compilation cost to prepare for answering a large number of queries in order

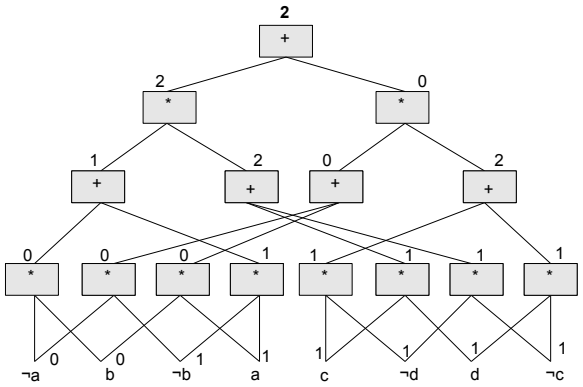


Figure 2.6: A counting graph

to amortize their cost. In this subsection, we describe how to compile a CNF formula into DNNF using the compiler c2d [29]. Interestingly enough, imposing decomposability on formulas consists in disconnecting the underlying formula into sub-formulas that do no share a variable by instantiating enough variables using conditioning. This process is then applied recursively until each sub-formula becomes decomposable [70].

The following theorem is the key of the compilation procedure underlying the compiler c2d [29]:

Theorem 2.1. ***Case Analysis****: Let α_1 and α_2 be two DNNF formulas. Let α be the formula $\bigvee_{\rho}((\alpha_1|\rho)\wedge(\alpha_2|\rho)\wedge\rho)$ where ρ are instantiations of variables mentioned in both α_1 and α_2. Then, α is a DNNF formula equivalent to $\alpha_1 \wedge \alpha_2$.*

The algorithm that converts a CNF formula α into a formula in DNNF is mainly based on this theorem:

Theorem 2.2. *Let α be a CNF formula, then*

$$DNNF(\alpha) = \begin{cases} c & \text{if } \alpha \text{ contains a single clause } c \\ \bigvee_{\rho} DNNF(\alpha_1|\rho) \wedge DNNF(\alpha_2|\rho) \wedge \rho & \text{otherwise} \end{cases}$$

where α_1 and α_2 is a partitioning of clauses in α and ρ is an instantiation of the variables mentioned in both α_1 and α_2.

Let us illustrate this theorem by the following example:

Example 2.10. *Let* $\alpha = (\neg a \vee b) \wedge (\neg b \vee c)$ *such that* $\alpha_1 = \neg a \vee b$ *and* $\alpha_2 = \neg b \vee c$. *Then, the formula* $((\neg a \vee b)|b \wedge (\neg b \vee c)|b \wedge b) \vee ((\neg a \vee b)|\neg b \wedge (\neg b \vee c)|\neg b \wedge \neg b) = (c \wedge b) \vee (\neg a \wedge \neg b)$ *is a DNNF formula equivalent to* α.

It is obvious that case analysis gives us the decomposability property since $(\alpha_1|\rho) \wedge (\alpha_2|\rho) \wedge \rho$ do not share variables. This is due to the conditioned formula $(\alpha_i|\rho)$ that do not mention any instantiation of ρ. However, satisfying the decomposability property is at the expense of increasing the size of the original formula.

Let us now detail the compilation process of the compiler c2d. In fact, given a clausal form α, c2d converts it into an equivalent DNNF by considering α as a conjunction of two sub-formulae α_1 and α_2 and applying the previous property, which gives us $\bigvee_{\rho}((\alpha_1|\rho) \wedge (\alpha_2|\rho) \wedge \rho)$. For each ρ, if $(\alpha_1|\rho)$ is a clause, nothing is done, else we perform the same treatment as we have done for α and so on for $(\alpha_2|\rho)$.

Two key observations should be mentioned. First, the size of the resulting DNNF is sensitive to the splitting way of α into two sub-formulae α_1 and α_2. Second, the above procedure is not *deterministic* in the sense that it does not specify how to split α. To make the procedure deterministic, Darwiche proposes to use a *decomposition tree*, which represents a recursive partitioning of α's clauses, defined as follows [27]:

Definition 2.5. *A decomposition tree* T *of a clausal form* α *is a binary tree composed of leaves corresponding to the clauses in* α *and internal nodes. If* d *is a leaf node in* T *corresponding to a clause* β *in* α, *then* $\alpha(d) = \{\beta\}$.

Some information should be associated to each node in the decomposition tree. First, for each internal node d, d_l and d_r denote the left and right children of d, respectively and $\alpha(d) = \alpha(d_l) \cup \alpha(d_r)$. Second, $vars(d)$ is defined as the set of variables appearing in clauses $\alpha(d)$. Finally, $vars^{\uparrow}(d)$ denotes the set of variables associated with leaf nodes which are not in the subtree rooted at d.

Example 2.11. *Figure 2.7 depicts a decomposition tree of the CNF formula* $\alpha = (\neg a \vee b) \wedge (\neg b \vee c) \wedge (\neg c \vee d)$. *From this decomposition tree, we can deduce that* $\alpha(d_1) = \{\neg a \vee b, \neg b \vee c\}$, $vars(d_1) = \{a, b, c\}$, $vars^{\uparrow}(d_1) = \{c, d\}$, $\alpha(d_l) = \{\neg a \vee b\}$ *and* $\alpha(d_r) = \{\neg b \vee c\}$.

The compilation of a CNF form α into a DNNF requires the construction of a decomposition tree T and the call of $DNNF(d_0, \top)$ with d_0 being the root of T, as outlined by Algorithm 1.

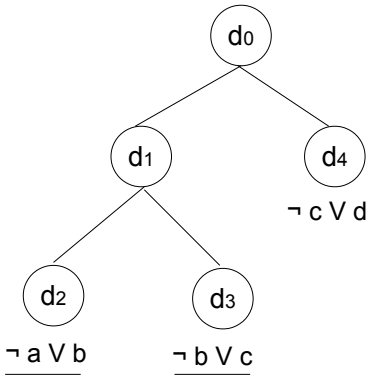


Figure 2.7: A decomposition tree

Algorithm 1: DNNF(d,ρ)

Data: Node in a decomposition tree d, Term ρ
Result: DNNF representation C_{DNNF}
begin
  if *d is a leaf node and $\alpha(d) = \beta$* **then**
    $C_{DNNF} \leftarrow \beta|\rho$
  else
    $C_{DNNF} \leftarrow \bigvee_{\gamma}(DNNF(d_l|\rho \wedge \gamma) \wedge DNNF(d_r|\rho \wedge \gamma) \wedge \gamma)$
    where $vars(\gamma) = (vars(d_l) \cap vars(d_r))\backslash vars(\rho)$
  return C_{DNNF}

Example 2.12. *Let $\alpha = (\neg a \vee b) \wedge (\neg b \vee c) \wedge (\neg c \vee d)$ be a CNF formula and Figure 2.7 be its decomposition tree. To compile α into DNNF, we should apply Algorithm 1 using d_0 and $\top$ as parameters (i.e., $DNNF(d_0, \top)$), which gives us:*

$DNNF(d_0, \top) = (c \wedge DNNF(d_1, c) \wedge DNNF(d_4, c)) \vee (\neg c \wedge DNNF(d_1, \neg c) \wedge DNNF(d_4, \neg c))$.

Since d_4 corresponds to a leaf node and $\alpha(d_4) = \neg c \vee d$, we can deduce that $DNNF(d_4, c) = (\neg c \vee d)|c \equiv d$ and $DNNF(d_4, \neg c) = (\neg c \vee d)|\neg c \equiv \top$.

Therefore, $DNNF(d_0, \top) = (c \wedge d \wedge DNNF(d_1, c)) \vee (\neg c \wedge DNNF(d_1, \neg c))$.

We shall now compute $DNNF(d_1, c)$ corresponding to $(b \wedge DNNF(d_2, c \wedge b) \wedge DNNF(d_3, c \wedge b)) \vee (\neg b \wedge DNNF(d_2, c \wedge \neg b) \wedge DNNF(d_3, c \wedge \neg b))$.

After simplification, we have $DNNF(d_1, c) = b \vee (\neg b \wedge \neg a)$ and $DNNF(d_1, \neg c) = b$.

Finally, $C_{DNNF} = DNNF(d_0, \top) = (c \wedge d \wedge (b \vee (\neg b \wedge \neg a))) \vee (\neg c \wedge b)$.

C_{DNNF} *is the DNNF representation of* α*, from which we can point out that conjuncts of any conjunction do not share variables.*

The complexity of building DNNFs depends on the width of the used decomposition tree, defined as follows [27]:

Definition 2.6. *Let* d *be a node in a decomposition tree* T*. The cluster of* d *is defined as follows:*

- *If* d *is a leaf node, then its cluster is* $vars(d)$.
- *If* d *is an internal node, then its cluster is* $(vars(d) \cap vars^{\uparrow}(d)) \cup (vars(d_r) \cap vars(d_l))$.

The width of a decomposition tree T *is the size of its maximal cluster minus one.*

The call of $DNNF(d_0, \top)$ for a clausal form α having n clauses and a decomposition tree T with width w takes $O(nw2^w)$ time and space. Therefore, the complexity of compiling a propositional theory into DNNF depends crucially on the quality (width) of the decomposition tree. The construction of good decomposition trees (ones with small widths) is still under study [3, 27, 33].

2.6 Conclusion

In this chapter, we have at first outlined the basic elements of propositional logic. Then, we have defined the principle of knowledge compilation which consists in pre-processing the fixed part of an input knowledge base in an off-line phase in order to efficiently respond to queries in an on-line phase. The mapping from one base to a new base is established using the so-called *target compilation languages*. Then, we have studied NNF languages and their subsets by restricting our study to decomposability, determinism and smoothness properties. Finally, we have mainly focused on the most succinct target compilation language, namely DNNF. The next chapter will be devoted to our new framework on compilation-based inference in possibilistic networks.

Part II

Compiling possibilistic graphical models

Chapter 3

Compilation-based inference in min-based possibilistic networks

3.1 Introduction

As we have seen in Chapter 2, knowledge compilation consists in transforming a problem offline into a tractable form which is then used to answer queries online. Applications of knowledge compilation include real-world problems, like model-based diagnosis [27], configuration [35], planning [64], Bayesian inference [30], etc. We are, in particular, interested in Compilation-based inference in Bayesian networks, which has been recently under intense investigation [23, 30, 74, 82].

The standard approach of Darwiche [30] is mainly based on encoding the *polynomial* associated to the network into a *Conjunctive Normal Form (CNF)* base, then retrieving answers to probabilistic inference queries in polytime by evaluating the *arithmetic circuit* associated to the compiled base. Our first objective in this chapter is to propose the possibilistic counterpart of this approach adapted to the qualitative context and named Π-DNNF. Then, we will develop a new purely possibilistic method, named DNNF-PKB, based on compiling possibilistic knowledge bases associated with possibilistic networks [12]. This latter is qualified as flexible since it permits to exploit efficiently all the existing propositional compilers. Both of proposed approaches are based on three sequential phases, namely: *encoding* phase, *compilation* phase and *inference* phase as sketched in Figure 3.1.

The remaining of this chapter is organized as follows: Section 3.2 provides

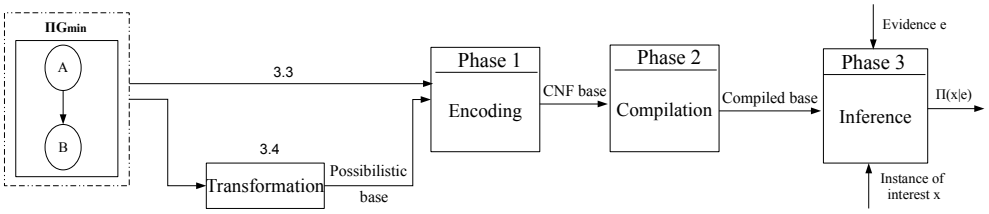


Figure 3.1: Principle of possibilistic compilation-based inference approaches (Lines annotated by 3.3 (resp. 3.4) are relative to the method Π-DNNF (resp. DNNF-PKB), while non annotated lines are shared by both methods)

the standard probabilistic compilation-based inference approach of Darwiche [30]. Section 3.3 presents its possibilistic adaptation using min-based possibilistic networks. Section 3.4 presents the purely possibilistic approach. Main results of this Chapter are published in [9].

3.2 Compilation of Bayesian networks

The topic of probabilistic compilation-based inference has been recently under intense investigation [23, 28, 30, 72, 73, 74, 82]. The idea behind probabilistic compilation-based inference is to encode the Bayesian network into a CNF formula and then perform *weighted model counting* is an effective method. This latter represents a generalization of model counting in which a weight is associated for each literal [10]. The reduction of probabilistic inference into a problem of model counting varies across two dimensions. The first dimension relates to compiling the CNF encoding of the Bayesian network into a structure that renders weighted model counting a polytime operation in the size of the compiled structure [23, 28, 30, 82]. The second dimension relates to whether weighted model counting is performed using a *search algorithm* on the CNF by splitting on the possible values of a chosen variable [72, 73, 74].

In this work, we are, in particular, interested in the first dimension and more precisely the standard probabilistic compilation-based inference approach [30], which is based on representing the Bayesian network using a *multi-linear function*, denoted by f_{MLF} and containing two types of variables, namely evidence indicators and network parameters, defined as follows:

- $\forall X_i \in V$, $\forall x_{ij} \in D_{X_i}$, we associate an *evidence indicator* $\lambda_{x_{ij}}$.

- $\forall X_i \in V$, $\forall x_{ij} \in D_{X_i}$, $\forall u_i \in D_{U_i}$ s.t. $u_i = \{u_{i1}, u_{i2}, ..., u_{im}\}$, we associate a *parameter variable* $\theta_{x_i|u_i}$ for each network parameter $P(x_i|u_i)$.

The multi-linear function contains a term for each instantiation of the network variables, and the term is the product of all indicators and parameters that are consistent with the instantiation. Formally, f_{MLF} is expressed by Equation (3.1).

$$f_{MLF} = \sum_{\mathbf{x}} \prod_{(x_i,u_i)\sim\mathbf{x}} \lambda_{x_i}\theta_{x_i|u_i} \tag{3.1}$$

where $\mathbf{x}$ represents instantiations of all network variables and $u_i \sim \mathbf{x}$ denotes the compatibility relationship among u_i and $\mathbf{x}$.

The multi-linear function f_{MLF} of the Bayesian network represents the probability distribution and allows to compute probability degrees of variables of interest. Namely, for any piece of evidence e which is an instantiation of some variables $E \in V$, f_{MLF} can be instantiated to return the probability of e. In fact, $P(e)$ is the result of replacing each evidence indicator λ_{x_i} in f_{MLF} with 1 if x_i is consistent with e, and with 0 otherwise.

Example 3.1. *Let us consider the Bayesian network of Figure 1.1 composed of two binary variables A and B. To represent the Bayesian network using a multi-linear function, we should at first set evidence indicators and network parameters given in Table 3.1 and Table 3.2, respectively.*

Instances	$\mathbf{f_{MLF}}$
a_1	λ_{a_1}
a_2	λ_{a_2}
b_1	λ_{b_1}
b_2	λ_{b_2}

Table 3.1: Instance indicators used in f_{MLF}

Then, the MLF corresponding to this network is as follows:
$f_{MLF} = \lambda_{a_1}\lambda_{b_1}\theta_{a_1}\theta_{b_1|a_1} + \lambda_{a_1}\lambda_{b_2}\theta_{a_1}\theta_{b_2|a_1} + \lambda_{a_2}\lambda_{b_1}\theta_{a_2}\theta_{b_1|a_2} + \lambda_{a_2}\lambda_{b_2}\theta_{a_2}\theta_{b_2|a_2}$.

Let us compute $P(b_1, a_2)$ from f_{MLF}. To this end, we should set λ_{b_2} and λ_{a_1} (resp. λ_{b_1} and λ_{a_2}) to 0 (resp. 1) and evaluate the reduced function as follows: $P(b_1, a_2) = \lambda_{a_2}\lambda_{b_1}\theta_{a_2}\theta_{b_1|a_2} = 1 * 1 * 0.6 * 0.2 = 0.12$.

Multi-linear functions have an exponential size as they include a term for each instantiation of the network variables [30]. To avoid this problem, Darwiche proposes to represent f_{MLF} using a new structure whose size may

Variables	Probability degrees	$\mathbf{f_{MLF}}$
A	$P(a_1) = 0.4$	θ_{a_1}
	$P(a_2) = 0.6$	θ_{a_2}
B	$P(b_1\|a_1) = 0.4$	$\theta_{b_1\|a_1}$
	$P(b_1\|a_2) = 0.2$	$\theta_{b_1\|a_2}$
	$P(b_2\|a_1) = 0.6$	$\theta_{b_2\|a_1}$
	$P(b_2\|a_2) = 0.8$	$\theta_{b_2\|a_2}$

Table 3.2: Network parameters used in f_{MLF}

not be exponential and leading the inference problem efficient. In fact, at first f_{MLF} should be encoded using the CNF propositional theory. Definition 3.1 outlines the CNF encoding of BN, denoted by C_{p*}.

Definition 3.1. *Using the set of evidence indicators and network parameters, C_{p*} contains the following clauses:*

- *Mutual exclusive clauses:*

$$\lambda_{x_{i1}} \vee \lambda_{x_{i2}} \vee \cdots \vee \lambda_{x_{in}} \tag{3.2}$$

$$\neg\lambda_{x_{ij}} \vee \neg\lambda_{x_{ik}}, j \neq k \tag{3.3}$$

- *Network parameter clauses:*
  $\forall\ \theta_{x_i|u_i}$, *we have:*

$$\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \theta_{x_i|u_i} \tag{3.4}$$

$$\theta_{x_i|u_i} \rightarrow \lambda_{x_i} \tag{3.5}$$

$$\theta_{x_i|u_i} \rightarrow \lambda_{u_{i1}}, \cdots, \theta_{x_i|u_i} \rightarrow \lambda_{u_{im}} \tag{3.6}$$

Once the CNF encoding is accomplished, it is then compiled into the most succinct target compilation language that supports model counting, namely d-DNNF. From this compiled base, an arithmetic circuit, denoted by CB_{*+}, is extracted that implements the encoded f_{MLF}.

Definition 3.2. *An arithmetic circuit of a compiled base CB, denoted by CB_{*+}, is a valued d-DNNF where $\wedge$ and $\vee$ are substituted by $*$ and $+$, respectively and each network parameter $\theta_{x_i|u_i}$ is replaced by the probability degree $P(x_i|u_i)$ it encodes.*

The circuit CB_{*+} can be used for linear-time inference since its evaluation after setting each evidence indicator to 1 or 0 and applying operators * and + in a bottom-up way corresponds to a weighted model counting problem.

Example 3.2. *The CNF encoding of the Bayesian network of Figure 1.1 is represented in Table 3.3 (20 clauses).*

Variables	**Mutual exclusive clauses**
A	$(\lambda_{a_1} \vee \lambda_{a_2}) \wedge (\neg\lambda_{a_1} \vee \neg\lambda_{a_2})$
B	$(\lambda_{b_1} \vee \lambda_{b_2}) \wedge (\neg\lambda_{b_1} \vee \neg\lambda_{b_2})$
Probability degrees of A	**Parameter clauses of A**
$P(a_1) = 0.4$	$(\lambda_{a_1} \rightarrow \theta_{a_1}) \wedge (\theta_{a_1} \rightarrow \lambda_{a_1})$
$P(a_2) = 0.6$	$(\lambda_{a_2} \rightarrow \theta_{a_2}) \wedge (\theta_{a_2} \rightarrow \lambda_{a_2})$
Probability degrees of B	**Parameter clauses of B**
$P(b_1\|a_1) = 0.4$	$(\lambda_{a_1} \wedge \lambda_{b_1} \rightarrow \theta_{b_1\|a_1}) \wedge (\theta_{b_1\|a_1} \rightarrow \lambda_{b_1}) \wedge (\theta_{b_1\|a_1} \rightarrow \lambda_{a_1})$
$P(b_1\|a_2) = 0.2$	$(\lambda_{a_2} \wedge \lambda_{b_1} \rightarrow \theta_{b_1\|a_2}) \wedge (\theta_{b_1\|a_2} \rightarrow \lambda_{b_1}) \wedge (\theta_{b_1\|a_2} \rightarrow \lambda_{a_2})$
$P(b_2\|a_1) = 0.6$	$(\lambda_{a_1} \wedge \lambda_{b_2} \rightarrow \theta_{b_2\|a_1}) \wedge (\theta_{b_2\|a_1} \rightarrow \lambda_{b_2}) \wedge (\theta_{b_2\|a_1} \rightarrow \lambda_{a_1})$
$P(b_2\|a_2) = 0.8$	$(\lambda_{a_2} \wedge \lambda_{b_2} \rightarrow \theta_{b_2\|a_2}) \wedge (\theta_{b_2\|a_2} \rightarrow \lambda_{b_2}) \wedge (\theta_{b_2\|a_2} \rightarrow \lambda_{a_2})$

Table 3.3: The CNF encoding of BN

*Suppose that we receive a certain information about A (i.e., $A = a_2$). Given this evidence, we will compute its impact on b_1, i.e., compute $P(b_1|a_2)$. To this end, we should compute both of $P(b_1, a_2)$ and $P(a_2)$ using the arithmetic circuit resulting from compiling the CNF encoding into a d-DNNF. Each probability degree is computed after assigning the appropriate values to evidence indicators and applying * and + in a bottom-up way. The value on the root of Figure 3.2 and Figure 3.3 corresponds to $P(b_1, a_2) = 0.12$ and $P(a_2) = 0.6$, respectively. We are now able to compute $P(b_1|a_2)$ which is equal to $\frac{P(b_1,a_2)}{P(a_2)} = \frac{0.12}{0.6} = 0.2$.*

3.3 Possibilistic adaptation of standard probabilistic inference approach

In this section, we will propose a direct possibilistic adaptation of the standard probabilistic approach presented in the previous section. In this approach, denoted by Π-DNNF, the min-based possibilistic networks are encoded and compiled into a new circuit specific to the qualitative framework.

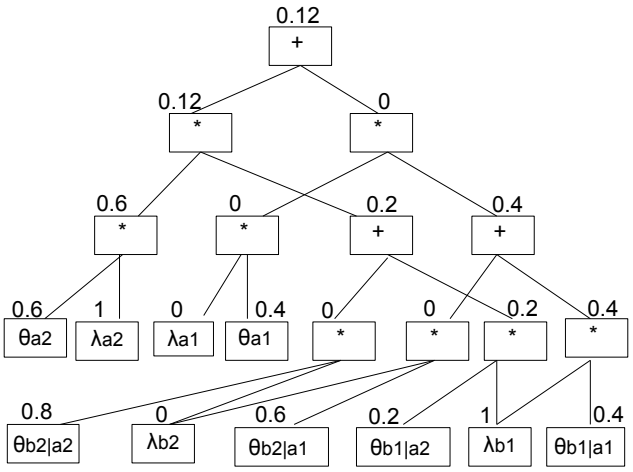


Figure 3.2: Computing $P(b_1, a_2)$

3.3.1 Encoding phase

The principle of encoding a min-based possibilistic network is to first transform it into a *Conjunctive Normal Form (CNF)* base. To this end, we need to represent instances of variables and also parameters using a set of propositional variables. More precisely, *instances indicators* are associated to different instances of network variables and *parameter variables* are relative to possibility degrees. Formally, the Π-DNNF method requires two types of propositional variables, namely:

- $\forall X_i \in V$, $\forall x_{ij} \in D_{X_i}$, we associate an *instance indicator* $\lambda_{x_{ij}}$.
- $\forall X_i \in V$, $\forall x_{ij} \in D_{X_i}$, $\forall u_i \in D_{U_i}$ s.t. $u_i = \{u_{i1}, u_{i2}, ..., u_{im}\}$, we associate a *parameter variable* $\theta_{x_i|u_{i1},u_{i2},...,u_{im}}$ for each network parameter $\Pi(x_i|u_{i1}, u_{i2}, ..., u_{im})$. Note that for any root node X_i, this parameter corresponds to θ_{x_i}.

When there is no ambiguity, we use λ_{x_i} (resp. $\theta_{x_i|u_i}$) instead of $\lambda_{x_{ij}}$ (resp. $\theta_{x_{ij}|u_{i1},u_{i2},...,u_{im}}$).

Example 3.3. *Let us consider the min-based possibilistic network* ΠG_{min} *of Figure 1.2. To encode* ΠG_{min} *into a CNF base, we should at first define the set of instance indicators and parameter variables represented respectively, in Table 4.3 and Table 4.4.*

The CNF encoding of ΠG_{min}, denoted by C_{min}, can be defined as follows:

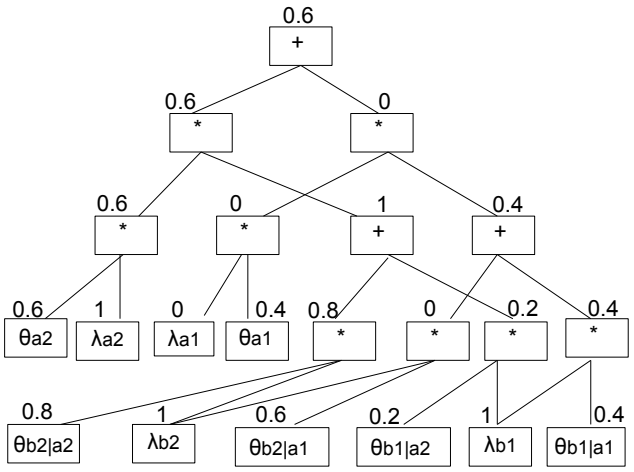


Figure 3.3: Computing $P(a_2)$

Instances	$\mathbf{C_{min}}$
a_1	λ_{a_1}
a_2	λ_{a_2}
b_1	λ_{b_1}
b_2	λ_{b_2}

Table 3.4: Instance indicators used in C_{min}

Definition 3.3. *Using the set of instance indicators and parameter variables, C_{min} contains the following clauses:*

- *Mutual exclusive clauses:*

$$\lambda_{x_{i1}} \vee \lambda_{x_{i2}} \vee \cdots \vee \lambda_{x_{in}} \tag{3.7}$$

$$\neg\lambda_{x_{ij}} \vee \neg\lambda_{x_{ik}}, j \neq k \tag{3.8}$$

- *Network parameter clauses:*
  $\forall\ \theta_{x_i|u_{i1},u_{i2},\ldots,u_{im}}$, *we have:*

$$\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \theta_{x_i|u_{i1},u_{i2},\ldots,u_{im}} \tag{3.9}$$

$$\theta_{x_i|u_{i1},u_{i2},\ldots,u_{im}} \rightarrow \lambda_{x_i} \tag{3.10}$$

$$\theta_{x_i|u_{i1},u_{i2},\ldots,u_{im}} \rightarrow \lambda_{u_{i1}}, \cdots, \theta_{x_i|u_{i1},u_{i2},\ldots,u_{im}} \rightarrow \lambda_{u_{im}} \tag{3.11}$$

Variables	Possibility degrees	$\mathbf{C_{min}}$
A	$\Pi(a_1) = 1$	θ_{a_1}
	$\Pi(a_2) = 0.4$	θ_{a_2}
B	$\Pi(b_1\|a_1) = 1$	$\theta_{b_1\|a_1}$
	$\Pi(b_1\|a_2) = 0.8$	$\theta_{b_1\|a_2}$
	$\Pi(b_2\|a_1) = 0.8$	$\theta_{b_2\|a_1}$
	$\Pi(b_2\|a_2) = 1$	$\theta_{b_2\|a_2}$

Table 3.5: Parameter variables used in C_{min}

The encoding C_{min} given in Definition 3.3 is in a CNF form i.e., a conjunction of all clauses induced by Equations (3.7), (3.8), (3.9), (3.10) and (3.11) where:

- Clauses (3.7) and (3.8) state that indicator variables are exclusive, i.e., exactly one indicator variable for each $X_i \in V$ is set to true in each model in C_{min},
- Clauses (3.9), (3.10) and (3.11) simply encode the fact that the possibility degree of $x_i|u_{i1}, ..., u_{im}$ (represented by the propositional formula $\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}}$) is equal (a logical equivalence $\Leftrightarrow$ in a logical setting) to $\Pi(x_i|u_{i1}, ..., u_{im})$ (represented by the propositional variable $\theta_{x_i|u_{i1},...,u_{im}}$).

  We call Equation (3.9) a right-side clause and Equations (3.10) and (3.11) left-side clauses since the parameter variable $\theta_{x_i|u_{i1},...,u_{im}}$ appears in the right (resp. left) -side of these Equations. The presence of these two types of clauses is simply due to the logical equivalence $\Leftrightarrow$ between indicator variables and parameter variables. From a logical point of view, clauses (3.9), (3.10) and (3.11) indicate that $\lambda_{x_i}, \lambda_{u_{i1}}, \cdots, \lambda_{u_{im}}$ are set to true in a model iff the parameter $\theta_{x_i|u_{i1},u_{i2},...,u_{im}}$ is also set to true in that model. Moreover, each $\theta_{x_i|u_{i1},u_{i2},...,u_{im}}$ only implies the set of indicator variables compatible with it.

Example 3.4. *Considering the network ΠG_{min} of Figure 1.2 and the set of instance indicators and parameter variables in Tables 4.3 and 4.4, respectively, then, the CNF encoding of ΠG_{min} using Definition 3.3 contains clauses of Table 3.6.*

From a possibilistic point of view, we can say that C_{min} recovers the min-based joint possibility distribution.

Proposition 3.1. *Let C_{min} be the CNF encoding of a ΠG_{min} using Definition 3.3. Let ω be an interpretation from Ω and λ be the conjunction of*

Variables	**Mutual exclusive clauses**
A	$(\lambda_{a_1} \vee \lambda_{a_2}) \wedge (\neg\lambda_{a_1} \vee \neg\lambda_{a_2})$
B	$(\lambda_{b_1} \vee \lambda_{b_2}) \wedge (\neg\lambda_{b_1} \vee \neg\lambda_{b_2})$
Possibility degrees of A	**Parameter clauses of A**
$\Pi(a_1) = 1$	$(\lambda_{a_1} \rightarrow \theta_{a_1}) \wedge (\theta_{a_1} \rightarrow \lambda_{a_1})$
$\Pi(a_2) = 0.4$	$(\lambda_{a_2} \rightarrow \theta_{a_2}) \wedge (\theta_{a_2} \rightarrow \lambda_{a_2})$
Possibility degrees of B	**Parameter clauses of B**
$\Pi(b_1\|a_1) = 1$	$(\lambda_{a_1} \wedge \lambda_{b_1} \rightarrow \theta_{b_1\|a_1}) \wedge (\theta_{b_1\|a_1} \rightarrow \lambda_{b_1}) \wedge (\theta_{b_1\|a_1} \rightarrow \lambda_{a_1})$
$\Pi(b_1\|a_2) = 0.8$	$(\lambda_{a_2} \wedge \lambda_{b_1} \rightarrow \theta_{b_1\|a_2}) \wedge (\theta_{b_1\|a_2} \rightarrow \lambda_{b_1}) \wedge (\theta_{b_1\|a_2} \rightarrow \lambda_{a_2})$
$\Pi(b_2\|a_1) = 0.8$	$(\lambda_{a_1} \wedge \lambda_{b_2} \rightarrow \theta_{b_2\|a_1}) \wedge (\theta_{b_2\|a_1} \rightarrow \lambda_{b_2}) \wedge (\theta_{b_2\|a_1} \rightarrow \lambda_{a_1})$
$\Pi(b_2\|a_2) = 1$	$(\lambda_{a_2} \wedge \lambda_{b_2} \rightarrow \theta_{b_2\|a_2}) \wedge (\theta_{b_2\|a_2} \rightarrow \lambda_{b_2}) \wedge (\theta_{b_2\|a_2} \rightarrow \lambda_{a_2})$

Table 3.6: The CNF encoding C_{min} of ΠG_{min}

indicator variables λ_{x_i} *related to* ω *(i.e.,* $\lambda \equiv \bigwedge_{x_i \in \omega} \lambda_{x_i}$*).*

Let us consider $k(\lambda)$ *be the result of conditioning of* C_{min} *on* λ *using instance indicators, i.e.,*

$$\lambda_{x_i} = \begin{cases} \top & \text{if } x_i \in \omega \\ \bot & \text{otherwise} \end{cases} \tag{3.12}$$

Then,

$$k(\lambda) \equiv (\bigwedge_{(x_i,u_i)\in\omega} \theta_{x_i|u_i}) \bigwedge (\bigwedge_{(x_i,u_i)\notin\omega} \neg\theta_{x_i|u_i})$$

Note that each $\neg\theta_{x_i|u_i}$ *should be ignored by replacing it by* $\top$ *since only positive literals, i.e.,* $\theta_{x_i|u_i}$ *encode possibility degrees. The resulted* $k(\lambda)$ *is equivalent to:*

$$k(\lambda) \equiv \bigwedge_{(x_i,u_i)\in\omega} \theta_{x_i|u_i}$$

Proof 3.1. *From* C_{min}*, we deduce* $\theta_{x_i|u_i}$ *and* $\neg\theta_{x_i|u_i}$ *as follows:*

- *If* $(x_i, u_i) \in \omega$*: From Equations (3.9), (3.10) and (3.11), we can deduce* $\theta_{x_i|u_i}$ *as follows:*

$$\begin{cases} \lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \theta_{x_i|u_i} \\ \theta_{x_i|u_i} \rightarrow \lambda_{x_i} \\ \theta_{x_i|u_i} \rightarrow \lambda_{u_i} \end{cases} \quad \overset{\lambda_{x_i}=\top}{\underset{\lambda_{u_i}=\top}{\Longrightarrow}} \quad \theta_{x_i|u_i}$$

- *If* $(x_i, u_i) \notin \omega$*: From Equations (3.9), (3.10) and (3.11), we can deduce* $\neg\theta_{x_i|u_i}$ *as follows:*

$$\begin{cases} \lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \theta_{x_i|u_i} \\ \theta_{x_i|u_i} \rightarrow \lambda_{x_i} \\ \theta_{x_i|u_i} \rightarrow \lambda_{u_i} \end{cases} \overset{\lambda_{x_i}=\bot}{\underset{\lambda_{u_i}=\bot}{\Longrightarrow}} \neg\theta_{x_i|u_i}$$

Thus, $k(\lambda) \equiv (\bigwedge_{(x_i,u_i)\in\omega} \theta_{x_i|u_i}) \bigwedge (\bigwedge_{(x_i,u_i)\notin\omega} \neg\theta_{x_i|u_i})$. ■

Let $Pk(\lambda)$ the positive part of Proposition 3.1 by replacing each negative literal $\neg\theta_{x_i|u_i}$ by $\top$ in $k(\lambda)$, i.e.,

$$Pk(\lambda) \equiv \bigwedge_{(x_i,u_i)\in\omega} \theta_{x_i|u_i} \tag{3.13}$$

Let $\pi_{C_{min}} : \Omega \rightarrow [0,1]$ be the possibility distribution computed from $Pk(\lambda)$ by replacing $\wedge$ by min and each $\theta_{x_i|u_i}$ by $\Pi(x_i|u_i)$.

Proposition 3.2. *Let ΠG_{min} be a min-based possibilistic network and C_{min} its CNF encoding using Definition 3.3. Let ω be an interpretation from Ω and $Pk(\lambda)$ be its CNF encoding resulting from incorporating ω into C_{min} using Equation (3.13). Then,*

$$\forall\omega \in \Omega, \pi_{min}(\omega) = \pi_{C_{min}}(\omega) \tag{3.14}$$

$$\forall\phi \subseteq \Omega, \Pi_{min}(\phi) = \Pi_{C_{min}}(\phi) \tag{3.15}$$

where π_{min} (resp. $\pi_{C_{min}}$) is given by Definition 1.5 (resp. Equation (3.13)) and Π_{min} (resp. $\Pi_{C_{min}}$) is derived from π_{min} (resp. $\pi_{C_{min}}$).

Proof 3.2. *By setting $\neg\theta_{x_i|u_i}$ to $\top$, $\pi_{C_{min}}(\omega)$ is computed using $Pk(\lambda)$ as follows:*

$$\begin{aligned} \pi_{C_{min}}(\omega) &= \min_{(x_i,u_i)\in\omega} \theta_{x_i|u_i} \\ &= \min_{(x_i,u_i)\in\omega} \Pi(x_i|u_i) \\ &= \pi_{min}(\omega). \end{aligned}$$

Thus, Equation (3.14) is established.

This result is relative to an interpretation ω, to generalize it to an event $\phi \subseteq \Omega$, we obtain:

$\max_{\omega\models\phi} \pi_{min}(\omega) = \max_{\omega\models\phi} \pi_{C_{min}}(\omega)$,

Thus, $\Pi_{min}(\phi) = \Pi_{C_{min}}(\phi)$. ■

This proposition is illustrated by the following example:

Example 3.5. *Let us consider the min-based possibilistic network* ΠG_{min} *of Figure 1.2. Let* $\omega = \{a_2, b_1\}$ *be an interpretation from* Ω *and* $\lambda \equiv \lambda_{a_2} \wedge \lambda_{b_1}$. *Then, conditioning* C_{min} *on* λ *using Equation (3.1) results in* $k(\lambda) = (\theta_{a_2} \wedge \theta_{b_1|a_2} \wedge \neg\theta_{a_1} \wedge \neg\theta_{b_1|a_1} \wedge \neg\theta_{b_2|a_1} \wedge \neg\theta_{b_2|a_2})$. *The positive part of* $k(\lambda)$ *using Equation (3.13) is equivalent to* $Pk(\lambda) \equiv (\theta_{a_1} \wedge \theta_{b_1|a_2})$.

Given $Pk(\lambda)$, *we are now able to compute* $\pi_{C_{min}}(\omega)$ *as follows:* $\pi_{C_{min}}(\omega) = min(\Pi(a_1), \Pi(b_1|a_1)) = min(0.4, 0.8) = 0.4 = \pi_{min}(\omega)$ *(row 3 of Table 1.6).*

3.3.2 Compilation phase

The investigated query refers to an efficient computation of a-posteriori possibility degrees given some evidence on a set of variables. The computation process of $\Pi(e)$ is mainly based on setting appropriate values to instance indicators which are used to record evidence e. These indicators have the effect of excluding the terms that are incompatible with the evidence, i.e., given an evidence e, indicators incompatible with e should be excluded and those consistent with e should be remained. Using compilation terms, the CNF encoding associated to the min-based possibilistic network should be conditioned on e and the resulting conditioned representation is then decoded on a valued expression using min and max operators from which we efficiently compute $\Pi(e)$. At first sight, the chosen target compilation language should support *conditioning*. Therefore, our choice will rely on the trade-off between succinctness and tractability criteria. Based on this trade-off, DNNF should be chosen since it has been proved in [27] that it is well suited for ensuring conditioning. The compilation from CNF to DNNF has been detailed in the previous chapter (see section 2.5.2). The resulting compiled base is denoted by CB.

3.3.3 Inference phase

Given the compiled base CB resulting from the previous phase, an instance of interest x of a variable $X \in V$ and an evidence e of some variables $E \subseteq V$, we should be able to efficiently compute the effect of e on x, namely $\Pi_c(x|e)$. Using Equation (1.11), it is clear that we should compute both of $\Pi_c(x, e)$ and $\Pi_c(e)$. The computation process is described in depth in the following steps in which we will compute $\Pi_c(x, e)$. Of course, $\Pi_c(e)$ should be computed in the same spirit as $\Pi_c(x, e)$.

Step 1: Updating instance indicators

This step serves to record the instance of interest x and the evidence e into instance indicators λ_{x_i}. It corresponds to conditioning the compiled base

CB on x and e. Formally:

$$\lambda_{x_i} = \begin{cases} \top & \text{if } x_i \sim e \text{ and } x_i \sim x \\ \bot & \text{otherwise.} \end{cases} \tag{3.16}$$

where $\sim$ denotes the compatibility relation, i.e., $x_i \sim x$ refers to the fact that x_i and x agree on values. For instance, if we consider a_1 the instance of interest of A, then $b_1 \sim a_1$, $b_2 \sim a_1$, while $a_2 \nsim a_1$. The conditioned compiled base is denoted by $[CB|x, e]$.

Step 2: Mapping logical representation into a numerical representation

In this step, we transform the logical compiled base resulting from the previous step into a numerical representation, named a *min-max circuit* and denoted by CB_{minmax}, from which we will be able to ensure an efficient possibilistic computation. By mapping, we mean:

- replacing $\vee$ and $\wedge$ by max and min, respectively,
- substituting each $\top$ (resp. $\bot$) by 1 (resp. 0),
- associating the possibility degree $\Pi(x_i|u_i)$ to each propositional variable $\theta_{x_i|u_i}$.

Definition 3.4. *A min-max circuit of a compiled base CB, denoted by CB_{minmax}, is a valued sentence where $\wedge$ and $\vee$ are substituted by min and max, respectively. Each propositional variable $\theta_{x_i|u_i}$ is replaced by the possibility degree $\Pi(x_i|u_i)$ it encodes. Moreover, each truth value associated to an instance indicator λ_{x_i} is replaced by 1 or 0. Leaf nodes correspond to circuit inputs (i.e., indicator and parameter variables), internal nodes correspond to min and max operators, and the root corresponds to the circuit output.*

Min-max circuits are considered a special case of *Valued Negation Normal Forms* (VNNFs) [44] which are valued languages used to represent a much more general class of functions than just Boolean ones, namely those ranging over an ordered scale. In these languages, the semantics of nodes are changed from logical operators (such as $\wedge$, $\vee$) to general operators ($\otimes$ and $\oplus$). In our case, operators in min-max circuits are restricted to minimum and maximum. The VNNF framework supports a larger family of queries, such as optimization, etc. It also supports several transformations, namely $\otimes$-variable elimination (a generalization of classical forgetting by using $\otimes$ instead of $\vee$ in Equation (2.1).

In what follows, the function *map*$([CB|x, e])$ will be used to map $[CB|x, e]$ into CB_{minmax}.

Step 3: Computation

The last step corresponds to evaluating the min-max circuit CB_{minmax} in order to compute $\Pi_c(x, e)$. In fact, evaluation consists in applying *min* and *max* operators in a bottom-up way and the final degree on the root represents $\Pi_c(x, e)$. Computation using CB_{minmax} corresponds to a *max-variable elimination* operation. Proposition 3.3 proves that the possibility degree computed using max-variable elimination and the one using the joint possibility distribution are the same.

The impact of e on x (i.e., $\Pi_c(x|e)$) is finally computed depending on the underlying definition of conditioning as outlined by Algorithm 2. It is clear that inference is guaranteed to be established in polytime since max-variable elimination is supported by min-max circuits. In what follows, we will use the function *evaluate*(CB_{minmax}) to evaluate CB_{minmax}.

Algorithm 2: Inference using CB_{minmax}

Data: CNF encoding C_{min}, instance of interest x, evidence e
Result: $\Pi_c(x|e)$
begin
  $CB \leftarrow$ compile(C_{min})
  $Int \leftarrow \{x, e\}$
  $\Pi_c(x, e) \leftarrow$ Computing (CB, Int)
  $Int \leftarrow \{e\}$
  $\Pi_c(e) \leftarrow$ Computing (CB, Int)
  if $\Pi_c(x, e) < \Pi_c(e)$ **then**
    $\Pi_c(x|e) \leftarrow \Pi_c(x, e)$
  else
    $\Pi_c(x|e) \leftarrow 1$
  return $\Pi_c(x|e)$

Algorithm 3: Computing

Data: CB, instance of interest Int
Result: $\Pi_c(Int)$
begin
  $CB|Int \leftarrow$ condition(CB, Int)
  $CB_{minmax} \leftarrow$ map$(CB|Int)$
  $\Pi_c(Int) \leftarrow$ evaluate(CB_{minmax})
  return $\Pi_c(Int)$

Example 3.6. *Let us consider the compiled base of Figure 3.4 and suppose that we receive a certain information about B (i.e., $B = b_1$). Given this evidence, we will compute its impact on a_2 (i.e., computing $\Pi_c(a_2|b_1)$). To this*

end, we should only compute $\Pi_c(a_2, b_1)$ *since* $\Pi_c(b_1)$ *is equal to 1. At first, we should record* a_2 *and* b_1 *into instance indicators by setting* λ_{a_2} *and* λ_{b_1} *(resp.* λ_{a_1} *and* λ_{b_2}*) to* $\top$ *(resp.* $\bot$*). Then, we map the logical compiled base arisen from Step 1 into a min-max circuit depicted by Figure 3.5. Finally, we apply min and max in a bottom-up fashion as shown in Figure 3.5. Thus,* $\Pi_c(a_2, b_1)$ *is equal to* 0.4*. This value corresponds to the one of Table 1.6 using the min-based joint distribution. Now, we are able to compute* $\Pi_c(a_2|b_1)$ *which is equal to* $\Pi_c(a_2, b_1) = 0.4$ *since* $\Pi_c(a_2, b_1) < \Pi_c(b_1)$*.*

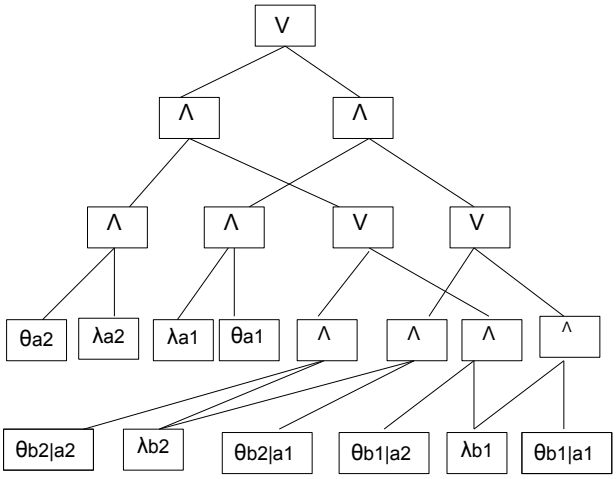


Figure 3.4: The compiled base CB of ΠG_{min}

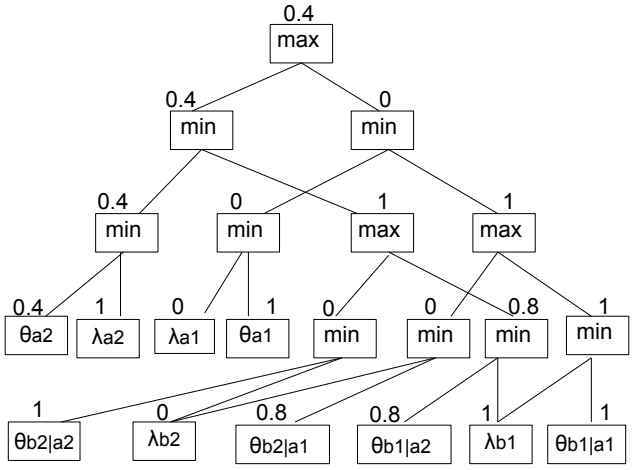


Figure 3.5: The min-max circuit CB_{minmax}

Proposition 3.3. *Let* ΠG_{min} *be a possibilistic network and* CB_{minmax} *be*

its min-max circuit. Let Int *be an instance of interest. Then,*

$$\Pi_c(Int) = \Pi_{VE}(Int) \tag{3.17}$$

where $\Pi_c(Int)$ *is computed using Algorithm 3 and* $\Pi_{VE}(Int)$ *uses the max-variable elimination process.*

Proof 3.3. $\Pi_{VE}(Int)$ *is computed by applying max-variable elimination, denoted by max-VE, to each propositional variable* $P_i \in CB_{minmax}$ *which can be either* λ_{x_i} *or* $\theta_{x_i|u_i}$ *as follows:*

- *If* $CB_{minmax} = min(\alpha, \beta)$, *then* $max\text{-}VE(CB_{minmax}, P_i) = \mathbf{min}(max\text{-}VE(\alpha, P_i), max\text{-}VE(\beta, P_i))$.
- $CB_{minmax} = max(\alpha, \beta)$, *then* $max\text{-}VE(CB_{minmax}, P_i) = \mathbf{max}(max\text{-}VE(\alpha, P_i), max\text{-}VE(\beta, P_i))$.
- *If* $CB_{minmax} = l$ *and* $l = P_i$, *then* $max\text{-}VE(CB_{minmax}, P_i) = \mathbf{value}(P_i)$ *which corresponds to the value that* P_i *encodes.*
- *If* $CB_{minmax} = l$ *and* $l \neq P_i$, *then* $max\text{-}VE(CB_{minmax}, P_i) = l$.
- *If* $CB_{minmax} = 1$, *then* $max\text{-}VE(CB_{minmax}, P_i) = 1$.
- *If* $CB_{minmax} = 0$, *then* $max\text{-}VE(CB_{minmax}, P_i) = 0$.

Thus, $\Pi_{VE}(Int) = max\text{-}VE(CB_{minmax}, P_i) = \Pi_c(Int)$. ■

Let us illustrate Proposition 3.3 by the following example:

Example 3.7. *Let us consider an excerpt of* CB_{minmax} *of Figure 3.5 corresponding to* $max(min(\lambda_{b_2}, \theta_{b_2|a_1}), min(\lambda_{b_1}, \theta_{b_1|a_1}))$. *Then, to compute* $\Pi_c(b_1)$, *we should substitute* λ_{b1}, λ_{b_1}, $\theta_{b_1|a_1}$ *and* $\theta_{b_2|a_1}$ *by 1, 0, 1 and 0.8, respectively and apply min and max operators, which results in* $\Pi_c(b_1) = 1$. *Let us check this result using the max-variable elimination process.*

By applying max-VE to each variable in CB_{minmax}, *we obtain:*

1. $max\text{-}VE(CB_{minmax}, \lambda_{b_2})$
   $= \mathbf{max}(max\text{-}VE(min(\lambda_{b_2}, \theta_{b_2|a_1}), \lambda_{b_2}), max\text{-}VE(min(\lambda_{b_1}, \theta_{b_1|a_1}), \lambda_{b_2})$
   $= \mathbf{max}(min(max\text{-}VE(\lambda_{b_2}, \lambda_{b_2}), max\text{-}VE(\theta_{b_2|a_1}, \lambda_{b_2}), min(max\text{-}VE(\lambda_{b_1}, \lambda_{b_2}), max\text{-}VE(\theta_{b_1|a_1}, \lambda_{b_2})))$
   $= \mathbf{max}(min(0, \theta_{b_2|a_1}), min(\lambda_{b_1}), \theta_{b_1|a_1})$
   $= \mathbf{max}(0, min(\lambda_{b_1}, \theta_{b_1|a_1}))$
   $= min(\lambda_{b_1}, \theta_{b_1|a_1})$
2. $max\text{-}VE(CB_{minmax}, \lambda_{b_1}) = \mathbf{min}(1, \theta_{b_1|a_1}) = \theta_{b_1|a_1}$
3. $max\text{-}VE(CB_{minmax}, \theta_{b_1|a_1}) = 1 = \Pi_{VE}(b_1)$.

Note that $\Pi_{VE}(b_1)$ *corresponds to* $\Pi_c(b_1)$ *computed using Algorithm 3.*

3.4 Possibilistic approach using possibilistic knowledge bases

Possibilistic logic bases and possibilistic networks are considered two different frameworks for representing knowledge. Using possibilistic logic bases, pieces of knowledge are expressed by logical formulas according to their levels of certainty, while possibilistic networks exhibit relationships between variables. These representations are semantically equivalent since they lead to the same possibility distribution. In a logical framework, data are given in terms of necessities, while in a graphical framework, data are given in terms of conditional possibilities [12].

In [12], authors have provided a transition of possibilistic networks into possibilistic logic bases. In another angle, Benferhat et al. in [18] have focused on the compilation of bases under the possibilistic logic policy in order to be able to process inference from it in a polynomial time. This is mainly accomplished by ensuring a CNF encoding of the possibilistic base, and compiling it using any target compilation language that supports conditioning. The combination of these methods allows us to propose an alternative approach to possibilistic inference in min-based possibilistic networks, denoted by DNNF-PKB. This is vindicated by the fact that the possibilistic logic reasoning machinery can be applied to directed possibilistic networks.

3.4.1 From a graphical to a logic-based representation

The starting point of the transformation from a graphical to a logic-based representation is that the possibilistic base associated to a possibilistic network is the result of the fusion of elementary bases. These elementary bases are composed of formulae associated to the prior and conditional possibilities attached to network's nodes. Definition 3.5 presents the transformation of a min-based possibilistic network into a possibilistic knowledge base [12]:

Definition 3.5. *Let ΠG_{min} be a min-based possibilistic network, then its possibilistic knowledge base is expressed by:*

$$\Sigma_{min} = \Sigma_{X_1} \cup \Sigma_{X_2} \cup \cdots \cup \Sigma_{X_N} \tag{3.18}$$

where $\forall X_i \in V$:

$$\Sigma_{X_i} = \{(\neg x_i \vee \neg u_i, a_i) : a_i = 1 - \Pi(x_i|u_i) \neq 0\} \tag{3.19}$$

Example 3.8. *Let us consider the min-based network ΠG_{min} of Figure 1.2. Then, the possibilistic knowledge base of ΠG_{min} is the following:* $\Sigma_{min} =$

$((a_1, 0.6), (a_2 \vee b_1, 0.2), (a_1 \vee b_2, 0.2))$. *We can deduce that* Σ_{min} *does not contain zero-weighted formulas corresponding to possibility degrees equal to 1.*

Note that the quantitative possibilistic base of a product-based possibilistic network can be obtained using the transformation of [14].

3.4.2 Compilation-based inference using possibilistic knowledges bases

After translating the possibilistic network to a possibilistic base Σ_{min} using definition 3.5, the idea is to encode this latter into a classical propositional base using the CNF representation language. This is performed by affecting new propositional variables for the different necessity degrees existing in the possibilistic knowledge base. More formally, let $A = \{a_1, ..., a_n\}$ with $a_1 \succ ... \succ a_n$ be the different weights used in Σ_{min}, a set of additional propositional variables, denoted by A_i, which correspond exactly to the number of different weights, are incorporated. For each formula (α_i, a_i) will correspond the propositional formula $\alpha_i \vee A_i$. Hence, the propositional encoding of Σ_{min}, denoted by K_Σ is expressed by:

$$K_\Sigma = \{\alpha_i \vee A_i : (\alpha_i, a_i) \in \Sigma_{min}\} \tag{3.20}$$

Note that the number of additional propositional variables in K_Σ is not high since we encode one variable per different degree instead of one variable per degree. Consequently, the encoding cost is also considered faible.

In [18], different inference queries have been taken into account but in our case, we are only interested in a particular query useful for possibilistic networks, namely *What is the possibility degree of an event* $X = x$ *given an evidence* $E = e$ *on a set of variables?* Therefore, we propose to adapt the algorithm given in [18] in order to respond to this query. Algorithm 4 outlines the new possibilistic approach in which both of *conditioning* and *clausal entailment* are required to compute a-posteriori possibility degrees. Hence, to ensure an efficient computation, K_Σ should be compiled into any target compilation language that supports conditioning and clausal entailment. The resulting compiled base is denoted by K_c. Our approach is qualified to be flexible since it takes advantage of existing propositional knowledge bases compilation methods [18].

In algorithm 4, we first start by testing the clausal entailment of the first propositional variable, equivalent to $K_c \nvDash A_1 \vee \neg e$. If this deduction is not satisfied, we condition K_c on $\neg A_1$ and then test if $K_c|\neg A_1$ entails $\neg x$. If this is the case, we compute $\Pi_c(x|e)$, else we move to the next propositional

variable and we re-itere the same treatment. This method is referred to by DNNF-PKB.

Algorithm 4: DNNF-PKB

Data: ΠG_{min}, instance of interest x, evidence e
Result: $\Pi_c(x|e)$
begin
  Transformation into K_Σ
  Let Σ_{min} be the possibilistic base of ΠG_{min} using Definition 3.5
  Let K_Σ be the CNF encoding of Σ_{min} using Equation (3.20)
  Inference
  Let K_c be the compilation of K_Σ
  StopCompute $\leftarrow$ false
  i $\leftarrow 1$
  $\Pi_c(x|e) \leftarrow 1$
  while *($K_c \not\models A_i \vee \neg e$)* ***and*** *($i \leq k$)* ***and*** *(StopCompute=false)* **do**
    $K_c|\neg A_i \leftarrow$ condition (K, $\neg A_i$)
    if $K_c|\neg A_i \models \neg x$ **then**
      StopCompute $\leftarrow$ true
      Let *degree*(i) be the weight associated to A_i
      $\Pi_c(x|e) \leftarrow$ 1-*degree*(i)
    else
      $i \leftarrow i + 1$
  return $\Pi_c(x|e)$

Due to the compilation step, this algorithm runs in a polynomial time. Moreover, the number of additional variables is low since it corresponds exactly to the number of priority levels existing in the base.

Example 3.9. *The CNF encoding of the possibilistic knowledge base* Σ_{min} *of Example 3.8 is shown in Table 3.7.*

Clauses of A	
$(a_1, 0.6)$	$(a_1 \vee A_1)$
Clauses of B	
$(a_2 \vee b_1, 0.2)$	$(a_2 \vee b_1 \vee A_2)$
$(a_1 \vee b_2, 0.2)$	$(a_1 \vee b_2 \vee A_2)$

Table 3.7: The CNF encoding K_Σ of Σ_{min}

The CNF encoding K_Σ *is then compiled into DNNF. The resulting compiled base is as follows:* $K_c = \{[(a_2 \wedge A_1) \wedge (b_2 \vee (b_1 \wedge A_2))] \vee [a_1 \wedge (b_1 \vee (b_2 \wedge A_2))\}$. *Let us compute the effect of the evidence* b_1 *on* a_2 *using* K_c. *The computation of* $\Pi_c(a_2|b_1)$ *requires only one iteration as follows:*

- $K_c \not\models b_2 \vee A_1 \Rightarrow K_c|\neg A_1 = [a_1 \wedge (b_1 \vee (b_2 \wedge A_2))]$,

– $(K_c|\neg A_1) \models a_1 \Rightarrow StopCompute \leftarrow true$.

Hence, $\Pi_c(a_2|b_1) = 1 - degree(1) = 1 - 0.6 = 0.4$ *where* $degree(1)$ *designates the weight associated to* A_1*, i.e., 0.6.*

Proposition 3.4. *Let* ΠG_{min} *be a min-based possibilistic network. Let* x *be an instance of interest of a variable* $X \in V$ *and* e *an evidence. Then:*

$$\Pi_c(x|e) = \Pi(x|e) \tag{3.21}$$

where $\Pi_c(x|e)$ *is computed using Algorithm 4 and* $\Pi(x|e)$ *is derived from the joint distribution associated to the possibilistic knowledge base of Equation (3.20).*

Proof 3.4. *K_c is the compiled base of K_Σ which is composed of clauses $(\neg x_i \vee \neg u_i \vee A_i)$ where A_i encodes the necessity degree $1 - \Pi(x_i|u_i)$. To compute $\Pi_c(x|e)$, we should, at first, test if $K_c \not\models A_i \vee \neg e$, this means that we cannot deduce $A_i \vee \neg e$. In this case, we keep formulas $\alpha_i = \neg x_i \vee \neg u_i$ by conditioning K_c on $\neg A_i$, i.e., replace $\alpha_i \vee A_i$ by $\alpha_i \vee \bot$, hence α_i. Then, from the conditioned base $K_c|\neg A_i$, we check if $K_c|\neg A_i \models \neg x$. If this is the case, then $\Pi_c(x|e) = 1 - degree(i) = \Pi(x_i|u_i)$, which in its turn corresponds to $\Pi(x, e)$. Thus, $\Pi_c(x|e) = \Pi(x|e)$.* ■

3.5 Conclusion

In this chapter, we have studied compilation-based inference in min-based possibilistic networks. In fact, we proposed a possibilistic adaptation of the standard probabilistic compilation method [30] (detailed in Section 3.2) consisting on encoding the network into a CNF base and compiling this latter to ensure inference in polytime. Then, we developed a new purely possibilistic method based on compiling possibilistic knowledge bases associated with possibilistic networks. The next chapter will deal with improvements and particular cases that can be investigated and exhibited into the possibilistic adaptation.

Chapter 4

Refined CNF encodings of min-based possibilistic networks

4.1 Introduction

As we have pointed out in the previous chapter, compilation-based inference is mainly based on encoding the min-based possibilistic network into a CNF base using a set of propositional variables. In fact, the CNF encoding of the possibilistic adaptation Π-DNNF associates a parameter variable per possibility degree, without taking into consideration any numerical value. In this chapter, we will refine such encoding by dealing with specific values of parameters, namely equal parameters and extreme values, i.e., 0 and 1. The objective behind refining the encoding is to reduce CNF variables and clauses and study the behavior of compiled bases when we exploit various encoding strategies and consequently compare the inference time.

More precisely, we will at first propose two types of encoding strategies. The first one, named *local structure* and used in both probabilistic and possibilistic networks, consists in assigning one propositional variable per equal parameters per possibility table. This encoding strategy does not take into account specific features of possibility theory such as the ordinal nature of uncertainty scale, which motivates us to propose a new encoding strategy, named *possibilistic local structure*. This latter is exclusively useful for min-based possibilistic networks since it exploits the idempotency property of the min operator. Then, we will take advantage of the particularity of binary variables and refine the n-ary encoding of the possibilistic adaptation when $n = 2$ in order to explore compilation-based inference in binary min-based possibilistic networks.

This chapter is organized as follows: Sections 4.2 and 4.3 provide local structure and possibilistic local structure encoding strategies, respectively. In Section 4.4, we will study the particular case of compiling binary networks. Main results of this Chapter are published in [8].

4.2 Local structure

The CNF encoding of the probabilistic inference approach proposed in [30] depends only on the network structure and variables domains. In other terms, if we have two networks having the same structure such that their variables have the same cardinalities, then we will obtain the same CNF encoding. This means that such encoding does not benefit from specific values of parameters. Chavira et al. [23] have improved this CNF encoding using the so-called *local structure*. By local structure, they mean specific values of network parameters, in particular, *logical constraints* corresponding to the extreme values 0 and 1 and *equal parameters*. This encoding strategy should induce a reduction of the size of both CNF encodings and compiled bases and consequently the time spent during the inference process. In this section, we will exploit local structure in the possibilistic inference approach Π-DNNF (see Section 3.3) by emphasizing on novelties of CNF encodings and compiled bases which are contributed by this strategy. The new possibilistic inference approach using local structure is denoted by Π-DNNF$_{LS}$.

4.2.1 CNF encoding

Exploiting local structure by incorporating logical constraints and equal parameters into the CNF encoding of Definition 3.3 has an impact on both CNF variables and clauses as we will detail below:

- *Logical constraints*: correspond to a conditional possibility degree equal to an extreme value, i.e., either 0 or 1. By exploiting such constraints, we can produce a more efficient CNF encoding with a reduced number of variables and clauses.
  - *Parameters equal to* 0: Each parameter variable $\theta_{x_i|u_i}$ equal to 0 can be dropped from the CNF encoding since any model that sets a variable to true and has a zero weight does not contribute to the possibilistic computation. Consequently, clauses (3.9), (3.10) and (3.11) associated to each $\theta_{x_i|u_i}$ equal to 0 should be substituted by a shorter clause involving only indicator variables. Formally:

$$\neg\lambda_{x_i} \vee \neg\lambda_{u_{i1}} \vee \cdots \vee \neg\lambda_{u_{im}} \tag{4.1}$$

– *Parameters equal to* 1: In the probabilistic case [23], the parameter variable $\theta_{x_i|u_i}$ associated to each parameter equal to 1 as well as its clauses can be dropped from the CNF encoding since they do not contribute to the probabilistic computation, even in the extreme case where all probabilities are equal to 1 and 0. In the possibilitic framework, we cannot omit such variables since they are overriding and represent the key of normalization.

- *Equal parameters*: Suppose that we have some equal parameters in a $C\Pi T_i$, then when we exploit them, we can collapse the number of generated propositional variables. The key idea is then to use the same propositional variable to represent equal parameters pertaining to the same conditional possibility table $C\Pi T_i$. The encoding strategy used in such case is *one variable per equal parameters per* $C\Pi T_i$. However, encoding the network using this strategy can involve in an inconsistent theory as illustrates Example 4.1.

  Example 4.1. *Let us consider* $\Pi(a_2|b_1)$ *and* $\Pi(a_1|b_2)$ *two equal possibility degrees pertaining to the same* $C\Pi T_i$*, then by associating the same parameter variable* θ *for both of them, we will obtain these clauses using Equations (3.10) and (3.11):* $\theta \rightarrow \lambda_{c_1}$ *and* $\theta \rightarrow \lambda_{c_2}$*. This means that if* θ *is set to* $\top$*, then* λ_{c_1} *and* λ_{c_2} *are also set to* $\top$*, which is inconsistent since* λ_{c_1} *and* λ_{c_2} *cannot be both true in the same model (as mutual exclusive clauses reveals).*

  In order to avoid this problem, we should move from a logical equivalence $\Leftrightarrow$ to a logical implication $\Rightarrow$ by dropping clauses (3.10) and (3.11) from the encoding in the case of equal parameters per $C\Pi T_i$.

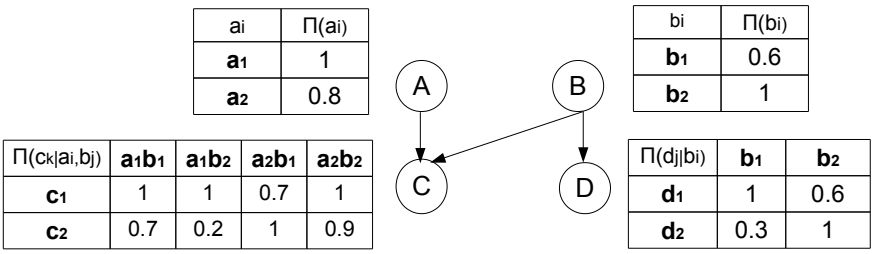
The new set of parameter variables associated to a min-based possibilistic network ΠG_{min} when we consider *local structure* is as follows:

- $\forall X_i \in V$, $\forall\ \Pi(x_i|u_i)$, we associate a *parameter variable*:

$$\begin{cases} \theta_j & \text{if } occ(\Pi(x_i|u_i), C\Pi T_i) > 1 \\ \\ \theta_{x_i|u_i} & \text{if } occ(\Pi(x_i|u_i), C\Pi T_i) = 1 \end{cases} \tag{4.2}$$

  where $occ(\Pi(x_i|u_i), C\Pi T_i)$ is the occurrence number of the parameter $\Pi(x_i|u_i)$ **per conditional possibility table**, i.e., $\mathbf{C\Pi T_i}$.

Example 4.2. *Let us consider the possibilistic network, depicted by Figure 4.1, containing four binary variables A, B, C and D. Using local structure, parameter variables associated to each possibility table* $C\Pi T_i$ *are represented in Table 4.1.*



ai	Π(ai)
a1	1
a2	0.8

bi	Π(bi)
b1	0.6
b2	1

Π(ck|ai,bj)	a1b1	a1b2	a2b1	a2b2
c1	1	1	0.7	1
c2	0.7	0.2	1	0.9

Π(dj|bi)	b1	b2
d1	1	0.6
d2	0.3	1

Figure 4.1: A possibilistic network of 4 nodes

Definition 4.1 outlines the CNF encoding of ΠG_{min} using *local structure*, denoted by C_{min}^{LS}.

Definition 4.1. *Using the set of instance indicators and parameter variables of Equation (4.2), the CNF encoding C_{min}^{LS} contains:*

- ***Mutual exclusive clauses**: $\forall X_i \in V$, we have:*

$$\lambda_{x_{i1}} \vee \lambda_{x_{i2}} \vee \cdots \vee \lambda_{x_{in}} \tag{4.3}$$

$$\neg\lambda_{x_{ij}} \vee \neg\lambda_{x_{ik}}, j \neq k \tag{4.4}$$

- ***Parameter clauses**: $\forall X_i \in V$:*
  - $\forall$ $\Pi(x_i|u_i) = 0$, *we have:*

$$\neg\lambda_{x_i} \vee \neg\lambda_{u_{i1}} \vee \cdots \vee \neg\lambda_{u_{im}} \tag{4.5}$$

  - $\forall$ $\theta_{x_i|u_i}$, *we have:*

$$\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \theta_{x_i|u_i} \tag{4.6}$$

$$\theta_{x_i|u_i} \rightarrow \lambda_{x_i} \tag{4.7}$$

$$\theta_{x_i|u_i} \rightarrow \lambda_{u_{i1}}, \cdots, \theta_{x_i|u_i} \rightarrow \lambda_{u_{im}} \tag{4.8}$$

  - $\forall$ θ_j, *we have:*

$$\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \theta_j \tag{4.9}$$

From a possibilistic point of view, we can say that C_{min}^{LS} recovers the min-based joint possibility distribution.

Variables	Possibility degrees	Parameter variables
A	$\Pi(a_1) = 1$	θ_{a_1}
	$\Pi(a_2) = 0.8$	θ_{a_2}
B	$\Pi(b_1) = 0.6$	θ_{b_1}
	$\Pi(b_2) = 1$	θ_{b_2}
D	$\Pi(d_1\|b_1) = 1$	θ_1
	$\Pi(d_1\|b_2) = 0.6$	$\theta_{d_1\|b_2}$
	$\Pi(d_2\|b_1) = 0.3$	$\theta_{d_2\|b_1}$
	$\Pi(d_2\|b_2) = 1$	θ_1
C	$\Pi(c_1\|a_1, b_1) = 1$	θ_2
	$\Pi(c_1\|a_1, b_2) = 1$	θ_2
	$\Pi(c_1\|a_2, b_1) = 0.7$	θ_3
	$\Pi(c_1\|a_2, b_2) = 1$	θ_2
	$\Pi(c_2\|a_1, b_1) = 0.7$	θ_3
	$\Pi(c_2\|a_1, b_2) = 0.2$	$\theta_{c_2\|a_1,b_2}$
	$\Pi(c_2\|a_2, b_1) = 1$	θ_2
	$\Pi(c_2\|a_2, b_2) = 0.9$	$\theta_{c_2\|a_2,b_2}$

Table 4.1: Parameter variables using local structure

Proposition 4.1. *Let C^{LS}_{min} be the CNF encoding of a ΠG_{min} using Definition 4.1. Let ω be an interpretation from Ω and λ be the conjunction of indicator variables λ_{x_i} related to ω (i.e., $\lambda \equiv \wedge_{x_i \in \omega}\ \lambda_{x_i}$).*
Let us consider $k_{LS}(\lambda)$ be the result of conditioning of C^{LS}_{min} on λ using Equation (3.12). Then,

$$k_{LS}(\lambda) \equiv (\underset{(x_i,u_i)\in\omega}{\wedge} \theta_{x_i|u_i}) \bigwedge (\underset{(x_i,u_i)\in\omega}{\wedge} \theta_j) \bigwedge (\underset{(x_i,u_i)\notin\omega}{\wedge} \neg\theta_{x_i|u_i}) \tag{4.10}$$

where θ_j encodes equal possibility degrees $\Pi(x_i|u_i)$ per $C\Pi T_i$.

After setting $\neg\theta_{x_i|u_i}$ to $\top$, $k_{LS}(\lambda)$ is equivalent to:

$$k_{LS}(\lambda) \equiv (\underset{(x_i,u_i)\in\omega}{\wedge} \theta_{x_i|u_i}) \bigwedge (\underset{(x_i,u_i)\in\omega}{\wedge} \theta_j) \tag{4.11}$$

Proof 4.1. *From C^{LS}_{min}, we deduce $\theta_{x_i|u_i}$, $\neg\theta_{x_i|u_i}$ and θ_j:*

- *If $(x_i, u_i) \in \omega$: From Equations (4.6), (4.7) and (4.8), we deduce $\theta_{x_i|u_i}$ as follows:*

$$\begin{cases} \lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \theta_{x_i|u_i} \\ \theta_{x_i|u_i} \rightarrow \lambda_{x_i} \\ \theta_{x_i|u_i} \rightarrow \lambda_{u_i} \end{cases} \quad \begin{matrix} \lambda_{x_i}=\top \\ \Longrightarrow \\ \lambda_{u_i}=\top \end{matrix} \quad \theta_{x_i|u_i}$$

From Equation (4.9), we deduce θ_j as follows:

$$\{\lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \theta_j \quad \begin{matrix} \lambda_{x_i}=\top \\ \Longrightarrow \\ \lambda_{u_i}=\top \end{matrix} \quad \theta_j$$

- *If $(x_i, u_i) \notin \omega$: From Equations (4.6), (4.7) and (4.8), we deduce $\neg\theta_{x_i|u_i}$ as follows:*

$$\begin{cases} \lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \theta_{x_i|u_i} \\ \theta_{x_i|u_i} \rightarrow \lambda_{x_i} \\ \theta_{x_i|u_i} \rightarrow \lambda_{u_i} \end{cases} \quad \begin{matrix} \lambda_{x_i}=\bot \\ \Longrightarrow \\ \lambda_{u_i}=\bot \end{matrix} \quad \neg\theta_{x_i|u_i}$$

Thus, $k_{LS}(\lambda) \equiv (\wedge_{(x_i,u_i)\in\omega}\ \theta_{x_i|u_i}) \bigwedge (\wedge_{(x_i,u_i)\in\omega}\ \theta_j) \bigwedge (\wedge_{(x_i,u_i)\notin\omega}\ \neg\theta_{x_i|u_i})$. ■

Let $Pk_{LS}(\lambda)$ be the positive part of Proposition 4.1 by replacing each $\neg\theta_{x_i|u_i}$ by $\top$ in $k_{LS}(\lambda)$, i.e.,

$$Pk_{LS}(\lambda) \equiv (\underset{(x_i,u_i)\in\omega}{\wedge} \theta_{x_i|u_i}) \bigwedge (\underset{(x_i,u_i)\in\omega}{\wedge} \theta_j) \tag{4.12}$$

Let $\pi_{C^{LS}_{min}} : \Omega \rightarrow [0,1]$ be the possibility distribution computed from $Pk_{LS}(\lambda)$ by replacing $\wedge$ (resp. $\theta_{x_i|u_i}$ and θ_j) by min (resp. the possibility degrees they encode).

Proposition 4.2. *Let ΠG_{min} be a min-based possibilistic network and C^{LS}_{min} be its CNF encoding using Definition 4.1. Let ω be an interpretation from Ω and $Pk_{LS}(\lambda)$ be its CNF encoding resulting from incorporating ω into C^{LS}_{min} using Equation (4.12). Then,*

$$\forall\omega \in \Omega, \pi_{min}(\omega) = \pi_{C^{LS}_{min}}(\omega) \tag{4.13}$$

$$\forall\phi \subseteq \Omega, \Pi_{min}(\phi) = \Pi_{C^{LS}_{min}}(\phi) \tag{4.14}$$

where π_{min} (resp. $\pi_{C^{LS}_{min}}$) is given by Definition 1.5 (resp. 4.12) and Π_{min} (resp. $\Pi_{C^{LS}_{min}}$) is derived from π_{min} (resp. $\pi_{C^{LS}_{min}}$), respectively.

Proof 4.2. *By setting* $\neg\theta_{x_i|u_i}$ *to* $\top$, $\pi_{C^{LS}_{min}}(\omega)$ *is computed using* $Pk_{LS}(\lambda)$ *as follows:*

$$\begin{aligned}\pi_{C^{LS}_{min}}(\omega) &= \boldsymbol{min}(\min_{(x_i,u_i)\in\omega} \theta_{x_i|u_i}; \min_{(x_i,u_i)\in\omega} \theta_j)\\ &= \min_{(x_i,u_i)\in\omega} \Pi(x_i|u_i)\\ &= \pi_{min}(\omega).\end{aligned}$$

Thus, Equation (4.13) is established.

This result is relative to an interpretation ω*, to generalize it to an event* $\phi \subseteq \Omega$*, we obtain:*

$\max_{\omega\models\phi} \pi_{min}(\omega) = \max_{\omega\models\phi} \pi_{C^{LS}_{min}}(\omega)$,

Thus, $\Pi_{min}(\phi) = \Pi_{C^{LS}_{min}}(\phi)$. ■

Let us illustrate Proposition 4.2 by the following example:

Example 4.3. *Considering the min-based possibilistic network* ΠG_{min} *of Figure 1.2, the interpretation* $\omega = \{a_2, b_1\}$ *and its logical counterpart* $\lambda \equiv \lambda_{a_2} \wedge \lambda_{b_1}$*. Then, conditioning* C^{LS}_{min} *on* λ *using Equation (4.10) gives us* $k_{LS}(\lambda) = (\theta_{a_2} \wedge \theta_2 \wedge \neg\theta_{a_1})$*. The positive part of* $k(\lambda)$ *is equivalent to* $Pk_{LS}(\lambda) \equiv (\theta_{a_2} \wedge \theta_2)$ *using Equation (4.12).*

We can now compute $\pi_{C^{LS}_{min}}(\omega)$ *from* $Pk_{LS}(\lambda)$ *as follows:* $\pi_{C^{LS}_{min}}(\omega) = min(0.4, 0.8) = 0.4 = \pi_{min}(\omega)$ *(row 3 of Table 1.6).*

4.2.2 Compiled base

Once the possibilistic network is encoded using local structure, the resulting CNF encoding is then transformed into a compiled base CB, which is afterwards mapped into a numerical representation as we have pinpointed in section 3.3.3. The *min-max circuit* arising from the mapping step is denoted by CB^{LS}_{minmax} and defined as follows:

Definition 4.2. *A min-max circuit with local structure is a valued sentence where* $\wedge$ *and* $\vee$ *are substituted by min and max, respectively and each propositional variable, either* $\theta_{\mathbf{x_i}|\mathbf{u_i}}$ *or* $\theta_{\mathbf{j}}$*, is replaced by the possibility degree it encodes. Moreover, the value 1 or 0 is associated to each instance indicator* λ_{x_i} *depending on its truth value.*

4.3 Possibilistic local structure

It is well known that the qualitative interpretation of the possibilistic scale that involves the use of the min and max operators is an important property in the possibility theory framework. In this interpretation, the joint possibility distribution can be computed by considering each parameter value only a once since the min operator is *idempotent*, i.e., $min(a, a) = a$. This means that redundancy is not prominent when we deal with a qualitative setting. The first proposed encoding strategy does not take into account specific features of possibility theory and more precisely, the ordinal nature of uncertainty scale. In this section, we propose a new encoding strategy of min-based possibilistic networks, taking advantage of the idempotency property of the min operator by associating a unique propositional variable per equal parameters per all conditional possibility tables. We show that this new encoding strategy, that we call *possibilistic local structure*, reduces the number of propositional variables and clauses required for encoding a min-based possibilistic network since it handles equal parameters from a global point of view.

4.3.1 CNF encoding

The CNF encoding of a min-based possibilistic network ΠG_{min} using *possibilistic local structure* requires a new set of parameter variables associated to possibility degrees of all conditional possibility tables $C\Pi T$. Formally,

- $\forall X_i \in V$, $\forall\ \Pi(x_i|u_i)$, we associate a *parameter variable*:

$$\begin{cases} \Pi\theta_j & \text{if } occ(\Pi(x_i|u_i), C\Pi T) > 1 \\ \\ \Pi\theta_{x_i|u_i} & \text{if } occ(\Pi(x_i|u_i), C\Pi T) = 1 \end{cases} \tag{4.15}$$

  where $occ(\Pi(x_i|u_i), C\Pi T)$ is the occurrence number of the parameter $\Pi(x_i|u_i)$ per **all conditional possibility tables**, i.e., **CΠT**.

Example 4.4. *Let us consider the possibilistic network of Figure 4.1. Then, parameter variables associated to all possibility tables using possibilistic local structure are represented in Table 4.2.*

As we have pointed out in Section 4.2, instance indicators and parameter variables can be joined using either a logical equivalence $\Leftrightarrow$ or a logical implication $\Rightarrow$. Using these connectors and parameter variables arising from using possibilistic local structure, we will propose two variants of CNF encodings, namely an encoding with left-side clauses and an encoding without left-side clauses.

Variables	Possibility degrees	Parameter variables
A	$\Pi(a_1) = 1$	$\Pi\theta_1$
	$\Pi(a_2) = 0.8$	$\Pi\theta_{a_2}$
B	$\Pi(b_1) = 0.6$	$\Pi\theta_2$
	$\Pi(b_2) = 1$	$\Pi\theta_1$
D	$\Pi(d_1\|b_1) = 1$	$\Pi\theta_1$
	$\Pi(d_1\|b_2) = 0.6$	$\Pi\theta_2$
	$\Pi(d_2\|b_1) = 0.3$	$\Pi\theta_{d_2\|b_1}$
	$\Pi(d_2\|b_2) = 1$	$\Pi\theta_1$
C	$\Pi(c_1\|a_1, b_1) = 1$	$\Pi\theta_1$
	$\Pi(c_1\|a_1, b_2) = 1$	$\Pi\theta_1$
	$\Pi(c_1\|a_2, b_1) = 0.7$	$\Pi\theta_3$
	$\Pi(c_1\|a_2, b_2) = 1$	$\Pi\theta_1$
	$\Pi(c_2\|a_1, b_1) = 0.7$	$\Pi\theta_3$
	$\Pi(c_2\|a_1, b_2) = 0.2$	$\Pi\theta_7$
	$\Pi(c_2\|a_2, b_1) = 1$	$\Pi\theta_{c_2\|a_2,b_1}$
	$\Pi(c_2\|a_2, b_2) = 0.9$	$\Pi\theta_{c_2\|a_2,b_2}$

Table 4.2: Parameter variables using possibilistic local structure

Encoding with left-side clauses

This variant of encoding deals with a logical equivalence $\Leftrightarrow$ (both of right-side clause and left-side clauses) and a logical implication $\Rightarrow$ (only the right-side clause) depending on the occurrence number of the possibility degree per $C\Pi T$. Interestingly enough, two cases should be highlighted:

- *Parameters appearing only a once per* $C\Pi T$: should be encoded using both the right-side clause and the left-side clauses. This refers to apply logical equivalence $\Leftrightarrow$.
- *Parameters appearing several times per* $C\Pi T$: should be encoded using only the right-side clause since left-side clauses can evoke an inconsistent theory. This means that we deal with a simple logical implication $\Rightarrow$.

Definition 4.3 outlines the CNF encoding of ΠG_{min} using *possibilistic local structure* and left-side clauses, and denoted by $C^{PLS}_{min_l}$. We denote the inference approach using $C^{PLS}_{min_l}$ by Π-DNNF$^l_{PLS}$.

Definition 4.3. *Using the set of instance indicators and parameter variables of Equation (4.15), the CNF encoding* $C^{PLS}_{min_l}$ *contains:*

- ***Mutual exclusive clauses***: $\forall X_i \in V$, *we have:*

$$\lambda_{x_{i1}} \vee \lambda_{x_{i2}} \vee \cdots \vee \lambda_{x_{in}} \tag{4.16}$$

$$\neg\lambda_{x_{ij}} \vee \neg\lambda_{x_{ik}}, j \neq k \tag{4.17}$$

- ***Parameter clauses***: $\forall X_i \in V$:
  - $\forall\ \Pi(x_i|u_i) = 0$, *we have:*

$$\neg\lambda_{x_i} \vee \neg\lambda_{u_{i1}} \vee \cdots \vee \neg\lambda_{u_{im}} \tag{4.18}$$

  - $\forall\ \Pi\theta_{x_i|u_i}$, *we have:*

$$\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \Pi\theta_{x_i|u_i} \tag{4.19}$$

$$\Pi\theta_{x_i|u_i} \rightarrow \lambda_{x_i} \tag{4.20}$$

$$\Pi\theta_{x_i|u_i} \rightarrow \lambda_{u_{i1}}, \cdots, \Pi\theta_{x_i|u_i} \rightarrow \lambda_{u_{im}} \tag{4.21}$$

  - $\forall\ \Pi\theta_j$, *we have:*

$$\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \Pi\theta_j \tag{4.22}$$

Proposition 4.3. *Let $C^{PLS}_{min_l}$ be the CNF encoding of a ΠG_{min} using Definition 4.3. Let ω be an interpretation from Ω and λ be the conjunction of indicator variables λ_{x_i} related to ω (i.e., $\lambda \equiv \wedge_{x_i \in \omega}\ \lambda_{x_i}$).*
Let $k_{PLS_l}(\lambda)$ be the conditioning result of $C^{PLS}_{min_l}$ on λ using Equation (3.12). Then,

$$k^l_{PLS}(\lambda) \equiv (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_{x_i|u_i}) \bigwedge (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_j) \bigwedge (\underset{(x_i,u_i)\notin\omega}{\wedge} \neg\Pi\theta_{x_i|u_i}) \tag{4.23}$$

where $\Pi\theta_j$ encodes equal possibility degrees $\Pi(x_i|u_i)$ per $C\Pi T$.

After setting $\neg\Pi\theta_{x_i|u_i}$ to $\top$, $k^l_{PLS}(\lambda)$ is equivalent to:

$$k^l_{PLS}(\lambda) \equiv (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_{x_i|u_i}) \bigwedge (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_j) \tag{4.24}$$

Proof 4.3. *From $C^{PLS}_{min_l}$, we deduce $\Pi\theta_{x_i|u_i}$, $\Pi\theta_j$ and $\neg\Pi\theta_{x_i|u_i}$:*

- *If $(x_i, u_i) \in \omega$: From Equations (4.19), (4.20) and (4.21), we deduce $\Pi\theta_{x_i|u_i}$ as follows:*

$$\begin{cases} \lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \Pi\theta_{x_i|u_i} \\ \Pi\theta_{x_i|u_i} \rightarrow \lambda_{x_i} \\ \Pi\theta_{x_i|u_i} \rightarrow \lambda_{u_i} \end{cases} \quad \begin{matrix} \lambda_{x_i}=\top \\ \Longrightarrow \\ \lambda_{u_i}=\top \end{matrix} \quad \Pi\theta_{x_i|u_i}$$

From Equation (4.22), we deduce $\Pi\theta_j$ as follows:

$$\left\{ \lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \Pi\theta_j \right. \quad \begin{matrix} \lambda_{x_i}=\top \\ \Longrightarrow \\ \lambda_{u_i}=\top \end{matrix} \quad \Pi\theta_j$$

- *If $(x_i, u_i) \notin \omega$: From Equations (4.19), (4.20) and (4.21), we deduce $\neg\Pi\theta_{x_i|u_i}$ as follows:*

$$\begin{cases} \lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \Pi\theta_{x_i|u_i} \\ \Pi\theta_{x_i|u_i} \rightarrow \lambda_{x_i} \\ \Pi\theta_{x_i|u_i} \rightarrow \lambda_{u_i} \end{cases} \quad \begin{matrix} \lambda_{x_i}=\bot \\ \Longrightarrow \\ \lambda_{u_i}=\bot \end{matrix} \quad \neg\Pi\theta_{x_i|u_i}$$

Thus, $k^l_{PLS}(\lambda) \equiv (\bigwedge\limits_{(x_i,u_i)\in\omega} \Pi\theta_{x_i|u_i}) \bigwedge (\bigwedge\limits_{(x_i,u_i)\in\omega} \Pi\theta_j) \bigwedge (\bigwedge\limits_{(x_i,u_i)\notin\omega} \neg\Pi\theta_{x_i|u_i})$.

■

Let $Pk^l_{PLS}(\lambda)$ be the positive part of Proposition 4.3 by replacing each $\neg\Pi\theta_{x_i|u_i}$ by $\top$ in $k^l_{PLS}(\lambda)$, i.e.,

$$Pk^l_{PLS}(\lambda) \equiv (\bigwedge_{(x_i,u_i)\in\omega} \Pi\theta_{x_i|u_i}) \bigwedge (\bigwedge_{(x_i,u_i)\in\omega} \Pi\theta_j) \tag{4.25}$$

Let $\pi_{C^{PLS}_{min_l}} : \Omega \rightarrow [0, 1]$ be the possibility distribution computed from $Pk^l_{PLS}(\lambda)$ by replacing $\wedge$ (resp. $\Pi\theta_{x_i|u_i}$ and $\Pi\theta_j$) by min (resp. the possibility degrees they encode).

Proposition 4.4. *Let ΠG_{min} be a min-based possibilistic network and $C^{PLS}_{min_l}$ its CNF encoding using Definition 4.3. Let ω be an interpretation from Ω and $Pk^l_{PLS}(\lambda)$ be its CNF encoding resulting from incorporating ω into $C^{PLS}_{min_l}$ using Equation (4.25). Then,*

$$\forall\omega \in \Omega, \pi_{min}(\omega) = \pi_{C^{PLS}_{min_l}}(\omega) \tag{4.26}$$

$$\forall\phi \subseteq \Omega, \Pi_{min}(\phi) = \Pi_{C^{PLS}_{min_l}}(\phi) \tag{4.27}$$

where π_{min} *(resp.* $\pi_{C^{PLS}_{min_l}}$*) is given by Definition 1.5 (resp. 4.25) and* Π_{min} *(resp.* $\Pi_{C^{PLS}_{min_l}}$*) is derived from* π_{min} *(resp.* $\pi_{C^{PLS}_{min_l}}$*), respectively.*

Proof 4.4. *By setting* $\neg\Pi\theta_{x_i|u_i}$ *to* $\top$, $\pi_{C^{PLS}_{min_l}}(\omega)$ *is computed using* $Pk^{l}_{PLS}(\lambda)$ *as follows:*

$$\begin{aligned}\pi_{C^{PLS}_{min_l}}(\omega) &= \boldsymbol{min}(\min_{(x_i,u_i)\in\omega} \Pi\theta_{x_i|u_i}; \min_{(x_i,u_i)\in\omega} \Pi\theta_j)\\ &= \min_{(x_i,u_i)\in\omega} \Pi(x_i|u_i)\\ &= \pi_{min}(\omega).\end{aligned}$$

Thus, Equation (4.26) is established.

This result is relative to an interpretation ω*, to generalize it to an event* $\phi \subseteq \Omega$*, we obtain:*

$\max_{\omega\models\phi} \pi_{min}(\omega) = \max_{\omega\models\phi} \pi_{C^{PLS}_{min_l}}(\omega)$,

Thus, $\Pi_{min}(\phi) = \Pi_{C^{PLS}_{min_l}}(\phi)$. ■

Proposition 4.4 will be illustrated by Example 4.5.

Encoding without left-side clauses

Encoding a min-based possibilistic network using possibilistic local structure and without left-side clauses refers to associating a parameter variable per equal parameters per **CΠT** and encoding each parameter, either appearing a once or several times per $C\Pi T$, using only the right-side clause. In other terms, we only deal with the logical implication $\Rightarrow$ since this connector suffices to join instance indicators and parameter variables. The CNF encoding of a ΠG_{min} using *possibilistic local structure* and without left-side clauses (see Definition 4.4) and its associated inference approach are denoted by C^{PLS}_{min} and Π-DNNF$_{PLS}$, respectively.

Definition 4.4. *Using the set of instance indicators and parameter variables of Equation (4.15), the CNF encoding* C^{PLS}_{min} *contains:*

- ***Mutual exclusive clauses**: $\forall X_i \in V$, we have:*

$$\lambda_{x_{i1}} \vee \lambda_{x_{i2}} \vee \cdots \vee \lambda_{x_{in}} \tag{4.28}$$

$$\neg\lambda_{x_{ij}} \vee \neg\lambda_{x_{ik}}, j \neq k \tag{4.29}$$

- ***Parameter clauses**: $\forall X_i \in V$:*

– $\forall$ $\Pi(x_i|u_i) = 0$, *we have:*

$$\neg\lambda_{x_i} \vee \neg\lambda_{u_{i1}} \vee \cdots \vee \neg\lambda_{u_{im}} \tag{4.30}$$

– $\forall$ $\Pi\theta_{x_i|u_i}$, *we have:*

$$\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \Pi\theta_{x_i|u_i} \tag{4.31}$$

– $\forall$ $\Pi\theta_j$, *we have:*

$$\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \Pi\theta_j \tag{4.32}$$

Proposition 4.5. *Let C^{PLS}_{min} be the CNF encoding of a ΠG_{min} using Definition 4.4. Let ω be an interpretation from Ω and λ be the conjunction of indicator variables λ_{x_i} related to ω (i.e., $\lambda \equiv \wedge_{x_i \in \omega} \lambda_{x_i}$).*
Let us consider $k_{PLS}(\lambda)$ be the result of conditioning of C^{PLS}_{min} on λ using Equation (3.12). Then,

$$k_{PLS}(\lambda) \equiv (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_{x_i|u_i}) \bigwedge (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_j) \tag{4.33}$$

where $\Pi\theta_j$ encodes equal possibility degrees $\Pi(x_i|u_i)$ within $C\Pi T$.

Proof 4.5. *From C^{PLS}_{min}, we deduce $\Pi\theta_{x_i|u_i}$ and $\Pi\theta_j$ when $(x_i, u_i) \in \omega$:*

- *From Equation (4.31), we deduce $\Pi\theta_{x_i|u_i}$ as follows:*

$$\Big\{ \lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \Pi\theta_{x_i|u_i} \qquad \overset{\lambda_{x_i}=\top}{\underset{\lambda_{u_i}=\top}{\Longrightarrow}} \qquad \Pi\theta_{x_i|u_i}$$

- *From Equation (4.32), we deduce $\Pi\theta_j$ as follows:*

$$\Big\{ \lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \Pi\theta_j \qquad \overset{\lambda_{x_i}=\top}{\underset{\lambda_{u_i}=\top}{\Longrightarrow}} \qquad \Pi\theta_j$$

Thus, $k_{PLS}(\lambda) \equiv (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_{x_i|u_i}) \bigwedge (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_j)$. ■

Let $Pk_{PLS}(\lambda)$ the result of Proposition 4.5, i.e.,

$$Pk_{PLS}(\lambda) \equiv (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_{x_i|u_i}) \bigwedge (\underset{(x_i,u_i)\in\omega}{\wedge} \Pi\theta_j) \tag{4.34}$$

Let $\pi_{C^{PLS}_{min}} : \Omega \rightarrow [0, 1]$ be the possibility distribution computed from $Pk_{PLS}(\lambda)$ by replacing $\wedge$ (resp. $\Pi\theta_{x_i|u_i}$ and $\Pi\theta_j$) by min (resp. the possibility degrees they encode).

Proposition 4.6. *Let ΠG_{min} be a min-based possibilistic network and C_{min}^{PLS} be its CNF encoding using Definition 4.4. Let ω be an interpretation from Ω and $Pk_{PLS}(\lambda)$ be its CNF encoding resulting from incorporating ω into C_{min}^{PLS} using Equation (4.34). Then,*

$$\forall \omega \in \Omega, \pi_{min}(\omega) = \pi_{C_{min}^{PLS}}(\omega) \tag{4.35}$$

$$\forall \phi \subseteq \Omega, \Pi_{min}(\phi) = \Pi_{C_{min}^{PLS}}(\phi) \tag{4.36}$$

where π_{min} (resp. $\pi_{C_{min}^{PLS}}$) is given by Definition 1.5 (resp. Equation (4.34)) and Π_{min} (resp. $\Pi_{C_{min}^{PLS}}$) is derived from π_{min} (resp. $\pi_{C_{min}^{PLS}}$), respectively.

Proof 4.6. *$\pi_{C^{PLS}_{min}}(\omega)$ is computed using $Pk_{PLS}(\lambda)$ as follows:*

$$\pi_{C^{PLS}_{min}}(\omega) = \boldsymbol{min}(\min_{(x_i,u_i)\in\omega} \Pi\theta_{x_i|u_i}; \min_{(x_i,u_i)\in\omega} \Pi\theta_j)$$
$$= \min_{(x_i,u_i)\in\omega} \Pi(x_i|u_i)$$
$$= \pi_{min}(\omega).$$

Thus, Equation (4.35) is established.

This result is relative to an interpretation ω, to generalize it to an event $\phi \subseteq \Omega$, we obtain:

$\max_{\omega\models\phi} \pi_{min}(\omega) = \max_{\omega\models\phi} \pi_{C^{PLS}_{min}}(\omega)$,

Thus, $\Pi_{min}(\phi) = \Pi_{C^{PLS}_{min}}(\phi)$. ■

The following example illustrates both of Propositions 4.4 and 4.6.

Example 4.5. *Let us consider the min-based possibilistic network ΠG_{min} of Figure 1.2. Let $\omega = \{a_1, b_2\}$ be an interpretation from Ω and $\lambda \equiv \lambda_{a_1} \wedge \lambda_{b_2}$. Then, the conditioning result of $C^{PLS}_{min_l}$ and C^{PLS}_{min} on λ is the following:*

- $\mathbf{C^{PLS}_{min_l}}$*: using Equation (4.23), $k^l_{PLS}(\lambda)$ is equivalent to $(\Pi\theta_{a_1} \wedge \neg\Pi\theta_{a_2} \wedge \Pi\theta_2)$. The positive part of k^l_{PLS} using Equation (4.25) corresponds to $Pk^l_{PLS}(\lambda) = (\Pi\theta_{a_1} \wedge \Pi\theta_2)$.*
- $\mathbf{C^{PLS}_{min}}$*: using Equation (4.34), $Pk_{PLS}(\lambda) = (\Pi\theta_{a_1} \wedge \Pi\theta_2)$. The negative literal $\neg\Pi\theta_{a_2}$ is not within deduced variables since the left-side clause $\neg\Pi\theta_{a_2} \vee \lambda_{a_2}$ is not included in C^{PLS}_{min}.*

This means that $Pk^l_{PLS}(\lambda)$ requires an additional step since deduced variables can be negative due to left-side clauses.

The possibility degrees $\pi_{C^{PLS}_{min_l}}(\omega)$ and $\pi_{C^{PLS}_{min}}(\omega)$ can be computed from $Pk^l_{PLS}(\lambda)$ and $Pk_{PLS}(\lambda)$ in the same spirit as follows: $\pi_{C^{PLS}_{min_l}}(\omega) = \pi_{C^{PLS}_{min}}(\omega) = min(1, 0.8) = 0.8 = \pi_{min}(\omega)$ (row 2 of Table 1.6).

4.3.2 Compiled base

The compiled base CB arising from compiling the CNF encoding associated to a min-based possibilistic network using *possibilistic local structure* should be used in the inference phase to efficiently compute the effect of an evidence e on an instance of interest x, i.e., $\Pi_c(x|e)$. When possibilistic local structure

is taken into consideration, the *min-max circuit*, denoted by $CB^{PLS}_{minmax_l}$ if we deal with left-side clauses and CB^{PLS}_{minmax} if only right-side clauses are considered, is defined as follows:

Definition 4.5. *A min-max circuit with possibilistic local structure, either $CB^{PLS}_{minmax_l}$ or CB^{PLS}_{minmax}, is a valued sentence where $\wedge$ and $\vee$ are substituted by min and max, respectively. Each parameter variable, either $\mathbf{\Pi\theta_{x_i|u_i}}$ or $\mathbf{\Pi\theta_j}$, is replaced by the possibility degree it encodes. Also, each instance indicator λ_{x_i} is set to 1 or 0 depending on its truth value.*

4.4 Illustrative example of encoding strategies

In the previous sections, we have proposed two strategies, namely local structure and possibilistic local structure to encode a min-based possibilistic network. In this section, we will apply them to the network ΠG_{min} of Figure 1.2 and compare the impact of each strategy on both CNF parameters and compiled bases parameters.

Example 4.6. *Let us consider the min-based possibilistic network ΠG_{min} of Figure 1.2. Before encoding the network, we should at first associate the set of instance indicators to instances of variables and parameter variables to possibility degrees as shown in Table 4.3 and Table 4.4, respectively. From Table 4.4, we can deduce that the number of parameter variables is equal to 4 when we exploit local structure, while it corresponds to 3 in the case of possibilistic local structure. This reduction of variables is obvious since the possibility degree 1, which appears in both tables of A and B, is encoded using the same parameter variable, namely $\Pi\theta_1$.*

We should then encode ΠG_{min} using Definitions 4.1, 4.3 and 4.4 as shown, respectively in columns 2,3 and 4 of Table 4.5. We can pinpoint that the number of parameter clauses is decreasing from one strategy to another. In fact, it is equal to 8, 7 and 6 in the case of C^{LS}_{min}, $C^{PLS}_{min_l}$ and C^{PLS}_{min}, respectively.

Instances	$\mathbf{C^{LS}_{min}, C^{PLS}_{min_l}, C^{PLS}_{min}}$
a_1	λ_{a_1}
a_2	λ_{a_2}
b_1	λ_{b_1}
b_2	λ_{b_2}

Table 4.3: Instance indicators used in C^{LS}_{min}, $C^{PLS}_{min_l}$ and C^{PLS}_{min}

Let us now compile the CNF encodings of Table 4.5 and map the resulting compiled bases into min-max circuits using Definitions 4.2 and 4.5. The

Variables	**Possibility degrees**	$\mathbf{C^{LS}_{min}}$	$\mathbf{C^{PLS}_{min_l}}$, $\mathbf{C^{PLS}_{min}}$
A	$\Pi(a_1) = 1$	θ_{a_1}	$\Pi\theta_1$
	$\Pi(a_2) = 0.4$	θ_{a_2}	$\Pi\theta_{a_2}$
B	$\Pi(b_1\|a_1) = 1$	θ_1	$\Pi\theta_1$
	$\Pi(b_1\|a_2) = 0.8$	θ_2	$\Pi\theta_2$
	$\Pi(b_2\|a_1) = 0.8$	θ_2	$\Pi\theta_2$
	$\Pi(b_2\|a_2) = 1$	θ_1	$\Pi\theta_1$

Table 4.4: Parameter variables used in C^{LS}_{min}, $C^{PLS}_{min_l}$ and C^{PLS}_{min}

Variables	**Mutual exclusive clauses**		
A	$(\lambda_{a_1} \vee \lambda_{a_2}) \wedge (\neg\lambda_{a_1} \vee \neg\lambda_{a_2})$		
B	$(\lambda_{b_1} \vee \lambda_{b_2}) \wedge (\neg\lambda_{b_1} \vee \neg\lambda_{b_2})$		
Possibility degrees	**Parameter clauses**		
A	$\mathbf{C^{LS}_{min}}$	$\mathbf{C^{PLS}_{min_l}}$	$\mathbf{C^{PLS}_{min}}$
$\Pi(a_1) = 1$	$(\lambda_{a_1} \rightarrow \theta_{a_1})$ $\wedge(\theta_{a_1} \rightarrow \lambda_{a_1})$	$(\lambda_{a_1} \rightarrow \Pi\theta_1)$	$(\lambda_{a_1} \rightarrow \Pi\theta_1)$
$\Pi(a_2) = 0.4$	$(\lambda_{a_2} \rightarrow \theta_{a_2})$ $\wedge(\theta_{a_2} \rightarrow \lambda_{a_2})$	$(\lambda_{a_2} \rightarrow \Pi\theta_{a_2})$ $\wedge(\Pi\theta_{a_2} \rightarrow \lambda_{a_2})$	$(\lambda_{a_2} \rightarrow \Pi\theta_{a_2})$
B	$\mathbf{C^{LS}_{min}}$	$\mathbf{C^{PLS}_{min_l}}$	$\mathbf{C^{PLS}_{min}}$
$\Pi(b_1\|a_1) = 1$	$(\lambda_{a_1} \wedge \lambda_{b_1} \rightarrow \theta_1)$	$(\lambda_{a_1} \wedge \lambda_{b_1} \rightarrow \Pi\theta_1)$	$(\lambda_{a_1} \wedge \lambda_{b_1} \rightarrow \Pi\theta_1)$
$\Pi(b_2\|a_1) = 0.8$	$(\lambda_{a_1} \wedge \lambda_{b_2} \rightarrow \theta_2)$	$(\lambda_{a_1} \wedge \lambda_{b_2} \rightarrow \Pi\theta_2)$	$(\lambda_{a_1} \wedge \lambda_{b_2} \rightarrow \Pi\theta_2)$
$\Pi(b_1\|a_2) = 0.8$	$(\lambda_{a_1} \wedge \lambda_{b_2} \rightarrow \theta_2)$	$(\lambda_{a_2} \wedge \lambda_{b_1} \rightarrow \Pi\theta_2)$	$(\lambda_{a_2} \wedge \lambda_{b_1} \rightarrow \Pi\theta_2)$
$\Pi(b_2\|a_2) = 1$	$(\lambda_{a_2} \wedge \lambda_{b_2} \rightarrow \theta_1)$	$(\lambda_{a_2} \wedge \lambda_{b_2} \rightarrow \Pi\theta_1)$	$(\lambda_{a_2} \wedge \lambda_{b_2} \rightarrow \Pi\theta_1)$

Table 4.5: The CNF encodings C^{LS}_{min}, $C^{PLS}_{min_l}$ and C^{PLS}_{min}

resulting min-max circuits are represented by sub-figures (a), (b) and (c) of Figure 4.2 when we deal with local structure, possibilistic local structure with left-side clauses and possibilistic local structure without left-side clauses, respectively. An important interpretation from this example is that compiled bases parameters follow the same behavior as those of CNF encodings. Indeed, the number of edges is equal to 22 and 20 in CB^{LS}_{minmax} *and* $CB^{PLS}_{minmax_l}$, CB^{PLS}_{minmax}, *respectively. The question that may arise is the following: Does this behavior remain the same when we deal with networks having a higher number of nodes? This question will be explored in the experimental study of Chapter 7.*

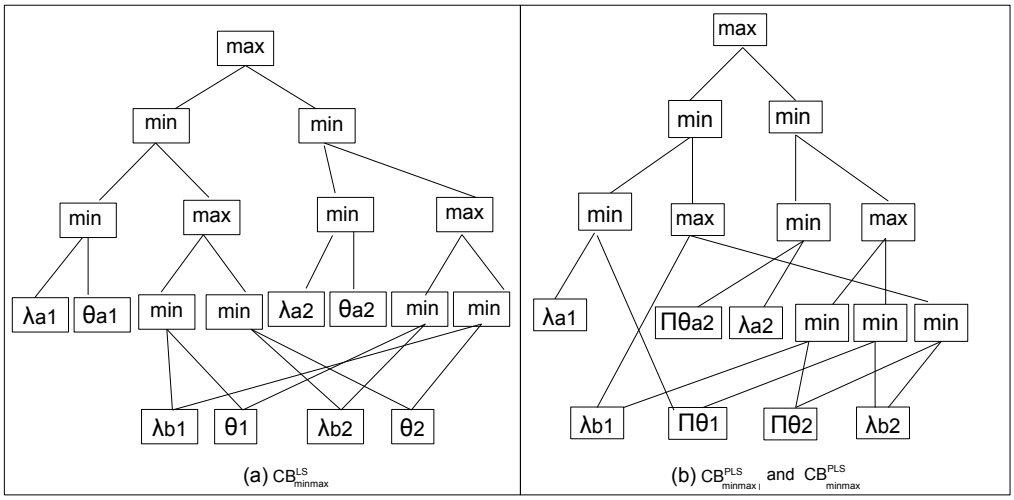


Figure 4.2: CB^{LS}_{minmax}, $CB^{PLS}_{minmax_l}$ and CB^{PLS}_{minmax}

4.5 Particular case of binary networks

The possibilistic method Π-DNNF and its variants using local structure and possibilistic local structure deal with n-ary min-based possibilistic networks while incorporating a set of mutual exclusive clauses stating that exactly one instance indicator λ_{x_i} for each $X_i \in V$ is set to true in each model in the CNF encoding. In the particular case of binary networks, the CNF encoding of Definition 3.3 requires to set n to 2 yielding supplementary propositional variables and clauses since the particularity of binary variables is not taken into consideration. In this section, we propose a new encoding especially dedicated for binary min-based possibilistic networks by emphasizing on novelties of both the CNF encoding and the compiled base which are contributed by this binary aspect. We denote the new binary approach by

Bin-Π-DNNF, which follows the same global reasoning as n-ary approaches, namely: from encoding to inference by going through compilation.

4.5.1 Binary CNF encoding

When the possibilistic network is only composed of binary variables, in this case we can encode both of its instances and parameters using a unique propositional variable. In other terms, we associate to each X_i's instances one instance indicator (i.e., λ_{x_i}) instead of two (i.e., $\lambda_{x_{i1}}$ and $\lambda_{x_{i2}}$) where the positive (resp. negative) instance is represented by λ_{x_i} (resp. $\neg\lambda_{x_i}$). Moreover, each parameter $\Pi(x_i|u_i)$ (resp. $\Pi(\neg x_i|u_i)$) will be encoded using $\theta_{x_i|u_i}$ (resp. $\neg\theta_{x_i|u_i}$). It is worthwhile to point out that by taking advantage of the particularity of binary variables, we *halve the number of instance indicators and network parameters* but also *release the need for mutual exclusive clauses* and some *parameter clauses* as we vindicate later.

The new binary encoding needs two types of propositional variables, namely:

- $\forall X_i \in V$, we associate one *instance indicator* λ_{x_i}, s.t. λ_{x_i} (resp. $\neg\lambda_{x_i}$) encodes the instance x_i (resp. $\neg x_i$).
- $\forall X_i \in V, \forall\, \Pi(X_i|u_i)$ s.t. $u_i = \{u_{i1}, u_{i2}, ..., u_{im}\}$, we associate a *parameter variable* as follows:

$$\begin{cases} \theta_{x_i|u_{i1},u_{i2},...,u_{im}} & \text{if } X_i = x_i \\ \\ \neg\theta_{x_i|u_{i1},u_{i2},...,u_{im}} & \text{if } X_i = \neg x_i \end{cases} \tag{4.37}$$

The new auxiliary encoding specific to binary networks, denoted by C_{bin}, is outlined by Definition 4.6.

Definition 4.6. *Let ΠG_{min} be a binary possibilistic network, then using the new set of binary instance indicators and parameter variables, the binary encoding C_{bin} contains the following clauses for each $X_i \in V$:*

$$\bigwedge \begin{cases} \lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \theta_{x_i|u_{i1},...,u_{im}} \\ \neg\lambda_{x_i} \wedge \lambda_{u_{i1}} \wedge \ldots \wedge \lambda_{u_{im}} \rightarrow \neg\theta_{x_i|u_{i1},...,u_{im}} \\ \quad\vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ \lambda_{x_i} \wedge \neg\lambda_{u_{i1}} \wedge \ldots \wedge \neg\lambda_{u_{im}} \rightarrow \theta_{x_i|u_{\bar{i}1},...,u_{\bar{i}m}} \\ \neg\lambda_{x_i} \wedge \neg\lambda_{u_{i1}} \wedge \ldots \wedge \neg\lambda_{u_{im}} \rightarrow \neg\theta_{x_i|u_{\bar{i}1},...,u_{\bar{i}m}} \end{cases} \tag{4.38}$$

The encoding C_{bin} is in a CNF form where each clause encodes the fact that the possibility degree of $x_i|u_{i1}, u_{i2}, ..., u_{im}$ represented by the propositional formula $literal(\lambda_{x_i}) \wedge literal(\lambda_{u_{i1}}) \wedge \ldots \wedge literal(\lambda_{u_{im}})$ is equal ($\Rightarrow$ in the logical setting) to $\Pi(x_i|u_{i1}, u_{i2}, ..., u_{im})$, which is in its turn represented by the parameter variable $\theta_{x_i|u_{i1},u_{i2},...,u_{im}}$ such that $literal(\lambda_{x_j})$ is expressed by:

$$literal(\lambda_{x_j}) = \begin{cases} \lambda_{x_j} & if \;\; x_j \in \{x_i, u_i\} \\ \neg\lambda_{x_j} & if \;\; \bar{x_j} \in \{x_i, u_i\} \end{cases} \tag{4.39}$$

The same explanation is also valuable for the negative part, i.e., $\Pi(\neg x_i|u_{i1}, u_{i2}, ..., u_{im})$ and $\neg\theta_{x_i|u_{i1},u_{i2},...,u_{im}}$. Let us now compare the n-ary encoding C_{min} of Definition 3.3 and the binary one C_{bin} of Definition 4.6.

By emphasizing on clauses of these two CNF encodings, we can point out that in the binary case, we only resort to clauses (4.38) which represent the binary counterpart of clause (3.9) while omitting left-side clauses and using the set of instance indicators and parameter variables of Equation (4.37). The question that may arise is: *why did we drop left-side clauses of Equations (3.10) and (3.11) in the binary encoding?* In fact, from a logical point of view, the negation of $\theta_{x_i|u_i}$ (i.e., $\neg\theta_{x_i|u_i}$) can concern x_i or u_i or both of x_i and u_i. In other terms, it implies $\theta_{\bar{x_i}|u_i} \vee \theta_{x_i|\bar{u_i}} \vee \theta_{\bar{x_i}|\bar{u_i}}$. For generality reasons, we assume that $\neg\theta_{x_i|u_i}$ only implies $\theta_{\bar{x_i}|u_i}$. Therefore, only the right-side clause should be considered and clauses (3.10) and (3.11) should be excluded since from a logical point of view $\neg\theta_{x_i|u_i}$ implies neither $\neg\lambda_{x_i}$ nor λ_{u_i}.

From a possibilistic point of view, we can say that the binary encoding C_{bin} recovers the min-based joint possibility distribution. Formally:

Proposition 4.7. *Let C_{bin} be the binary encoding of a binary min-based possibilistic network ΠG_{min} using Definition 4.6. Let ω be an interpretation from Ω and λ be the conjunction of $literal(\lambda_{x_i})$ related to ω (i.e., $\lambda \equiv \wedge_{x_i \in \omega} \; literal(\lambda_{x_i})$) s.t.,*

$$literal(\lambda_{x_i}) = \begin{cases} \lambda_{x_i} & if \;\; x_i \in \omega \\ \neg\lambda_{x_i} & if \;\; \neg x_i \in \omega \end{cases} \tag{4.40}$$

Let $Bk(\lambda)$ be the result of conditioning of C_{bin} on λ by setting each $literal(\lambda_{x_i})$ of λ to:

$$\lambda_{x_i} = \begin{cases} \top & if \;\; x_i \in \omega \\ \bot & otherwise \end{cases} \tag{4.41}$$

$$\neg\lambda_{x_i} = \begin{cases} \top & if \;\; \neg x_i \in \omega \\ \bot & otherwise \end{cases} \tag{4.42}$$

Then,

$$Bk(\lambda) \equiv \bigwedge_{(PosNeg(x_i),u_i)\in\omega} literal(\theta_{x_i|u_i})$$

where:

$$PosNeg(x_i) = \begin{cases} x_i & if \ \ x_i \in \omega \\ \neg x_i & if \ \ \neg x_i \in \omega \end{cases} \tag{4.43}$$

and

$$literal(\theta_{x_i|u_i}) = \begin{cases} \theta_{x_i|u_i} & if \ \ PosNeg(x_i) = x_i \\ \neg\theta_{x_i|u_i} & if \ \ PosNeg(x_i) = \neg x_i \end{cases} \tag{4.44}$$

Proof 4.7. *From C_{bin}, we deduce $\theta_{x_i|u_i}$ and $\neg\theta_{x_i|u_i}$:*

- *If $PosNeg(x_i) = x_i$ and $(x_i, u_i) \in \omega$:*
  From Equation (4.38), we deduce $\theta_{x_i|u_i}$ as follows:

$$\Big\{\lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \theta_{x_i|u_i} \qquad \begin{matrix}\lambda_{x_i}=\top \\ \Longrightarrow \\ \lambda_{u_i}=\top\end{matrix} \qquad \theta_{x_i|u_i}$$

- *If $PosNeg(x_i) = \neg x_i$ and $(\neg x_i, u_i) \in \omega$:*
  From Equation (4.38), we deduce $\neg\theta_{x_i|u_i}$ as follows:

$$\Big\{\neg\lambda_{x_i} \wedge \lambda_{u_i} \rightarrow \neg\theta_{x_i|u_i} \qquad \begin{matrix}\lambda_{x_i}=\bot \\ \Longrightarrow \\ \lambda_{u_i}=\top\end{matrix} \qquad \neg\theta_{x_i|u_i}$$

Thus, $Bk(\lambda) \equiv \bigwedge_{(PosNeg(x_i),u_i)\in\omega} literal(\theta_{x_i|u_i})$. ■

Let $Bk(\lambda)$ be the result of conditioning of C_{bin} on λ as shown in Proposition 4.7. Formally:

$$Bk(\lambda) \equiv \bigwedge_{(PosNeg(x_i),u_i)\in\omega} literal(\theta_{x_i|u_i}) \tag{4.45}$$

Let $\pi_{C_{bin}} : \Omega \rightarrow [0, 1]$ be the possibility distribution computed from $Bk(\lambda)$ by replacing $\wedge$ (resp. each $literal(\theta_{x_i|u_i})$) by min (resp. $\Pi(x_i|u_i)$ or $\Pi(\neg x_i|u_i)$).

Proposition 4.8. *Let ΠG_{min} be a binary possibilistic network and C_{bin} be its binary encoding using Definition 4.38. Let $Bk(\lambda)$ be the encoding resulting from conditioning C_{bin} on λ using Equation (4.45). Then,*

$$\forall\omega \in \Omega, \pi_{min}(\omega) = \pi_{C_{bin}}(\omega) \tag{4.46}$$

$$\forall\phi \subseteq \Omega, \Pi_{min}(\phi) = \Pi_{C_{bin}}(\phi) \tag{4.47}$$

where π_{min} (resp. $\pi_{C_{bin}}$) is given by Definition 1.5 (resp. Equation (4.45)) and Π_{min} (resp. $\Pi_{C_{bin}}$) is derived from π_{min} (resp. $\pi_{C_{bin}}$), respectively.

Proof 4.8. *By associating to each* $literal(\theta_{x_i|u_i})$ *the possibility degree* $\Pi(x_i|u_i)$ *or* $\Pi(\neg x_i|u_i)$ *it encodes,* $\pi_{C_{bin}}(\omega)$ *is computed using* $Bk(\lambda)$ *as follows:*

$$\pi_{C_{bin}}(\omega) = \min_{(PosNeg(x_i),u_i)\in\omega} literal(\theta_{x_i|u_i})$$
$$= \min_{(PosNeg(x_i),u_i)\in\omega} \Pi(PosNeg(x_i)|u_i):$$

where

$$\Pi(PosNeg(x_i)|u_i) = \begin{cases} \Pi(x_i|u_i) & if \;\; PosNeg(x_i) = x_i \\ \Pi(\neg x_i|u_i) & if \;\; PosNeg(x_i) = \neg x_i \end{cases}$$

$$= \pi_{min}(\omega).$$

Thus, Equation (4.46) is established.

This result is relative to an interpretation ω*, to generalize it to an event* $\phi \subseteq \Omega$*, we obtain:*

$$\max_{\omega\models\phi} \pi_{min}(\omega) = \max_{\omega\models\phi} \pi_{C_{bin}}(\omega),$$

Thus, $\Pi_{min}(\phi) = \Pi_{C_{bin}}(\phi)$. ■

Example 4.7. *Let us consider the binary possibilistic network* ΠG_{min} *of Figure 1.2. Let* $\omega = \{\neg a, b\}$ *be an interpretation from* Ω *and* $\lambda \equiv literal(\lambda_a) \wedge literal(\lambda_b) \equiv \neg\lambda_a \wedge \lambda_b$*. To compute* $Bk(\lambda)$*, we need to define at first:*

$$\begin{cases} PosNeg(a) = \neg a & since \;\; \neg a \in \omega \\ PosNeg(b) = b & since \;\; b \in \omega \end{cases}$$

Hence:

$$\begin{cases} literal(\theta_a) = \neg\theta_a & since \;\; PosNeg(a) = \neg a \\ literal(\theta_{b|\neg a}) = \theta_{b|\neg a} & since \;\; PosNeg(b) = b \end{cases}$$

Using Equation (4.7), the encoding $Bk(\lambda)$ *is then equivalent to* $literal(\theta_a) \wedge literal(\theta_{b|\neg a}) \equiv \neg\theta_a \wedge \theta_{b|\neg a}$*.*

Now, if we want to compute $\pi_{C_{bin}}(\omega)$ *from* $Bk(\lambda)$*, we should at first substitute each* $\wedge$ *by min. Then, we should assign the appropriate possibility degree for each of* $\neg\theta_a$ *and* $\theta_{b|\neg a}$*. Hence,* $\pi_{C_{bin}}(\omega) = min(0.4, 0.8) = 0.4 = \pi_{min}(\omega)$ *(row 3 of Table 1.6).*

It is important to pinpoint that the binary encoding C_{bin} of Definition 4.6 can be refined using *local structure* or *possibilistic local structure* as shown in what follows:

Local structure in binary encoding

Incorporating local structure into C_{bin} requires to use parameter variables θ_j or $\theta_{x_i|u_i}$ of Equation (4.2). Moreover, each network parameter $\theta_{x_i|u_i}$ or $\neg\theta_{x_i|u_i}$ equal to 0 should be encoded using the following Equation:

$$\neg literal(\lambda_{x_i}) \vee \neg literal(\lambda_{u_{i1}}) \vee \cdots \vee \neg literal(\lambda_{u_{im}}) \tag{4.48}$$

The resulting CNF encoding exploiting local structure and its associated binary inference approach are denoted by C_{bin}^{LS} and Bin-Π-DNNF$_{LS}$, respectively.

Possibilistic local structure in binary encoding

Possibilistic local structure can be also involved into the binary encoding C_{bin}. However, instead of using $\theta_{x_i|u_i}$ and $\neg\theta_{x_i|u_i}$, we should handle parameter variables $\Pi\theta_j$ or $\Pi\theta_{x_i|u_i}$ of Equation (4.15). Furthermore, each zero parameter should be encoded using Equation (4.48).

The new binary CNF encoding and the inference approach using *possibilistic local structure* are denoted by C_{bin}^{PLS} and Bin-Π-DNNF$_{PLS}$, respectively.

4.5.2 Binary compiled base

Once the encoding phase is achieved, the resulting CNF encoding (i.e., C_{bin}) is then compiled into a DNNF base CB. Before transforming it into a numerical representation, we should update instance indicators (λ_{x_i} and $\neg\lambda_{x_i}$) depending on the instance of interest x and the evidence e of the possibility degree $\Pi(x, e)$ as follows:

$$\lambda_{x_i} = \begin{cases} \top & \text{if } x_i \sim e \text{ and } x_i \sim x \\ \bot & \text{otherwise} \end{cases} \tag{4.49}$$

$$\neg\lambda_{x_i} = \begin{cases} \top & \text{if } \neg x_i \sim e \text{ and } \neg x_i \sim x \\ \bot & \text{otherwise} \end{cases} \tag{4.50}$$

After that, we should transform the conditioned binary compiled base into a *binary min-max circuit*, denoted by CB_{bin}, in which $\wedge$ and $\vee$ are substituted by *min* and *max*, respectively, parameter variables $\theta_{\mathbf{x_i|u_i}}$ and $\neg\theta_{\mathbf{x_i|u_i}}$ are replaced by $\Pi(x_i|u_i)$ and $\Pi(\neg x_i|u_i)$, respectively and instance indicators λ_{x_i} and $\neg\lambda_{x_i}$ are set to 1 or 0 depending on their truth values.

When we deal with local structure, the binary min-max circuit, denoted by CB_{bin}^{LS}, is obtained by replacing $\theta_{\mathbf{x_i|u_i}}$ and $\theta_{\mathbf{j}}$ by the possibility degrees they encode.

In the case of possibilistic local structure, we obtain the binary min-max circuit, denoted by CB_{bin}^{PLS}, by substituting propositional variables $\mathbf{\Pi}\theta_{\mathbf{x_i|u_i}}$ and $\mathbf{\Pi}\theta_{\mathbf{j}}$ by their numerical values.

4.6 Illustrative example of binary approaches

This section illustrates the binary approaches and compares CNF parameters and compiled bases parameters with those of the n-ary case where $n = 2$ (see Example 4.6).

Example 4.8. *Let us encode the min-based possibilistic network ΠG_{min} of Figure 1.2 using the set of instance indicators and parameter variables of Table 4.6 and Table 4.7, respectively. The binary CNF encoding is given by Table 4.8. We can notice that the number of instance indicators is halved comparing to those of Example 4.6. Moreover, the number of parameter variables remains the same (equal to 3) in the three encodings (C_{bin}, C_{bin}^{LS} and C_{bin}^{PLS}) but each one has its own parameter variables.*

Instances	$\mathbf{C_{bin}}, \mathbf{C_{bin}^{LS}}, \mathbf{C_{bin}^{PLS}}$
a_1	λ_a
a_2	$\neg\lambda_a$
b_1	λ_b
b_2	$\neg\lambda_b$

Table 4.6: Instance indicators used in C_{bin}, C_{bin}^{LS} and C_{bin}^{PLS}

Variables	**Possibility degrees**	$\mathbf{C_{bin}}$	$\mathbf{C_{bin}^{LS}}$	$\mathbf{C_{bin}^{PLS}}$
A	$\Pi(a_1) = 1$	θ_a	θ_a	$\Pi\theta_1$
	$\Pi(a_2) = 0.4$	$\neg\theta_a$	$\neg\theta_a$	$\Pi\theta_3$
B	$\Pi(b_1\|a_1) = 1$	$\theta_{b\|a}$	θ_1	$\Pi\theta_1$
	$\Pi(b_1\|a_2) = 0.8$	$\theta_{b\|\neg a}$	θ_2	$\Pi\theta_2$
	$\Pi(b_2\|a_1) = 0.8$	$\neg\theta_{b\|a}$	θ_2	$\Pi\theta_2$
	$\Pi(b_2\|a_2) = 1$	$\neg\theta_{b\|\neg a}$	θ_1	$\Pi\theta_1$

Table 4.7: Parameter variables used in C_{bin}, C_{bin}^{LS} and C_{bin}^{PLS}

The compiled bases resulting from compiling the binary CNF encodings are represented by Figure 4.3. The number of edges is equal to 22 in CB_{bin}

A	$\mathbf{C_{bin}}$	$\mathbf{C_{bin}^{LS}}$	$\mathbf{C_{bin}^{PLS}}$
$\Pi(a_1) = 1$	$(\lambda_a \rightarrow \theta_a)$	$(\lambda_a \rightarrow \theta_a)$	$(\lambda_a \rightarrow \Pi\theta_1)$
$\Pi(a_2) = 0.4$	$(\neg\lambda_a \rightarrow \neg\theta_a)$	$(\neg\lambda_a \rightarrow \neg\theta_a)$	$(\neg\lambda_a \rightarrow \Pi\theta_3)$
B	$\mathbf{C_{bin}}$	$\mathbf{C_{bin}^{LS}}$	$\mathbf{C_{bin}^{PLS}}$
$\Pi(b_1\|a_1) = 1$	$(\lambda_a \wedge \lambda_b \rightarrow \theta_{b\|a})$	$(\lambda_a \wedge \lambda_b \rightarrow \theta_1)$	$(\lambda_a \wedge \lambda_b \rightarrow \Pi\theta_1)$
$\Pi(b_2\|a_1) = 0.8$	$(\lambda_a \wedge \neg\lambda_b \rightarrow \neg\theta_{b\|a})$	$(\lambda_a \wedge \neg\lambda_b \rightarrow \theta_2)$	$(\lambda_a \wedge \neg\lambda_b \rightarrow \Pi\theta_2)$
$\Pi(b_1\|a_2) = 0.8$	$(\neg\lambda_a \wedge \lambda_b \rightarrow \theta_{b\|\neg a})$	$(\neg\lambda_a \wedge \lambda_b \rightarrow \theta_2)$	$(\neg\lambda_a \wedge \lambda_b \rightarrow \Pi\theta_2)$
$\Pi(b_2\|a_2) = 1$	$(\neg\lambda_a \wedge \neg\lambda_b \rightarrow \neg\theta_{b\|\neg a})$	$(\neg\lambda_a \wedge \neg\lambda_b \rightarrow \theta_1)$	$(\neg\lambda_a \wedge \neg\lambda_b \rightarrow \Pi\theta_1)$

Table 4.8: The CNF encodings C_{bin}, C_{bin}^{LS} and C_{bin}^{PLS}

and CB_{bin}^{LS}, while it is equal to 20 in CB_{bin}^{PLS}.

By considering the network composed of 4 nodes, depicted by Figure 4.1, the number of edges is equal to 60, 61 and 64 in CB_{bin}, CB_{bin}^{LS} and CB_{bin}^{PLS}, respectively. According to a four-node network, using a parameter variable $\theta_{x_i|u_i}$ and its negation $\neg\theta_{x_i|u_i}$ performs better than exploiting local structure and possibilistic local structure.

We can point out that the behavior of compiled bases do not remain the same when the number of nodes is increased.

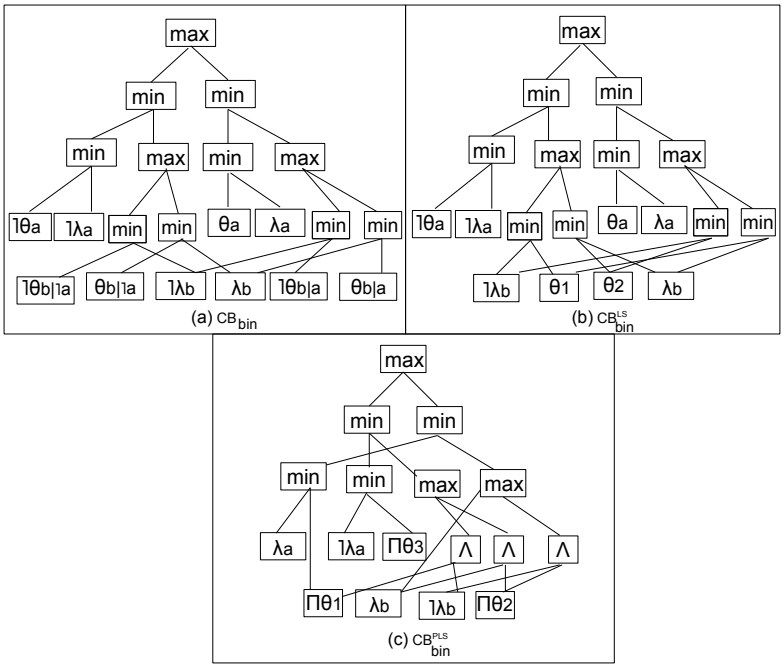


Figure 4.3: CB_{bin}, CB_{bin}^{LS} and CB_{bin}^{PLS}

Further experiments aiming to compare binary approaches will be performed in Chapter 7.

4.7 Conclusion

In this chapter, we have refined the CNF encoding of the Π-DNNF method proposed in Chapter 3. In fact, we have explored two variants of encoding strategies. The first one, named *local structure* consisting in assigning one propositional variable per equal parameters per possibility table. We have also proposed a new encoding strategy, named *possibilistic local structure* dealing with equal parameters from a global point of view. Moreover, we have studied the particular case of binary networks, which can be encoded using a more refined CNF base taking advantage of the particularity of binary variables. Next chapter will focus on compiling possibilistic causal networks to efficiently compute the effect of both observations and interventions.

Chapter 5

Handling interventions under compilation

5.1 Introduction

Possibilistic causal networks [17] make reference to causality in the possibility theory framework. The intriguing aspects of such networks are: *observations* which are results of testing some variables and *interventions* which correspond to external actions forcing some variables to have some specific values. From a reasoning point of view, an intervention on a variable A is represented using the so-called *mutilation*, by ignoring relations between the intervened variable A and its direct causes. From a representational point of view, an intervention is depicted by a new extra node added as a parent-node to each intervened variable. Inference in causal networks, which focuses on determining the impact of either an observation or an intervention on the remaining variables, is known as a hard problem [17, 69].

In [15], authors propose a new representation format, called *hybrid possibilistic causal networks*, where local uncertainty is no longer represented by conditional possibility distributions but by possibilistic knowledge bases. The main advantage of this representation concerns space complexity. An adaptation of the junction tree inference algorithm was proposed for hybrid possibilistic causal networks.

In Chapter 3, we have proposed two compilation-based inference methods for min-based possibilistic networks that only deal with observations. Our idea in this chapter is to enrich these methods to handle interventions in min-based possibilistic causal networks using a compilation setting. This idea has not been explored yet in the possibility theory, neither on the probability theory. In fact, each method studied in Chapter 3 will be extended

to deal with interventions twofold: *mutilation* and *augmentation* as depicted by Figure 5.5. More precisely, we will propose two mutilated-based methods that require the mutilation of symbolic compiled bases. This avoids re-compiling the network each time an intervention is occurred, which is intractable. We will also suggest augmented-based methods that do not apply this constraint due to the new extra node. After compiling the network and handling interventions either by mutilation or augmentation, an efficient computation of the effect of both observations and interventions should be ensured using compiled bases.

This chapter is organized as follows: Section 5.2 presents a refresher on possibilistic causal networks. Sections 5.3 and 5.4 are dedicated to mutilated-based approaches and augmented-based approaches, respectively. Main results of this Chapter are published in [4, 5].

5.2 Refresher on possibilistic causal networks

The notion of *causality* is a crucial concept in artificial intelligence when we describe, interpret and analyze information and phenomena of our environment. An intervention is a crucial notion in *causality*. It is an external event, coming from outside the system and forcing some variables to take a specific value. In our work, we focus on graphical representations for handling interventions.

In the probabilistic framework, Pearl's work [69] on causal Bayesian networks is considered one of the most prominent ones where a causal Bayesian network models the effect of both observations (i.e., evidences) and interventions, while a Bayesian network only handles observations. Pearl proposed two graphical representations of interventions [69]. The first one consists in mutilating the network by deleting direct links pointing to the variable of interest. The second adds a parent node to the variable concerned by the intervention. Such extra node describes the behavior of the variable of interest. The effect of interventions on the remaining variables consists in applying conditioning using new networks resulting from *mutilation* or *augmentation*.

In the probabilistic framework, when interventions are dealt with augmentation, a main problem resides, which consists in the inability to express the non-intervention. In other words, if there is no intervention, the augmented network does not recover the probability distribution on variables in the initial Bayesian network. *Is it also the case in the possibility theory framework?*

The following subsection shows how interventions can be handled in the possibility theory framework while dealing with possibilistic causal networks.

5.2.1 Possibilistic causal networks

A possibilistic causal network is a possibilistic network such that its *graphical component* is a DAG where nodes represent variables and edges encode not only dependencies between variables but also direct causal relationships [16]. The parent set U_i of any variable $X_i \in V$ represents all direct causes for X_i.

Example 5.1. *Let us consider the network of Figure 5.1*[1] *modeling this situation:*

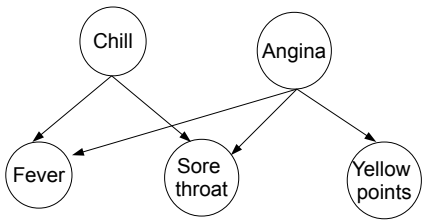


Figure 5.1: An Example of a causal network

- *A sore throat could result from a chill or angina.*
- *The chill can cause fever and a sore throat.*
- *Angina can cause these symptoms, and yellow points in the throat.*

Such network is causal since edges represent direct causality and oriented from the cause to the effect.

Causal networks are updated in the presence of two types of information: a set of *observations* (evidences) which are results of testing so me variables, and a set of *interventions* which represent external events, coming from outside the system and forcing some variables to take some specific values [69]. Interventions, denoted by $do(x_I)$, may have two different interpretations depending on whether we focus on the representational or on the reasoning issue.

Mutilation

From a reasoning point of view, an intervention is handled by the so-called *mutilation* operation [69], which refers to altering the network structure by

[1] From http://www.matthieuamiguet.ch/media/documents/MA-IARTI-05-ResBayesiens.pdf.

excluding all direct causes related to the variable of interest and maintaining the remaining variables unchanged [69]. The intuition behind such mutilation is that interventions are results of external actions and hence beliefs on direct causes of the intervened variable should not change. The possibility distribution associated with the mutilated network ΠG_{mut} is denoted by π_m. In possibility theory, the effect of $do(x_I)$ is to transform $\pi(\omega)$ into $\pi_m(\omega|x_I)$, which gives us [16]:

$$\forall\omega; \pi_m(\omega|x_I) = \pi(\omega|do(x_I)). \tag{5.1}$$

By mutilating the network, parents of X_I become independent of X_I. Moreover, the event that attributes the value x_I to X_I becomes sure after performing intervention $do(x_I)$. More formally, $\pi_m(x_I) = 1$ and $\forall x_i, x_i \neq x_I, \pi_m(x_i) = 0$. The effect of $do(x_I)$ on π is given as follows, $\forall\omega$:

$$\pi(\omega|do(x_I)) = \begin{cases} \min\limits_{i\neq I} \ \pi(x_i|u_i) & if\ \omega[X_i] = x_I \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

The effect of such interventions over the remaining variables is computed by applying the definition of conditioning on the mutilated network.

Example 5.2. *Let us consider the possibilistic network ΠG_{min} of Figure 5.2. Let B be the variable in ΠG_{min} forced to take the value b_1 by the intervention $do(b_1)$. Such intervention is reflected graphically by deleting the edge between A and B since the parent A is no longer responsible of the state of B after intervention. The resulting mutilated network is depicted by Figure 5.3.*

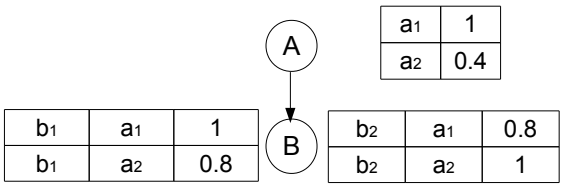


Figure 5.2: A possibilistic causal network ΠG_{min}

Augmentation

From a representational point of view, an intervention can be depicted by *augmentation*, an alternative but equivalent approach for handling interventions. This allows to represent interventions as observations on special new variables. More precisely, the augmenting process consists in viewing interventions as observations on new variables added to the system [51, 68].

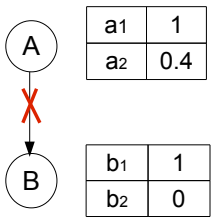


Figure 5.3: The mutilated network ΠG_{mut}

This leads to add new links $DO_I \rightarrow X_I$ where DO_I represents the new intervention taking k values $\{do(x_I) : \forall x_I \in D_{X_I}\} \cup \{do_{I-NoAct}\}$. The value $do_{I-NoAct}$ means that there is no intervention performed for X_I, while values $do(x_I)$ intend that the system forces the value x_I for X_I. We denote by do_I any value of DO_I.

The possibility distribution associated with the augmented network ΠG_{aug} is denoted by π_a. The new parent set of X_I is represented by $U'_I = U_I \cup DO_I$. The new possibility distribution of X_I after performing $do(x_I)$ is given by [16]:

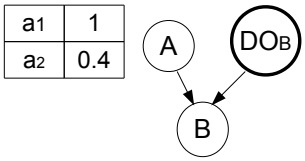
$$\pi(x_i|u'_i) = \begin{cases} \pi(x_i|u_i) & if\ DO_I = do_{I-NoAct} \\ 1 & if\ x_i = x_I \\ 0 & if\ x_i \neq x_I \end{cases} \tag{5.3}$$

However, possibility distributions associated with added nodes DO_I are not a priori stated. In [17], it has been proposed to define $\pi_a(do_{I-NoAct}) = 1$, while $\forall x_I \in D_{X_I}, \pi_a(do(x_I)) = \epsilon$ s.t., ϵ is a very small positive number close to 0. This allows to express that by default there is no intervention without excluding future interventions [16].

Example 5.3. *Let us re-consider the possibilistic network ΠG_{min} of Figure 5.2, then the augmented network resulting from the intervention $do(b_1)$ on B is represented by Figure 5.4. The new possibility distribution of B is obtained using Equation (5.3). For the distribution of DO_B, we should just find the smallest possibility degree in the distributions of the initial network of Figure 5.2, namely 0.4 and assign it to both of $\pi(do(b_1))$ and $\pi(do(b_2))$.*

It has been proved in [16] that the two ways of handling interventions are equivalent in the possibility theory framework. More formally,

Definition 5.1. *Let ΠG_{min} be a min-based possibilistic causal network. Let $do(x_I)$ be an intervention forcing X_I to take the value x_I. Let ΠG_{mut} (resp. ΠG_{aug}) be the mutilated (resp. augmented) network obtained after mutilation (resp. augmentation). Then, two situations are considered:*



a_1	1
a_2	0.4

do(B-NoAct)	$do(b_1)$	$do(b_2)$
1	0.4	0.4

a_1	b_1	do(B-NoAct)	1
a_1	b_1	$do(b_1)$	1
a_1	b_1	$do(b_2)$	0
a_1	b_2	do(B-NoAct)	0.8
a_1	b_2	$do(b_1)$	0
a_1	b_2	$do(b_2)$	1

a_2	b_1	do(B-NoAct)	0.8
a_2	b_1	$do(b_1)$	1
a_2	b_1	$do(b_2)$	0
a_2	b_2	do(B-NoAct)	1
a_2	b_2	$do(b_1)$	0
a_2	b_2	$do(b_2)$	1

Figure 5.4: The augmented network ΠG_{aug}

1. *No intervention:* $\forall \omega, \forall x_I \in D_{X_I}$, $\pi(\omega) = \pi_a(\omega | DO_I = do_{I-NoAct})$.
2. *An intervention* $do(x_I)$ *occurs:* $\pi(\omega | do(x_I)) = \pi_m(\omega | X_I = x_I) = \pi_a(\omega | DO_I = do(x_I))$.

Example 5.4. *Let us re-consider the* ΠG_{min} *of Figure 5.2. Let* B *be the variable in* ΠG_{min} *forced to take the value* b_1 *by the intervention* $do(b_1)$*. This latter implies:*

- *Mutilation :* $\pi_m(b_1) = 1$ *and* $\pi_m(b_2) = 0$.
- *Augmentation :* $\pi_a(do(b_1)) = 1$ *and* $\pi_a(do(B\text{-}NoAct)) = \pi_a(do(b_2)) = 0$.

It is easy to check that $\pi_m(a_2, b_1)$ *computed from* ΠG_{mut} *is equal to* $\pi_a(a_2, b_1)$ *computed from* ΠG_{aug}*. Effectively,* $\pi_m(a_2, b_1) = min(\pi_m(a_2), \pi_m(b_1)) = \pi_a(a_2, b_1) = max(\pi_a(a_2, b_1, do(B\text{-}NoAct)), \pi_a(a_2, b_1, do(b_1)), \pi_a(a_2, b_1, do(b_2))) = 0.4$*, which confirms Definition 5.1.*

In the following, we will present mutilated-based approaches and augmented-based approaches as summarized by Figure 5.5.

5.3 Mutilated-based approaches

In Chapter 3, we proposed two compilation-based inference methods for min-based possibilistic networks. The first method is a possibilistic adaptation of the so-called arithmetic circuit method [30]. The second method is a

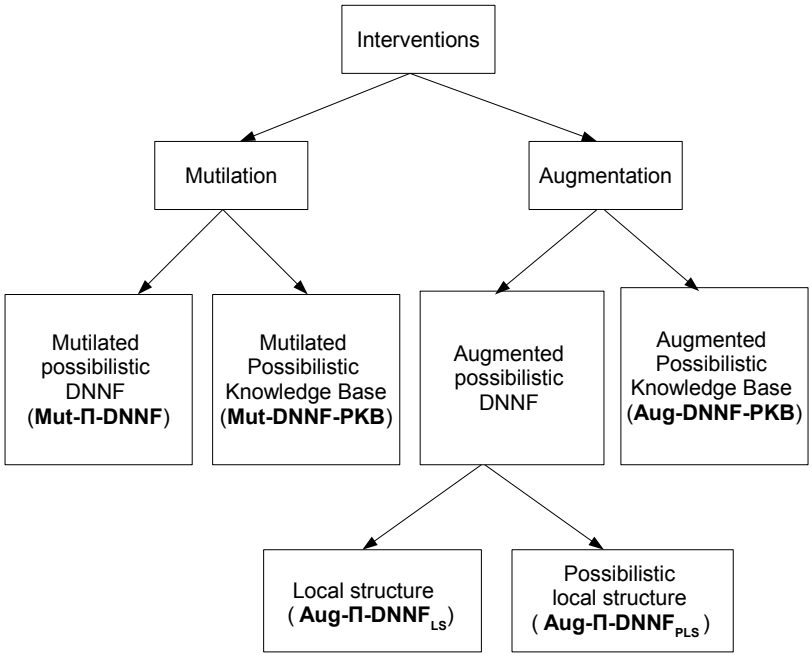


Figure 5.5: Summary of proposed methods using mutilation and augmentation

purely possibilistic inference method, which is not grounded on encoding probabilistic works. In Chapter 4, we have focused on the structure exhibited by network parameters locally using the so-called *local structure* and globally using *possibilistic local structure*.

In this section, we will extend these methods to deal with both observations and interventions using mutilation which gives rise to two mutilated-based approaches, namely: Mut-Π-DNNF and Mut-DNNF-PKB. Figure 5.6 depicts the general principle of mutilated-based approaches, which is based on three sequential phases: i) *encoding and compilation* phase, ii) *mutilation* phase and iii) *inference* phase such that non annotated lines are shared by both methods, while annotated ones concern one method, either Mut-Π-DNNF (a) or Mut-DNNF-PKB (b).

5.3.1 Mutilated Π-DNNF

One immediate way for handling sets of interventions consists first in mutilating the possibilistic network ΠG_{min}, encoding the mutilated network using the CNF propositional theory and then compiling it to offer a poly-time handling of queries. However, such a way is not efficient since it needs

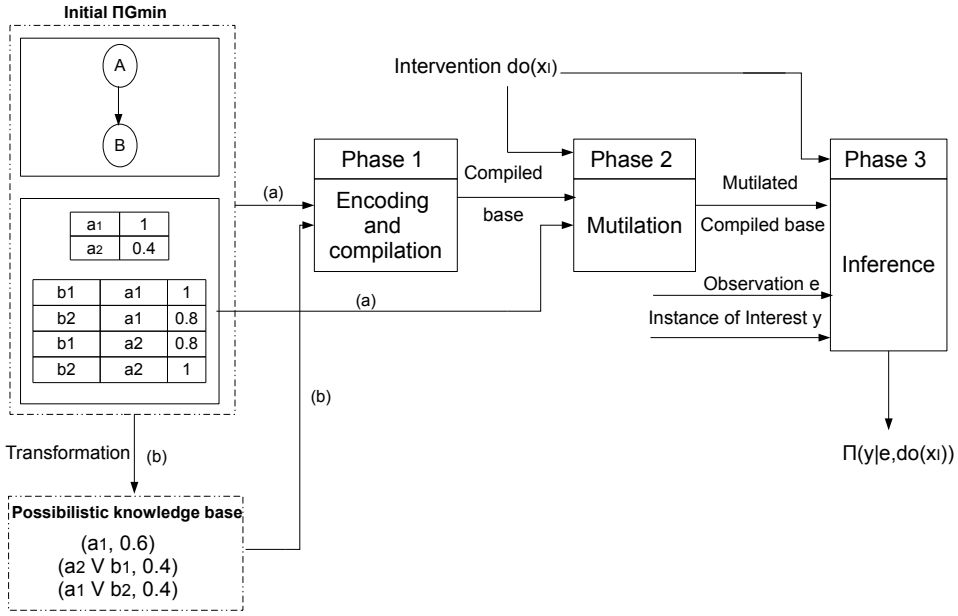


Figure 5.6: Principle of mutilated-based approaches (Lines labeled with (a) (resp. (b)) are relative to Mut-Π-DNNF (resp. Mut-DNNF-PKB) and unlabeled lines are relative to to both methods)

a re-compilation of the network each time an intervention occurs. The mutilated Π-DNNF method, denoted by Mut-Π-DNNF, proposed here avoids this problem by handling both observations and interventions without recompiling the initial network. We detail in what follows the three phases mentioned in Figure 5.6 (by considering lines annotated with (a)).

Phase 1: Encoding and compilation phase

The starting point of Mut-Π-DNNF method is the CNF encoding of the possibilistic network ΠG_{min} using two types of propositional variables namely, *instance indicators* λ_{x_i} for recording instances of variables and *parameter variables* $\theta_{x_i|u_i}$ for recording possibility degrees. Each parameter $\Pi(x_i|u_i)$ should be encoded using a unique $\theta_{x_i|u_i}$, regardless of its numerical value. The use of the encoding strategy *one variable per parameter* represents the key of handling interventions by mutilation under a compilation framework, as we will vindicate next. The CNF encoding handling n-ary variables is given by Definition 3.3. The encoding C_{min} is then compiled into DNNF as decribed in Section 3.3.2. The resulting compiled base is *symbolic* since we do not take into consideration any numerical value while encoding the network.

Phase 2: Mutilation phase

The mutilation phase proceeds with three inputs:
(i) the compiled base resulting from the previous step (CB),
(ii) the intervention $do(x_I)$ forcing the variable X_I to take the value x_I,
(iii) the initial possibility degrees of ΠG_{min}.

Given these inputs, we need to express that after intervention, the value of X_I is x_I for sure using CB. Technically speaking, this is established by conditioning the compiled base CB on x_I. Formally, $\forall\ \theta_{x_i|u_i}$ relative to X_I, we have:

$$\theta_{x_i|u_i} = \begin{cases} \top & \text{if } x_i = x_I \\ \bot & \text{otherwise (i.e., } x_i \neq x_I) \end{cases} \tag{5.4}$$

The resulting mutilated compiled base is denoted by CB^{mut}. By conditioning CB, as if we assign 1 to $\pi_m(x_I)$ and 0 to $\pi_m(x_i), \forall x_i \neq x_I$, i.e., we alter X_I's parameters to 1 ($\top$) and 0 ($\bot$). Hence, given interventions, new possibility degrees are affected to propositional variables $\theta_{x_i|u_i}$ corresponding to the intervened variable X_I which results in a new compiled base CB^{mut} as illustrated by Figure 5.6. In the following, the function *mutilate(CB)* will be used to mutilate CB.

It is worth to point out that the factor that makes this phase achievable is the strategy *one variable per parameter* providing a symbolic compiled base restricted to a set of symbols (propositional variables) without regard to their numerical values. So, the so-called *local structure* and *possibilistic local structure* enhancements related to equal parameters cannot be explored. More precisely, we cannot attribute the same propositional variable even for equal parameters within $C\Pi T_i$ or $C\Pi T$. For instance, assuming that we have $\theta_{b_2|a_1} = \theta_{b_1|a_2} = 0.8$. Then, after performing intervention $do(b_1)$, we should set $\theta_{b_2|a_1}$ (resp. $\theta_{b_1|a_2}$) to 0 (resp. 1). This is infeasible when we use the same propositional variable θ for both of $\theta_{b_2|a_1}$ and $\theta_{b_1|a_2}$.

By using this strategy, we do not need to re-compile the network for any new intervention, but we just need to 'logically' mutilate the current compiled base by applying conditioning.

Phase 3: Inference phase

Given the mutilated compiled base CB^{mut} resulting from the previous phase, an instance of interest y of a variable $X \in V$, an observation e and an intervention $do(x_I)$, we should be able to efficiently compute the effect of e and $do(x_I)$ on y, namely $\Pi_c(y|e, do(x_I))$. Using Equation (1.11), it is clear that we should first compute $\Pi_c(y, e, do(x_I))$ and $\Pi_c(e, do(x_I))$ following these three steps:

Step 1: Updating instance indicators: This step serves to record the instance of interest y, the observation e and the intervention $do(x_I)$ into instance indicators λ_{x_i}. It corresponds to conditioning the mutilated compiled base using instance indicators. Formally:

- CB^{mut} is conditioned on e and y, i.e., $\forall\ \lambda_{x_i}$ of $X_i \neq X_I$, we have:

$$\lambda_{x_i} = \begin{cases} \top & \text{if } x_i \sim e \text{ and } x_i \sim y \\ \bot & \text{otherwise} \end{cases} \tag{5.5}$$

  where $\sim$ denotes the compatibility relation.

- CB^{mut} is conditioned on x_I, i.e., $\forall\ \lambda_{x_i}$ of X_I, we have:

$$\lambda_{x_i} = \begin{cases} \top & \text{if } x_i = x_I \\ \bot & \text{otherwise} \end{cases} \tag{5.6}$$

The resulting compiled base is denoted by $\left[CB^{mut}|e, y, x_I\right]$.

Step 2: Mapping from logical to numerical representation: In this step, we transform the logical compiled base resulting from the previous step into a mutilated min-max circuit ΠCB^{mut} by:

- replacing $\vee$ and $\wedge$ by max and min, respectively,
- substituting each $\top$ (resp. $\bot$) by 1 (resp. 0),
- associating $\Pi(x_i|u_i)$ to each $\theta_{x_i|u_i}$ related to $\forall X_i \neq X_I$.

It is obvious that the mapping from logical to numerical representation is established in a polynomial time since it corresponds to a set of trivial substitution operations. The function *map(*$\left[CB^{mut}|e, y, x_I\right]$*)* will be used to map $\left[CB^{mut}|e, y, x_I\right]$ into ΠCB^{mut}.

Step 3: Computation: The last step corresponds to evaluating ΠCB^{mut} in order to efficiently compute $\Pi_c(y, e, do(x_I))$ and $\Pi_c(e, do(x_I))$ by applying min and max operators in a bottom-up way. In what follows, we will use the function *evaluate(*ΠCB^{mut}*)* to evaluate the mutilated min-max circuit ΠCB^{mut}.

Example 5.5. *Considering the network* ΠG_{min} *of Figure 5.2, its CNF encoding using Definition 3.3 contains the clauses of Table 3.6. It is clear that the degree 0.8 which appears twice in the distribution of* B *is encoded by two different propositional variables, namely* $\theta_{b_1|a_2}$ *and* $\theta_{b_2|a_1}$*. The CNF encoding* C_{min} *is then compiled into* CB *as shown in Figure 5.7.*

Let $do(b_1)$ be an intervention forcing the variable B to take the value b_1. Then, applying Equation (5.4) to CB consists in assigning $\theta_{b_1|a_1}$ and $\theta_{b_1|a_2}$ (resp. $\theta_{b_2|a_1}$ and $\theta_{b_2|a_2}$) to $\top$ (resp. $\bot$). The resulted mutilated compiled base is depicted by Figure 5.7.

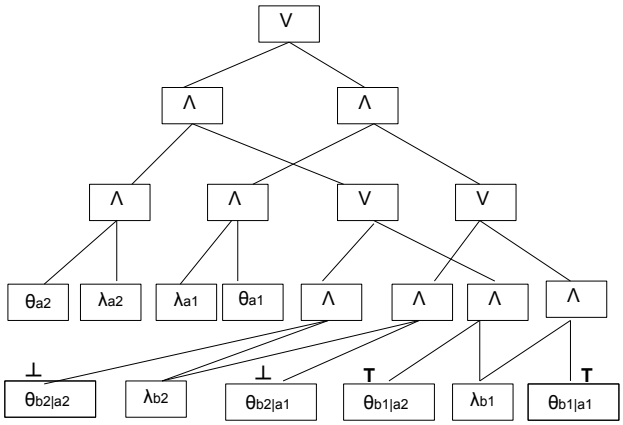


Figure 5.7: The mutilated compiled base CB^{mut}

Let us now compute $\Pi_c(a_2|do(b_1))$. We should then compute $\Pi_c(a_2, do(b_1))$ as follows:

1. *Record a_2 and $do(b_1)$ into instance indicators using Equation (5.6). This consists in setting λ_{a_2} and λ_{b_1} (resp. λ_{a_1} and λ_{b_2}) to $\top$ (resp. $\bot$). The resulting compiled base $CB^{mut}|a_2, b_1$ is shown by sub-figure (a) of Figure 5.8.*

2. *Transform the logical compiled base $\Pi CB^{mut}|a_2, do(b_1)$ into a mutilated min-max circuit ΠCB^{mut} (see sub-figure (b) of Figure 5.8).*

3. *Evaluate ΠCB^{mut} leading to $\Pi_c(a_2, do(b_1)) = 0.4$ as shown in sub-figure (c) of Figure 5.8.*

Thus, $\Pi_c(a_2|do(b_1)) = 0.4$ since $\Pi_c(a_2, do(b_1)) = 0.4 < \Pi_c(do(b_1)) = 1$.

Mut-Π-DNNF algorithm

We can conclude that Mut-Π-DNNF does not depend on interventions, i.e., even if the number of interventions is increased, the complexity is not altered since the mutilation process, which is a conditioning operation, is a linear task and the computation of interventions effects is polynomial with respect to the compiled base size. As a negative side, the constraint of *one*

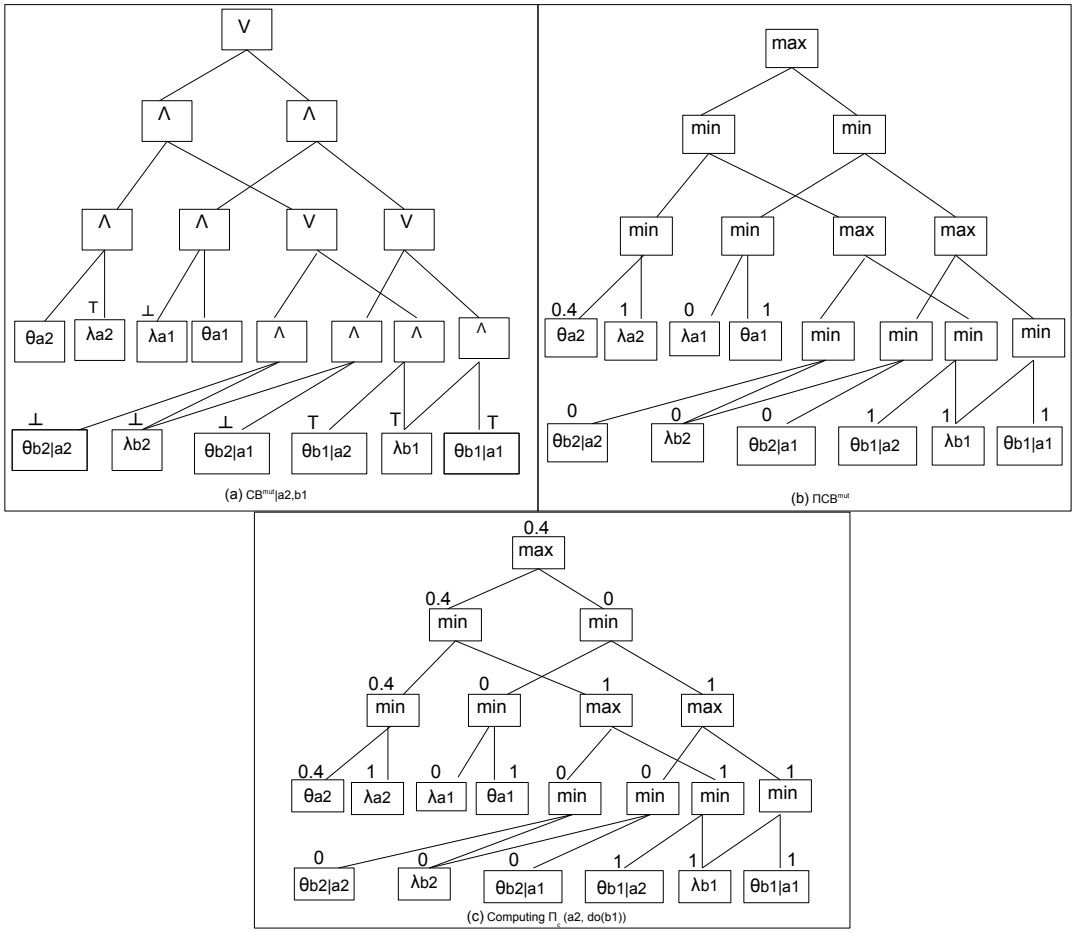


Figure 5.8: $CB^{mut}|a_2, b_1$, ΠCB^{mut} and $\Pi_c(a_2, do(b_1))$

variable per parameter should be applied even for equal parameters since the environment is qualified as dynamic when we deal with interventions.

The whole Mut-Π-DNNF approach, outlined by Algorithm 5, guarantees the equivalence between possibility degrees computed using Mut-Π-DNNF and the joint distribution as shown by the following proposition:

Algorithm 5: Mut-Π-DNNF

Data: ΠG_{min}, instance of interest y, evidence e, intervention $do(x_I)$
Result: $\Pi_c(y|e, do(x_I))$
begin
  % **Encoding and compilation phase**
  $C_{min} \leftarrow$ encode(ΠG_{min})
  $CB \leftarrow$ compile(C_{min})
  % **Mutilation phase**
  $CB^{mut} \leftarrow$ mutilate(CB)
  % **Inference phase**
  $Int \leftarrow \{y, e, do(x_I)\}$
  $\Pi_c(y, e, do(x_I)) \leftarrow$ Computing-mut $(CB^{mut}, Int, do(x_I))$
  $Int \leftarrow \{e, do(x_I)\}$
  $\Pi_c(e, do(x_I)) \leftarrow$ Computing-mut $(CB^{mut}, Int, do(x_I))$
  if $\Pi_c(y, e, do(x_I)) < \Pi_c(e, do(x_I))$ **then**
    $\Pi_c(y|e, do(x_I)) \leftarrow \Pi_c(y, e, do(x_I))$
  else
    $\Pi_c(y|e, do(x_I)) \leftarrow 1$
  return $\Pi_c(y|e, do(x_I))$

Algorithm 6: Computing-mut

Data: CB^{mut}, instance of interest Int, intervention $do(x_I)$
Result: $\Pi_c(Int, do(x_I))$
begin
  $CB^{mut}|Int, x_I \leftarrow$ condition(CB^{mut}, Int, x_I)
  $\Pi CB^{mut} \leftarrow$ map($CB^{mut}|Int, do(x_I)$)
  $\Pi_c(Int, do(x_I)) \leftarrow$ evaluate(ΠCB^{mut})
  return $\Pi_c(Int, do(x_I))$

Proposition 5.1. *Let CB be the compiled base of a possibilistic network ΠG_{min}. Let $do(x_I)$ be an intervention that forces the variable X_I to take the value x_I.*

i) $\forall \omega \in \Omega$, *we have:*

$$\pi_c(\omega|do(x_I)) = \pi_m(\omega|do(x_I)) \tag{5.7}$$

where $\pi_c(\omega|do(x_I))$ *(resp.* $\pi_m(\omega|do(x_I))$*) is computed using Algorithm 6 (resp. Equation (5.2)).*

ii) Let y *be an instantiation of a variable* $Y \in V$ *and* e *be an instantiation of a set of variables* $E \subseteq V$*. Then:*

$$\Pi_c(y|e, do(x_I)) = \Pi_m(y|e, do(x_I)) \tag{5.8}$$

where $\Pi_c(y|e, do(x_I))$ *(resp.* $\Pi_m(y|e, do(x_I))$*) is computed using Algorithm 5 (resp. Equation (5.2)).*

Proof 5.1. *From Equation (5.2), it is clear that given* $do(x_I)$*,* $\pi_m(\omega|do(x_I))$ *is equal to the minimum of possibility degrees compatible with* x_I*, 0 otherwise. From a logical point of view,* CB *is mutilated by associating* $\top$ *to each* $\theta_{x_i|u_i}$ *where* $x_i = x_I$ *and* $\bot$ *otherwise using Equation (5.4).*
From a numerical point of view, the possibility degree associated to $\theta_{x_i|u_i}$ *is 1 or 0.*
We obtain, $\pi_c(\omega|do(x_I)) = \pi_m(\omega|do(x_I))$*. Thus, Equation (5.7) is established.*

This result is relative to an interpretation ω*, to generalize it to any instantiation* y *of a variable* $Y \in V$*, we obtain:*
$\max_{\omega \models y} \pi_c(\omega|do(x_I)) = \max_{\omega \models y} \pi_c(\omega|do(x_I))$
Thus, $\Pi_c(y|do(x_I)) = \Pi_m(y|do(x_I))$*.*
When we deal with an observation e*, we obtain:*
$\Pi_c(y|e, do(x_I)) = \Pi_m(y|e, do(x_I))$*.* ■

5.3.2 Mutilated compiled possibilistic knowledge bases

The mutilated compiled possibilistic knowledge bases (denoted by Mut-DNNF-PKB) is a purely possibilistic inference method based on the transformation of the initial network into a possibilistic knowledge base [12] and then using this secondary structure as an input of the encoding and compilation phase. The remaining details the three main phases of Mut-DNNF-PKB presented in Figure 5.6 (by considering lines annotated with (b)).

Phase 1: Encoding and compilation phase

The principle of the transformation of the initial ΠG_{min} into Σ_{min} is to associate to each X_i a local possibilistic knowledge base Σ_{X_i} and to combine

them into a global possibilistic knowledge base Σ_{min}. Formally, this transformation is described by Definition 3.5. The function *transform(ΠG_{min})* will transform ΠG_{min} into Σ_{min}.

The min-based possibilistic knowledge base Σ_{min} is then encoded into a CNF base by affecting a new propositional variable, denoted by A_i, for each $N(\alpha_i)$ where $\alpha_i \in \Sigma_{min}$ without taking into consideration equal necessity values per base. The set of propositional variables, denoted by $\{A_1, ..., A_n\}$, encodes $\{N(\alpha_1), ..., N(\alpha_n)\}$. The propositional encoding of Σ_{min}, denoted by K_Σ is expressed by Equation (3.20). The strategy one variable per parameter is required in Mut-DNNF-PKB since some parameters values are not stable and will be updated depending on interventions. This is the fundamental difference between handling only observations and handling both of observations and interventions. In what follows, the function *encode-PKB(Σ_{min})* will be used to encode Σ_{min} into K_Σ.

The CNF encoding K_Σ of Σ_{min} is then compiled into any target compilation language supporting both of conditioning and clausal entailment, which are the required operations to compute the effect of observations and interventions. Thanks to these operations which make Mut-DNNF-PKB *flexible*. In what follows, we pick on the most succinct target compilation language, i.e., DNNF and we denote the resulting compiled base by K_c.

Phase 2: Mutilation phase

Given an intervention $do(x_I)$ performed on X_I, the compiled base K_c should be mutilated as depicted by Figure 5.6 (by considering lines annotated with (b)). This phase makes the connection between mutilating ΠG_{min} and mutilating K_c by updating the necessity degrees of variables A_i related to the variable of interest X_I. In fact, the necessity degree 1 should be assigned for formulas of $\neg x_i$ s.t. $x_i \neq x_I$ (since $\pi_m(\neg x_i) = 0$) and 0 for formulas of $\neg x_I$ (since $\pi_m(x_I) = 1$). The mutilated compiled base is denoted by K_c^{mut}.

It is worth pointing out that before the mutilation step we cannot attribute the same propositional variable A_i even for equal degrees in Σ_{min}. For instance, assuming that we have the following formulae $(a_2 \vee b_1, A_1)$ and $(a_1 \vee b_2, A_1)$ such that A_1 encodes the necessity degree 0.2. Let $do(b_1)$ be an intervention that forces B to take the value b_1. By mutilation, we mean setting the degree 0 (resp. 1) to the A_i corresponding to $(a_1 \vee b_2, 0.2)$ (resp. $(a_2 \vee b_1, 0.2)$). However, this is infeasible since the degree 0.2 is encoded twice using the same propositional variable, i.e., A_1.

Phase 3: Inference phase

After the mutilation phase, we should sort variables A_i by associating a new variable B_j to variables A_i encoding equal degrees in K_c^{mut}. To this end, let us consider g the number of different degrees in K_c after mutilation. Then, we will associate the set of propositional variables $B = \{B_1, \ldots, B_g\}$ for all different degrees pertaining to K_c after mutilation. The function *sort(A)* will sort the set A and the new set of variables is represented by B.

Once we sort propositional variables A_i, we should compute $\Pi_c(y|e, do(x_I))$ using the compiled base K_c^{mut} given an instance of interest y, an observation e and an intervention $do(x_I)$ performed on X_I. The computation process is established in an iterative manner by applying entailment and conditioning as outlined by Algorithm 7. In the worst case, computation is performed $g-1$ times since the last variable B_g encodes the degree 0.

Example 5.6. *Let us consider the possibilistic network ΠG_{min} of Figure 5.2. The CNF encoding of the possibilistic knowledge base Σ_{min} of ΠG_{min} is shown in Table 5.1.*

Clauses of A	
$(a_1, 0.6)$	$(a_1 \vee A_1)$
Clauses of B	
$(a_2 \vee b_1, 0.2)$	$(a_2 \vee b_1 \vee A_2)$
$(a_1 \vee b_2, 0.2)$	$(a_1 \vee b_2 \vee A_3)$

Table 5.1: The CNF encoding K_Σ of Σ_{min} of Figure 5.2

The CNF encoding K_Σ is then compiled into DNNF. The resulting compiled base is as follows: $K_c = \{[(a_2 \wedge A_1) \wedge (b_2 \vee (b_1 \wedge A_3))] \vee [a_1 \wedge (b_1 \vee (b_2 \wedge A_2))\}$.

Let B be the variable forced to take the value b_1 by the intervention $do(b_1)$. Then, mutilating the compiled base K_c consists in updating the degree of A_2 (resp. A_3) corresponding to $(a_2 \vee b_1, 0.2)$ (resp. $(a_1 \vee b_2, 0.2)$) from 0.2 to 1 (resp. 0.2 to 0).

Let us now compute the effect of $do(b_1)$ on a_2. First, we should sort propositional variables after mutilation. The new set of variables is the following: $B = \{B_1(1), B_2(0.6), B_3(0)\}$. *Then, we should compute $\Pi_c(a_2|do(b_1))$ as follows:*

- ***Iteration 1***: $K_c^{mut} \not\models b_2 \vee B_1 \Rightarrow (K_c^{mut}|\neg B_1) \not\models a_1 \Rightarrow i \leftarrow i + 1$,
- ***Iteration 2***: $K_c^{mut} \not\models b_2 \vee B_2 \Rightarrow (K_c^{mut}|\neg B_2) \models a_1 \Rightarrow StopCompute \leftarrow true$.

This means that $\Pi_c(a_2|do(b_1)) = 1 - degree(2) = 1 - 0.6 = 0.4$ *where* $degree(2)$ *designates the weight associated to* B_2*, i.e., 0.6.*

Mut-DNNF-PKB algorithm

The Mut-DNNF-PKB method, outlined by Algorithm 7, guarantees the equivalence between possibility degrees computed using Mut-DNNF-PKB and the joint distribution.

Proposition 5.2. *Let* K_c *be the compiled base of a possibilistic network* ΠG_{min}*. Let* $do(x_I)$ *be an intervention that forces the variable* X_I *to take the value* x_I*.*

i) $\forall \omega \in \Omega$*, we have:*

$$\pi_c(\omega|do(x_I)) = \pi_\Sigma(\omega|do(x_I)) \tag{5.9}$$

where $\pi_c(\omega|do(x_I))$ *(resp.* $\pi_\Sigma(\omega|do(x_I))$*) is computed using Algorithm 7 (resp. Equation (5.2)).*

ii) Let y *be an instantiation of a variable* $Y \in V$ *and* e *be an instantiation of any variables* $E \subseteq V$*. Then:*

$$\Pi_c(y|e, do(x_I)) = \Pi_\Sigma(y|e, do(x_I)) \tag{5.10}$$

where $\Pi_c(y|e, do(x_I))$ *(resp.* $\Pi_\Sigma(y|e, do(x_I))$*) is computed using Algorithm 7 (resp. Equation (5.2)).*

Proof 5.2. *From Equation (1.15), we have* $\pi(\omega) = \pi_\Sigma(\omega)$*. By mutilating* ΠG_{min}*, its means setting 1 to* $\pi(x_I)$ *and 0 to* $\pi(x_i), \forall x_i \neq x_I$*. Using CNF encodings of possibilistic knowledge bases, mutilation associates 1 or 0 to propositional variables* A_i *encoding necessity degrees.*
We obtain, $\pi_c(\omega|do(x_I)) = \pi_\Sigma(\omega|do(x_I))$*. Thus, Equation (5.9) is established.*

This result is relative to an interpretation ω*, to generalize it to any instantiation* y *of a variable* $Y \in V$*, we obtain:*
$\max_{\omega \models y} \pi_c(\omega|do(x_I)) = \max_{\omega \models y} \pi_c(\omega|do(x_I))$
Thus, $\Pi_c(y|do(x_I)) = \Pi_\Sigma(y|do(x_I))$*.*
When we deal with an observation e*, we obtain:*
$\Pi_c(y|e, do(x_I)) = \Pi_\Sigma(y|e, do(x_I))$*.* ■

Algorithm 7: Mut-DNNF-PKB

Data: ΠG_{min}, instance of interest y, evidence e, intervention $do(x_I)$
Result: $\Pi_c(y|e, do(x_I))$
begin
  % **Encoding and compilation phase**
  $\Sigma_{min} \leftarrow$ transform(ΠG_{min})
  $K_\Sigma \leftarrow$ encode-PKB(Σ_{min})
  $K_c \leftarrow$ compile(K_Σ)
  % **Mutilation phase**
  $K_c^{mut} \leftarrow$ mutilate(K_c)
  % **Inference phase**
  Let $B = \{B_1, \ldots, B_g\} \leftarrow$ sort(A)
  i $\leftarrow 1$, StopCompute $\leftarrow$ false, $\Pi_c(y|e, do(x_I)) \leftarrow 1$
  while *($K_c^{mut} \not\models B_i \vee \neg e \vee \neg x_I$)* ***and*** *($i < g$)* ***and***
  (StopCompute=false) **do**
    $K_c^{mut}|\neg B_i \leftarrow$ condition (K_c^{mut}, $\neg B_i$)
    if *($K_c^{mut}|\neg B_i) \models \neg y$* **then**
      StopCompute $\leftarrow$ true
      Let *degree*(i) be the weight associated to B_i
      $\Pi_c(y|e, do(x_I)) \leftarrow$ 1-*degree*(i)
    else $i \leftarrow i + 1$
  return $\Pi_c(y|e, do(x_I))$

5.4 Augmented-based approaches

In this section, we will explore augmented-based approaches for handling interventions. The focus will be on the new extra node. Figure 5.9 shows the general principle of augmented-based approaches which are based on two phases: i) *encoding and compilation* phase and ii) *inference* phase.

5.4.1 Augmented Π-DNNF

Augmented-based approaches of the possibilistic adaptation (see Figure 5.9 by considering lines annotated with (a)) deals with interventions from the first phase (i.e., encoding and compilation phase) by adding a new extra node to the possibilistic network. Contrarily to Mut-Π-DNNF, the Π-DNNF method under augmentation can benefit from *local structure* and *possibilistic local structure*. Using the structure exhibited by parameters values, three variants can be explored, namely Aug-Π-DNNF$_{LS}$, Aug-Π-DNNF$^l_{PLS}$ and Aug-Π-DNNF$_{PLS}$ when we deal with local structure, possibilistic local structure with left-side clauses and possibilistic local structure without left-side

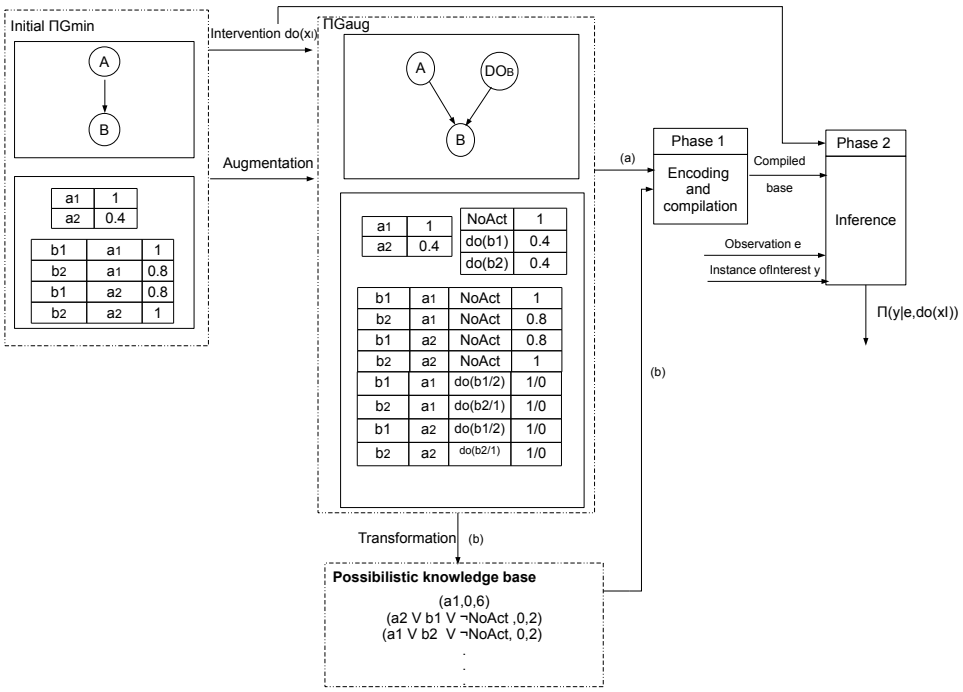


Figure 5.9: Principle of augmented-based approaches ((a): Augmented-based approaches of the possibilistic adaptation), (b): Augmented-based approach of the possibilistic logic counterpart))

clauses, respectively. These methods are composed of two phases as detailed below.

Phase 1: Encoding and compilation phase

Augmented-based approaches proceed by encoding the augmented possibilistic network arisen from performing intervention on the initial network. The question that may arise is: *Can we handle interventions before the encoding and compilation phase?* We have seen that the answer is NO when using mutilated-based approaches, which is not the case in augmented-based approaches since the new extra node DO_I enables us to focus on both observations and interventions using the augmented network.

Once we obtain the augmented network ΠG_{aug}, it should be encoded into a CNF base, but we will focus on ΠG_{aug} instead of ΠG_{min}. In other words, we should just consider the new extra node DO_I as a variable of V (i.e., $V = V \cup DO_I$). The key point that should be highlighted is the strategy of encoding, which depends on the handled node either an extra node DO_I or

a node $X_i \in V$. In fact, for any node $X_i \in V$ except DO_I, parameters values are constant and do not depend on interventions. For this reason, specific parameters values can be addressed. If we deal with *local structure*, each $X_i \in V$ except DO_I should be encoded using Definition 4.1. When we apply *possibilistic local structure*, Definitions 4.3 or 4.4 can be used depending on the used clauses (with or without left-side clauses).

Regarding the new extra node DO_I, neither possibilistic local structure nor local structure can be exploited due to the variability of DO_I's parameters. For this reason, DO_I should be encoded using symbolic propositional variables without taking into consideration any numerical value, which results in a local symbolic encoding associated with DO_I. More explicitly, the strategy *one variable per parameter* will be adopted regardless of whether there are some equal or zero parameters in DO_I's $\text{C}\Pi T_i$. This is the key point that allows us to express that by default there is no intervention and does not exclude future interventions by just setting the appropriate values to DO_I's parameter variables. The CNF encoding of ΠG_{aug}, denoted by C^{aug}_{LS}, $C^{aug}_{PLS_l}$ and C^{aug}_{PLS} in Aug-Π-DNNF$_{LS}$, Aug-Π-DNNF$^{l}_{PLS}$ and Aug-Π-DNNF$_{PLS}$ methods, respectively, should be then compiled into DNNF. The augmented compiled base is denoted by CB^{aug}.

Phase 2: Inference phase

The inference phase consists in computing efficiently the effect of both observations and interventions using the compiled base CB^{aug}. Hence, given CB^{aug}, an instance of interest y, an observation e and an intervention $do(x_I)$, we should compute both of $\Pi_c(y, e, do(x_I))$ and $\Pi_c(e, do(x_I))$ by applying the following steps, to have $\Pi_c(y|e, do(x_I))$.

Step 1: Updating instance indicators: The first step consists in updating instances indicators according to e, y and $do(x_I)$. We should at first condition CB^{aug} on e and y using Equation (5.5) as in Mut-Π-DNNF. A slight modification should be performed in both of these Equations by replacing $\forall X_i \neq X_I$ by $\forall X_i \neq DO_I$. Then, we should set each λ_{do_I} of DO_I to $\top$ or $\bot$ depending on $do(x_I)$.

- Intervention $do(x_I)$: $\forall\ \lambda_{do_I}$ of DO_I, we have:

$$\lambda_{do_I} = \begin{cases} \top & \text{if } do_I = do(x_I) \\ \bot & \text{otherwise (i.e., } do_I \neq do(x_I)) \end{cases} \tag{5.11}$$

- No Interventions: $\forall\ \lambda_{do_I}$ of DO_I, we have:

$$\lambda_{do_I} = \begin{cases} \top & \text{if } do_I = do_{I-NoAct} \\ \bot & \text{otherwise (i.e., } do_I \neq do_{I-NoAct}) \end{cases} \tag{5.12}$$

Step 2: Mapping from logical to numerical representation: This step transforms the logical representation arisen from Step 1 into an augmented min-max circuit, denoted by ΠCB^{aug}_{LS}, $\Pi CB^{aug}_{PLS_l}$ and ΠCB^{aug}_{PLS} in Aug-Π-DNNF$_{LS}$, Aug-Π-DNNF$^{l}_{PLS}$ and Aug-Π-DNNF$_{PLS}$, respectively. This mapping consists in replacing each $\vee$, $\wedge$, $\top$ and $\bot$ by max, min, 1 and 0, respectively. Also, each $\theta_{x_i|u_i}$ of $X_i \neq DO_I$ should be replaced by the appropriate possibility degree it encodes. An additional substitution should be performed for DO_I's parameter variables. In fact, if we only focus on observations, we should set initial possibility values (i.e., 1 and ϵ which is a very small positive number close to 0) to DO_I's network parameters. However, if an intervention occurs, the degree 1 or 0 is assigned to each DO_I's parameter variable depending on $do(x_I)$.

Step 3: Computation: The last step corresponds to computing $\Pi_c(y, e, do(x_I))$ and $\Pi_c(e, do(x_I))$ using ΠCB^{aug}_{LS}, $\Pi CB^{aug}_{PLS_l}$ and ΠCB^{aug}_{PLS} by applying min and max operators in a bottom-up way.

Example 5.7. *Considering the network of Figure 5.4. At first, we should define the set of instance indicators and network parameters given respectively, in Table 5.2 and Table 5.3. Then, the CNF encoding of* ΠG_{aug} *using local structure contains clauses of Table 5.4.*

Instances	C_{aug}
a_1	λ_{a_1}
a_2	λ_{a_2}
b_1	λ_{b_1}
b_2	λ_{b_2}
$do(B\text{-}NoAct)$	$\lambda_{do(B-NoAct)}$
$do(b_1)$	$\lambda_{do(b_1)}$
$do(b_2)$	$\lambda_{do(b_2)}$

Table 5.2: Instance indicators used in C_{aug}

It is obvious that we are using the strategy one variable per parameter per DO_B*'s* $C\Pi T_i$*, without taking into account any numerical value, i.e., 1, 0.4 and 0.4. The compiled base of* C^{aug}_{min} *is depicted by Figure 5.10.*

Let a_2 *be an instance of interest and* $do(b_1)$ *be an intervention forcing* B *to take the value* b_1*, then* $\Pi_c(a_2|do(b_1))$ *is obtained by comparing both of* $\Pi_c(a_2, do(b_1))$ *and* $\Pi_c(do(b_1))$*. To compute* $\Pi_c(a_2, do(b_1))$*:*

1. *We should first set* λ_{a_2}, $\lambda_{do(b_1)}$, λ_{b_1} *to* $\top$ *and* $\lambda_{do(B-NoAct)}$, λ_{a_1}, λ_{b_2}, $\lambda_{do(b_2)}$ *to* $\bot$,
2. *We should replace each* $\wedge$ *and* $\vee$ *by min and max, respectively,*

Variables	Possibility degrees	C_{aug}
A	1	θ_{a_1}
	0.4	θ_{a_2}
B	$\Pi(b_1 \mid a_1, do(B\text{-}NoAct) = 1$	θ_1
	$\Pi(b_1 \mid a_1, do(b_1)) = 1$	θ_1
	$\Pi(b_1 \mid a_1, do(b_2)) = 0$	-
	$\Pi(b_1 \mid a_2, do(B\text{-}NoAct)) = 0.8$	θ_2
	$\Pi(b_1 \mid a_2, do(b_1)) = 1$	θ_1
	$\Pi(b_1 \mid a_2, do(b_2)) = 0$	-
	$\Pi(b_2 \mid a_1, do(B\text{-}NoAct)) = 0.8$	θ_2
	$\Pi(b_2 \mid a_1, do(b_1)) = 0$	-
	$\Pi(b_2 \mid a_1, do(b_2)) = 1$	θ_1
	$\Pi(b_2 \mid a_2, do(B\text{-}NoAct)) = 1$	θ_1
	$\Pi(b_2 \mid a_2, do(b_1)) = 0$	-
	$\Pi(b_2 \mid a_2, do(b_2)) = 1$	θ_1
DO_B	$\Pi(do(B\text{-}NoAct))$ (not initialized)	$\theta_{do(B-NoAct)}$
	$\Pi(do(b_1))$ (not initialized)	$\theta_{do(b_1)}$
	$\Pi(do(b_2))$ (not initialized)	$\theta_{do(b_2)}$

Table 5.3: Network parameters used in C_{aug}

3. *Also, we should set the possibility degree 1 to* $\theta_{do(b_1)}$ *and 0 for both of* $\theta_{do(b_2)}$ *and* $\theta_{do(B-NoAct)}$,

4. *Finally, we compute* $\Pi_c(a_2, do(b_1))$ *in a bottom-up way as shown in Figure 5.11.*

Hence, $\Pi_c(a_2 \mid do(b_1)) = 0.4$ *since* $\Pi_c(a_2 \mid do(b_1)) = 0.4 < \Pi_c(do(b_1)) = 1$.

Augmented-based algorithms

Algorithm 8 outlines Aug-Π-DNNF$_{LS}$, Aug-Π-DNNF$^{l}_{PLS}$ and Aug-Π-DNNF$_{PLS}$ methods. For generality reasons, we will use C^{aug} to denote C^{aug}_{LS} or $C^{aug}_{PLS_l}$ or C^{aug}_{PLS} and ΠCB^{aug} to denote ΠCB^{aug}_{LS} or $\Pi CB^{aug}_{PLS_l}$ or ΠCB^{aug}_{PLS}.

Mutual Exclusive clauses	
A	$(\lambda_{a_1} \vee \lambda_{a_2}) \wedge (\neg\lambda_{a_1} \vee \neg\lambda_{a_2})$
B	$(\lambda_{b_1} \vee \lambda_{b_2}) \wedge (\neg\lambda_{b_1} \vee \neg\lambda_{b_2})$
DO_B	$(\lambda_{do(B-NoAct)} \vee \lambda_{do(b_1)} \vee \lambda_{do(b_2)})$ $\wedge(\neg\lambda_{do(B-NoAct)} \vee \neg\lambda_{do(b_1)})$ $\wedge(\neg\lambda_{do(B-NoAct)} \vee \neg\lambda_{do(b_2)})$ $\wedge(\neg\lambda_{do(b_1)} \vee \neg\lambda_{do(b_2)})$
Parameter Clauses of A	
$\Pi(a_1) = 1$	$(\lambda_{a_1} \rightarrow \theta_{a_1}) \wedge (\theta_{a_1} \rightarrow \lambda_{a_1})$
$\Pi(a_2) = 0.4$	$(\lambda_{a_2} \rightarrow \theta_2) \wedge (\theta_{a_2} \rightarrow \lambda_{a_2})$
Parameter Clauses of B	
$\Pi(b_1 \mid a_1, do(B\text{-}NoAct)) = 1$	$\lambda_{a_1} \wedge \lambda_{b_1}) \wedge \lambda_{do(B-NoAct)} \rightarrow \theta_1$
$\Pi(b_1 \mid a_1, do(b_1)) = 1$	$\lambda_{a_1} \wedge \lambda_{b_1} \wedge \lambda_{do(b_1)} \rightarrow \theta_1$
$\Pi(b_1 \mid a_1, do(b_2)) = 0$	$\neg\lambda_{a_1} \vee \neg\lambda_{b_1} \vee \neg\lambda_{do(b_2)}$
$\Pi(b_1 \mid a_2, do(B\text{-}NoAct)) = 0.8$	$\lambda_{a_2} \wedge \lambda_{b_1} \wedge \lambda_{do(B-NoAct)} \rightarrow \theta_2$
$\Pi(b_1 \mid a_2, do(b_1)) = 1$	$\lambda_{a_2} \wedge \lambda_{b_1} \wedge \lambda_{do(b_1)} \rightarrow \theta_1$
$\Pi(b_1 \mid a_2, do(b_2)) = 0$	$\neg\lambda_{a_2} \vee \neg\lambda_{b_2} \vee \neg\lambda_{do(b_2)}$
$\Pi(b_2 \mid a_1, do(B\text{-}NoAct)) = 0.8$	$\lambda_{a_1} \wedge \lambda_{b_2} \wedge \lambda_{do(B-NoAct)} \rightarrow \theta_2$
$\Pi(b_2 \mid a_1, do(b_1)) = 0$	$\neg\lambda_{a_1} \vee \neg\lambda_{b_2} \vee \neg\lambda_{do(b_1)}$
$\Pi(b_2 \mid a_1, do(b_2)) = 1$	$\lambda_{a_1} \wedge \lambda_{b_2} \wedge \lambda_{do(b_2)} \rightarrow \theta_1$
$\Pi(b_2 \mid a_2, do(B\text{-}NoAct)) = 1$	$\lambda_{a_2} \wedge \lambda_{b_2} \wedge \lambda_{do(B-NoAct)} \rightarrow \theta_1$
$\Pi(b_2 \mid a_2, do(b_1)) = 0$	$\neg\lambda_{a_2} \vee \neg\lambda_{b_2} \vee \neg\lambda_{do(b_1)}$
$\Pi(b_2 \mid a_2, do(b_2)) = 1$	$\lambda_{a_2} \wedge \lambda_{b_2} \wedge \lambda_{do(b_2)} \rightarrow \theta_1$
Parameter Clauses of DO_B	
$\Pi(do(B\text{-}NoAct))$	$(\lambda_{do(B-NoAct)} \rightarrow \theta_{do(B-NoAct)})$ $\wedge(\theta_{do(B-NoAct)} \rightarrow \lambda_{do(B-NoAct)})$
$\Pi(do(b_1))$	$(\lambda_{do(b_1)} \rightarrow \theta_{do(b_1)}) \wedge (\theta_{do(b_1)} \rightarrow \lambda_{do(b_1)})$
$\Pi(do(b_2))$	$(\lambda_{do(b_2)} \rightarrow \theta_{do(b_2)}) \wedge (\theta_{do(b_2)} \rightarrow \lambda_{do(b_2)})$

Table 5.4: The CNF encoding C_{aug} of ΠG_{aug} of Figure 5.4

Algorithm 8: Aug-Π-DNNF

Data: ΠG_{aug}, instance of interest y, evidence e, intervention $do(x_I)$
Result: $\Pi_c(y|e, do(x_I))$
begin
  % **Encoding and compilation phase**
  $C^{aug} \leftarrow$ encode(ΠG_{aug})
  $CB^{aug} \leftarrow$ compile(C^{aug})
  % **Inference phase**
  $Int \leftarrow \{y, e, do(x_I)\}$
  $\Pi_c(y, e, do(x_I)) \leftarrow$ Computing-Aug $(CB^{aug}, Int, do(x_I))$
  $Int \leftarrow \{e, do(x_I)\}$
  $\Pi_c(y, do(x_I)) \leftarrow$ Computing-Aug $(CB^{aug}, Int, do(x_I))$
  if $\Pi_c(y, e, do(x_I)) < \Pi_c(e, do(x_I))$ **then**
    $\Pi_c(y|e, do(x_I)) \leftarrow \Pi_c(y, e, do(x_I))$
  else
    $\Pi_c(y|e, do(x_I)) \leftarrow 1$
  return $\Pi_c(y|e, do(x_I))$

Algorithm 9: Computing-Aug

Data: CB^{aug}, instance of interest Int, intervention $do(x_I)$
Result: $\Pi_c(Int, do(x_I))$
begin
  $CB^{aug}|Int, do(x_I) \leftarrow$ Condition(CB^{aug}, Int, $do(x_I)$)
  $\Pi CB^{aug} \leftarrow$ map($CB^{aug}|Int, do(x_I)$)
  $\Pi_c(Int, do(x_I)) \leftarrow$ evaluate(ΠCB^{aug})
  return $\Pi_c(Int, do(x_I))$

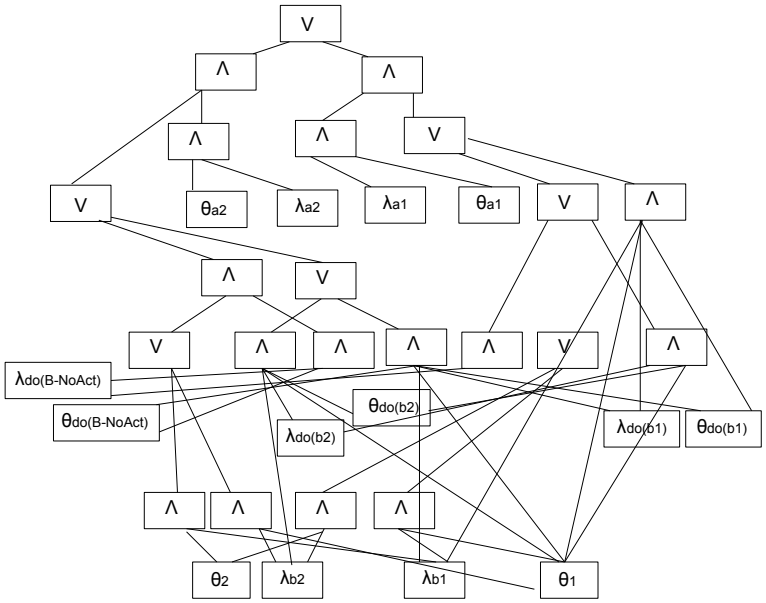


Figure 5.10: The augmented compiled base C^{aug}_{DNNF}

Proposition 5.3. *Let CB^{aug} be the compiled base of an augmented network ΠG_{aug}. Let $do(x_I)$ be an intervention that forces the variable X_I to take the value x_I.*

i) $\forall \omega \in \Omega$, we have:

$$\pi_c(\omega|do(x_I)) = \pi_a(\omega|do(x_I)) \tag{5.13}$$

where $\pi_c(\omega|do(x_I))$ (resp. $\pi_a(\omega|do(x_I))$) is computed using Algorithm 9 (resp. Definition (5.1)).

ii) Let y be an instantiation of a variable $Y \in V$ and e be an instantiation of any variables $E \subseteq V$. Then:

$$\Pi_c(y|e, do(x_I)) = \Pi_a(y|e, do(x_I)) \tag{5.14}$$

where $\Pi_c(y|e, do(x_I))$ (resp. $\Pi_a(y|e, do(x_I))$) is computed using Algorithm 8 (resp. Definition (5.1)).

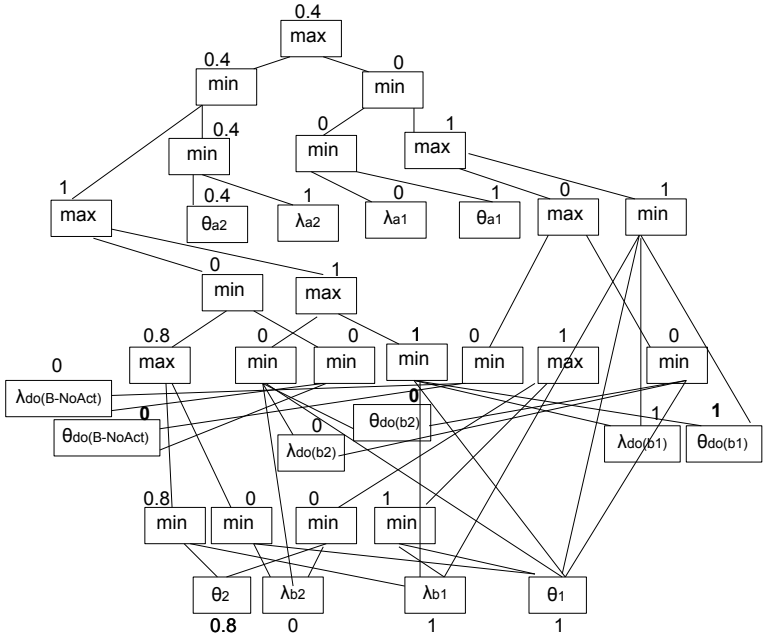


Figure 5.11: Computing $\Pi_c(a_2, do(b_1))$

Proof 5.3. *When we use:*

- ***Local structure**:*
  For any min-based possibilistic network ΠG_{min}*, from Proposition 4.2, we have* $\pi_{min}(\omega) = \pi_{C^{LS}_{min}}(\omega)$.
  Thus, $\pi_a(\omega|do(x_I)) = \pi_{C^{aug}_{LS}}(\omega|do(x_I))$ *since* $\Pi G_{aug} = \Pi G_{min} \cup DO_I$.
- ***Possibilistic local structure with left-side clauses**:*
  For any min-based possibilistic network ΠG_{min}*, from Proposition 4.4, we have* $\pi_{min}(\omega) = \pi_{C^{PLS}_{min_l}}(\omega)$.
  Thus, $\pi_a(\omega|do(x_I)) = \pi_{C^{aug}_{PLS_l}}(\omega|do(x_I))$ *since* $\Pi G_{aug} = \Pi G_{min} \cup DO_I$.
- ***Possibilistic local structure without left-side clauses**:*
  For any min-based possibilistic network ΠG_{min}*, from Proposition 4.6, we have* $\pi_{min}(\omega) = \pi_{C^{PLS}_{min}}(\omega)$.
  Thus, $\pi_a(\omega|do(x_I)) = \pi_{C^{aug}_{PLS}}(\omega|do(x_I))$ *since* $\Pi G_{aug} = \Pi G_{min} \cup DO_I$.

In the three cases, we obtain:
$\pi_c(\omega|do(x_I)) = \pi_a(\omega|do(x_I))$. *Thus, Equation (5.13) is established.*

This result is relative to an interpretation ω*, to generalize it to any instantiation* y *of a variable* $Y \in V$*, we obtain:*

$\max_{\omega \models y} \pi_c(\omega | do(x_I)) = \max_{\omega \models y} \pi_c(\omega | do(x_I))$,
Thus, $\Pi_c(y|do(x_I)) = \Pi_a(y|do(x_I))$.
When we deal with an observation e*, we obtain:*
$\Pi_c(y|e, do(x_I)) = \Pi_a(y|e, do(x_I))$. ■

5.4.2 Augmented compiled possibilistic knowledge bases

The starting point of the augmented compiled possibilistic knowledge bases approach (denoted by Aug-DNNF-PKB) is a transformation of the augmented possibilistic network into a possibilistic knowledge base which will be used to ensure the phases depicted by Figure 5.9 (by considering lines (b)).

Phase 1: Encoding and compilation phase

The transformation of ΠG_{aug} into an augmented possibilistic knowledge base Σ_{aug} consists in associating a local possibilistic knowledge base for both of DO_I and $\forall X_i \in V$. Formally, this transformation is described by Definition 5.2.

Definition 5.2. *Let* ΠG_{min} *be a possibilistic causal network. Let* $do(x_I)$ *be an intervention that forces the variable* X_I *to take the value* x_I*. Let* ΠG_{aug} *be the augmented network and* Σ_{aug} *be its possibilistic knowledge base expressed by:*

$$\Sigma_{aug} = \cup_{X_i \in V} \Sigma_{X_i} \bigcup \Sigma_{DO_I} \tag{5.15}$$

where $\Sigma_{X_i} = \{(\neg x_i \vee \neg u_i, a_i) : a_i = 1 - \Pi(x_i|u_i) \neq 0\}$ *and* $\Sigma_{DO_I} = \{(\neg do_I, b_i)\}$

Let $\pi_{\Sigma_{aug}} : \Omega \rightarrow [0, 1]$ be the possibility distribution associated with Σ_{aug}. Then,

$$\forall \omega \in \Omega, \pi_a(\omega) = \pi_{\Sigma_{aug}}(\omega). \tag{5.16}$$

where π_a is the possibility distribution associated with ΠG_{aug} using Equation (1.21). Clearly, in Definition 5.2, Σ_{DO_I} and Σ_{X_i}, $\forall X_i \in V$ are handled separately since these latters do not require the same encoding strategy. Note also that we should incorporate the mutual exclusive clauses with a necessity degree of 1 for each non-binary variable.

After the transition from a graphical to a logic-based representation, an encoding phase is established as in *Mut-DNNF-PKB*. However, the encoding

strategy does not remain the same in Aug-DNNF-PKB since it depends on the node related to each possibilistic formula. In fact, possibilistic formulas α_i related to each $X_i \in V$ are encoded using propositional variables A_i such that A_i may encode a set of equal parameters a_i pertaining to any $\Sigma_{X_i}, \forall X_i \in V$. However, each possibilistic formula related to DO_I should be encoded using a unique propositional variable B_i without considering any necessity value. Interestingly enough, possibilistic local structure per base is only exploited for non-extra nodes (i.e., $\forall X_i \in V$ except DO_I) since necessity degrees associated for their possibilistic formulas are constant and do not depend on $do(x_I)$. This is not the case for parameters of the new extra node DO_I which are unstable and depend on $do(x_I)$. More formally, the CNF encoding of Σ_{aug} is expressed by:

$$K_{\Sigma}^{aug} = \{(\alpha_i \vee A_i : (\alpha_i, a_i) \in \Sigma_{X_i}, \forall X_i \in V)\} \cup \{(\alpha_i \vee B_i : (\alpha_i, b_i) \in \Sigma_{DO_I})\} \quad (5.17)$$

It is worth pointing out that we should use the strategy one variable per parameter per DO_I's formula without taking into account any numerical value b_i, which is not the case for each $X_i \in V$. This is the key point that allows us to handle both the non-intervention and the intervention cases in a representational framework. After encoding the possibilistic knowledge base, we should then compile it into DNNF. The augmented compiled base is denoted by K_c^{aug}.

Phase 2: Inference phase

Once we compiled the augmented possibilistic knowledge base, we should at first update the necessity degree of each propositional variable B_i associated with Σ_{DO_I} depending on $do(x_I)$. In fact, we should assign the necessity degree 1 to variables B_i corresponding to formulae of $\neg do(I - NoAct)$ and $\neg do(x_i), i \neq I$, (since $\pi_a(do(I - NoAct)) = 0$ and $\pi_a(do(x_i)) = 0$) while variables B_i corresponding to formula of $\neg do(x_I)$ should encode the necessity degree 0 (since $\pi_a(do(x_I)) = 1$). In what follows, the function *update(B)* will update necessity degrees of each $B_i \in B$ depending on $do(x_I)$.

Then, propositional variables A_i and B_i should be unified into one set of variables $A = \{A_1, \ldots, A_g\}$, where g is the new number of variables. This guarantees that the same necessity degree is not encoded using two different propositional variables. The function *unify(A, B)* will be used to perform this task. Finally, we should compute efficiently the effect of the intervention $do(x_I)$ and the observation e on the instance of interest y by applying entailment and conditioning in an iterative manner as shown by Algorithm 10.

Example 5.8. *Let us re-consider the network* ΠG_{aug} *of Figure 5.4. The set*

of propositional variables used to encode necessity degrees of the possibilistic knowledge base Σ_{aug} associated with ΠG_{aug} is presented in Table 5.5.

Variables	Necessity degrees	$K_{\Sigma_{aug}}$
A	0.6	A_2
B	1	A_1
	0.3	A_3
DO_B	(Not initialized)	B_1
	(Not initialized)	B_2
	(Not initialized)	B_3

Table 5.5: Propositional variables used in $K_{\Sigma_{aug}}$

The CNF encoding $K_{\Sigma_{aug}}$ of the possibilistic base of ΠG_{aug} is presented in Table 5.6 s.t. B_1, B_2 and B_3 encode DO_B's degrees. Compiling $K_{\Sigma_{aug}}$ into DNNF results in an augmented compiled base represented by Figure 5.12.

Mutual Exclusive clauses	
A	$(a_1 \vee a_2 \vee A_1) \wedge (\neg a_1 \vee \neg a_2 \vee A_1)$
B	$(b_1 \vee b_2 \vee A_1) \wedge (\neg b_1 \vee \neg b_2 \vee A_1)$
DO_B	$(do(B\text{-}NoAct) \vee do(b_1) \vee do(b_2) \vee A_1)$ $\wedge(\neg do(B\text{-}NoAct) \vee \neg do(b_1) \vee A_1)$ $\wedge(\neg do(B\text{-}NoAct) \vee \neg do(b_2) \vee A_1)$ $\wedge(\neg do(b_1) \vee \neg do(b_2) \vee A_1)$
Clauses of A	
$(\neg a_2, 0.6)$	$(\neg a_2 \vee A_2)$
Clauses of B	
$(\neg a_1 \vee \neg b_1 \vee \neg do(b_2), 1)$	$(\neg a_1 \vee \neg b_1 \vee \neg do(b_2) \vee A_1)$
$(\neg a_2 \vee \neg b_1 \vee \neg do(B\text{-}NoAct), 0.3)$	$(\neg a_2 \vee \neg b_1 \vee \neg do(B\text{-}NoAct) \vee A_3)$
$(\neg a_2 \vee \neg b_1 \vee \neg do(b_2), 1)$	$(\neg a_2 \vee \neg b_1 \vee \neg do(b_2) \vee A_1)$
$(\neg a_1 \vee \neg b_2 \vee \neg do(B\text{-}NoAct), 0.3)$	$(\neg a_1 \vee \neg b_2 \vee \neg do(B\text{-}NoAct) \vee A_3)$
$(\neg a_1 \vee \neg b_2 \vee \neg do(b_1), 1)$	$(\neg a_1 \vee \neg b_2 \vee \neg do(b_1) \vee A_1)$
$(\neg a_2 \vee \neg b_2 \vee \neg do(b_1), 1)$	$(\neg a_2 \vee \neg b_2 \vee \neg do(b_1) \vee A_1)$
Clauses of DO_B	
$\neg do(B\text{-}NoAct)$	$(\neg do(B\text{-}NoAct) \vee B_1)$
$\neg do(b_1)$	$(\neg do(b_1) \vee B_2)$
$\neg do(b_2)$	$(\neg do(b_2) \vee B_3)$

Table 5.6: The CNF encoding $K_{\Sigma_{aug}}$ of Σ_{aug} of ΠG_{aug} of Figure 5.4

Let $do(b_1)$ be the intervention on B. First, we should update the necessity degree of DO_B's parameters. In fact, B_1 and B_3 will encode the degree 1 and B_2 will encode the degree 0. After that, we unify variables A_i and B_i such that each B_1 and B_3 (resp. B_2) appearing in $K_{\Sigma_{aug}}$ is

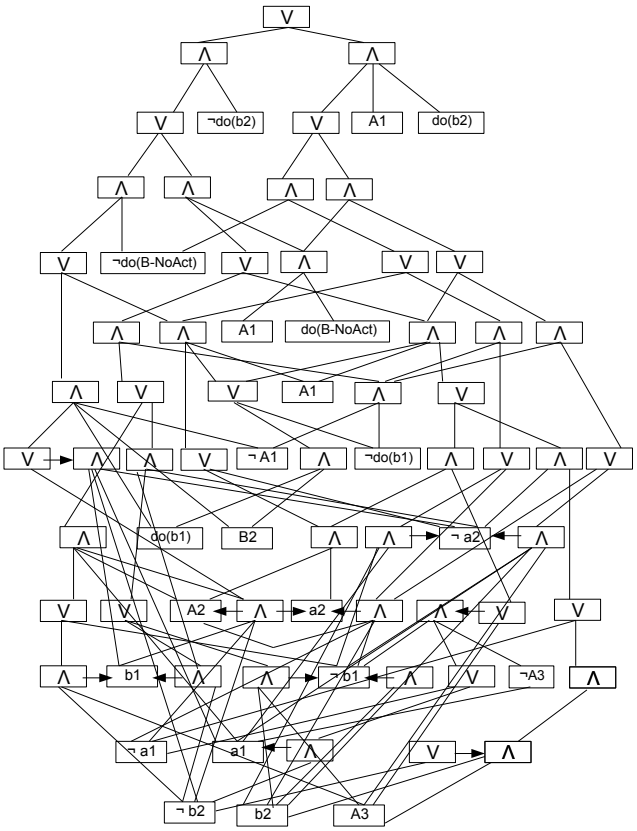


Figure 5.12: The augmented compiled base K_c^{aug}

substituted by A_1 *(resp.* A_4*). The new set of variables is the following:* $A = \{A_1(1), A_2(0.6), A_3(0.3), A_4(0)\}$*. Finally, the computation process of* $\Pi_c(a_2|do(b_1))$ *is ensured as follows:*

- ***Iteration 1****:* $K_c^{aug} \not\models \neg do(b_1) \vee A_1 \Rightarrow (K_c^{aug}|\neg A_1) \not\models a_1 \Rightarrow i \leftarrow i + 1$.
- ***Iteration 2****:* $K_c^{aug} \not\models \neg do(b_1) \vee A_2 \Rightarrow (K_c^{aug}|\neg A_2) \models a_1 \Rightarrow StopCompute \leftarrow$ *true.*

Hence, $\Pi_c(a_2|do(b_1)) = 1 - degree(2) = 1 - 0.6 = 0.4$ *where* $degree(2)$ *designates the necessity degree associated to* A_2*.*

Aug-DNNF-PKB algorithm

The Aug-DNNF-PKB method, outlined by Algorithm 10, guarantees the equivalence between possibility degrees computed using Aug-DNNF-PKB and the joint distribution.

Algorithm 10: Aug-DNNF-PKB

Data: ΠG_{aug}, instance of interest x, observation e, intervention $do(x_I)$
Result: $\Pi_c(y|e, do(x_I))$
begin
  % **Encoding and compilation phase**
  $\Sigma_{aug} \leftarrow$ transform(ΠG_{aug})
  $K_{\Sigma}^{aug} \leftarrow$ encode-PKB(Σ_{aug})
  $K_c^{aug} \leftarrow$ compile(K_{Σ}^{aug})
  % **Inference phase**
    % *A : the set of propositional variables of* $\Sigma_{X_i}, \forall X_i \in V$
    % *B : the set of propositional variables of* Σ_{DO_I}
  $B \leftarrow$ update(B)
  $A = \{A_1, \ldots, A_g\} \leftarrow$ unify(A, B)
  i $\leftarrow 1$, StopCompute $\leftarrow$ false, $\Pi_c(y|e, do(x_I)) \leftarrow 1$
  while *(*$K_c^{aug} \not\models A_i \vee \neg e \vee \neg do(x_I)$*)* ***and*** *(*$i < g$*)* ***and*** *(StopCompute=false)* **do**
    $K_c^{aug}|\neg A_i \leftarrow$ condition(K_c^{aug}, $\neg A_i$)
    if $K_c^{aug}|\neg A_i \models \neg y$ **then**
      StopCompute $\leftarrow$ true
      Let *degree*(i) be the weight associated to A_i
      $\Pi_c(y|e, do(x_I)) \leftarrow$ 1-*degree*(i)
    else $i \leftarrow i + 1$
  return $\Pi_c(y|e, do(x_I))$

Proposition 5.4. *Let* K_c^{aug} *be the compiled base of an augmented network* ΠG_{aug}*. Let* $do(x_I)$ *be an intervention that forces the variable* X_I *to take the value* x_I*.*

i) $\forall \omega \in \Omega$*, we have:*

$$\pi_c(\omega|do(x_I)) = \pi_a(\omega|do(x_I)) \tag{5.18}$$

where $\pi_c(\omega|do(x_I))$ *(resp.* $\pi_a(\omega|do(x_I))$*) is computed using Algorithm 10 (resp. Definition (5.1)).*

ii) Let y *be an instantiation of a variable* $Y \in V$ *and* e *be an instantiation of a set of variables* $E \subseteq V$*. Then:*

$$\Pi_c(y|e, do(x_I)) = \Pi_a(y|e, do(x_I)) \tag{5.19}$$

where $\Pi_c(y|e, do(x_I))$ *(resp.* $\Pi_a(y|e, do(x_I))$*) is computed using Algorithm 10 (resp. Definition (5.1)).*

Proof 5.4. *For any min-based possibilistic network* ΠG_{min}*, from Equation*

(1.15), we have $\pi_{min}(\omega) = \pi_{\Sigma}(\omega)$.
We obtain, $\pi_a(\omega|do(x_I)) = \pi_{\Sigma_{aug}}(\omega|do(x_I))$ *since* $\Pi G_{aug} = \Pi G_{min} \cup DO_I$.
Thus, Equation (5.18) is established.

This result is relative to an interpretation ω*, to generalize it to any instantiation* y *of a variable* $Y \in V$*, we obtain:*
$\max_{\omega \models y} \pi_c(\omega|do(x_I)) = \max_{\omega \models y} \pi_c(\omega|do(x_I))$.
Thus, $\Pi_c(y|do(x_I)) = \Pi_a(y|do(x_I))$.
When we deal with an observation e*, we obtain:*
$\Pi_c(y|e, do(x_I)) = \Pi_a(y|e, do(x_I))$. ■

5.5 Conclusion

In this chapter, we addressed the inference problem in possibilistic causal networks where we handle both observations and interventions. We proposed, at first, mutilated-based approaches consisting in mutilating the compiled bases resulting from compiling CNF bases of possibilistic networks obtained using the strategy one variable per parameter. This allows to efficiently compute the effect of both observations and interventions without re-compiling the network each time an intervention occurs. We also proposed augmented-based approaches that do not follow the same principle than mutilated-based approaches since each node, except the extra node in augmented possibilistic networks, should be encoded using either local structure or possibilistic local structure. However, the new node cannot benefit from the structure exhibited by parameters values. In the next chapter, we will study the inference implementation of product-based possibilistic networks and we will explore the decisional aspect under compilation.

Chapter 6

Beyond compiling min-based possibilistic networks

6.1 Introduction

The previous chapters were dedicated to reasoning with state variables in an ordinal setting. Our objective in this chapter is to study two variants of min-based possibilistic networks, the first one concerns the numerical setting of possibility theory (i.e., product-based possibilistic networks) and the second deals with decision graphical models, namely possibilistic influence diagrams.

More precisely, this chapter proposes the inference implementation of product-based possibilistic networks under compilation. We emphasize on similarities and differences between product-based possibilistic networks, min-based possibilistic networks and Bayesian networks under compilation, while using the same DAG structure. In the second part, we propose to go one step further and explore the decisional aspect. In fact, we extend the compilation concepts to the decisional aspect by evaluating possibilistic influence diagrams [47, 52] which are the possibilistic counterpart of standard influence diagrams [54]. Such decisional models present a compact model representing the decision maker's belief and preferences about a sequence of decisions to be made.

This Chapter is organized as follows: Section 6.2 presents compilation-based inference in product-based possibilistic networks. Section 6.3 is dedicated to the compilation-based evaluation approach of possibilistic influence diagrams. Main results of this Chapter are published in [6, 7].

6.2 Compilation-based inference in product-based possibilistic networks

Previously, our emphasis was placed on inference in min-based possibilistic networks under compilation. The main idea consists in compiling the *Conjunctive Normal Form (CNF)* encoding associated with the possibilistic network into the DNNF language, then compiling it to efficiently compute the effect of an evidence on a set of variables of interest. In this section, we propose to study the numerical counterpart of Π-*DNNF* method using product-based possibilistic networks. The proposed method, denoted by *Prod*-Π-*DNNF*, requires two phases as detailed below:

6.2.1 Encoding and compilation phase

The starting point of Prod-Π-DNNF is the CNF encoding of the product-based possibilistic network ΠG_*, denoted by C_*^{LS}. The difference between compiling min-based and product-based possibilistic networks resides in the required encoding strategy. In fact, when we deal with min-based possibilistic networks, we can go beyond the standard local structure by exploiting *possibilistic local structure* due to the idempotency of the min operator. However in a numerical setting, the conjunctive operator is the product instead of the minimum. For this reason, only the so-called *local structure* can be handled to address specific values of network parameters in a ΠG_*. Formally, parameters $\Pi(x_i|u_i)$ should be encoded using Equation (4.2) and the CNF encoding C_*^{LS} of ΠG_* is given by Definition 4.1.

Exploiting parameters values in the encoding phase has a significant impact on CNF parameters. From a theoretical point of view, we propose two results. The first one concerns the comparison between CNF encodings of product-based possibilistic networks and min-based possibilistic networks, having the same DAG structure. The second result refers to comparing CNF encodings of Bayesian networks and possibilistic networks in the extreme case, i.e., where all uncertainty degrees are different (except 1 in possibility). The choice of such case is argued by the fact that it is not appropriate to compare probabilistic and possibilistic networks even if they share the same DAG structure due to the semantic of their uncertainty degrees. Formally:

Proposition 6.1. *Let ΠG_{min} be a min-based possibilistic network. Let ΠG_* be a product-based possibilistic network sharing the same graphical (i.e., DAG) and numerical (i.e., possibility degrees) components as ΠG_{min}. Let C_{min}^{PLS} (resp. C_*^{LS}) be the CNF encoding of ΠG_{min} (resp. ΠG_*). Then,*

$$Size(C_{min}^{PLS}) < Size(C_*^{LS}) \tag{6.1}$$

where $Size(C)$ denotes the size of the CNF encoding C measured by its number of propositional variables and clauses.

Proof 6.1. *The proof is twofold:*

- *Worst case (all parameters are different per $C\Pi T$ except 1): Let s be the number of parameters per $C\Pi T_i$. Let $card(U_{ij})$ be the cardinality of the parent U_{ij} of any $X_i \in V$, then the number of parameters equal to 1 per $C\Pi T_i$, denoted by nb_1, is equal to $*_{j=\{1..m\}}\ card(U_{ij})$. Let N be the number of conditional possibility tables. Then, the number of parameter variables is equal to $N * (s - nb_1) + N$ (resp. $N * (s - nb_1) + 1$) in C_*^{LS} (resp. C_{min}^{PLS}).*
- *Best case (all parameters are equal to 1): The number of parameter variables is equal to N (resp. 1) in C_*^{LS} (resp. C_{min}^{PLS}).*

$\Rightarrow Size(C_{min}^{PLS}) < Size(C_*^{LS})$. ■

Proposition 6.2. *Let BN be a Bayesian network and $\Pi G_\otimes$ be a possibilistic network having the same DAG structure as BN where $\otimes = \{min, *\}$. Let C_{min}^{PLS}, C_*^{LS} and $C_{p_*}^{LS}$ be the CNF encoding of ΠG_{min}, ΠG_* and BN, respectively. Then, in the extreme case where all parameters values are different except 1 in $\Pi G_\otimes$, we have:*

$$Size(C_{min}^{PLS}) < Size(C_*^{LS}) < Size(C_{p_*}^{LS}) \tag{6.2}$$

Proof 6.2. *We have:*

- *From Proof 6.1,* $Size(C_{min}^{PLS}) < Size(C_{*}^{LS})$.
- *The number of parameter variables is equal to* s *(resp.* $s - nb_1 + 1$*) in* $C_{p_*}^{LS}$ *(resp.* C_{*}^{LS}*). Thus,* $Size(C_{*}^{LS}) < Size(C_{p_*}^{LS})$. ■

Once ΠG_* is logically represented into C_{*}^{LS}, this latter is then compiled into the most succinct language DNNF. The compiled base is denoted by CB.

6.2.2 Inference phase

Given the compiled base CB resulting from Phase 1, an instance of interest x of some variables $X \in V$ and an evidence e of some variables $E \subseteq V$, we can efficiently compute the effect of e on x, namely $\Pi_c(x|e)$. Using Equation (5.3), it is clear that we should compute both of $\Pi_c(x, e)$ and $\Pi_c(e)$ while following these three steps:

Updating instance indicators

This step serves to record the instance of interest x and the evidence e into instance indicators λ_{x_i}. It corresponds to conditioning the compiled base CB on e and x. The conditioned compiled base is denoted by $[CB|x, e]$.

Mapping logical representation into a product-based representation

In this step, we transform the logical compiled base resulting from the previous step into a valued representation. In this step, it is important to note that the valued compiled bases, named *arithmetic circuits* and used in the probabilistic method [30] cannot be applied in our case since its operators (i.e., * and +) are different from those that should be used in the product-based case (i.e., * and max). For this reason, we propose to use a new *prod-max circuit* suitable for the product-based case. In fact, given the conditioned compiled base resulting from the previous step, we should apply the following operations: (i) replace $\vee$ and $\wedge$ by max and *, respectively, (ii) substitute each $\top$ (resp. $\bot$) by 1 (resp. 0) and (iii) associate $\Pi(x_i|u_i)$ to each parameter variable.

Definition 6.1. *A prod-max circuit of a DNNF sentence CB, denoted by CB^{LS}_{*max}, is a valued sentence where $\wedge$ and $\vee$ are substituted by * and max, respectively. Leaf nodes correspond to indicator and parameter variables, internal nodes correspond to max and * operators, and the root corresponds to the circuit output.*

It is obvious that the mapping from logical to numerical representation is established in a polynomial time since it corresponds to a set of trivial substitution operations. The use of *valued DNNFs* with max and * operators (i.e., prod-max circuits relative to product-based possibilistic networks) differs from the probabilistic case since inference in Bayesian networks requires *valued d-DNNFs* with + and * operators (i.e., arithmetic circuits). This is essentially due to the fact that probabilistic computations require the determinism property (i.e., d) to ensure polytime model counting [34]. Following the succinctness relation between DNNF and d-DNNF stating that DNNF is strictly more succinct than d-DNNF [27], we can give this important result, asserting that from a theoretical point of view, prod-max circuits are considered more compact than arithmetic circuits.

Proposition 6.3. *Let BN be a Bayesian network and CB^{LS}_{*+} be its arithmetic circuit using the probabilistic compilation method proposed in [30]. Let ΠG_* be a product-based possibilistic network sharing the same DAG structure as BN. Let CB^{LS}_{*max} be the prod-max circuit of ΠG_*. Then, from a theoretical point of view, we have:*

$$Size(CB^{LS}_{*max}) < Size(CB^{LS}_{*+}) \tag{6.3}$$

Proof 6.3. *From the knowledge map of [34], DNNF is strictly more succinct than d-DNNF. From a numerical point of view, prod-max circuit CB^{LS}_{*max} is a valued DNNF and arithmetic circuits CB^{LS}_{*+} are valued d-DNNF. Thus, $Size(CB^{LS}_{*max}) < Size(CB^{LS}_{*+})$.* ■

Computation

The prod-max circuit is a special case of VNNFs where operators only restrict to max and *. In literature, prod-max circuits have been used under different notations to answer different queries. In fact, in [22], authors have explored circuits with max and *, called *maximizer circuits* to answer the *Most Probable Explanation*[1] (MPE) probabilistic query. While in [19], authors proposed *decision circuits* which add the operator + to max and * of

[1] A MPE is a complete variable instantiation with the highest probability given the current evidence.

maximizer circuits in order to *evaluate influence diagrams*. In our case, we are interested in computing $\Pi_c(x, e)$ and $\Pi_c(e)$ by evaluating CB^{LS}_{*max}. Even we are in presence of a prod-max circuit similar to maximizer and decision circuits, we cannot use neither MPE nor evaluation of influence diagrams queries to compute $\Pi_c(x, e)$ and $\Pi_c(e)$. In fact, evaluating CB^{LS}_{*max} consists in applying max and * operators in a bottom-up way. Inference is guaranteed to be established in polytime since it corresponds to a simple propagation from leaves to root, and more precisely to a *max-variable elimination* operation.

It is worth pointing out that the query used for evaluating CB^{LS}_{*max} does not correspond to a *model counting problem* as in the probabilistic case [30]. This means that under compilation product-based possibilistic networks are not close to probabilistic ones and do not share the same features as Bayesian networks.

Example 6.1. *Considering the product-based possibilistic network ΠG_* of Figure 1.2. Column 2 of Table 4.5 represents its CNF encoding C^{LS}_* using Definition 4.1 and Figure 6.1 represents its prod-max circuit.*

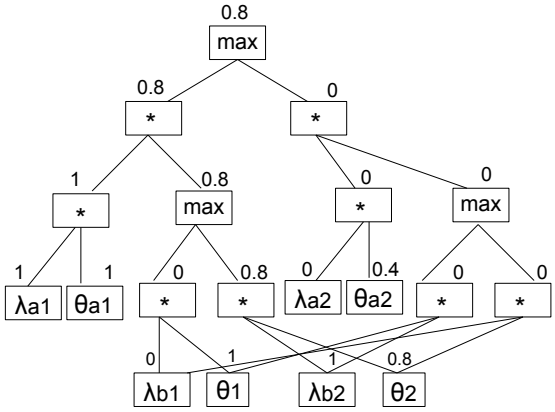


Figure 6.1: The prod-max circuit CB^{LS}_{*max}

Let us compute the effect of the evidence a_1 on the instance of interest b_2 (i.e., $\Pi_c(b_2|a_1)$). To compute $\Pi_c(a_1, b_2)$, we should first record a_1 and b_2 into instance indicators by setting λ_{a_1} and λ_{b_2} (resp. λ_{a_2} and λ_{b_1}) to $\top$ (resp. $\bot$).

*By mapping the logical compiled base into a prod-max circuit and applying max and * in a bottom-up fashion as shown in Figure 6.1, we can deduce that*

$\Pi_c(a_1, b_2)$ *is equal to* 0.8. *This value corresponds to the one of Table 1.6 using the product-based joint distribution. Hence,* $\Pi_c(b_2|a_1) = \frac{0.8}{1} = 0.8$.

6.3 Evaluation of possibilistic influence diagrams

Above, we have explored compilation-based inference in min-based and product-based possibilistic networks which are dedicated to reasoning with state variables. In this section, we will go one step further and explore the decisional aspect. In fact, we will take advantage of the transformation of possibilistic influence diagrams into possibilistic networks [52] and our results on compilation of min-based possibilistic networks to evaluate possibilistic influence diagrams using compilation. We start by defining influence diagrams, then we move to the evaluation algorithm under compilation.

6.3.1 Possibilistic influence diagrams

Possibilistic influence diagrams [47] are possibilistic counterpart of standard influence diagrams [54] presenting a compact popular framework modeling a decision maker's belief and preferences about a sequence of decisions to be made.

A possibilistic influence diagram, denoted by IIID, has two components: a graphical and a numerical ones while adding decision and utility nodes. Formally, a IIID is a DAG containing three kinds of nodes:

- *Decision* nodes $D = \{D_1, \cdots, D_p\}$, drawn as rectangles, representing decision variables which have necessarily a temporal order meaning that D_i succeeds D_{i-1} and precedes D_{i+1},
- *Chance* nodes $C = \{C_1, \cdots, C_n\}$, drawn as circles, representing state variables (relevant uncertain factors for the decision problem),
- *Utility* nodes $V_u = \{V_1, \cdots, V_q\}$, drawn as diamonds, representing utilities to be maximized.

A combination $c = \{c_1, \cdots, c_n\}$ of state variables represents a state. A combination $d = \{d_1, \cdots, d_p\}$ of values represents a decision. $pa(D_i)$ denotes parents of D_i comprising decision and chance variables.

Chance nodes are quantified by conditional possibility distributions, while decision nodes are not quantified. Regarding utility nodes, possibility theory offers several ways to represent preferences of the decision maker, e.g. binary

utilities [49], ordinal utilities [63], cardinal utilities [2]. A *strategy* σ for a IIID specifies the value of every decision variable as a function of all (or part of) the known values of the variables at the time the decision is taken.

Example 6.2. *Let us consider the possibilistic influence diagram of Figure 6.2. It is composed of two chance nodes (A and B), one decision node (D) and one utility node (V). As shown in this figure, both of A and B are quantified using conditional possibility distributions, this is not the case of the decision node D. Regarding the utility node V, preferences of the decision maker are represented using ordinal utilities by means of the following preference order:* $(D = Act_2 \wedge A = a_2) \geq (D = Act_1 \wedge A = a_1) \geq (D = Act_1 \wedge A = a_2) \geq (D = Act_2 \wedge A = a_1)$.

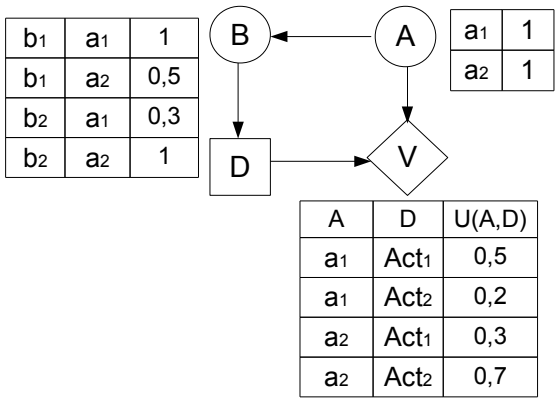


Figure 6.2: A possibilistic influence diagram

Commonly, two structural assumptions on the DAG structure are considered: (i) utility nodes have no children and (ii) there exists a directed path comprising all decision nodes. Furthermore, we accept the *no forgetting assumption*, which implies that all values of variables that have been instantiated before d_i is chosen are still known at the time of the choice of d_i. Different combinations between the quantification of chance and utility nodes offer several kinds of possibilistic influence diagrams with different semantics [52].

Contrarily to standard influence diagrams where evaluation means maximizing decision maker's utilities using the MEU criterion. In the possibility theory framework, we can evaluate the same IIID using a panoply of possibilistic decision criteria in order to obtain the optimal strategy σ^*. We can, in particular, mention these criteria: *possibilistic qualitative utilities*, *possibilistic likely dominance* and *Possibilistic choquet integrals*. Obviously, the choice of the decision criteria depends on the semantic underlying the possibilistic influence diagram on hand. In our current proposal, we maintain the

same qualitative framework used in the second part of this thesis which implies that only qualitative possibilistic decision criteria can be used under the assumption that the utility scale and the possibility scale are commensurable and purely ordinal.

6.3.2 Compilation-based evaluation of possibilistic IDs

Evaluating possibilistic influence diagrams can be done in a direct manner or in an indirect one (i.e., by transforming it into a secondary structure). Regarding indirect methods, [47] proposed to transform them into *decision trees*, while [52] transformed them into *possibilistics networks*. The idea here is to take advantage of refined compilation-based inference methods proposed in the second part in order to evaluate min-based ΠIDs. In fact, we first re-use the polynomial *transformation* phase of [52] in order to morph the initial min-based possibilistic influence diagram into a min-based possibilistic network and propose an *evaluation* phase in which we generate the optimal strategy as depicted by Figure 6.3.

The principle of the transformation phase is to transform each decision node into a chance node quantified via a uniform possibility distribution stating a total ignorance (i.e., all values are equal to 1). Besides, each utility node should be converted into a new binary chance node having two values, i.e., $\{True(T), False(F)\}$ and will be evaluated via a possibility distribution generated from the original preference relation. The resulting network is denoted by ΠG^{ID}_{min}. In the evaluation phase, we propose to encode at first the possibilistic network ΠG^{ID}_{min} issued from the transformation phase into a CNF base, then compile the resulting encoding only once in order to generate the optimal decision strategy.

In order to illustrate our approach, we propose to resort to the ordinal counterpart of expected utility [41] which offers an optimistic and a pessimistic variants defined by:

$$u_{opt}(\sigma) = \max_{c \in C} \ min(\pi_\sigma(c), \mu_\sigma(c)) \tag{6.4}$$

$$u_{pes}(\sigma) = \min_{c \in C} \ max(n^2(\pi_\sigma(c), \mu_\sigma(c))) \tag{6.5}$$

where $u_{opt}(\sigma)$ (resp. $u_{pes}(\sigma)$) corresponds to an optimistic (resp. pessimistic) attitude in front of uncertainty, π_σ represents the possibility distribution on C and μ_σ denotes the possibility distribution associated to decision maker's preferences using the utility scale.

[2] The order-reversing map of the possibility scale.

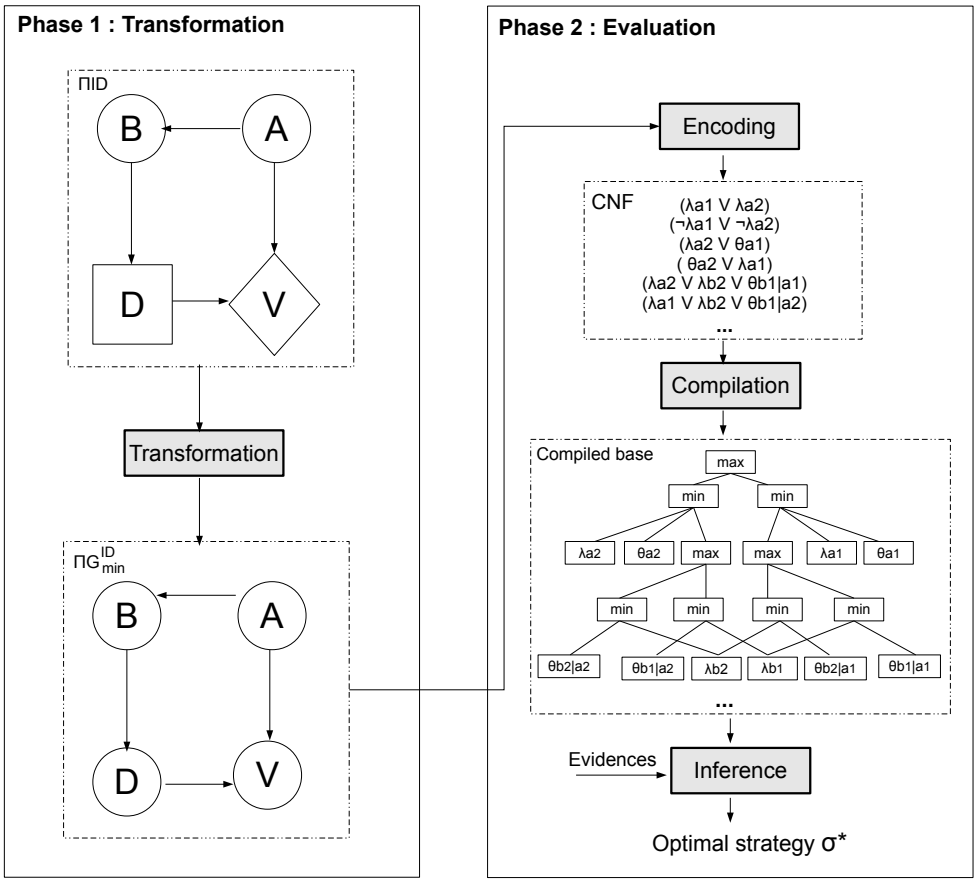


Figure 6.3: Principle of compilation-based evaluation

Let E be a set containing chance nodes with known values (i.e., evidences) including those decisions $\{D_1, ..., D_{i-1}\}$ in D that have been already made. Let D_i represent the remaining decision to be made which means that all variables in $pa(D_i)$ are already instantiated and are therefore in E. At this stage, we should determine the optimal instantiation of D_i depending on the selected criterion (U_{opt} or U_{pes}) as follows:

$$U^*_{opt}(D_i, E) = \max_{D_i} \Pi(V = T|D_i, E) \tag{6.6}$$

$$U^*_{pes}(D_i, E) = \min_{D_i} \Pi(V = T|D_i, E) \tag{6.7}$$

Roughly speaking, Equations (6.6) and (6.7) should be applied recursively so that to fix at each stage i ($i = 1, \ldots, p$) the optimal decision

regarding D_i and to consider it as an evidence for the remaining stages $(i+1,\ldots,p)$.

To ensure this computation step, we will use the min-max circuit (i.e., the already compiled base) associated to ΠG^{ID}_{min} in a bottom-up way. The advantage behind such evaluation is twofold: (i) the expensive compilation phase is performed only once on ΠG^{ID}_{min} and (ii) computation is performed in a polynomial time in the circuit size due to the polytime compilation-based inference. It is also important to point out that there is no need to re-define a new circuit appropriate to decision problems under qualitative uncertainty in possibility theory framework since evaluating min-based possibilistic influence diagrams requires the same aggregation operators (i.e., max and min) as those used to infer min-based possibilistic networks.

Example 6.3. *Let us re-consider the possibilistic influence diagram of Figure 6.2. We suppose that we are in an optimistic setting i.e., the decision criterion is* U_{opt}.

To transform the possibilistic ID into a possibilistic network, we should first transform the decision node D into a chance node by setting each $\Pi(d_i|B)$ *to 1 where* $d_i = \{Act_1, Act_2\}$. *Then, we should transform the utility node V into a binary chance one. The possibility distribution associated to the new chance node is presented in Table 6.1.*

V	A	D	$\Pi(V\|A,D)$
T	a_1	Act_1	0.5
T	a_1	Act_2	0.2
T	a_2	Act_1	0.3
T	a_2	Act_2	0.7
F	a_1	Act_1	1
F	a_1	Act_2	1
F	a_2	Act_1	1
F	a_2	Act_2	1

Table 6.1: The possibility distribution of the utility node

Once we obtain ΠG^{ID}_{min}, *we should at first encode it into a CNF base. To illustrate the encoding, we use Definition 4.4 to obtain* C^{PLS}_{min} *as shown in Table 6.2 where* θ_1, θ_2, θ_3, $\theta_{V=T|a_1,Act_2}$ *and* $\theta_{V=T|a_2,Act_2}$ *encode, respectively possibility degrees 1, 0.5, 0.3, 0.2 and 0.7. Then, we should compile* C^{PLS}_{min} *in a min-max circuit as depicted by Figure 6.4.*

Suppose that we receive a certain information saying that the variable A takes the value a_2, *then using the optimistic attitude (Equation (6.6)) and the min-max circuit of Figure 6.4, we should compute the maximum of*

Mutual exclusive clauses	
Variables	**Clauses**
A	$(\lambda_{a_1} \vee \lambda_{a_2}) \wedge (\neg\lambda_{a_1} \vee \neg\lambda_{a_2})$
B	$(\lambda_{b_1} \vee \lambda_{b_2}) \wedge (\neg\lambda_{b_1} \vee \neg\lambda_{b_2})$
D	$(\lambda_{Act_1} \vee \lambda_{Act_2}) \wedge (\neg\lambda_{Act_1} \vee \neg\lambda_{Act_2})$
V	$(\lambda_{V=T} \vee \lambda_{V=F}) \wedge (\neg\lambda_{V=T} \vee \neg\lambda_{V=F})$
Parameter clauses of A	
Possibility degrees	**Clauses**
$\Pi(a_1) = 1$	$\lambda_{a_1} \rightarrow \theta_1$
$\Pi(a_2) = 1$	$\lambda_{a_2} \rightarrow \theta_1$
Parameter clauses of B	
$\Pi(b_1\|a_1) = 1$	$\lambda_{a_1} \wedge \lambda_{b_1} \rightarrow \theta_1$
$\Pi(b_1\|a_2) = 0.5$	$\lambda_{a_2} \wedge \lambda_{b_1} \rightarrow \theta_2$
$\Pi(b_2\|a_1) = 0.3$	$\lambda_{a_1} \wedge \lambda_{b_2} \rightarrow \theta_3$
$\Pi(b_2\|a_2) = 1$	$\lambda_{a_2} \wedge \lambda_{b_2} \rightarrow \theta_1$
Parameter clauses of D	
$\Pi(Act_1\|b_1) = 1$	$\lambda_{Act_1} \wedge \lambda_{b_1} \rightarrow \theta_1$
$\Pi(Act_1\|b_2) = 1$	$\lambda_{Act_1} \wedge \lambda_{b_2} \rightarrow \theta_1$
$\Pi(Act_2\|b_1) = 1$	$\lambda_{Act_2} \wedge \lambda_{b_1} \rightarrow \theta_1$
$\Pi(Act_2\|b_2) = 1$	$\lambda_{Act_2} \wedge \lambda_{b_2} \rightarrow \theta_1$
Parameter clauses of V	
$\Pi(V = T\|a_1, Act_1) = 0.5$	$\lambda_{V=T} \wedge \lambda_{a_1} \wedge \lambda_{Act_1} \rightarrow \theta_2$
$\Pi(V = T\|a_1, Act_2) = 0.2$	$\lambda_{V=T} \wedge \lambda_{a_1} \wedge \lambda_{Act_2} \rightarrow \theta_{V=T\|a_1,Act_2}$
$\Pi(V = T\|a_2, Act_1) = 0.3$	$\lambda_{V=T} \wedge \lambda_{a_2} \wedge \lambda_{Act_1} \rightarrow \theta_3$
$\Pi(V = T\|a_2, Act_2) = 0.7$	$\lambda_{V=T} \wedge \lambda_{a_2} \wedge \lambda_{Act_2} \rightarrow \theta_{V=T\|a_2,Act_2}$
$\Pi(V = F\|a_1, Act_1) = 1$	$\lambda_{V=F} \wedge \lambda_{a_1} \wedge \lambda_{Act_1} \rightarrow \theta_1$
$\Pi(V = F\|a_1, Act_2) = 1$	$\lambda_{V=F} \wedge \lambda_{a_1} \vee \lambda_{Act_2} \rightarrow \theta_1$
$\Pi(V = F\|a_2, Act_1) = 1$	$\lambda_{V=F} \wedge \lambda_{a_2} \vee \lambda_{Act_1} \rightarrow \theta_1$
$\Pi(V = F\|a_2, Act_2) = 1$	$\lambda_{V=F} \wedge \lambda_{a_2} \wedge \lambda_{Act_2} \rightarrow \theta_1$

Table 6.2: The CNF encoding C_{min}^{PLS} of ΠG_{min}^{ID}

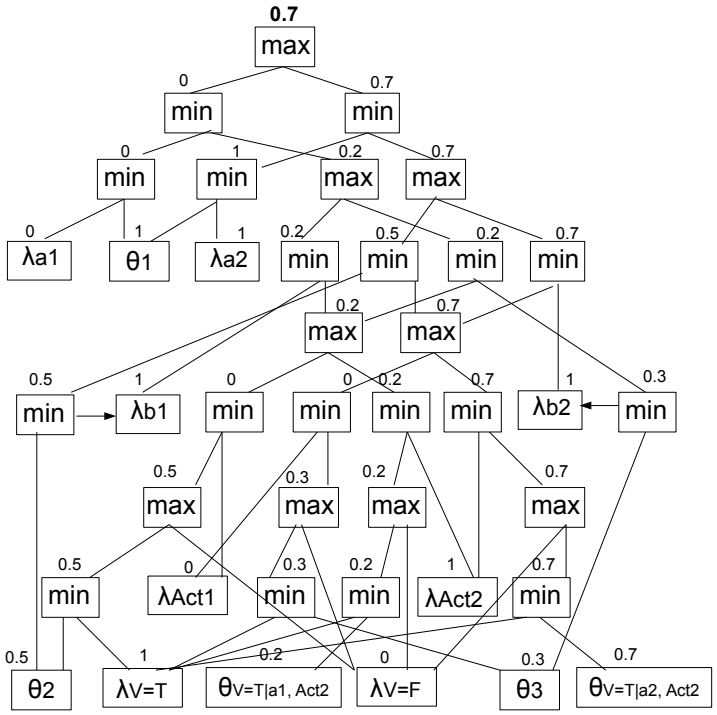


Figure 6.4: The min-max circuit of ΠG^{ID}_{min}

$\Pi(V = T|Act_1, a_2)$ *and* $\Pi(V = T|Act_2, a_2)$. *Figure 6.4 depicts the computation process of* $\Pi(V = T|Act_2, a_2)$ *by setting* $\lambda_{V=T}$, λ_{a_2}, λ_{Act_2}, λ_{b_1}, λ_{b_2} *to 1;* $\lambda_{V=F}$, λ_{Act_1}, λ_{a_1} *to 0 and applying min and max operators in a bottom-up way. The root value i.e., 0.7 corresponds to* $\Pi(V = T|Act_2, a_2)$. *The possibility degree* $\Pi(V = T|Act_1, a_2)$ *which is equal to 0.3 is computed in the same spirit by setting* λ_{Act_1} *to 1 and* λ_{Act_2} *to 0. Thus,* $U^*_{opt}(D, E) = 0.7$, *which means that the optimal decision is* Act_2.

6.4 Conclusion

In this Chapter, we have studied compilation-based inference using product-based possibilistic networks. The encoding strategy that should be used in this case is *local structure*. This means that possibilistic local structure is not tolerated in a numerical setting, while min-based possibilistic networks can benefit from both local structure and possibilistic local structure. We also focused on theoretical common points between probabilistic and possibilistic approaches and unveil the differences between them under compilation. Interestingly enough, our experimental study shows that the product-based approach outperforms the probabilistic approach and this is especially due

to the particularity of possibility values and the normalization constraint in possibility theory. Finally, we explored the decisional aspect by developing a compilation-based evaluation method of possibilistic influence diagrams, mainly based on a compilation-based inference method of min-based possibilistic networks. In the next chapter, we will study inference methods for min-based possibilistic networks and product-based possibilistic networks from an experimental point of view.

Chapter 7

Implementation and Experimentations

7.1 Introduction

This Chapter proposes an experimental study aiming to compare possibilistic compilation-based inference algorithms proposed throughout this thesis in terms of CNF parameters, compiled bases parameters and the inference time w.r.t the one of the junction tree method. Also, a comparison between mutilated-based approaches and augmented-based approaches will be established. To this end, we implement different CNF encodings of possibilistic networks using Matlab R2010, then compile these latters using the state of the art c2d compiler[1] [29, 31] and finally implement inference using the resulting compiled bases. Experiments ran on a 2.27 GHz Core i3 processor with 4 GO of memory.

This Chapter is organized as follows: Section 7.2 gives details of the used experimental protocol. Section 7.3 provides experimental results regarding n-ary and binary approaches proposed in Chapters 3 and 4 and compares the inference time w.r.t the standard junction tree method. Section 7.4 compares mutilated-based approaches and augmented-based approaches proposed in Chapter 5. Section 7.5 emphasizes on differences regarding product-based possibilistic networks and Bayesian networks behaviors under compilation.

[1] Available at http://reasoning.cs.ucla.edu/c2d/.

7.2 Experimental protocol

Our objective is to study the behavior of our compilation-based inference methods from an experimental point of view. Figure 7.1 is a summary of different methods compared in this Chapter.

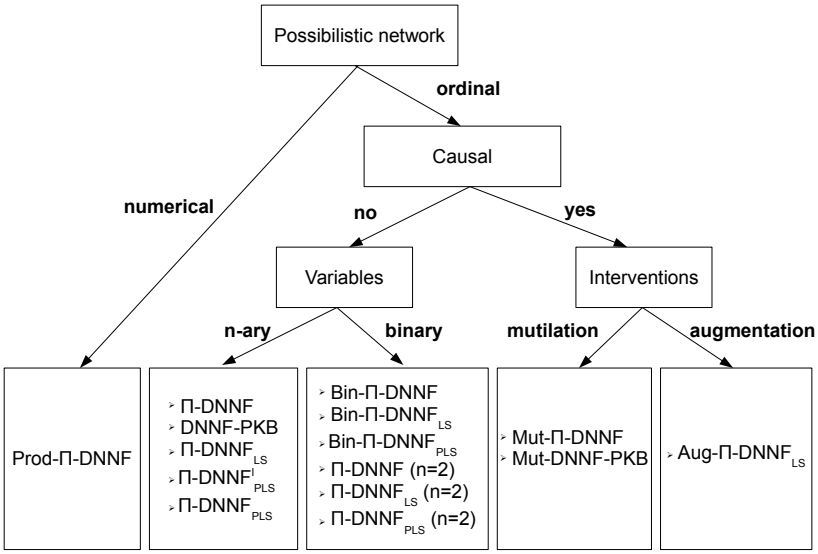


Figure 7.1: Summary of studied methods

In our comparison, we use the following criteria: number of propositional variables and clauses of CNF encodings, number of edges of compiled bases and the inference time.

This section gives relevant variables of our experimentation. As possibilistic networks have a graphical component and a numerical one, then it is judicious to specify which kind of networks to use during the experimental process.

7.2.1 DAG structure

As shown in Sections 3.4, 4.2 and 4.3, our compilation-based inference approaches are very sensitive to parameters since they depend on the occurrence number of parameters per table $C\Pi T_i$ or tables $C\Pi T$. The primordial question that should be explored, at first, is the following: *Should we vary both of DAG structures and possibility distributions parameters to have a pertinent experimentation?* The answer can be found in the following example.

Example 7.1. *Let us consider Figure 7.2 containing three DAG structures of a possibilistic network composed of three binary nodes A, B and C. Clearly, networks (a) and (b) require the same number of parameters (i.e., 10), contrarily to (c) which requires 12 parameters.*

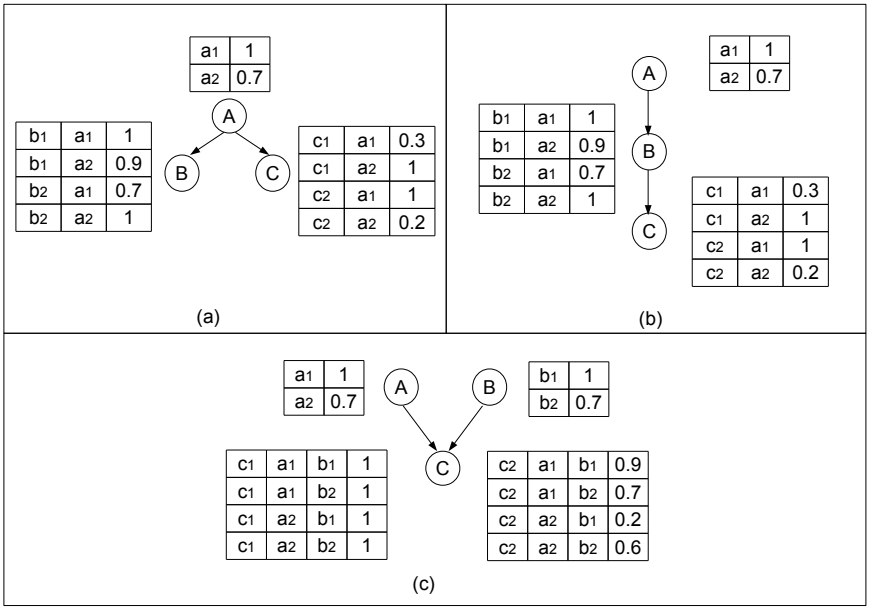


Figure 7.2: Three-nodes DAG structures

Table 7.1 shows the behaviors of Π*-DNNF*$^{I}_{PLS}$, Π*-DNNF*$_{PLS}$ *and DNNF-PKB in terms of number of variables, clauses and edges.*

We can see that the number of variables, clauses and edges of network (c) in the three methods are higher than those of networks (a) and (b). This is especially due to the number of parents per node, which has an impact on the size of conditional possibility tables. Moreover, even if possibility distributions of (a) and (b) share the same degrees (i.e., 1, 0.9, 0.7, 0.3, 0.2), CNF and compiled bases parameters are not the same. This means that the DAG structure influences the size of compiled bases, then we should take into account this information in our interpretation.

It is clear from Example 7.1 that varying DAG structures can distort the interpretation. Hence, we propose to fix the DAG structure and vary parameters. In fact, we generate randomly a possibilistic network by setting the number of nodes to 50, the maximum number of parents per node to 3, the number of instances per variable to 2 and the number of roots to 10.

Network	Method	Variables	Clauses	Edges
(a)	Π-DNNF$^{l}_{PLS}$	10	20	63
	Π-DNNF$_{PLS}$	10	16	58
	DNNF-PKB	9	11	45
(b)	Π-DNNF$^{l}_{PLS}$	10	20	59
	Π-DNNF$_{PLS}$	10	16	56
	DNNF-PKB	9	11	49
(c)	Π-DNNF$^{l}_{PLS}$	11	27	70
	Π-DNNF$_{PLS}$	11	18	59
	DNNF-PKB	10	12	52

Table 7.1: CNF and compiled bases parameters of the network of Figure 7.2

7.2.2 DAG parameters

Given a random possibilistic network, we vary values of possibility distributions (except for the normalization value 1) using two parameters:

1. ($EP_{C\Pi T}$ (%)): *the percent of equal parameters within conditional possibility tables (i.e., $C\Pi T$)*,
2. ($EP_{C\Pi T_i}$): *the occurrence number of a parameter in a conditional possibility table $C\Pi T_i$ relative to X_i.*

We set $EP_{C\Pi T}$ to $\{0\%, 10\%, 30\%, 50\%, 70\%, 100\%\}$. When $EP_{C\Pi T}$ is equal to 50%, this means that each possibility degree appears in 50% of $C\Pi T$. The extreme case 0% states that each possibility degree, except for 1, appears in a unique conditional possibility table, i.e., $C\Pi T_i$. While the case of 100% means that there are two degrees, including the normalization one, which appear in all conditional possibility tables, i.e., $C\Pi T$.

Of course when we affect equal parameters per $C\Pi T$, we should specify which tables are involved by $EP_{C\Pi T}$. Example 7.2 shows that the location of parameters may influence results.

Example 7.2. *Let us consider Figure 7.3 containing three quantifications of a possibilistic network composed of four binary nodes A, B, C and D. The possibility degree 0.7, shown in bold, appears in variables $\langle A, B, C\rangle$, $\langle A, B, D\rangle$ and $\langle B, C, D\rangle$ in networks (a), (b) and (c), respectively.*

Table 7.2 shows the behavior of Π-DNNF$^{l}_{PLS}$, Π-DNNF$_{PLS}$ and DNNF-PKB in terms of number of variables, clauses and edges. We can point out that edges of network (c) in the three methods are higher than those of networks (a) and (b). This is especially due to the table $C\Pi T_i$ affected by

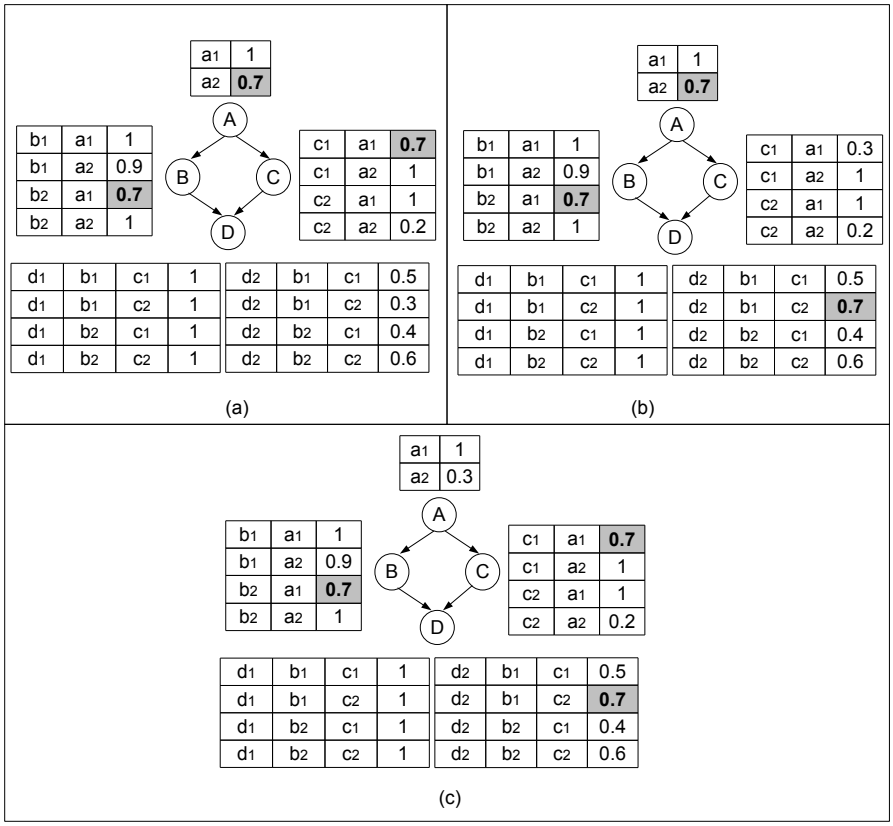


Figure 7.3: Three different locations of a redundant degree

the degree 0.7 and consequently, number of parents of the current node. This means that even with the same DAG structure and the same $EP_{C\Pi T}$ (70 %), the location of parameters affects results. So, it is opportune to take into account this information when affecting equal parameters per $C\Pi T$.

Example 7.2 points out that results depend on locations of equal parameters per tables. In order to vary parameters positions, we propose to generate randomly indexes of tables involved by $EP_{C\Pi T}$. We perform this process 100 times, for each percentage of $EP_{C\Pi T}$.

The parameter $EP_{C\Pi T}$ measures the amount of possibilistic local structure. To take into consideration local structure, we use the parameter $EP_{C\Pi T_i}$ having two values, namely:

- (1): *Each possibility degree appears once in any conditional possibility table $C\Pi T_i$ (except the degree 1 needed for normalization).*
- (> 1): *Each possibility degree appears more than once in a random*

Network	Method	Variables	Clauses	Edges
(a)	Π-DNNF$^{l}_{PLS}$	16	26	97
	Π-DNNF$_{PLS}$	16	42	138
	DNNF-PKB	15	17	87
(b)	Π-DNNF$^{l}_{PLS}$	16	26	98
	Π-DNNF$_{PLS}$	16	41	136
	DNNF-PKB	15	17	88
(c)	Π-DNNF$^{l}_{PLS}$	16	26	109
	Π-DNNF$_{PLS}$	16	40	148
	DNNF-PKB	15	17	93

Table 7.2: CNF and compiled bases parameters of the network of Figure 7.3

manner in the conditional possibility table $C\Pi T_i$.

Clearly, the first case (1) deals with possibilistic local structure since parameters are only redundant per $C\Pi T$, contrarily to the second case (> 1) that exploits local structure by dealing with equal parameters per $C\Pi T_i$.

7.3 Comparing compilation-based inference approaches

Using the experimental protocol described in the previous section, we start with comparing n-ary approaches, then we move to binary ones. More precisely, we compare CNF and compiled bases parameters averaged over 100 different randomly generated parameters locations. Interestingly enough, we establish a comparison covering the inference time of our inference approaches and the well known *junction tree* method, averaged over 30 different randomly generated evidence sets.

7.3.1 n-ary approaches

The experimental results of n-ary approaches are shown in Table 7.3. A row presents results of an approach in terms of CNF parameters and compiled bases edges, while columns correspond to $EP_{C\Pi T_i} = 1$ and $EP_{C\Pi T_i} > 1$.

We note that we have studied both cases of $EP_{C\Pi T_i}$ for Π-DNNF$_{LS}$ method (row 2 of Table 7.3) since it is sensitive to the number of equal parameters per table $C\Pi T_i$.

		$EP_{C\Pi T_i} = 1$				$EP_{C\Pi T_i} > 1$			
Method	$EP_{C\Pi T}$	Variables	Clauses	Edges	Inference time (sec)	Variables	Clauses	Edges	Inference time (sec)
Π-DNNF	0 10 30 50 70 100	358	1048	3428	0.489	-	-	-	-
Π-DNNF$_{LS}$	0	278	684	**2504**	**0.377**	211	372	**963**	**0.219**
	10	276	668	**2412**	**0.367**	210	371	**922**	**0.213**
	30	271	632	**2353**	**0.356**	209	365	**885**	**0.208**
	50	262	574	**2121**	**0.306**	206	368	**868**	**0.201**
	70	254	520	**1951**	**0.295**	201	368	**858**	**0.180**
	100	200	358	**1148**	**0.235**	197	362	**855**	**0.176**
Π-DNNF$^{l}_{PLS}$	0	179	473	97618	34.501	-	-	-	-
	10	127	362	255311	221.883				
	30	112	362	31811	6.528				
	50	108	362	7716	1.097				
	70	107	362	3948	0.432				
	100	102	358	608	0.152				
Π-DNNF$_{PLS}$	0	179	358	76695	32.254	-	-	-	-
	10	127	358	251712	212.355				
	30	112	358	30151	5.775				
	50	108	358	7232	0.859				
	70	107	358	3856	0.287				
	100	102	358	608	0.152				
DNNF-PKB	0	**178**	**229**	76414	31.563	-	-	-	-
	10	**126**	**229**	245368	201.809				
	30	**111**	**229**	30003	4.701				
	50	**107**	**229**	7019	0.748				
	70	**106**	**229**	3623	0.269				
	100	**101**	**229**	502	0.138				

Table 7.3: Π-DNNF vs Π-DNNF$_{LS}$ vs Π-DNNF$^{l}_{PLS}$ vs Π-DNNF$_{PLS}$ vs DNNF-PKB (better values are in bold)

A deep analysis of these results are established for each criterion separately. We consider $Var(method)$ and $Cl(method)$ the number of variables and clauses of the CNF encoding and $Edg(method)$ (resp. $Inf(method)$) the number of edges of the compiled base (resp. the inference time using the compiled base).

CNF variables

Let us analyze the variables behavior of each method, depicted by Figure 7.4.

- Π-*DNNF*: Row 1 of Table 7.3 shows that the number of variables remains unaltered even if $EP_{C\Pi T}$ is rising. Obviously, this is an expected result since Π-DNNF does not take into consideration any numerical value by encoding each possibility degree by a parameter variable, regardless of its value.

- Π-$DNNF_{LS}$: The variables gain is equal to 1.411 as shown in Table 7.4. This proves that local structure exploited in Π-DNNF$_{LS}$ has a positive impact on CNF variables since equal parameters, especially the normalization degree 1, are encoded by the same parameter variable θ_i. It is also clear that the number of variables is reduced for each increase of $EP_{C\Pi T}$, as shown in Figure 7.4.

  Column 2 of Π-DNNF$_{LS}$'s row shows that the variables gain is equal to 1.246. Effectively, when we consider the case of $EP_{C\Pi T_i} > 1$, the number of equal parameters, other than the normalization degree 1, per $C\Pi T_i$ increases and consequently the number of parameter variables is reduced.

- Π-$DNNF^{l}_{PLS}$ *and* Π-$DNNF_{PLS}$: Both of these methods have the same number of variables since they use the same encoding strategy. In comparison to Π-DNNF$_{LS}$, the variables gain is equal to 2.151, as shown in Table 7.4. This proves that CNF variables are increasingly reduced since equal parameters per $C\Pi T$ are increased for each percentage of $EP_{C\Pi T}$. When $EP_{C\Pi T} = 100\%$, the number of variables is equal to 102 since we have 100 indicator variables and 2 parameter variables.

- *DNNF-PKB*: Row 5 of Table 7.3 shows that the number of variables of DNNF-PKB is reduced by one comparing to those of Π-DNNF$^{l}_{PLS}$ and Π-DNNF$_{PLS}$. Indeed, the possibility degree equal to 1 in Π-DNNF$^{l}_{PLS}$ and Π-DNNF$_{PLS}$ is not encoded in DNNF-PKB since it corresponds to a necessity degree equal to 0, which is not represented in possibilistic knowledge bases.

$EP_{C\Pi T}$	$\frac{Var(\Pi-DNNF)}{Var(\Pi-DNNF_{LS})}$	$\frac{Var(\Pi-DNNF_{LS(=1)})}{Var(\Pi-DNNF_{LS(>1)})}$	$\frac{Var(\Pi-DNNF_{LS})}{Var(\Pi-DNNF^{l}_{PLS})}$	$\frac{Var(\Pi-DNNF^{l}_{PLS})}{Var(\Pi-DNNF_{PLS})}$	$\frac{Var(\Pi-DNNF_{PLS})}{Var(DNNF-PKB)}$
0	1.287	1.317	1.553	1	1.005
10	1.297	1.314	2.173	1	1.007
30	1.321	1.296	2.419	1	1.009
50	1.366	1.271	2.425	1	1.009
70	1.409	1.263	2.373	1	1.009
100	1.790	1.015	1.960	1	1.009
$ratio_{Var}$	1.411	1.246	**2.151**	1	1.008

Table 7.4: Variables ratio

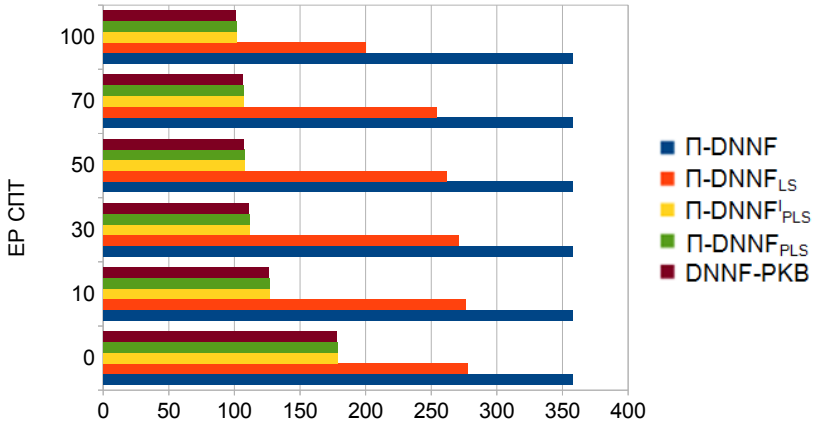


Figure 7.4: Number of variables

CNF clauses

The behavior of CNF clauses of each compilation-based method of Table 7.3 is depicted by Figure 7.5 and detailed below:

- Π-*DNNF*: As shown in Table 7.3, the number of clauses does not depend on $EP_{C\Pi T}$ since each parameter is encoded using a right-side clause and a set of left-side clauses, regardless of its numerical value.
- Π-$DNNF_{LS}$: The number of clauses per parameter depends on its occurrence number per $C\Pi T_i$. In fact, by increasing the number of equal parameters per $C\Pi T_i$, the number of clauses is reduced since these parameters are encoded using only right-side clauses. Effectively, as shown in Table 7.5, the clauses gain is equal to 1.921 (resp. 1.555) when $EP_{C\Pi T_i} = 1$ (resp. $EP_{C\Pi T_i} > 1$).

- Π-$DNNF^{l}_{PLS}$ *and* Π-$DNNF_{PLS}$: Table 7.5 shows that the clauses gain is equal to 1.509. This means that the number of clauses is increasingly reduced in Π-DNNF$^{l}_{PLS}$ comparing to those of Π-DNNF$_{LS}$ since the number of equal parameters per $C\Pi T$ is rising for each $EP_{C\Pi T}$ and consequently they are encoded using only right-side clauses. In Π-DNNF$_{PLS}$, the number of clauses is smaller since there are no left-side clauses.

- *DNNF-PKB*: Each clause encoding the possibility degree 1 in Π-DNNF$^{l}_{PLS}$ and Π-DNNF$_{PLS}$ is not within DNNF-PKB's clauses since it corresponds to a zero-weighted clause. This justifies the gain of clauses equal to 1.563 and shown in Table 7.5.

$EP_{C\Pi T}$	$\frac{Cl(\Pi-DNNF)}{Cl(\Pi-DNNF_{LS})}$	$\frac{Cl(\Pi-DNNF_{LS}(=1))}{Cl(\Pi-DNNF_{LS}(>1))}$	$\frac{Cl(\Pi-DNNF_{LS})}{Cl(\Pi-DNNF^{l}_{PLS})}$	$\frac{Cl(\Pi-DNNF^{l}_{PLS})}{Cl(\Pi-DNNF_{PLS})}$	$\frac{Cl(\Pi-DNNF_{PLS})}{Cl(DNNF-PKB)}$
0	1.532	1.838	1.446	1.321	1.563
10	1.568	1.800	1.845	1.011	1.563
30	1.658	1.731	1.745	1.011	1.563
50	1.825	1.559	1.585	1.011	1.563
70	2.015	1.413	1.436	1.011	1.563
100	2.927	0.98	1	1	1.563
$ratio_{Cl}$	**1.921**	1.555	1.509	1.060	1.563

Table 7.5: Clauses ratio

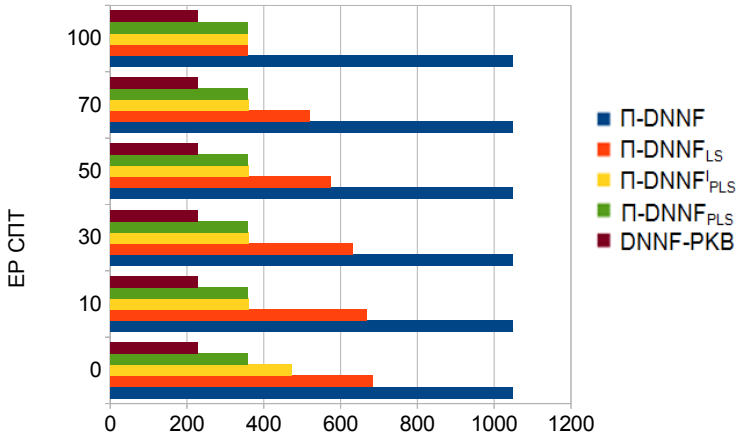


Figure 7.5: Number of clauses

Compiled bases edges

Let us now study and interpret Figures 7.6 and 7.7 showing the impact of CNF encoding strategies on compiled bases edges.

- Π-*DNNF*: As shown in row 1 of Table 7.3 and subfigure (a) of Figure 7.7, the number of edges is equal to 3428. This value remains the same for each $EP_{C\Pi T}$, as for CNF variables and clauses.
- Π-$DNNF_{LS}$: The edges gain of Π-DNNF$_{LS}$ is equal to 1.767 as shown in Table 7.6. This proves that encoding equal parameters per table using a unique parameter variable has a positive impact on compiled bases edges. This behavior follows the one of CNF variables and clauses.

  When $EP_{C\Pi T_i} > 1$, the edges gain is equal to 2.322, which means that rising the number of equal parameters per $C\Pi T_i$ improves increasingly compiled bases edges, comparing to those of $EP_{C\Pi T_i} = 1$.
- Π-$DNNF^{l}_{PLS}$: Table 7.6 shows that the edges ratio is equal to 0.461. In other terms, compiled bases edges are higher when we deal with possibilistic local structure, as shown in Figure 7.6. By paying more attention on row 3, column 5 of Table 7.3 and subfigure (b) of Figure 7.7, we can remark that compiled bases edges depend on $EP_{C\Pi T}$. In fact, when $EP_{C\Pi T} = 10\%$, generated compiled bases have more edges than those of $EP_{C\Pi T} = 0\%$. However, edges decrease from $EP_{C\Pi T} = 30\%$ until $EP_{C\Pi T} = 100\%$.
- Π-$DNNF_{PLS}$: By comparing edges of Π-DNNF$^{l}_{PLS}$ and Π-DNNF$_{PLS}$, we point out that there is a slight reduction in edges (edges ratio = 1.072 from Table 7.6). Such enhancement is due to the reduction of CNF clauses of Π-DNNF$_{PLS}$ comparing to those of Π-DNNF$^{l}_{PLS}$.
- *DNNF-PKB*: From Table 7.6, the edge ratio is equal to 1.056. This proves that DNNF-PKB has a number of edges smaller than those of Π-DNNF$_{PLS}$. This is especially due to the reduction of CNF variables and clauses.

Figures 7.6 and 7.7 do not reveal intersections between methods. Figure 7.8 shows that compiled bases edges of Π-$DNNF^{l}_{PLS}$, Π-$DNNF_{PLS}$ and DNNF-PKB decrease from $EP_{C\Pi T} = 70\%$, and then two breaking points become obvious. The first is situated at around $EP_{C\Pi T} = 80\%$, meaning that purely possibilistic approaches reach Π-DNNF when $EP_{C\Pi T}$ is around 80%. The second breaking point is present between Π-$DNNF_{LS}$ and purely possibilistic approaches when $EP_{C\Pi T}$ is around 90 %.

$EP_{C\Pi T}$	$\frac{Edg(\Pi-DNNF)}{Edg(\Pi-DNNF_{LS})}$	$\frac{Edg(\Pi-DNNF_{LS}(=1))}{Edg(\Pi-DNNF_{LS}(>1))}$	$\frac{Edg(\Pi-DNNF_{LS})}{Edg(\Pi-DNNF^{l}_{PLS})}$	$\frac{Edg(\Pi-DNNF^{l}_{PLS})}{Edg(\Pi-DNNF_{PLS})}$	$\frac{Edg(\Pi-DNNF_{PLS})}{Edg(DNNF-PKB)}$
0	1.369	2.600	0.025	1.272	1.003
10	1.421	2.616	0.009	1.014	1.025
30	1.456	2.658	0.073	1.055	1.004
50	1.616	2.443	0.274	1.066	1.030
70	1.757	2.273	0.494	1.023	1.064
100	2.986	1.342	1.888	1	1.211
$ratio_{Edg}$	**1.767**	**2.322**	0.461	1.072	1.056

Table 7.6: Edges ratio

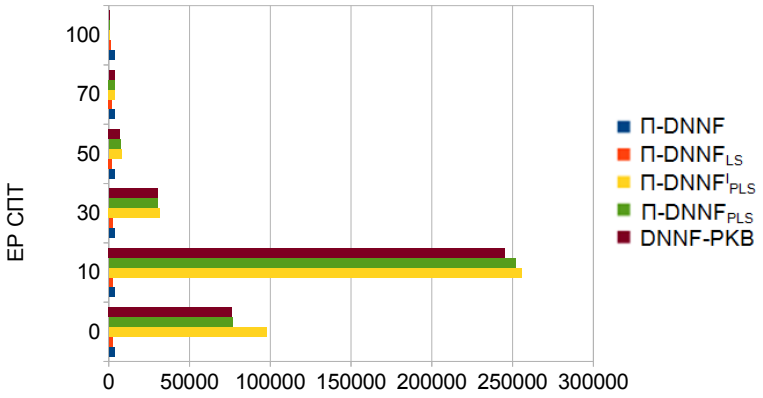


Figure 7.6: Number of edges

A deep analysis of these results shows that the reduction of CNF variables and clauses does not always imply more compact compiled bases. Indeed, using less CNF parameters while exploiting the encoding strategy local structure induces compiled bases with less edges. This is not the case of possibilistic local structure, which increases compiled bases parameters even with a reduced number of CNF parameters. In what follows, we will analyze such behavior while responding to these questions:

1. *Why does possibilistic local structure increase compiled bases edges?*

   - Theorem 2.1 states that satisfying decomposability from CNF bases requires performing case analysis over the variables shared by sub-formulas. When we use local structure, equal parameters per table $C\Pi T_i$ are encoded using the same parameter variable. In this case, the compiler c2d splits common variables pertain-

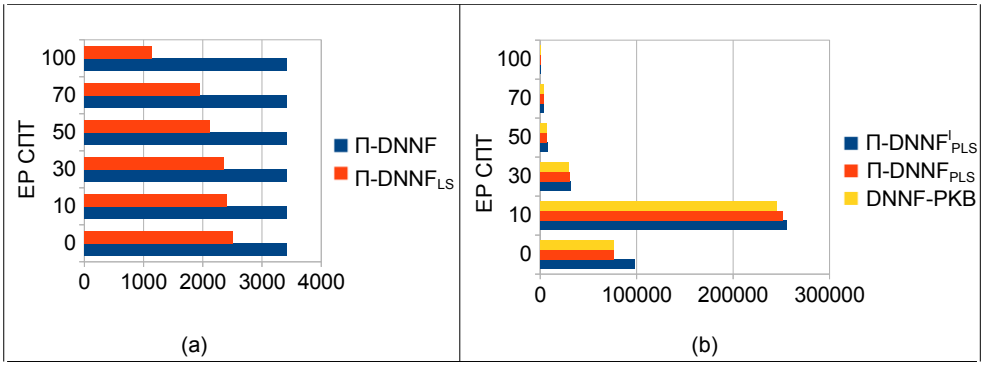


Figure 7.7: (a): Edges of Π-$DNNF$ and Π-$DNNF_{LS}$, (b) Edges of Π-$DNNF^{l}_{PLS}$, Π-$DNNF_{PLS}$ and DNNF-PKB

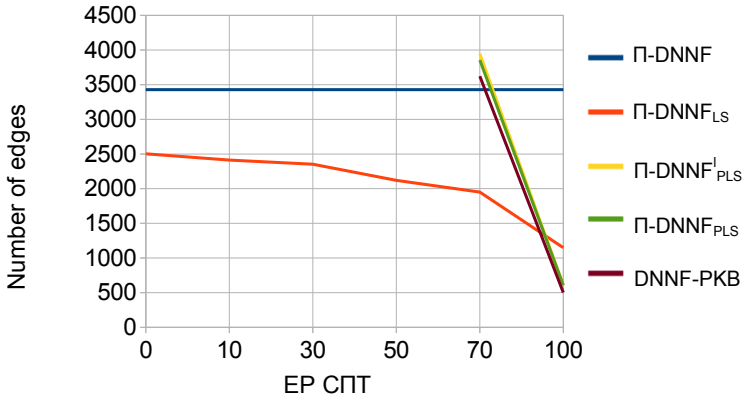


Figure 7.8: Breaking points between methods

ing to the same conditional possibility table, which implies that a local interaction between clauses is performed. However, when we use possibilistic local structure the number of shared variables is increased since equal parameters are handled from a global point of view (i.e., per $C\Pi T$). Such encoding strategy introduces many interactions among clauses corresponding to different conditional possibility tables, which makes the resulting knowledge base harder to compile.

- As we have mentioned above, we used the state of the art c2d compiler [29, 31] initially proposed to generate d-DNNFs. This compiler uses the case analysis technique that efficiently enforces the property of decomposability while enforcing determinism as

well. Example 2.10 shows that converting the CNF $\alpha = (\neg a \vee b) \wedge (\neg b \vee c)$ into the following DNNF $(c \wedge b) \vee (\neg a \wedge \neg b)$ using case analysis gives indeed a d-DNNF since disjunctions $(c \wedge b)$ and $(\neg a \wedge \neg b)$ are logically contradictory. This means that we are subjected to the determinism property as a side effect of this compiler, which is useless in the possibility theory framework.

Generating DNNFs remains a real challenge [62]. In fact, any available compilation algorithm which outputs a DNNF formula computes a *deterministic* DNNF (d-DNNF) formula [27, 31] or a *Structured* DNNF formula based on a *structured* version of decomposability [71].

2. *Why do compiled bases edges rise when* $EP_{C\Pi T} = 10\%$*?*

   By paying more attention on purely possibilistic approaches, we point out that these methods are very sensitive to equal parameters per tables (i.e., $C\Pi T$).

   Let us interpret the impact of each percentage of $EP_{C\Pi T}$:

   - $EP_{C\Pi T} = 0\%$: As shown in rows 4 and 5 of Table 7.3, the number of propositional variables in Π-DNNF$^{l}_{PLS}$ and Π-DNNF$_{PLS}$ is equal to 179. Since we associate an instance indicator for each instance of $X_i \in V$ and knowing that we deal with 50 nodes, then the number of instance indicators is equal to 100. Consequently, we have 79 parameter variables where 78 encode different possibility degrees appearing once per $C\Pi T$ and only one parameter variable θ_1 encodes the possibility degree 1 pertaining to all conditional possibility tables. The resulting base is more hard to compile than the one with local structure since there is an interaction between all clauses weighted by θ_1.
   - $EP_{C\Pi T} = 10\%$: From row 4 of Table 7.3, we can deduce that the number of redundant parameter variables in Π-DNNF$^{l}_{PLS}$ and Π-DNNF$_{PLS}$ is equal to 27, in which one parameter variable encoding the degree 1 appears in all tables and 26 ones appear in 10% of $C\Pi T$ (i.e., 5 tables). For DNNF-PKB method, there are 26 parameter variables encoding degrees different from 1.

     In such a case, the compiler c2d performs case analysis for each parameter variable θ_i holding 5 tables and pertaining to the 26 ones. Since the number of parameter variables θ_i encoding equal parameters per 5 tables is increased, interactions among clauses is rising and consequently, the base is more hard to compile.
   - $EP_{C\Pi T} = 30\%, \cdots, 70\%$: We can point out from row 4 of Table 7.3 that the number of parameter variables appearing in 15, 25 and 35 tables is equal to 11, 7 and 6 when $EP_{C\Pi T} = 30\%$,

$EP_{C\Pi T} = 50\%$ and $EP_{C\Pi T} = 70\%$, respectively. In such cases, c2d deals with a reduced number of shared variables, which explains the reduction of compiled bases edges.

- $EP_{C\Pi T} = 100\%$: In this case, we can deduce from row 4 of Table 7.3 that the number of parameter variables is equal to 2 such that each one appears in all tables. The reduction of shared variables facilitates the generation of compiled bases.

Note that 10 further experiments on different DAG structures have been established and the same behavior and conclusions have been observed.

Inference time

Table 7.7 gives us an idea about the inference ratio. We can see that better values concern Π-$DNNF_{LS}(= 1)$ and Π-$DNNF_{LS}(> 1)$ in both Tables 7.6 and 7.7. This means that the inference ratio follows exactly the same behavior as edges ratio, as shown in Figure 7.9. In other terms, the smaller is the compiled base the faster inference is.

$EP_{C\Pi T}$	$\frac{Inf(\Pi-DNNF)}{Inf(\Pi-DNNF_{LS})}$	$\frac{Inf(\Pi-DNNF_{LS}(=1))}{Inf(\Pi-DNNF_{LS}(>1))}$	$\frac{Inf(\Pi-DNNF_{LS})}{Inf(\Pi-DNNF^{l}_{PLS})}$	$\frac{Inf(\Pi-DNNF^{l}_{PLS})}{Inf(\Pi-DNNF_{PLS})}$	$\frac{Inf(\Pi-DNNF_{PLS})}{Inf(DNNF-PKB)}$
0	1.297	1.721	0.010	1.069	1.021
10	1.332	1.723	1.001	1.044	1.052
30	1.373	1.711	0.054	1.130	1.228
50	1.598	1.522	0.278	1.277	1.148
70	1.657	1.638	0.682	1.505	1.066
100	2.080	1.335	1.546	1	1.101
$ratio_{Inf}$	**1.556**	**1.608**	0.429	1.171	1.103

Table 7.7: Inference ratio

Let us now compare the inference time of the junction tree method and the most compact compilation-based inference method, namely Π-DNNF$_{LS}$.

We can deduce from Table 7.3 that the inference time in Π-DNNF$_{LS}$ decreases each time $EP_{C\Pi T}$ is rising. Yet the time for on-line inference in both Π-$DNNF_{LS}(= 1)$ and Π-$DNNF_{LS}(> 1)$ ranges from 0.1 to 0.3 milliseconds. This illustrates the extent to which local structure can improve the inference time.

Using the junction tree, the inference time ranges from 1.008 second to 1.211 as shown in Table 7.8, but it does not follow a stationary behavior since $EP_{C\Pi T_i}$ and $EP_{C\Pi T}$ are not influential factors. This experimental result confirms that the junction tree is structure-based. It depends on the network topology and is invariant to parameters.

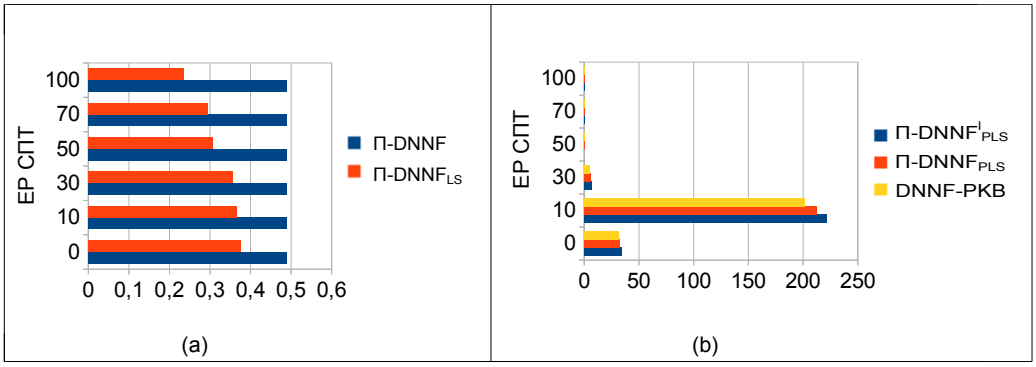


Figure 7.9: (a): Inference time of Π-$DNNF$ and Π-$DNNF_{LS}$, (b) Inference time of Π-$DNNF^{l}_{PLS}$, Π-$DNNF_{PLS}$ and DNNF-PKB

$EP_{C\Pi T}$	$EP_{C\Pi T_i} = 1$ (sec)	$EP_{C\Pi T_i} > 1$ (sec)
0	1.078	1.139
10	1.173	1.008
30	1.112	1.056
50	1.059	1.081
70	1.211	1.025
100	1.133	1.050

Table 7.8: Inference time of junction tree approach

7.3.2 Binary approaches

In the particular case of binary networks, we can either use n-ary approaches with $n = 2$ or use binary approaches, namely Bin-Π-DNNF, Bin-Π-DNNF$_{LS}$ and Bin-Π-DNNF$_{PLS}$. In what follows, we study their CNF parameters, compiled bases edges and inference time, depicted by Table 7.9, while comparing them to n-ary approaches Π-DNNF, Π-DNNF$_{LS}$ and Π-DNNF$_{PLS}$.

1. Π-*DNNF vs Bin*-Π-*DNNF*: Row 1 of Table 7.9 shows that Bin-Π-DNNF does not take into consideration any numerical value in the encoding phase. Comparing row 1 of both Table 7.3 and Table 7.9 gives us variables gain, clauses gain, edges gain and inference gain equal to 2, 4.062, 2.380 and 1.866, respectively. This highlights that Bin-Π-DNNF outperforms Π-DNNF and this is especially due to the reduction of: i) CNF variables by using only one propositional variable and its negation to encode binary variables and ii) CNF clauses by encoding each parameter by a unique right-side clause instead of both right-side clause and left-side clauses while disregarding parameters values.

Method	$EP_{C\Pi T}$	Variables	Clauses	Edges	Inf
Bin-Π-DNNF	0 10 30 50 70 100	179	258	1440	0.262
Bin-Π-DNNF$_{LS}$	0	228	258	1290	**0.235**
	10	226	258	1120	**0.206**
	30	221	258	1105	**0.198**
	50	212	258	1078	**0.187**
	70	204	258	1054	**0.179**
	100	150	258	1002	**0.168**
Bin-Π-DNNF$_{PLS}$	0	129	258	66040	23.881
	10	77	258	235692	186.121
	30	64	258	27146	5.278
	50	61	258	6566	0.772
	70	60	258	3200	0.367
	100	52	258	109	0.09

Table 7.9: Bin-Π-DNNF vs Bin-Π-DNNF$_{LS}$ vs Bin-Π-DNNF$_{PLS}$

2. *Bin*-Π-*DNNF vs Bin*-Π-*DNNF*$_{LS}$ *vs Bin*-Π-*DNNF*$_{PLS}$: As it is the case in Π-DNNF and Π-DNNF$_{LS}$, exploiting local structure in Bin-Π-DNNF has a positive effect on CNF variables, compiled bases edges and consequently the inference time as shown in Table 7.10.

$EP_{C\Pi T}$	$\frac{Var(Bin-\Pi-DNNF)}{Var(Bin-\Pi-DNNF_{LS})}$	$\frac{Edg(Bin-\Pi-DNNF)}{Edg(Bin-\Pi-DNNF_{LS})}$	$\frac{Inf(Bin-\Pi-DNNF)}{Inf(Bin-\Pi-DNNF_{LS})}$
0	1.273	1.116	1.114
10	1.262	1.285	1.271
30	1.234	1.303	1.323
50	1.184	1.335	1.401
70	1.139	1.366	1.463
100	0.837	1.437	1.559
ratio	1.155	**1.307**	**1.355**

Table 7.10: Bin-Π-DNNF vs Bin-Π-DNNF$_{LS}$

Regarding possibilistic local structure, compiled bases edges and inference time of Bin-Π-DNNF$_{PLS}$ increase when $EP_{C\Pi T} \leq 70$, contrarily to CNF variables as shown in Table 7.11. This confirms that using binary encoding while exploiting possibilistic local structure by means of two variables generates very compact compiled bases.

$EP_{C\Pi T}$	$\frac{Var(Bin-\Pi-DNNF_{LS})}{Var(Bin-\Pi-DNNF_{PLS})}$	$\frac{Edg(Bin-\Pi-DNNF_{LS})}{Edg(Bin-\Pi-DNNF_{PLS})}$	$\frac{Inf(Bin-\Pi-DNNF_{LS})}{Inf(Bin-\Pi-DNNF_{PLS})}$
0	1.767	0.019	0.009
10	2.935	0.004	0.001
30	3.453	0.040	0.037
50	3.475	0.164	0.242
70	3.400	0.329	0.487
100	2.884	9.192	1.866
ratio	**2.985**	1.625	0.440

Table 7.11: Bin-Π-DNNF$_{LS}$ vs Bin-Π-DNNF$_{PLS}$

3. Π-*DNNF*$_{LS}$ *vs* *Bin*-Π-*DNNF*$_{LS}$: Table 7.12 shows that variables gain, clauses gain, edges gain and inference gain are equal to 1.246, 2.219, 1.864 and 1.644, respectively. This confirms that Bin-Π-DNNF$_{LS}$ outperforms Π-DNNF$_{LS}$ and proves that encoding binary variables using one variable and its negation, while exploiting local structure and encoding each parameter using one clause have a positive impact on results.

$EP_{C\Pi T}$	$\frac{Var(\Pi-DNNF_{LS})}{Var(Bin-\Pi-DNNF_{LS})}$	$\frac{Cl(\Pi-DNNF_{LS})}{Cl(Bin-\Pi-DNNF_{LS})}$	$\frac{Edg(\Pi-DNNF_{LS})}{Edg(Bin-\Pi-DNNF_{LS})}$	$\frac{Inf(\Pi-DNNF_{LS})}{Inf(Bin-\Pi-DNNF_{LS})}$
0	1.219	2.651	1.941	1.604
10	1.221	2.589	2.153	1.781
30	1.226	2.449	2.129	1.797
50	1.235	2.224	1.967	1.636
70	1.245	2.015	1.851	1.648
100	1.333	1.387	1.145	1.398
ratio	1.246	**2.219**	**1.864**	1.644

Table 7.12: Π-DNNF$_{LS}$ vs Bin-Π-DNNF$_{LS}$

4. Π-DNNF$_{PLS}$ vs Bin-Π-DNNF$_{PLS}$: By paying attention on Table 7.13, we can see that edges gain is equal to 1.870. This highlights that using binary encoding with possibilistic local structure generates compiled bases with less edges, comparing to those of Π-DNNF$_{PLS}$.

As a conclusion, we can say that binary approaches follow the same behavior as n-ary approaches, while requiring less CNF parameters and compiled bases edges. This is especially to the reduction of CNF variables by using only one propositional variable and its negation to encode binary variables.

$EP_{C\Pi T}$	$\frac{Var(\Pi-DNNF_{PLS})}{Var(Bin-\Pi-DNNF_{PLS})}$	$\frac{Cl(\Pi-DNNF_{PLS})}{Cl(Bin-\Pi-DNNF_{PLS})}$	$\frac{Edg(\Pi-DNNF_{PLS})}{Edg(Bin-\Pi-DNNF_{PLS})}$	$\frac{Inf(\Pi-DNNF_{PLS})}{Inf(Bin-\Pi-DNNF_{PLS})}$
0	1.387	1.387	1.161	1.350
10	1.649	1.387	1.067	1.140
30	1.75	1.387	1.110	1.094
50	1.77	1.387	1.101	1.112
70	1.783	1.387	1.205	0.782
100	1.961	1.387	5.577	1.688
ratio	1.717	1.387	**1.870**	1.194

Table 7.13: Π-DNNF$_{PLS}$ vs Bin-Π-DNNF$_{PLS}$

7.4 Comparing interventions-based methods under compilation

In Chapter 5, we proposed mutilated-based approaches and augmented-based approaches dealing with interventions under a compilation framework. In this section, we study these methods from an experimental point of view. To this end, we should at first set the number of interventions and describe experimental data.

As we have outlined in Section 5.3, mutilated-based approaches are not sensitive to the number of interventions since we mutilate the compiled base instead of the initial possibilistic network. For this reason, we generate random possibilistic networks by setting the number of nodes to 50, variables cardinality to 2 and 3 and maximum number of parents per node to 3. This is not the case of augmented-based approaches that depend strongly on the number of interventions. In such a scenario, we generate augmented possibilistic networks given a number of interventions varying from 1 to 50.

Table 7.14 depicts experimental results of Mut-Π-DNNF, Mut-DNNF-PKB and Aug-Π-DNNF$_{LS}$ by considering one intervention, then 10, 20, 30, 40 and 50 interventions. When the number of occurred interventions is not well-known in an instant t, the price to be paid is to consider 50 extra nodes and compile a network composed of 100 nodes (50 network variables and 50 extra nodes). We call this particular situation the extreme case.

Note that we only deal with Aug-Π-DNNF$_{LS}$ since it turns out from subsection 7.3 that Π-DNNF$_{LS}$ is the most efficient and compact method. Regarding possibility distributions values, we exploit local structure using the parameter $EP_{C\Pi T_i} = \{0, 10, 30, 50, 70, 100\}$ s.t. $EP_{C\Pi T_i}$=50% means that a possibility degree appears in 50% of the possibility degrees different from 1 in the current $C\Pi T_i$.

Let us now study in depth results of Table 7.14 via the following pairwise

Method	Nb interventions	EP_{CIIT_i}	Variables	Clauses	Edges
Mut-Π-DNNF	-	$0 \cdots 100$	734	2573	29943
Mut-DNNF-PKB	-	$0 \cdots 100$	478	502	196179
Aug-Π-DNNF$_{LS}$	1	0	507	1687	**11892**
		10	453	1383	**9272**
		30	357	1065	**6246**
		50	297	942	**4266**
		70	285	896	**3986**
		100	238	856	**1571**
Aug-Π-DNNF$_{LS}$	10	0	571	2414	**14080**
		10	560	2326	**13265**
		30	484	2032	**10627**
		50	445	1953	**9959**
		70	437	1922	**9795**
		100	399	1886	**2743**
Aug-Π-DNNF$_{LS}$	20	0	631	2718	**15058**
		10	620	2630	**14243**
		30	556	2392	**11783**
		50	523	2331	**11137**
		70	519	2316	**10975**
		100	489	2296	**3826**
Aug-Π-DNNF$_{LS}$	30	0	691	3178	**16569**
		10	660	3040	**15187**
		30	654	3015	**15014**
		50	628	2975	**13857**
		70	611	2745	**13184**
		100	591	2737	**4707**
Aug-Π-DNNF$_{LS}$	40	0	785	3637	**18742**
		10	774	3549	**18006**
		30	759	3521	**17313**
		50	751	3504	**16803**
		70	748	3492	**16625**
		100	738	3476	**11521**
Aug-Π-DNNF$_{LS}$	50	0	849	4054	**19995**
		10	838	3970	**19180**
		30	824	3952	**18439**
		50	819	3938	**18126**
		70	801	3872	**18006**
		100	771	3764	**12682**

Table 7.14: Mut-Π-DNNF vs Mut-DNNF-PKB vs Aug-Π-DNNF$_{LS}$ (better values are in bold

comparison:

- *Mut*-Π-*DNNF vs Mut-DNNF-PKB*: It is well attested that both of Mut-Π-DNNF and Mut-DNNF-PKB do not exploit parameters values. This means that a propositional variable is associated for each parameter, regardless of its numerical value. As shown in rows 1 and 2 of Table 7.14, Mut-Π-DNNF, which has a number of variables and clauses higher than those of Mut-DNNF-PKB, gives rise to more compact compiled bases having less edges. This proves that dealing with symbolic bases using only right-side clauses makes the resulting knowledge base harder to compile. Consequently, we can conclude that both of left-side clauses and right-side clauses play a primary role in the compilation process since each parameter variable is encoded using a logical equivalence and therefore cannot imply instance indicators other than those compatible with variables instantiation.

- Mut-Π-DNNF vs Aug-Π-DNNF$_{LS}$: From Figure 7.10, it is clear that Aug-Π-DNNF$_{LS}$ outperforms Mut-Π-DNNF even in the extreme case in which an extra node DO_i is associated for each network variable. This means that even if Aug-Π-DNNF$_{LS}$ deals with additional extra nodes DO_I, it produces more compact compiled bases having less edges than those of Mut-Π-DNNF. In the extreme case, a key result that should be highlighted is that augmenting a network by 50 extra nodes and compiling 100 nodes with local structure gives better results than compiling 50 nodes without taking into account any numerical value. This means that increasing the size of possibility distributions of involved variables using parameters 1 and 0 and adding nodes DO_I which are encoded without taking into consideration any numerical value do not degrade the quality of compiled bases, in comparison with Mut-Π-DNNF. Interestingly enough, local structure plays a leading role in augmentation. By paying more attention on edges of Aug-Π-DNNF$_{LS}$, we can see that the number of edges varies from 11892 when we deal with a unique intervention to 19995 in the case of 50 interventions. This means that given the compiled base of a network augmented by a unique intervention, it is easy to obtain the compiled base of the extreme case. To strengthen this result, we have established another experiment for networks composed of 100 nodes. We have obtained compiled bases composed of 104339 (resp. 127821) edges when the number of interventions is equal to 1 (resp. 100). This experiment serves to demonstrate that augmentation is linear with respect to the compiled base size of a network augmented by a unique intervention.

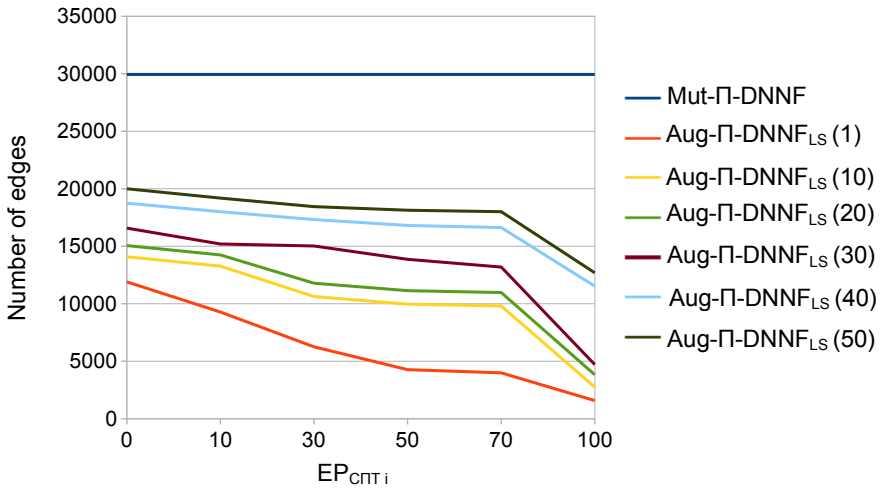


Figure 7.10: Mut-Π-DNNF vs Aug-Π-DNNF$_{LS}$

7.5 Bayesian networks vs product-based possibilistic networks under compilation

In the previous set of experiments, we restrict ourselves to min-based possibilistic networks. In this section, we emphasize on CNF encodings and compiled bases parameters of the probabilistic approach, denoted by Prob-d-DNNF [30] and the quantitative possibilistic approach Prod-Π-DNNF.

It is well known that product-based possibilistic networks are close to Bayesian networks [20]. We propose to check this statement under compilation by comparing Prod-Π-DNNF to Prob-d-DNNF. To this end, we consider randomly generated Bayesian networks and product-based possibilistic networks by setting the number of nodes to 50, the maximum number of parents per node to 3 and the number of instances per variable to 2 and 3. We vary probability (resp. possibility) distributions using one parameter, namely: *the percentage of equal parameters per conditional uncertainty table* $\mathbf{CUT_i}$, denoted by $EP_{CUT_i} = \{0\%, 10\%, 30\%, 50\%, 70\%, 100\%\}$. The interpretation of EP_{CUT_i} differs depending on the framework where the uncertainty degree ud_i corresponds to probability degrees (P) in Bayesian networks and possibility degrees (Π) in possibilistic networks:

- *Possibilistic case*: Let Nb_p be the number of possibility degrees different from the normalization degree i.e., 1 in a CUT_i. Then, for instance

EP_{CUT_i}=50% means that a possibility degree ud_i appears in 50% of Nb_p. The extreme case 0% (resp. 100%) states that a degree ud_i appears once (in 100% of Nb_p) in the current CUT_i.

- *Probabilistic case*: Let *set* be the probability degrees ud_i having $\sum_{ud_i} = 1$ in a CUT_i. Let $Nb_{set} = *_{j=\{1..m\}}\ card(U_{ij})$ be the number of *set* per CUT_i. Then, EP_{CUT_i}=50% means that a *set* appears in 50% of Nb_{set} in the current CUT_i. The extreme case 0% (resp. 100%) states that a *set* appears once (in 100% of Nb_{set}) per CUT_i.

The experimental results are shown in Table 7.15. From this table, we can derive values of Table 7.16 showing *Variable_ratio*, *Clause_ratio* and *Edge_ratio* expressing the average of variables, clauses and edges, respectively for Prod-Π-DNNF and Prob-d-DNNF, while varying EP_{CUT_i}.

As shown in Table 7.16, the number of CNF variables and clauses of Prod-Π-DNNF is smaller than those of Prob-d-DNNF, even if both of them exploit the same encoding strategy. In other terms, the normalization constraint offered by possibility theory gives rise to less CNF variables since only one parameter is repeated per table. We can say that such constraint is very suitable for local structure and of course proves the emphasis of possibility degrees in comparison to probability degrees, which in turn confirms the theoretical result of Proposition 6.2.

Regarding compiled bases edges, we can see from Table 7.16 that the *Edge_ratio* is around 1.141. This means that Prod-Π-DNNF is characterized by a lower number of edges comparing to those of Prob-d-DNNF. Hence, we can point out that compiled bases parameters follow the same behavior as those of CNF encodings.

	Prod-Π-DNNF			**Prob-d-DNNF**		
	C_*^{LS}		CB_{*max}^{LS}	C_{p*}^{LS}		CB_{*+}^{LS}
EP_{CUT_i}	Variables	Clauses	Edges	Variables	Clauses	Edges
0	**482**	**1518**	**12838**	660	2321	16573
10	**471**	**1469**	**12450**	649	2253	16208
30	**410**	**1211**	**10674**	553	1739	12290
50	**377**	**1121**	**9565**	498	1531	10177
70	**300**	**890**	**8049**	354	1027	8144
100	**225**	**642**	**3454**	248	765	3520

Table 7.15: Prod-Π-DNNF vs Prob-d-DNNF (better values are in bold)

EP_{CUT_i}	$\frac{Variables(C^{LS}_{p*})}{Variables(C^{LS}_{*})}$	$\frac{Clauses(C^{LS}_{p*})}{Clauses(C^{LS}_{*})}$	$\frac{Edges(CB^{LS}_{*+})}{Edges(CB^{LS}_{*max})}$
0	1.369	1.528	1.290
10	1.377	1.533	1.301
30	1.348	1.436	1.151
50	1.320	1.365	1.063
70	1.18	1.153	1.011
100	1.102	1.191	1.027
	Variable_ratio	**Clause_ratio**	**Edge_ratio**
	1.283	1.368	1.141

Table 7.16: Average of Prod-Π-DNNF and Prob-d-DNNF parameters

7.6 Conclusion

Our experimental results show that taking into consideration numerical values while encoding the min-based possibilistic network has a considerable impact on both CNF parameters and compiled bases and consequently the inference time. However, this relies deeply on the used encoding strategy. In fact, *possibilistic local structure*, which reduces increasingly CNF parameters, rises compiled bases parameters. This is not the case of local structure which has a positive impact on compiled bases. We also studied the effect of interventions on compiling possibilistic networks. Our experiments show that augmented-based approaches outperform mutilated-based approaches even in the extreme case. Finally, we established a comparison between compiling Bayesian networks and product-based possibilistic networks proving the emphasis of possibility degrees in comparison to probability degrees while using the same encoding strategy, i.e., local structure.

Conclusion

This thesis has contributed to the development of graphical models for reasoning and decision under compilation using the possibility theory framework. In this theory, uncertainty is either encoded *numerically* using the unit interval $[0, 1]$ or *qualitatively* using a total pre-order between events. This leads to two kinds of possibilistic networks, namely *product-based possibilistic networks* and *min-based possibilistic networks.*

At first, we have addressed the inference topic in min-based possibilistic networks using compilation techniques. In fact, we have adapted the standard probabilistic compilation-based inference approach into the possibility theory framework and we have developed a new purely possibilistic method based on compiling possibilistic knowledge bases associated with possibilistic networks. Then, a set of refinements have been suggested dealing with specific parameters values while encoding the network, namely local structure, possibilistic local structure and binary encodings. Our results point out that purely possibilistic approaches are the most compact ones in terms of CNF encodings parameters since they have less variables and clauses than remaining approaches. However, dealing with equal parameters from a global point of view generates compiled bases with higher edges. This is especially due to the higher number of shared variables incurring several interactions among clauses. This study points out that the inference time relies strongly on the compiled base size. This means that the smaller the compiled base is the faster inference is.

Another main contribution of this thesis is the proposal of mutilated-based methods and augmented-based methods that efficiently compute the effect of both observations and interventions for min-based possibilistic causal networks. In fact, mutilated-based methods require the mutilation of symbolic compiled bases, while augmented-based methods involve encoding augmented networks while ignoring numerical values of new extra nodes. Mutilated-based approaches are not sensitive to the number of interventions since the compiled base is mutilated instead of the initial possibilistic network, which enables the handling of a set of interventions without the need for re-compiling the network each time an intervention occurs. This is not

the case of augmented-based approaches since the augmented network is compiled after performing the set of interventions. Our results show that augmented-based approaches outperform mutilated-based approaches even in the extreme case in which an extra node is associated for each network variable.

In the last part of this work, compilation-based inference in product-based possibilistic networks has been studied. Moreover, we have explored the decisional aspect and extended compilation concepts to evaluate possibilistic influence diagrams by taking advantage of our compilation-based inference methods proposed in the first part. Our idea consists in re-using the polynomial *transformation* phase of [52] to morph the initial min-based possibilistic influence diagram into a min-based possibilistic network and proposing an *evaluation* phase in which we generate the optimal strategy under compilation.

An interesting future work is to develop a compiler that generates DNNFs (without the need for either the determinism property i.e., d as it is the case in the standard publicly available c2d compiler [31] or the structured decomposability of [71]). This remains a real challenge [62]. Another line of research will be to explore the idea of arithmetic circuits augmented with maximization nodes using variable elimination proposed for standard influence diagrams [19] in the possibility theory framework. From an applicative point of view, we will explore the issue of *fault detection*. In fact, in complex systems, fast detection of faults and accurate diagnosis of root cause of failure is a crucial aspect of operational safety and considered as a critical factor in reducing system down-time. We are thinking about the use of compilation in qualitative models which have proved their efficiency by providing a realistic representation and accurate diagnosis in [45].

Bibliography

[1] A. Ajroud, M. N. Omri, S. Benferhat, and H. Youssef. An approximate propagation algorithm for product-based possibilistic networks. In *Proceedings of the 10th International Conference on Enterprise Information Systems*, pages 321–326, Spain, 2008.

[2] A. Akari. *Microeconomie du consommateur et du producteur*. Etigraph editor, 2000.

[3] M. Aschinger, C. Drescher, G. Gottlob, P. Jeavons, and E. Thorstensen. Structural decomposition methods and what they are good for. In *Proceedings of the 28th Symposium on Theoretical Aspects of Computer Science*, pages 12–28, 2011.

[4] R. Ayachi, N. Ben Amor, and S. Benferhat. An augmented-based approach for compiling min-based possibilistic causal networks. In *Proceedings of 23rd IEEE International Conference on Tools with Artificial Intelligence*, pages 675–678, USA, 2011.

[5] R. Ayachi, N. Ben Amor, and S. Benferhat. Compiling min-based possibilistic causal networks: A mutilated-based approach. In *Proceedings of 11th European Conference of Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 700–712, Belfast-UK, 2011.

[6] R. Ayachi, N. Ben Amor, and S. Benferhat. Experimental comparative study of compilation-based inference in bayesian and possibilitic networks. In *Proceedings of 9th International Workshop on Fuzzy Logic and Applications*, pages 155–163, 2011.

[7] R. Ayachi, N. Ben Amor, and S. Benferhat. Inference using compiled product-based possibilistic networks. In *Proceedings of 14th International Conference on Information Processing and Management of Uncertainty*, pages 470–480, 2012.

[8] R. Ayachi, N. Ben Amor, and S. Benferhat. Possibilistic local structure for compiling min-based networks. In *Proceedings of 6th International*

Conference on Soft Methods in Probability and Statistics, pages 479–487, 2012.

[9] R. Ayachi, N. Ben Amor, S. Benferhat, and R. Haenni. Compiling possibilistic networks : Alternative approaches to possibilistic inference. In *Proceedings of 26th Conference on Uncertainty on Artificial Intelligence*, pages 40–47, California, 2010. AUAI Press.

[10] F. Bacchus, S. Dalmao, and T. Pitassi. Solving #sat and bayesian inference with backtracking search. *Journal of Artificial Intelligence Research*, 34:391–442, 2009.

[11] N. Ben Amor, S. Benferhat, and K. Mellouli. Anytime propagation algorithm for min-based possibilistic graphs. *Soft Computing*, 8(2):150–161, 2003.

[12] S. Benferhat, D. Dubois, L. Garcia, and H. Prade. On the transformation between possibilistic logic bases and possibilistic causal networks. *International Journal of Approximate Reasoning*, 29(2):135–173, 2002.

[13] S. Benferhat, D. Dubois, H. Prade, and M. Williams. A practical approach to fusing and revising prioritized belief bases. In *Proceedings of 9th Portuguese Conference on Artificial Intelligence*, pages 222–236, 1999.

[14] S. Benferhat, F. Khellaf Haned, and A. Mokhtari. Product-based causal networks and quantitative possibilistic bases. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 13(05):469–493, 2005.

[15] S. Benferhat and S. Smaoui. Possibilistic networks with locally weighted knowledge bases. In *Proceedings of the 4th International Symposium on Imprecise Probabilities and Their Applications*, pages 41–50, 2005.

[16] S. Benferhat and S. Smaoui. Possibilistic causal networks for handling interventions: A new propagation algorithm. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 373–378, 2007.

[17] S. Benferhat and S. Smaoui. Inferring interventions in product-based possibilistic causal networks. *Fuzzy Sets and Systems*, 169:26–50, April 2011.

[18] S. Benferhat, S. Yahi, and H. Drias. On the compilation of stratified belief bases under linear and possibilistic logic policies. In *International Journal of Approximate Reasoning*, pages 2425–2430, 2007.

[19] D. Bhattacharjya and R. Shachter. Evaluating influence diagrams with decision circuits. In *Proceedings of 23th Conference on Uncertainty in Artificial Intelligence*, pages 9–16, Canada, 2007. AUAI Press.

[20] C. Borgelt, J. Gebhardt, and R. Kruse. Possibilistic graphical models. In *Proceedings of International School for the Synthesis of Expert Knowledge*, Italy, 1998.

[21] M. Cadoli and F. M. Donini. A survey on knowledge compilation. *AI Communications–The European Journal for Artificial Intelligence*, (10):137–150, 1998.

[22] H. Chan and A. Darwiche. On the robustness of most probable explanations. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 63–71. Morgan Kaufmann, 2006.

[23] M. Chavira and A. Darwiche. Compiling bayesian networks with local structure. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1306–1312, 2005.

[24] M. Chavira and A. Darwiche. Compiling Bayesian networks using variable elimination. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2443–2449, 2007.

[25] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks (research note). *Artificial Intelligence*, 42(2-3):393–405, 1990.

[26] A. Darwiche. Compiling knowledge into decomposable negation normal form. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 284–289, California, 1999.

[27] A. Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4):608–647, 2001.

[28] A. Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2):11–34, 2001.

[29] A. Darwiche. A compiler for deterministic, decomposable negation normal form. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 627–634, California, 2002. AAAI Press.

[30] A. Darwiche. A logical approach to factoring belief networks. In *Proceedings of the 8th International Conference on Principles and Knowledge Representation and Reasoning*, pages 409–420, Toulouse, 2002.

[31] A. Darwiche. New advances in compiling CNF to decomposable negation normal form. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 328–332, 2004.

[32] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge Univ. Press, 2009.

[33] A. Darwiche and M. Hopkins. Using recursive decomposition to construct elimination orders, jointrees, and dtrees. In *Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 180–191, London, 2001. Springer-Verlag.

[34] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[35] A. Darwiche and P. Marquis. Compiling propositional weighted bases. *Artificial Intelligence*, 157:2004, 2004.

[36] G. De Cooman and E. E. Kerre. A new approach to possibilistic independence. In *Proceedings of the 3rd IEEE International Conference on Fuzzy Systems*, pages 1446–1451, 1994.

[37] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In *Handbook on Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 439–513. Oxford University press, 1994.

[38] D. Dubois and H. Prade. Possibility theory: qualitative and quantitative aspect. In *Handbook on Defeasible Reasoning and Uncertainty Management Systems*.

[39] D. Dubois and H. Prade. *Possibility theory:An approach to computerized, Processing of uncertainty*. Plenium Press, New York, 1988.

[40] D. Dubois and H. Prade. Possibility theory and data fusion in poorly informed environnment. *Control Ingeneering Practice*, 2:811–823, 1994.

[41] D. Dubois and H. Prade. Possibility theory as a basis for qualitative decision theory. In *Proceedings of the 14th international joint conference on Artificial intelligence*, pages 1924–1930, San Francisco, 1995. Morgan Kaufmann.

[42] D. Dubois and H. Prade. Possibility theory. In *Encyclopedia of Complexity and Systems Science*. Springer, 2009.

[43] D. Dubois, H. Prade, and Ph. Smets. Representing partial ignorance. *IEEE System Machine and Cybernetic*, 26:361–377, 1996.

[44] H. Fargier and P. Marquis. On valued negation normal form formulas. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 360–365, USA, 2007.

[45] A. Feldman, J. Pietersma, and A. Gemund. All roads lead to fault diagnosis: Model-based reasoning with lydia. In *Proceedings of the 8th Belgium-Netherlands Conference on Artificial Intelligence*, pages 123–131, 2006.

[46] P. Fonck. *Réseaux d'inférence pour le raisonnement possibiliste.* PhD thesis, Université de Liège, Faculté des Sciences, 1994.

[47] L. Garcia and R. Sabbadin. Possibilistic influence diagrams. In *Proceedings of the 17th European Conference on Artificial Intelligence*, pages 372–376, 2006.

[48] J. Gebhardt and R. Kruse. Background and perspectives of possibilistic graphical models. In *Proceedings of the 4th European Conference of Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 108–121, Germany, 1997.

[49] P. H. Giang and P. P. Shenoy. Two axiomatic approaches to decision making using possibility theory. *European journal of operational research*, 162(2):450–467, 2005.

[50] G. Gogic, H. Kautz, C. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proceedings of the 14th international joint conference on Artificial intelligence*, pages 862–869, 1995.

[51] M. Goldszmidt and J. Pearl. Rank-based systems: A simple approach to belief revision, belief update, and reasoning about evidence and actions. In *Proceedings of the 3rd International Conference on Principles and Knowledge Representation and Reasoning*, pages 661–672, 1992.

[52] W. GueGuez, N. Ben Amor, and K. Mellouli. Qualitative possibilistic influence diagrams based on qualitative possibilistic utilities. *European Journal of Operational Research*, 195(1):223–238, 2009.

[53] E. Hisdal. Conditional possibilities independence and non interaction. *Fuzzy Sets and Systems*, 1, 1978.

[54] R.A. Howard and J.E. Matheson. Influence diagrams. *The principles and applications of decision analysis*, 2:719–762, 1984.

[55] C. Huang and A. Darwiche. Inference in belief networks: a procedural guide. *International Journal of Approximate Reasoning*, 11:1–158, 1994.

[56] F. V. Jensen. *Introduction to Bayesien networks.* UCL Press, 1996.

[57] A. Kaufmann and M. Gupta. *Introduction to fuzzy arithmetic.* Theory and Publications, New York, 1985.

[58] J. H. Kim and J. Pearl. A computational model for causal and diagnostic reasoning in inference systems. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 190–193, Germany, 1983.

[59] J. Lang. Possibilistic logic: complexity and algorithms. In *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, volume 5, page 179Ũ220. 2001.

[60] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Readings in uncertain reasoning*, pages 415–448, 1990.

[61] P. Marquis. Knowledge compilation: a sightseeing tour. *Tutorial notes, 18th European Conference on Artificial Intelligence*, 2008.

[62] P. Marquis. Existential closures for knowledge compilation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 996–1001, 2011.

[63] J. V. Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1948.

[64] A. Niveau, H. Fargier, C. Pralet, and G. Verfaillie. Knowledge compilation using interval automata and applications to planning. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 459–464, 2010.

[65] C. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[66] J. Pearl. Fusion propagation and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

[67] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[68] J. Pearl. *Comment: graphical models, causality and intervention*. Statistical Science 8, 1993.

[69] J. Pearl. *Causality: Models, reasoning and inference*. Cambridge University Press, 2000.

[70] K. Pipatsrisawat and A. Darwiche. On decomposability and interaction functions. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 9–14, 2010.

[71] K Pipatsrisawat and A. Darwiche. Top-down algorithms for constructing structured dnnf: Theoretical and practical implications. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 3–8, 2010.

[72] T. Sang, F. Bacchus, P. Beame, H. Kautz, and T. Pitassi. Combining component caching and clause learning for effective model counting. In *The 7th International Conference on Theory and Applications of Satisfiability Testing*, Vancouver, 2004.

[73] T. Sang, P. Beame, and H. Kautz. Heuristics for fast exact model counting. In *The 8th International Conference on Theory and Applications of Satisfiability Testing*, page 226Ű240, 2005.

[74] T. Sang, P. Beame, and H. Kautz. Solving bayesian networks by weighted model counting. In *In Proceedings of the Twentieth National Conference on Artificial Intelligence*, volume 1, pages 475–482, 2005.

[75] G. Shafer. *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press., 1976.

[76] P. Smets. Belief functions. In P. Smets, A. Mandani, and H. Prade D. Dubois, editors, *Non Standard Logics for Automated Reasoning*, pages 253–286. Academic Press, London, 1988.

[77] P. Smets and R. Kennes. The transferable belif model. *Artificial Intelligence*, 66:191–234, 1990.

[78] W. Spohn. A general non-probabilistic theory of inductive reasoning. In *Proceedings of the 4th Conference on Uncertainty in Artificial Intelligence*.

[79] W. Spohn. Ordinal conditional functions: a dynamic theory of epistemic states causation in decision. In *Belief Changes and Statistics*.

[80] C. Starr, P. Shi, Defence Science, and Technology Organisation (Australia). Systems Sciences Laboratory. *An Introduction to Bayesian Belief Networks and Their Applications to Land Operations*. Technical note. DSTO Systems Sciences Laboratory, 2004.

[81] J. Vomlel. Two applications of bayesian networks. In *Proceedings of the 2nd Conference Znalosti*, pages 73–82, Ostrava, 2003.

[82] M. Wachter and R. Haenni. Logical compilation of bayesian networks with discrete variables. In *Proccedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 536–547, 2007.

[83] R. R. Yager. On the specificity of a possibility distribution. *Fuzzy Sets and Systems*, 50:279–292, 1992.

[84] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[85] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 100:9–34, 1999.