THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*
Cotutelle internationale *avec l'Institut Supérieur de Gestion de Tunis*

Présentée et soutenue le *30/10/2017* **par :**

Héla GOUIDER

Graphical Preference Representation under a Possibilistic Framework

---

JURY

Nahla BEN AMOR	Professeur d'Université	Directrice de thèse
Sébastien DESTERCKE	Directeur de Recherche	Examinateur
Didier DUBOIS	Directeur de Recherche	Examinateur
Zied ELOUEDI	Professeur à l'Université	Président
Sylvain LAGRUE	Maître de conférences habilité	Rapporteur
Patrice PERNY	Professeur d'Université	Rapporteur
Henri PRADE	Directeur de Recherche	Directeur de thèse

École doctorale et spécialité :
MITT : Domaine STIC : Intelligence Artificielle
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)
Directrice(s) de Thèse :
Nahla BEN AMOR et *Henri PRADE*
Rapporteurs :
Sylvain LAGRUE et *Patrice PERNY*

I dedicate this thesis to my dear family

Acknowledgments

First of all, I would like to thank all those who contributed to the preparation and writing of my thesis. Obviously, it is difficult to thank everyone, each by his name because it was thanks to the help of many people that I was able to carry out this thesis to its end.

In the first place, I would like to express my gratitude to all my family, thanks to my father and mother who have been the luminous stars for me all along the road and who have always believed in me. I would like to thank my two sisters Dhouha and Houda, my fiance Nizar and all my family for their sincere love and during these years.

My thanks go also to the reviewers, Sylvain Lagrue and Patrice Perny, to whom I associate the members Didier Dubois, Sébastien Destercke and Zied Elouedi, for honoring me with taking time to read in detail my work and whose kindness, interest and advice to continue and improve my research are invaluable.

I would like to thank my thesis directors Nahla Ben Amor and Henri Prade for having accepted to supervise me during the preparation of my thesis. I am delighted to have been able to work in their company because in addition to their scientific support and the human side, they have always been there to support me and advise me throughout the development of my work especially during difficult times. A special thank goes also to Didier Dubois for always being there whenever I had questions, and for his patience and immense knowledge.

A special thank you to Sahar and Amina: you were always there for me. It is almost impossible for me to imagine the last three years without my good friends Fatma and Zeineb, who gave me moral support during the years of this thesis.

Thanks, finally, to all the members of the ADRIA team, and more generally to IRIT. I must also add here that I deeply appreciate the LARODEC lab., every single one of it, for their involvement in my personal well-being.

Résumé

La modélisation structurée de préférences, fondée sur les notions d'indépendance préférentielle, a un potentiel énorme pour fournir des approches efficaces pour la représentation et le raisonnement sur les préférences des décideurs dans les applications de la vie réelle. Cette thèse soulève la question de la représentation des préférences par une structure graphique. Nous proposons une nouvelle lecture de réseaux possibilistes, que nous appelons π-pref nets, où les degrés de possibilité représentent des degrés de satisfaction. L'approche utilise des poids de possibilité non-instanciés (appelés poids symboliques), pour définir les tables de préférences conditionnelles. Ces tables donnent naissance à des vecteurs de poids symboliques qui codent les préférences qui sont satisfaites et celles qui sont violées dans un contexte donné. Nous nous concentrons ensuite sur les aspects théoriques de la manipulation de ces vecteurs. En effet, la comparaison de ces vecteurs peut s'appuyer sur différentes méthodes: celles induites par la règle de chaînage basée sur le produit ou celle basée sur le minimum que sous-tend le réseau possibiliste, les raffinements du minimum le discrimin, ou leximin, ainsi que l'ordre Pareto, et le Pareto symétrique qui le raffine. Nous prouvons que la comparaison par produit correspond exactement au celle du Pareto symétrique et nous nous concentrons sur les avantages de ce dernier par rapport aux autres méthodes. En outre, nous montrons que l'ordre du produit est consistant avec celui obtenu en comparant des ensembles de préférences satisfaites des tables. L'image est complétée par la proposition des algorithmes d'optimisation et de dominance pour les π-pref nets.

Dans ce travail, nous discutons divers outils graphiques pour la représentation des préférences. Nous nous focalisons en particulier sur les CP-nets car ils partagent la même structure graphique que les π-pref nets et sont basés sur la même nature de préférences. Nous prouvons que les ordres induits par les CP-nets ne peuvent pas contredire ceux des π-pref nets et nous avons fixé les contraintes nécessaires pour raffiner les ordres des π-pref nets afin de capturer les contraintes Ceteris Paribus des CP-nets. Cela indique que les CP-nets représentent potentiellement une sous-classe des π-pref nets avec des contraintes. Ensuite, nous fournissons une comparaison approfondie entre les différents modèles graphiques qualitatifs et quantitatifs, et les π-pref nets. Nous en déduisons que ces derniers peuvent être placés à mi-chemin entre les modèles qualitatifs et les modèles quantitatifs puisqu'ils ne nécessitent pas une instanciation complète des poids symboliques alors que des informations supplémentaires sur l'importance des poids peuvent être prises en compte.

La dernière partie de ce travail est consacrée à l'extension du modèle proposé pour représenter les préférences de plusieurs agents. Dans un premier temps, nous proposons l'utilisation de réseaux possibilistes où les préférences sont de type *tout ou rien* et nous définissons le conditionnement dans le cas de distributions booléennes. Nous montrons par ailleurs que ces réseaux multi-agents ont une contrepartie logique utile pour vérifier la cohérence des agents. Nous expliquons les étapes principales pour transformer ces réseaux en format logique. Enfin, nous décrivons une extension pour représenter des préférences nuancées et fournissons des algorithmes pour les requêtes d'optimisation et de dominance.

Mots-clés: Théorie de possibilité, réseaux de préférences, poids symboliques, multi-agents

Abstract

Structured modeling of preference statements, grounded in the notions of preferential independence, has tremendous potential to provide efficient approaches for modeling and reasoning about decision maker preferences in real-life applications. This thesis raises the question of representing preferences through a graphical structure. We propose a new reading of possibilistic networks, that we call π-pref nets, where possibility weights represent satisfaction degrees. The approach uses non-instantiated possibility weights, which we call symbolic weights, to define conditional preference tables. These conditional preference tables give birth to vectors of symbolic weights that reflect the preferences that are satisfied and those that are violated in a considered situation. We then focus on the theoretical aspects of handling of these vectors. Indeed, the comparison of such vectors may rely on different orderings: the ones induced by the product-based, or the minimum-based chain rule underlying the possibilistic network, the discrimin, or leximin refinements of the minimum-based ordering, as well as Pareto ordering, and the symmetric Pareto ordering that refines it. We prove that the product-based comparison corresponds exactly to symmetric Pareto and we focus on its assets compared to the other ordering methods. Besides, we show that product-based ordering is consistent with the ordering obtained by comparing sets of satisfied preference tables. The picture is then completed by the proposition of algorithms for handling optimization and dominance queries.

In this work we discuss various graphical tools for preference representation. We shed light particularly on CP-nets since they share the same graphical structure as π-pref nets and are based on the same preference statements. We prove that the CP-net orderings cannot contradict those of the π-pref nets and we found suitable additional constraints to refine π-pref net orderings in order to capture Ceteris Paribus constraints of CP-nets. This indicates that CP-nets potentially represent a subclass of π-pref nets with constraints. Finally, we provide an thorough comparison between the different qualitative and quantitative graphical models and π-pref nets. We deduce that the latter can be positioned halfway between qualitative and quantitative models since they do not need a full instantiation of the symbolic weights while additional information about the relative strengths of these weights can be taken into account.

The last part of this work is dedicated to extent the proposed model to represent multiple agents preferences. As a first step, we propose the use of possibilistic networks for representing all or nothing multiple agents preferences and define conditioning in the case of Boolean possibilities. These multiple agents networks have a logical counterpart helpful for checking agents consistency. We explain the main steps for transforming multiple agents networks into logical format. Finally, we outline an extension with priority levels of these networks and provide algorithms for handling optimization and dominance queries.

Keywords: Possibility theory, preference networks, symbolic weights, multiple agent

Contents

List of Figures

List of Tables

List of Algorithms

INTRODUCTION

Preferences have been studied in philosophy, economics, psychology and computer science and have a wide range of applications, including e-commerce, recommender systems and control. Modelling preferences is inherent in such applications and is considered as a prerequisite for any kind of thorough decision analysis. Handling such information needs a clear distinction between the knowledge about a state of the world and the preferences of an agent about it. Research on preferences in Artificial Intelligence (AI) has offered various ways of tackling the problem of representing preferences, from their acquisition to their formal representation and manipulation. Roughly speaking, two distinct ways to construct such models are possible, each focussing on a different aspect of acquiring preferences. Preference elicitation methods need the user interaction for the construction of the preference formalism while preference learning applies machine learning techniques on available data to predict a model following the specific characteristics of the chosen model. Once the preference model is available, two queries are usually considered: Optimization and dominance queries for finding the optimal solution and for comparing solutions respectively.

In this thesis, we are interested in preference models that exhibit a graphical structure and are based on an independence relation. In fact, structural properties of preferences can help to reduce the dimensionality of the value function, and thus achieve a compact representation of preferences and, most importantly, a compact representation reduces computation tasks and the elicitation burden.

In classical decision theory, specifying preferences comes down to determining a value function that enables to compare all possible situations. Since the late 1990s, Artificial Intelligence is interested in the representation of partially specified and contextually expressed preferences. The problem, thus, is to reconstruct, if not a value function, at least an order relation between all possible situations. This is called compact representation of preferences.

Roughly speaking, one may distinguish between qualitative and quantitative settings. In quantitative models, such as Generalized Additive Independence networks (GAI nets) [Gonzales and

Perny, 2005] representing preferences comes down to constructing a value function that enables us to compare all possible situations. However, decision-makers are rarely able to express their preferences directly in terms of numerical local value functions due to the considerable cognitive burden of determining accurate numerical values. Instead, qualitative models such as Conditional Preference networks (*CP-nets*) [Boutilier et al., 2004a, Boutilier et al., 2004b] allow the representation of partially specified and contextually expressed preference relations.

Indeed, Conditional preference networks (CP-nets) [Boutilier et al., 2004a, Boutilier et al., 2004b] are a popular example of qualitative models. They offer a graphical approach and ordinal representation of preferences, and provide a simple and compact way to specify them. But despite their success, CP-nets may induce debatable priorities between decision variables and lack a logical counterpart.

Possibility theory offers a natural and simple model to handle uncertain information. It is an appropriate framework for experts to express their opinion about uncertainty numerically using possibility degrees (finite scale values) or qualitatively (ordinal) using total preorder on the universe of discourse. This theory has been used in different areas such as default reasoning [Benferhat et al., 1997], qualitative decision [Dubois and Prade, 1995] and preference representation [Benferhat et al., 2001b]. Possibilistic logic [Dubois and Prade, 2004] is a possibilistic framework that can be used for preference representation [Benferhat et al., 2001b]. Beside its capability to express knowledge efficiently and reason with it, this logic is, as well, very efficient to deal with preferences. It induces a total pre-order thanks to its semantics in terms of possibility distributions [Dubois et al., 2006].

This thesis explores the representation of preferences by possibilistic networks, outlined in [Ben Amor et al., 2014] and establishes formal results about them. In fact, during this thesis we are particularly interested in the compact representation of preferences. Our main goal is to conduct thorough study of the possibilistic network based model, define its properties and develop its potential extensions.

In the first part of this thesis, Chapter 1 and 2 offer necessary background on graphical models for representing preferences. Moreover, we recall the basic notions of preferences and orderings.

Chapter 3 is dedicated to our proposed preference representation model. Our focus is, at first, to define the main properties of the model and to highlight its assets. This preference model relies on non-instantiated symbolic degrees which leads to associating each solution to a vector of weights. One can rank-order the different solutions in many different ways. Regarding to this fact, we propose a comparative discussion between the ordering offered by the possible ordering relations.

In addition to this theoretical contribution, we consider the importance to compare the expressive power of our model with other existing models. We aim to highlight some possible transformations between some well-known graphical and logical models for preferences. Chapter 4 is dedicated to this purpose.

Nowadays, collective preference models form an attractive field that attracts much interest in AI community In several areas, it is necessary to have a synthetic view of the preferences of groups of agents. To the best of our knowledge, all the contributions provided in this domain manage the agents based on their proportions not only in terms of profiles. This raises the problem of the extension of possibilistic networks model for preference that can represent the preferences of a group of agents. This model should be able to process these preferences with or without graded satisfaction degrees. Besides, this intended model should allow indifference and the inconsistency of some agents.

In this dissertation, we address various theoretical and applicative issues. Our main contributions are:

- Review all well-known graphical models,
- Set the characteristic of preference possibilistic networks and present their possible queries,
- Compare the different ordering relations that may be induced by the model and choose the most appropriate one from our point of view
- Compare the expressive power of our model to CP-net and some other graphical and logical models for preferences,
- Propose a new qualitative multiple agents graphical model corresponding to an extension of the aforementioned model,
- Extend qualitative multiple agents graphical model to represent gradual preferences,
- Propose a toolbox that permits the reasoning with the proposed graphical models.

Papers representing the results of this thesis are listed at the end of this thesis [1].

This thesis is organized as follows:

- Chapter 1 introduces the basic concepts relative to preferences and reviews all well known qualitative graphical models,
- Chapter 2 is devoted to some graphical models for representing quantitative preferences,
- Chapter 3 details a new graphical model based on possibilistic networks. In this chapter, we focus on needed information to construct the models, the use of symbolic weights and the possible queries that may be performed on the model.
- Chapter 4 aims to position the proposed models with regards other existing qualitative and quantitative models.

[1] References are [Ben Amor et al., 2015] [Ben Amor et al., 2016b] [Ben Amor et al., 2016a] [Ben Amor et al., 2017b] [Ben Amor et al., 2017a]

- Chapter 5 proposes a new multiple agent graphical model for representing collective and qualitative preferences. We propose algorithms and a quantitative extension for the latter.
- Chapter 6 proposes a toolbox implemented in Matlab that offers reasoning algorithms for the proposed graphical models

CHAPTER 1

Graphical Preferential Qualitative Models

Contents

1.1 Introduction

Modeling preferences is essential in any decision analysis task. However, eliciting these preferences becomes non trivial as soon as configurations are described by a Cartesian product of multiple features such that a configuration is defined as a complete instantiation of all decision variables. Indeed, the direct assessment of a preference relation between these alternatives is usually not feasible due to its combinatorial nature. Thus, the Artificial Intelligence community has carried out extensive research on this topic and has produced a huge number of preference representation approaches [Ben Amor et al., 2016a, Kaci et al., 2014, Domshlak et al., 2011], differing either by their nature (i.e quantitative or qualitative), type (i.e. graphical or logical), or their represented preferences (eg. *Ceteris Paribus*). A general description of preferential models from the Artificial Intelligence standpoint would be to divide such models into three components; the language,

corresponding to the form of the preference information communicated by the user, the model, defined by the induced preference relations between the solutions, and finally the queries performed on the model and exploiting its ability of aggregation [Domshlak, 2008].

A Preference model should achieve a good compromise between three conflicting aspects. First, the need for sufficient flexibility to describe sophisticated decision behaviors. Second, the practical necessity of keeping the elicitation effort at an admissible level as well as the need for efficient procedures to solve preference-based optimization problems. In a preference model, two types of queries are commonly used: namely, optimization queries for finding the optimal configuration(s) (i.e. those which are not dominated by others) and dominance queries for comparing configurations. The other important task is the elicitation of the model which corresponds to collecting user preferences and building the model.

Fortunately, the decision maker can express contextual preferences that exhibit some independence relations between decision variables, which allows us to be represent her/his preferences in a compact graphical manner. Moreover, graphical representations facilitate preference elicitation, as well as the construction of an ordering of configurations from these contextual local preferences. This use of graphical preference representations has been inspired by the success of Bayesian networks as a rigorous and computationally tractable uncertainty management device [Pearl, 1988]. Graphical preference models may be classified based on one main criterion. Namely, preferences can be represented in a *qualitative* or *quantitative* manner. Most of practically used preferential graphical models are qualitative [Santhanam et al., 2016, Ben Amor et al., 2016a] since they are easy to elicit. In the sequel, we detail some of the most important ones, namely, Conditional Preference networks (CP-nets) [Boutilier et al., 1999], their extension Tradeoffs-enhanced CP-nets [Brafman and Domshlak, 2002] and Preference trees (P-tree) [Liu, 2016]. These models allow to express various kinds of preference relations over variables or/and their domains. We detail the semantics behind such networks and the induced preference graph that encodes an order between configurations from a given set of preferences. Chapter 2 is dedicated to graphical models where preferences are quantitative.

The chapter is organized as follows. Section 1.2 sketches a general scheme for reasoning about preferences from AI point of view. Section 1.3 offers a reminder of some binary relations that may exist between two configurations (i.e. two complete instantiation of the decision variables used for describing alternatives) and reviews some classical orderings on a set of configurations. In Section 2.4, we present a well-known graphical qualitative model based on Ceteris Paribus independence, entitled CP-nets. Section 2.5 addresses an extension of CP-nets that admit some importance relations between variables. Section 2.6 is devoted to P-trees; a new tree-structured qualitative model that constructs a total ordering between configurations.

1.2 Preference models

The literature of preference modelling is very vast. This can be justified by the various disciplines that are interested in the question of modelling preferences. We can cite for instance; economics [Debreu, 1959], psychology [Tversky and Kahneman, 1975] and artificial intelligence (AI) [Kushmerick et al., 1995].

The general approach of AI to reason about a user preferences has 3 major components, and was proposed in [Domshlak, 2008]:

- The language: In each preference model, choosing the language which is the closest and the most intuitive to the user is important. For instance, pairwise comparisons are very appealing for users since no ambiguity in the interpretation of the information is possible. However, such language turns to be less appealing when the number of configurations is large. As an example, its clear that a user is unable to directly specify an ordering between one hundred of vacation house configurations. Therefore, other preference specification languages should be considered. It was noticed that users are generally willing to provide so-called *generalizing* statements of preference (i.e., partially instantiated preference statements applying to a large class of situations). For instance, I prefer to spend my vacations next to the beach than downtown. This indicates that the user when generalizing refers to some characteristics of the configurations offered. More precisely, the user here prefers the configurations of vacation houses that have the characteristic *next to the beach* to the others. Various ways on how to interpret generalizing preference statements have been proposed in philosophy and AI. Whereas, no agreed-upon generalization is suggested to be the best. In this chapter, we will be concerned by graphical models where generalization (completion) of the contextual preference statements amounts to comparing configurations. For instance CP-nets adopt the Ceteris Paribus completion while possibilistic reasoning suggests some other completion principle based in the Markov property (Chapter 3).
- The preference ordering (called 'model' in [Domshlak, 2008]): It consists in the ordering underlying a preference representation. One should make an assumption on the resulting ordering of configurations i.e. the ordering can be partial or total, weak or strict. Generally, in qualitative models, the ordering induced from the model corresponds to the comparisons constructed after the completion step or by transitivity.
- The algorithms: To reason with preference models various types of query are suggested. Generally, the graphical structure of the models plays an important role in reasoning about preferences, even if the user may not be aware of its existence. Most important queries correspond to finding the best configuration (the set of the best configurations) or finding the preference between two configurations. Other variants of such queries are sometimes considered.

Figure 1.1 presents the AI methodology for reasoning about preferences. This general model/

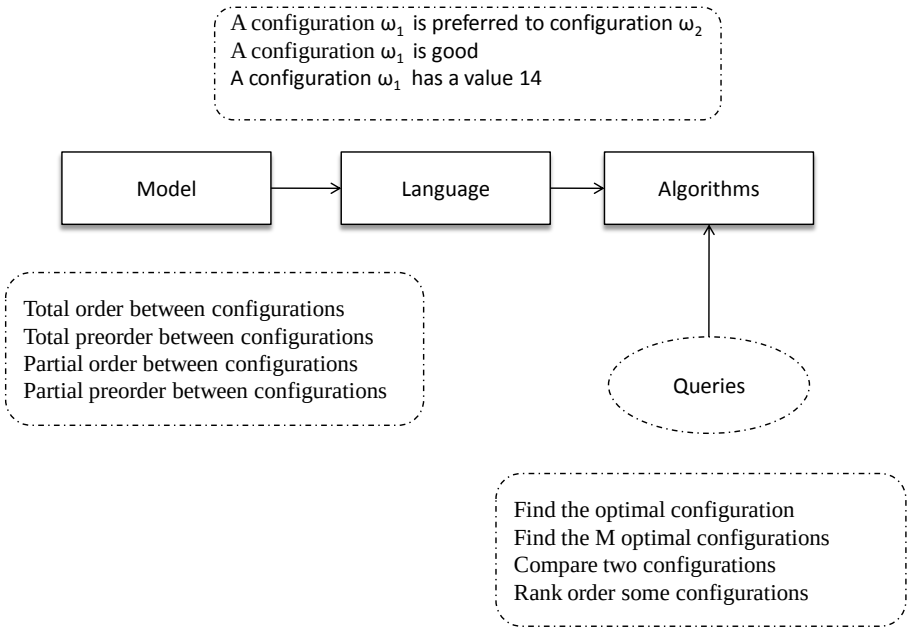


Figure 1.1: AI methodology for reasoning about preferences [Domshlak, 2008]

language/algorithms scheme can be instantiated in several ways, and each of these instantiations does unavoidably face some critiques from one side or another.

Finally, preference models are formal representations of preference relations between solutions. Such models should be established through the use of a formal language capturing the structure of the described preference and the manipulation of it. Considering formal logic as such a language is well-founded. However, the need of representing such preferences in a succinct, user-friendly way has urged the AI community to offer graphical model representations for this aim. Indeed, graphical models enable the representation of dependence/independence between variables which helps the problem to be decomposable. They permit a local processing of elementary preferences by exploiting the structural independence, represented by their graphical component, of the preferential relations. More precisely, the compactness of the language comes from the exploitation of preferential independence. In fact, thanks to the some observed independences among variables, a preference relation over a set of variables can be considered as preference relation pertaining to subsets of these variables, and thus, shown in a more compact way.

Qualitative preferences can correspond either to intra-variable preference relations over the values of a variable or to relative importance preference relations over variables. More precisely, relative importance preferences define priorities between variables, for instance satisfying the preferences of the variable A_1 is more important than satisfying the preferences of the variable A_2. While intra-variable preferences with respect to a variable consist on specifying an order relation between the values that may take this variable.

Prior to the problem of the preference representation, the problem of the elicitation or learning of these models arises. Regarding elicitation, building preference models needs an interaction with the user in order to acquire the necessary information and to offer a satisfactory result. Elicitation depends on the chosen representation language and exploits the structure of preferences to reduce the amount of information elicited and the cognitive effort of communication. Therefore, to facilitate the elicitation process, it is important that the preferential language is as intuitive as possible. In contrast, learning preference networks from data clearly differs from elicitation. Indeed, the learning system does not interact with the user by asking some specific requests in order to lead him to express all his preferences. However, the model is built from some observed past user behaviors

1.3 Ordering relations

Binary relations [Bouyssou and Vincke, 2010] are the central tool of most qualitative models. In this section we give an overview of mathematical and computational concepts that will be used throughout the rest of this thesis. First, since preference relations are modelled as binary relations, we recall the definition of the latter and some of their key properties. Then, based on these properties, we present several types of preference structures.

1.3.1 Binary preference relations

Let $\mathcal{V} = \{A_1, \ldots, A_N\}$ be a set of N variables. Each variable A_i has a value domain $D(A_i)$. Elements $a_i \in D(A_i)$ denote values of A_i. $\Omega = \{\omega_1, \ldots, \omega_n\}$ denotes the universe of discourse, which is the Cartesian product of all variable domains in V. Each element $\omega_i \in \Omega$ is called a configuration (or a solution). It corresponds to a complete instantiation of the variables in $\mathcal{V}$. We call a binary relation on Ω a subset of the Cartesian product $R \subseteq \Omega \times \Omega$. We write $\omega_i R \omega_j$ for $(\omega_i, \omega_j) \in R$ and $\omega_i \neg R \omega_j$ for $(\omega_i, \omega_j) \notin R$.

Let us first remind some properties of binary relations:
A binary relation R is said to be:

- Reflexive $\Leftrightarrow \forall \omega_i \in \Omega$, $\omega_i R \omega_i$
- Irreflexive $\Leftrightarrow \forall \omega_i \in \Omega$, $\omega_i \neg R \omega_i$
- Symmetric $\Leftrightarrow \forall \omega_i,\ \omega_j \in \Omega, \omega_i R \omega_j \Rightarrow\ \omega_j R \omega_i$
- Anti-symmetric $\Leftrightarrow \forall \omega_i,\ \omega_j \in \Omega, \omega_i R \omega_j \wedge \omega_j R \omega_i \Rightarrow\ \omega_i = \omega_j$
- Asymmetric $\Leftrightarrow \omega_i R \omega_j \Rightarrow \omega_j \neg R \omega_i$
- Transitive $\Leftrightarrow \forall \omega_i,\ \omega_j,\ \omega_z \in \Omega, \omega_i R \omega_j \wedge \omega_j R \omega_z \Rightarrow\ \omega_i R \omega_z$

- Complete $\Leftrightarrow \forall \omega_i,\ \omega_j \in \Omega$, $\omega_i R \omega_j$ or $\omega_j R \omega_i$
- Weakly complete $\Leftrightarrow \forall \omega_i,\ \omega_j \in \Omega$, $\omega_i R \omega_j$ or $\omega_j R \omega_i$ or $\omega_i = \omega_j$
- Acyclic $\Leftrightarrow \forall\ k > 2$, $\forall\ \omega_i \in \Omega$, $i = 1, 2, \ldots, k$, $\omega_1 R \omega_2$ and $\omega_2 R \omega_3$ and $\ldots$, $\omega_{k-1} R \omega_k$ $\Rightarrow \omega_1 \neq \omega_k$

The above mentioned properties are not always independent [Bouyssou and Vincke, 2010]. Indeed, we can check that:

- Asymmetric relation $\Leftrightarrow$ irreflexive and antisymmetric relation.
- Complete relation $\Leftrightarrow$ reflexive and weakly complete relation.
- Asymmetric relation $\Leftrightarrow$ irreflexive and antisymmetric relation.

For each pair of configurations (ω_i, ω_j), we are in one of the following four cases:

- $\omega_i R \omega_j$ and $\omega_j R \omega_i$, denoted by $\omega_i \sim \omega_j$, is interpreted as 'ω_i is indifferent to ω_j'.
- $\omega_i \neg R \omega_j$ and $\omega_j \neg R \omega_i$, denoted by $\omega_i \pm \omega_j$, is interpreted as 'ω_i is incomparable to ω_j'.
- $\omega_i R \omega_j$ and $\omega_j \neg R \omega_i$, denoted by $\omega_i \succ \omega_j$, is interpreted as 'ω_i is strictly preferred to ω_j'.
- $\omega_i \neg R \omega_j$ and $\omega_j R \omega_i$, denoted by $\omega_i \prec \omega_j$, is interpreted as 'ω_j is strictly preferred to ω_i'.

$\succ$ is the asymmetric part of $\succeq$ and $\sim$ its symmetrical part. The relation $\succeq$ is said to be a preference relation in the wide sense, $\succ$ a strict preference relation and $\sim$ an indifference relation. We note the opposite relations $\prec$ and $\preceq$ preference relation such as:

$$\omega_i \prec \omega_j \ \Leftrightarrow\ \omega_j \succ \omega_i \ and \ \omega_i \preceq \omega_j \ \Leftrightarrow\ \omega_j \succeq \omega_i$$

If the preference relation of the decision-maker is reflexive and transitive, these preferences meet the strong assumptions of the decision maker rationality defined as follows:

Definition 1.1. *A preference relation respects the strong assumptions of the decision maker rationality, if and only if:*

- *The intersection between $\sim$ and $\succ$ is empty;*
- *$\sim$ is reflexive and symmetric;*
- *$\succ$ is asymmetric;*
- *$\succeq$ is transitive.*

Preference structure	Properties
Total order	complete antisymmetric transitive
Total preorder	complete transitive
Partial order	reflexive antisymmetric transitive
Preorder	reflexive transitive

Table 1.1: Classical preference structures

1.3.2 Classical preference structures

The total order consists on ranking configurations without the possibility of indifference.

Definition 1.2. *A total order is a transitive, antisymmetric and complete binary relation.*

In a total order, incomparability relations are forbidden and the indifference is only found between two identical configurations.

Definition 1.3. *A partial order is a reflexive, antisymmetric and transitive binary relation.*

In a partial order, two distinct configurations are either strictly preferred or the two configurations are incomparable, such that the strict preference is transitive.

Definition 1.4. *A partial preorder is a reflexive and transitive binary relation.*

Definition 1.5. *A total preorder is a partial preorder that is complete.*

A total order is a total preorder which is antisymmetric. Preorders are more general than partial orders, which are special cases of preorders. Indeed, they allow indifference between distinct elements, having the indifference relation as transitive. Table 1.1 summarizes the characteristics of the preference relations presented above.

These notions are illustrated with several examples of Figure 1.2 [Liu, 2016]:

Example 1.1. *We assume that the directed edges correspond to strict preferences from a less preferred configuration to a more preferred one, i.e.* $\omega_i \rightarrow \omega_j$ *means that* ω_j *is strictly preferred to* ω_i*. Note that if a node contains more than one configuration then the preference relation between these configurations is indifference. For Figures 1.2a and 1.2b we consider that we have*

4 configurations $\omega_1, \ldots, \omega_4$. It can be checked that Figure 1.2a is a partial order since ω_2 and ω_4 are non compared, Figure 1.2b is a total order since all the configurations can be compared and there is no equality between configurations. The preference ordering expressed by Figure 1.2c and 1.2d is over 8 configurations, namely, $\omega_1, \ldots, \omega_8$. Figure 1.2c represents a partial preorder since ω_8 and ω_7 are equally preferred while ω_8 and ω_4 are non-compared and Figure 1.2d is a total preorder.

1.4 Conditional Preference Networks (CP-nets)

The direct specification of a binary preference relation on $\Omega = \{\omega_1, \ldots, \omega_n\}$ is rather difficult, as it requires the user to compare up to $O(n^2)$ pairs of configurations, which is huge in terms of time (n is exponentially large) and effort. Therefore, qualitative preference models allow for a local and concise specification of preferences.

The user is assumed to express preferences under the form of comparisons between values of each variable, conditioned on some other instantiated variables. CP-nets deal with strict *preference statements*. Unconditional statements are of the form: "I prefer a^+ to a^-", where $a^+, a^- \in \{a, \neg a\}$ and $a^- = \neg a^+$, and we denote them by $a^+ \succ a^-$. When $A = a^+$, we say that the *quality* of the choice for A is good, and is bad otherwise. If the preference on A depends on other variables $\mathcal{P}(A)$ called the *parents* of A, and $p(A)$ is an instantiation of $\mathcal{P}(A)$, conditional preference statements are of the form "in the context $p(A)$, I prefer a^+ to a^-", denoted by $p(A) : a^+ \succ a^-$. To each variable we associate a table representing the local preferences on its domain values in each parent context (the value of a^+, respectively a^-, depends on the parents context).

1.4.1 Preference independence: Ceteris Paribus

CP-nets, initially introduced in [Boutilier et al., 2004a], are considered as an efficient model to manage qualitative preferences. This preference model is based on a preferential independence property often referred to as a Ceteris Paribus assumption such that a partial configuration is preferred to another everything else being equal. Formally, it is defined as follows:

Definition 1.6 (Preferential independence)**.** *Let $\mathcal{V}$ be a set of variables and X be a subset of $\mathcal{V}$. X is said to be preferentially independent from its complement $Y = \mathcal{V} \setminus X$ iff for any instantiations, $y,\ y',\ x,\ x'$,*

$$(y, x) \succ (y', x) \Leftrightarrow (y, x') \succ (y', x')$$

Preferential independence is asymmetric. Indeed, it might happen, e.g., for disjoint sets X, Y and Z of variables that X is preferentially independent (Definition 1) from Y given Z without having

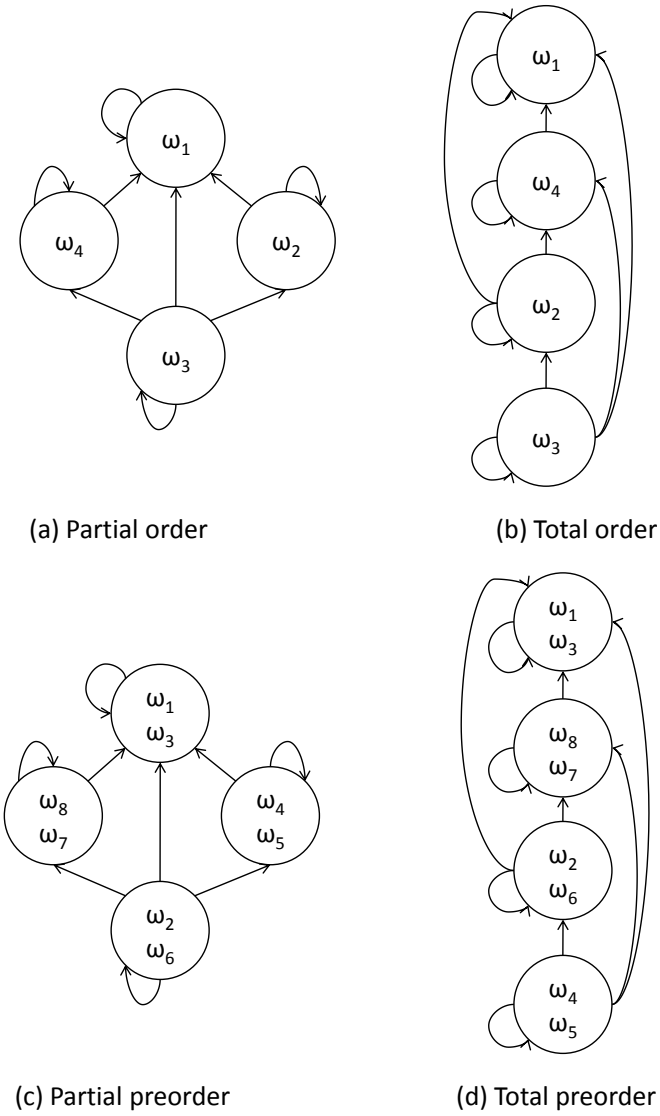


Figure 1.2: Illustrations of preference order relations

Y preferentially independent from X. This independence is at a work in the graphical structure underlying CP-nets.

1.4.2 Model definition and semantics

Preference networks can be viewed as a qualitative counterpart of Bayesian nets [Pearl, 1988]. More formally:

Definition 1.7 (CP-nets)**.** *A CP-net consists of a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges. A node corresponds to a variable. Edges represent the preference dependencies between the variables. To each variable A_i we associate a conditional preference table that corresponds to a strict total order between the values of A_i, for all instantiations $p(A_i)$ of parent variables $\mathcal{P}(A_i)$.*

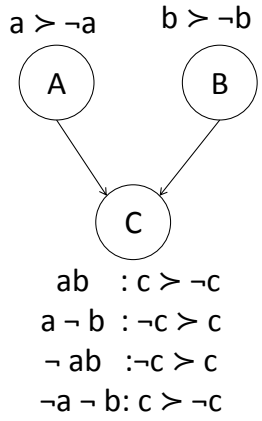


Figure 1.3: An example of a CP-net

Here, preferences over values of a variable depend only on the parent(s) context, and are preferentially independent from the rest of variables. Contrarily to Bayesian nets, CP-nets may be cyclic (without necessarily encoding inconsistent preferences). Each preference statement in the CP-net implicitly corresponds to a set of comparisons between pairs configurations. For instance, consider two independent variables House type= $\{flat, villa\}$ and size= $\{large, small\}$, if the user says "I prefer flats to villas, we can deduce that large flats are preferred to large villas and small flats are preferred to small villas. However, we cannot deduce that small flats are preferred to large villas. In fact, the preference relation is valid provided that the rest of the variables are instantiated in the same way.

A CP-net is said to be satisfiable if there exists at least one partial order of configurations that satisfies it. Note that every acyclic CP-net is satisfiable and leads to a unique partial order [Boutilier et al., 2004a].

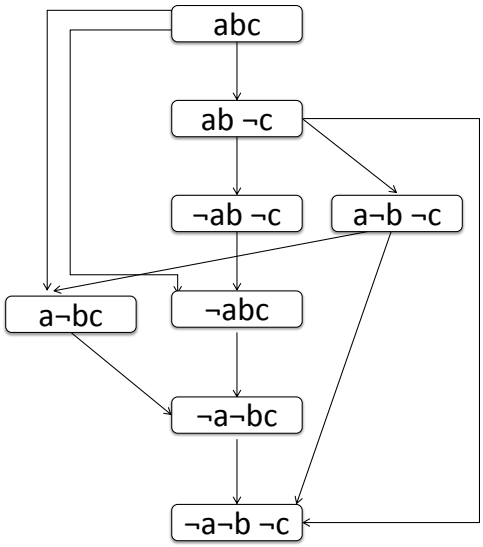


Figure 1.4: The worsening flips graph of the CP-net of Figure 1.3

Example 1.2. *Let us consider the simple CP-net of Figure 1.3, with 3 variables. This CP-net is acyclic thus it is satisfiable and leads to a unique partial order. The building of the worsening flips graph (Figure 1.4) leads to the partial ordering:* $abc \succ_{CP} ab\neg c \succ_{CP} \neg ab\neg c \succ_{CP} \neg abc \succ_{CP} \neg a\neg bc \succ_{CP} \neg a\neg b\neg c$, $ab\neg c \succ_{CP} a\neg b\neg c \succ_{CP} \neg a\neg b\neg c$, $abc \succ_{CP} a\neg bc \succ_{CP} \neg a\neg bc$ *which means that the optimal configuration is* abc.

1.4.3 Reasoning with CP-nets

Given a CP-net, two queries can be considered, namely, optimization query for finding the best configuration and dominance queries defined by finding a preference relation between two configurations. More precisely:

- As mentioned above, acyclic CP-nets have a unique optimal configuration. Finding it amounts to looking for a configuration where all the conditional preferences are satisfied. It can be done by a simple forward sweeping procedure where, for each node, we assign the most preferred value according to the parents context. For acyclic CP-nets, this procedure is linear w.r.t. the number of variables [Boutilier et al., 2004a]. In contrast, for cyclic ones answering this query needs an NP-hard algorithm and may lead to more than one optimal configuration [Goldsmith et al., 2008].
- Dominance queries are more complex. Using the information in the CP-Tables and applying the *Ceteris Paribus* principle, when one *flips* one variable value in a configuration one may obtain either an improved configuration, or a worsened one. These swap pairs can be organized into a collection of worsening (directed) paths with a unique root corresponding to

the best configuration and where the other extremity is the worst one. Thus, a configuration is preferred to another if there exists a chain (directed path) of worsening flips between them [Boutilier et al., 1999]. Note that if for any variable $A_i \in \mathcal{V}$, A_i is preferentially independent from $\mathcal{V} \setminus A_i$, then the CP-net graph is disconnected and many configurations cannot be compared. Testing dominance is PSPACE-complete for unrestricted CP-nets, NP-hard for acyclic ones, and quadratic for tree-structures [Goldsmith et al., 2008]. In a recent work [Xin and Liu, 2016], authors have proved that there is a connection between dominance querying and single source shortest path problem [Orlin et al., 2010]. They proposed a new exact dominance algorithm entitled *Johnson algorithm* for any binary-valued CP-net which runs in a time complexity equal to $O(2^n * n^2)$.

In general, the ordering induced by a CP-net is strict and partial, since several configurations may remain incomparable (i.e. no worsening flips chain exists between them). Clearly, acyclic CP-nets cannot exhibit any ties.

Randomly generating CP-nets is helpful for learning experiments [Allen et al., 2014]. Indeed, it enables experimental analysis of CP-net reasoning algorithms, understanding their properties and simulating some social choice experiments. Some research works gave importance to this topic such as [Allen et al., 2016, Allen et al., 2014]. In fact, recently, a novel algorithm for provably generating acyclic CP-nets uniformly at random was proposed in [Allen et al., 2016]. The proposed method allows for multi-valued domains and an arbitrary bounds on the indgree of the dependency graph.

1.4.4 Expressivity of CP-nets

In CP-nets, a parent preference tends to be more important than a child one [Boutilier et al., 1999]. In other words, violating a preference associated with a father node is more important than violating a preference associated with a child one; this priority *implicitly* given by the application of *Ceteris Paribus* may be debatable. For instance, in the previous example, configuration $ab\neg c$ is preferred to configuration $\neg abc$. Moreover, this kind of priority is not transitive in the sense that CP-nets cannot always decide whether violating preferences of two children nodes is preferred to violating preferences associated with one child and one grandson node respectively (which might have been expected as being less damaging than violating two children preferences) [Dubois et al., 2013b]. This limitation is problematic since these priorities cannot be questioned and modified. Generally, there are partial preference orderings that CP-nets cannot express, see [Ben Amor et al., 2015] for counterexamples.

Only few works discuss the expressivity and succinctness of CP-nets. Investigation on the expressive power of CP-nets in a quantitative manner are proposed in [Liu and Liao, 2015]. More precisely, the authors address a new concept called expressive efficiency that quantifies the trade-off between expressivity (i.e. the possible preference relations that the model can express) and succinctness (i.e. how much space is needed to store all the preferences) of CP-nets. Moreover, the

authors investigate the *expressivity* of two kinds of binary-valued CP-nets namely, set-structured and equal difference CP-nets. Indeed, the preference relations expressed by CP-nets are directly related to the dependency graph. Set-structured CP-nets are specified by a graphical structure where all the nodes are independent (i.e. there is no edges). It was proved that such networks can express $3^N - 2^N$ preference relations [Liu and Liao, 2015]. Besides, since all nodes are independent, there is only one preference statement per variable. This means that only N space is needed to store the preference statements. The other studied kind of CP-net is equal difference CP-nets. These latter are defined by a structure where each node connects to all its followings i.e., the root node is directly related to all the rest of the variables which means that the network has only one root and only one leaf. They proved that from this kind of structure one can construct a total order between configurations. That is, based on this structure, one can express all the preference relations between the configurations. Thus, we have $2^{N-1} * (2^N - 1)$ preference relations and $2^N - 1$ space is needed to store the conditional preference statements. However, the number of preference relations expressed by an arbitrary structured CP-net remains an open problem [Liu and Liao, 2015].

In general, CP-nets are restricted to binary-valued variables. This would restrict the expressivity of CP-nets. Even though the semantics allows for such representations, algorithms for multi-valued acyclic CP-nets were neglected. Roughly, optimization query is easy even for multi-valued CP-nets, however, dominance queries are hard. In fact, the number of nodes of the configuration graph built for finding a worsening path from a configuration to another grows exponentially with the number of configurations [Yaman and Desjardins, 2008]. A methods for handling such variables is to reduce their number into group of multiple values. In [Yaman and Desjardins, 2008], authors have identified a class of multi-valued CP-nets, entitled more-or-less CP-nets, that have the same computational complexity as binary-valued CP-nets. In fact, more-or-less CP-nets consider ordinal variables, i.e monotonic variables, and assume the existence of a single critical point per variable. To reason with such networks, one should aggregate the preferences of a range of values together. Then, after a slight modification of binary dominance algorithm, one can execute the task with the same computational complexity (i.e. NP-complete to PSPACE). However, this class of CP-nets is too restrictive and reasoning with multi-valued CP-nets is still an open question.

The expressivity of CP-net is also restricted by the fact that standard CP-nets cannot express indifference. In fact, each preference statement must be defined as a total order between the values of the variable. This can be explained by the fact that a CP-net that bears indifference may be non satisfiable as illustrated in Example 1.3.

Example 1.3. *Let us consider a CP-net over two variables A and B, such that A is the parent of B. With the following preference tables: $a \sim \neg a$, $a : b \succ \neg b$, $\neg a : \neg b \succ b$. This asserts that the user is indifferent between the values that A may take. The following preferences is deduced: $ab \succ a \sim b \succeq \neg a \neg b \succ \neg ab \sim ab$. These statements are not consistent with any preference ranking, hence this network is not satisfiable*

Some researchers have proposed to extend CP-nets in order to express indifference [Allen,

2013], since the assumption of a strict preference is too strong from a psychological standpoint and is often too restrictive even for simple preferential problems. [Allen, 2013] proposed to learn CP-nets where preference statements may include incomparability and indifference. For instance, 'big house $\sim$ small house' means that the user is indifferent about the size of the house. Authors consider a more general question for dominance corresponding to *weak* dominance i.e. whether $\omega_1 \succeq \omega_2$ or $\omega_1 \preceq \omega_2$. This entails searching for a monotonically worsening flipping sequence such that a worsening flip either corresponds to strict preference or indifference. If after the transitive closure and in the absence of a rule such that $\omega_1 \succeq \omega_2$ or $\omega_1 \preceq \omega_2$ then the two configurations remain non compared $\omega_1 \pm \omega_2$.

Constraints on the structure of CP-nets may restrict their expressivity. Indeed, cyclic CP-nets [Brafman and Dimopoulos, 2004] emerge naturally when there is a set of variables that are mutually dependent. In certain situations, it may be more natural to express cyclic preferences even if an acyclic representation could be used, for instance, for preference-based configuration of web page content [Domshlak et al., 2001]. It was noted that such cyclic networks are sometimes consistent and the semantics of CP-nets allows for such cycles (i.e. nothing in the semantics of the CP-net model forces it to be acyclic). It is important to note that cyclic CP-nets do not always lead to a cyclic order of configurations. Testing consistency of general cyclic CP-nets was proposed in [Goldsmith et al., 2008] and proved to be PSPACE-complete along with the dominance testing.

Other extensions that may somewhat enhance the expressivity of CP-nets (including Probabilistic CP-nets (PCP-nets) [Bigot et al., 2013] and Multiple agents CP-nets (mCP-nets) [Rossi et al., 2004] are not covered here since they enlarge the representation to other features, namely uncertainty or multiple agents).

1.4.5 How to build CP-nets?

CP-nets can be constructed by an expert, elicited from users, or learned from data. Each of these methods may show some strengths and weaknesses. The first method needs a good expertise and knowledge. Indeed, for complex domains even if the graphical structure may seem to be intuitive, finding dependencies and choosing the variables is not always practical.

The second method needs a long process of intake before the model can be employed. [Lang and Mengin, 2009] were interested in passive learning of separable CP-nets. These are set-structured CP-nets whose variables have no conditional dependency, that is to say that the graph has no arc. After that, [Koriche and Zanuttini, 2010] have been interested in eliciting boolean tree-structured acyclic CP-nets. Then, a heuristic algorithm for learning CP-nets from user query was proposed in [Guerin et al., 2013, Allen, 2014]. It has two phases. The first corresponds to constructing a separable CP-net with default CP-tables. Then, iteratively, it refines the model by adding edges and forming more complex CP-tables. While assuming a bound on the number of parents, the algorithm succeeds to construct a CP-net in time $O(N^p)$ such that N is the number of nodes and p is the bound on the number of parents of each node. The algorithms guarantees that

the preferences it outputs are always consistent with the user preferences.

An empirical study was held to argue about the suitability and tractability of elicitation and validation algorithms with humans [Allen et al., 2015]. In fact, preference information is often messy, not openly available, thus it needs a huge work to collect. Besides, the reliably of such preferences is not always ensured because of the difficulty of interpretation. Moreover, users may express preferences that are not representable by the preferential model. The experimental research aims to test whether users subjectively represent preferences in a way that is consistent with CP-nets. Following this line of stepping back and looking at decision-support systems from the user's point of view, [Patel, 2016] designed a tool which uses comparative preference statements on the basis of psychological, linguistic and personal considerations.

The last method requires weeks of observations in order to construct a model satisfying the preferences of a user. An algorithm for learning the CP-net from outcome comparisons data was presented in [Dimopoulos et al., 2009]. The algorithm takes as input a set of variables and a set of comparisons between outcomes and outputs a CP-net that is consistent with preferences afforded. Indeed, as a first step it initializes an empty CP-net and adds a node per iteration starting by finding the independent nodes. This learning is proved to be NP-hard even under some simplifying hypothesis.

Preference elicitation is a serious problem in many preference modeling tasks. Models based on Ceteris Paribus independence were designed to make this process simpler and more intuitive which is enhanced by the graphical structure that CP-nets have.

1.5 Tradeoffs-enhanced CP-nets (TCP-nets)

As mentioned above, the expressive power of CP-nets is limited. In particular, we are unable to specify importance relations between variables, beside those implicitly imposed between parents and children. Tradeoffs-enhanced CP-nets (TCP-nets) [Brafman and Domshlak, 2002] are an extension of CP-nets that adds a notion of importance between the variables by enriching the network with new arcs. These arcs express importance relations for stating the priority of a node over another (i.e.,"preference about the values of A_1 is more important than preference about the values of A_2"). Such priority statements may be conditioned on the values of other variables, e.g., "if the variable A_3 has value a_3, the preference about values of A_1 is more important than the preference about the values of A_2." Formally, TCP-nets are annotated graphs with three types of edges and are defined as below.

Definition 1.8 (TCP-nets)**.** *A TCP-net $\mathcal{G}'$ over a set $\mathcal{V}$ of variables is a CP-net $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ augmented with two types of arcs:*

1. *A set of directed i-$arcs$ (where i stands for importance). An i-$arc\langle\overrightarrow{A_i, A_j}\rangle$ belongs to $\mathcal{G}'$ iff A_i is more important than A_j, which is denoted by $A_i \triangleright A_j$.*

2. *A set of undirected ci-$arcs$ (where ci stands for conditional importance). A ci-$arc\,(A_i, A_j)$ belongs to $\mathcal{G}'$ iff the relative importance of A_i and A_j is conditioned on Z s.t. $Z \subseteq \mathcal{V}\backslash\{A_i, A_j\}$. Each ci-$arc\,(A_i, A_j)$ is associated with a mapping from a subset of D_Z to strict total orders over the set $\{A_i, A_j\}$.*

Let us turn to the expressive power of TCP-nets. TCP-nets obey the preference statements induced by *Ceteris Paribus*, since the ordering obtained is a refinement of the CP-nets ordering. In fact, the refinement brought by TCP-nets cannot override the implicit priority in favor of parents nodes. Indeed, in case one would add a $i-$, or a $ci-$ arc yielding a preference in favor of a son with respect to a parent (at least in some context), one would face an inconsistency between a worsening I-flip and a worsening CP-flip that act in opposite directions, thus we would have inconsistent TCP-nets.

The main issue for TCP-nets is the challenge of performing queries with this representation. Some first proposals are presented in [Brafman et al., 2006]. For consistent TCP-nets, the optimization procedure works like CP-nets. Indeed, the relative importance relations do not play a role in this case. The dominance problem can be also be treated as a search for an improving flipping sequence, where the notion of flipping sequence is extended. In fact, a flip corresponds either to a CP-flip like CP-nets or to an I-flip ("importance flip"). We assume that $\omega[X]$ denote the restriction of ω to variables in X. Let ω and ω' be two configurations, such that ω differs from ω' in the value of exactly two variables A_j and A_k, and such that $\omega[A_j] \succ \omega'[A_j]$ and $\omega[A_k] \prec \omega'[A_k]$ (given the same values of $\mathcal{P}(A_j)$ and $\mathcal{P}(A_k)$ in ω and ω'). Then, a worsening I-flip from ω to ω' takes place when there is a priority of A_j over A_k conditional (or not) on a subset of variables Z such that Z takes the same values in ω and ω'. However, no general algorithm is known for dominance query since results in the context of CP-nets do not seem to be immediately adaptable to TCP-nets.

Example 1.4. *Let us consider the TCP-net in Figure 1.5. An unconditioned importance $a \triangleright b$ is added. Indeed, a new arc i-$arc\langle\overrightarrow{A, B}\rangle$ is added with respect to the CP-net in Figure 1.3. The ordering given by the worsening flips graph in Figure 1.6 is refined, compared to the CP-net. Indeed, $a\neg b\neg c \succ_{TCP-net} \neg ab\neg c$ and $a\neg bc \succ_{TCP-net} \neg abc$, while these configurations comparable by I-flips, are not comparable in the CP-net, see Figure 1.3(b). In place of the previous unconditioned importance statement, one may exhibit an example of ci-$arc\,(A, B)$ by stating that A is more important than B if $C = c$, and B is more important than A if $C = \neg c$. Then, we would have $a\neg b\neg c \prec_{TCP-net} \neg ab\neg c$ and $a\neg bc \succ_{TCP-net} \neg abc$.*

TCP-nets also yield partial orderings that, from the same CP-net preference statements, are refinements of the ordering induced by the corresponding CP-nets.

Example 1.5. *Let us consider the following preferences over variables A and B with $D(A) = \{a, \neg a\}$ and $D(B) = \{b, \neg b\}$: (i) In all cases a is preferred to $\neg a$; (ii) b is preferred to $\neg b$. The CP-net view yields the order: $ab \succ_{CP} a\neg b \pm_{CP} \neg ab \succ_{CP} \neg a\neg b$. No CP-net yields the refined order $ab \succ a\neg b \succ \neg ab \succ \neg a\neg b$, while it can be represented with a TCP-net, with the additional information "A is more important than B".*

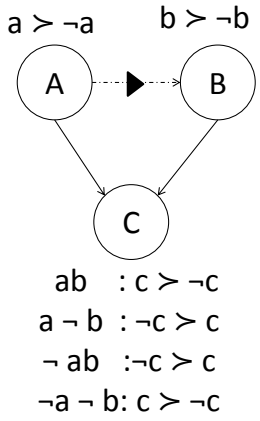


Figure 1.5: An example of a TCP-net

Some other related languages such as CI-networks [Bouveret et al., 2009] and CP-theories [Wilson, 2004] allow the statement of relative importance among sets of variables. Particularly, CP-theories, allow statements of the form "A_1 is more important than $\{A_2, A_3\}$". CI-nets are dedicated to the comparison of sets of goods rather than configurations. In fact, CI-nets allow unconditional and monotonic intra-attribute preferences and conditional relative importance preferences over sets of variables of the form "$\{A_1, A_2\}$ is more important that $\{A_3, A_4\}$". Formally, having $\mathcal{V}$ the set of binary variable, S^+, S^-, S_1 and S_2 subsets of arbitrary variables, a conditional importance statement on $\mathcal{V}$ is a quadruple $\gamma = (S^+, S^-, S_1, S_2)$ written as $S^+, S^-, S_1 \triangleright S_2$. In other words, this means that 'if I have all variables in S^+ and none of the variables in S^-, I prefer obtaining all variables in S_1 rather than obtaining all variables in S_1 Ceteris Paribus. CI-nets compare sets of variables of arbitrary sizes while TCP-nets can only express importance statements between single objects Ceteris Paribus [Bouveret et al., 2009]. Besides, in CP-nets, TCP-nets and CP-theories each preference statement can be represented in terms of preferences over values of a single variable, while CI-net expresses preference statements between sets of variables. Note that CI-networks do not express any conditional intra-variable preferences. This is explained by the fact that the larger the set of goods the better for the user (monotonicity w.r.t. set inclusion). Besides, CI-nets generalize TCP-nets since they can bear preferences on arbitrary sets of variables, and not only singletons. Since this chapter is dedicated to graphical models, CP-theories will be detailed in Chapter 4.

Table 1.2 summarizes the preference relations expressed by the preference models that are based on the Ceteris paribus independence. A check mark (✓) means that the model allows the corresponding preference relation type.

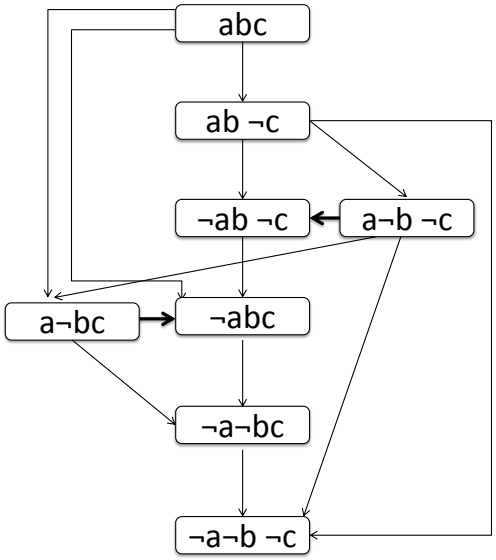


Figure 1.6: The worsening flips graph of Figure 1.5

Preference relation type	CP-nets	TCP-nets	CI-networks	CP-theories
Unconditional intra-variable preference		✓	✓	✓
Conditional intra-variable preference	✓	✓		✓
Conditional relative importance		✓	✓	✓
One-many relative importance			✓	✓
Many-many relative importance				✓

Table 1.2: Expessivity of *ceteris paribus* based models [Santhanam et al., 2016]

1.6 Preference trees

Conditionally lexicographic preferences are preferences where both relative importance between variables and intra-variable preferences are dependent on the values taken by some more important variables. A graphical representation of these preferences is presented by trees over binary variables with sometimes conditional preference tables. In this section, we present two tree structured models to represent lexicographic preferences, namely, LP-trees and their extension P-trees.

1.6.1 Lexicographic preference trees (LP-trees)

Lexicographic preference trees (LP-trees for short) [Liu and Truszczynski, 2013,Lang et al., 2012] offer a qualitative way to express total orders between configurations. Since, explicitly specifying a strict preference order on Ω becomes impossible for combinatorial domains with more than 8 configurations. A P-tree over Ω is a binary tree where each node is labelled by a variable. LP-trees have a depth equal to N with an implicit level $N+1$ is implicit (i.e. not always represented in the graphical structure and in dotted lines in Figure 1.7) and corresponds to non-decision nodes (the leaves of the tree) representing configurations.

Definition 1.9. *A lexicographic preference tree (LP-tree) $\mathcal{T}$ over a set of binary variables $\mathcal{V}$ is a labelled binary tree. Each node of $\mathcal{T}$ corresponds to a variable and is associated to a total order (conditioned or not) between the values of the variable. Each variable appears exactly once of each path from the root to a leaf.*

Therefore, a LP-tree is composed of a tree and a collection of conditional preference tables. Each node has two incident arcs such that the left branch indicates that the preference table at the node is satisfied and the right one indicates its violation, see Figure 1.7.

Example 1.6. *Let us consider a user preferences over three binary variables $\mathcal{V} = \{A, B, C\}$. Figure 1.7 illustrates the LP-tree corresponding to the user preferences. It can be checked that the importance between B and C depends on the instantiation of variable A. More precisely, when A is instantiated to a, satisfying the preferences of B is more important than those of C. Besides, in the left subtree its clear that the preference between the values of C is conditionally dependent of the values of B. For instance, when B is instantiated to b, then $c \succ \neg c$.*

Intuitively, the variable in the root of the tree has the highest importance. Then, configurations with the preferred value are preferred to those with the less preferred one. Iteratively, each node refines the ordering between these configurations such that left sub-trees correspond to the preferred configurations and right sub-trees to the non-preferred ones. (implicit) Leaf nodes correspond to a total order between the configurations. Note that the roots of each subtree do not need to be same. Indeed, the relative importance of variables depends on the values of variables on the path

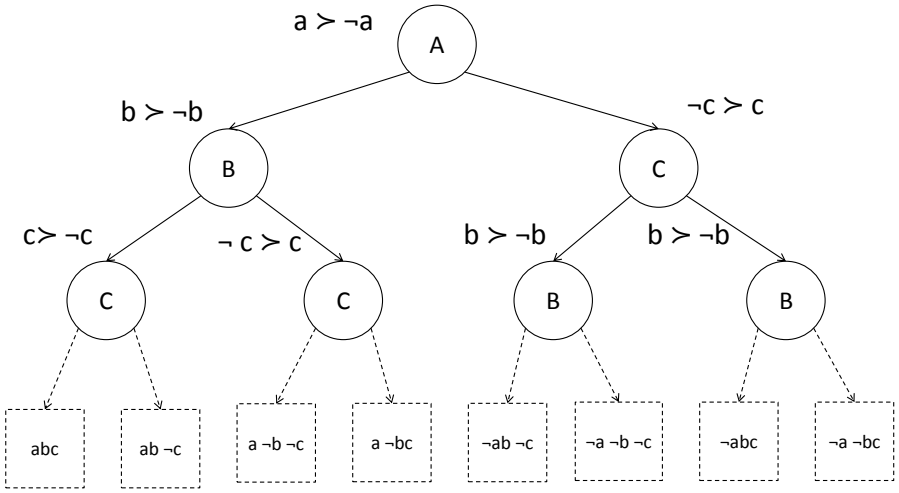


Figure 1.7: An example of a LP-tree

to the root. In other words, the preference of one variable or one value to another depends on the ancestor nodes.

To find the rank of a configuration, one should traverse the tree from the root to a leaf representing its order. More precisely, at each node A_i if the configuration satisfies the preference of the variable we follow down to the left subtree (left links always correspond to the preferred value). Otherwise, we follow down to the right subtree. The more to the left is situated the configuration the more preferred it is. For instance in Figure 1.7, the total order induced is:

$$abc \succ ab\neg c \succ a\neg b\neg c \succ a\neg bc \succ \neg a\neg cb \succ \neg a\neg c\neg b \succ \neg acb \succ \neg ac\neg b$$

Despite the difficulty of learning the model which is conjectured to be NP-complete in general [Liu and Truszczynski, 2015], LP-trees are quite simple to reason with. Since LP-trees induce total orderings, their consistency is trivial. Furthermore, optimization queries are defined by following each time the left subtree of the model until reaching the optimal configuration. Regarding dominance, the query can be executed in polynomial time.

Sometimes, LP-trees can be represented in more compact format. In fact, when having two identical subtrees, we could collapse them to a single subtree with the same importance preference. The conditional preference table of the collapsed subtree is retained into the remaining one.

Example 1.7. *The LP-tree illustrated by Figure 1.7 can be presented in a more concise format as in Figure 1.8. Subtrees in the second level have their nodes collapsed. The left subtree (with the root B) contains two preference statements conditioned on B, thus the retained node in Figure 1.8 is associated to the two conditional preference statements. The right subtree of Figure 1.7 (rooted by C) contains nodes that are not only identical but also have the same preference information*

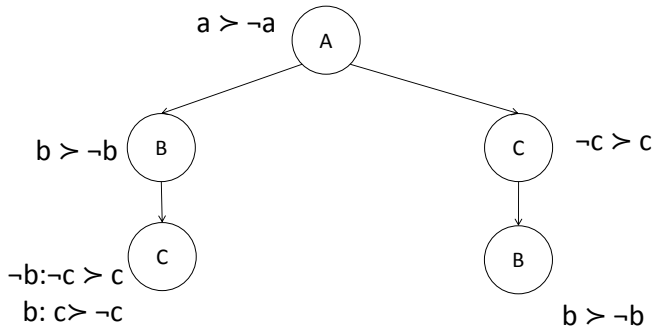


Figure 1.8: A collapsed representation of the LP-tree of Figure 1.8

(i.e. the preference at that level are not conditional). Thus, after the collapsing we can see that the size of the conditional preference table does not change as presented by Figure 1.8.

Such model do not allow for indifference between values of the same variable. Besides, LP-tree is able to capture only a small class of preference ordering. Precisely, an LP-tree over N variables is able to represent $\prod_{k=0}^{N-1}(N-k)^{2^k} \times 2^{2^k}$ compared to $2^N!$ possible preference relation [Liu and Truszczynski, 2015].

It is important to note that LP-trees use the same preference statements as CP-nets. However, it is clear that LP-trees may express preference statements that CP-nets cannot represent since they allow importance relations to be conditioned on the values of the parents (remember that the leftest configurations are the most preferred ones). Besides, although general LP-trees cannot be represented by TCP-nets, these latter succeed to catch the unconditional importance relations of LP-trees.

In general representing preferences by LP-trees is impractical since the size of the representation in the worst case (i.e. there may not exist any collapsible subtrees to obtain a more concise structure) is of the same order as an explicit preference ordering. Yet, some structures of LP-trees containing identical sub-trees may be represented more compactly as presented in the next section.

Finally, LP-trees may be aggregated to represent a social choice scheme, for instance issue-by-issue voting [Fargier et al., 2012] or sequential majority voting rule [Lang et al., 2012]. Basics on multiple agents preference are presented in Chapter 5.

1.6.2 A more general representation: Preference trees

Preference trees [Liu and Truszczynski, 2014,Liu, 2016] generalize LP-trees and are more expressive. LP-trees allow to construct total preorder between configurations in terms of the desirable properties. In fact, nodes in a P-tree correspond to propositional formulas such that each formula

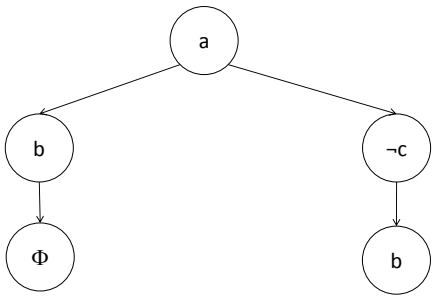


Figure 1.9: The P-tree corresponding to the LP-tree of Figure 1.7

defines a total preorder where configurations satisfying it are more preferred than those violating it. Note that configurations within these groups are considered as equally preferred.

Definition 1.10. *A preference tree (P-tree for short) over $\mathcal{V}$ is binary tree where all nodes other than leaves labelled with propositional formulas over $\mathcal{V}$. Each P-tree defines a total preorder on the set of its leaves as the order of their enumeration from left to right.*

Note that some leaves may be empty. This can be explained by the fact that the conjunction of the formulas corresponding to the same path is unsatisfiable. These leaves and the nodes whose descendants are empty leaves can be pruned from the P-tree to obtain a more compact representation. Similarly to LP-trees, P-trees may exhibit some special structure that allow some subtrees to be collapsed. In many cases, the result is much smaller than the original one.

Each LP-tree is representable by a P-tree where each node corresponds to a literal. Precisely, a node will be labelled a if a is preferred to $\neg a$ and and $\neg a$ otherwise. Figure 1.9 exemplifies the transformation of the LP-tree of Figure 1.7 to a simplified P-tree, having $\phi = (\neg b \wedge \neg c) \vee (b \wedge c)$. Dominance testing can be solved in time linear to the height of the P-tree. In fact, a preference relation between two configurations can be determined at the first non-leaf node where the two configurations evaluate the logic formula differently. If no such formula exist then the two configuration are equally preferred.

1.7 Conclusion

In this chapter, we outlined the general scheme underlying graphical preference models. We have shown why it was important to represent preferences compactly. Besides, we outlined the most known compact qualitative representation models. The tools discussed in this chapter search for a compromise between the expressivity of the model and the ease of elicitation. However, the balance between these two conflicting aspects differs from one model to another.

Roughly, we can say that the advantage of qualitative models is their ease of elicitation which does not need much effort from the user. However, its important to know that qualitative models generally lead to partial orderings (apart from P-trees and LP-trees), which is sometimes problematic specifically when applications must be precise. Another obstacle to using these models is dominance queries which are generally complex in contrast with quantitative models that will be detailed in next chapter.

CHAPTER 2

Graphical Preferential Quantitative models

Contents

2.1 Introduction

In classical decision theory, utility functions are generally used to represent a decision maker preferences since they provide a quantitative valuation of the desirable solutions. Despite the fact that the acquisition of such numerical values is often time consuming and requires too much effort from the experts as well as from the decision maker, models based on numerical values are of use in many situations. Specifically, when applications need accurate information such as medical applications.

It is often convenient to have preferences expressed in numerical terms, since it enables an easy comparison of possible choices contrarily to the qualitative models presented in Chapter 1. It is therefore interesting to consider quantitative graphical models for preferences. These latter are generally based on utility functions corresponding to a mapping from the Cartesian product of variables domains to numerical values, namely $u : \Omega \mapsto \mathbb{R}$. These utilities correspond to a total ordering s.t., for two configurations ω and ω', $\omega \succ \omega'$ (respectively $\omega \sim \omega'$) if and only if

$u(\omega) > u(\omega')$ (respectively $u(\omega) = u(\omega')$). In this Chapter, we review the most known graphical models based on several types of utility independence such as, GAI-networks [Gonzales and Perny, 2005], UCP-nets [Boutilier et al., 2001] and Marginal Utility networks [Brafman and Engel, 2010]. Besides, we present OCF-networks [Eichhorn et al., 2016] advocated to represent preferences by the use of ordinal conditional functions (OCF) as valuation functions. OCF frameworks have been basically used for modeling belief revision. In a Bayesian style like, OCF-networks independence satisfies the markov properties. Such networks support efficient optimization and dominance algorithms.

This chapter is organized as follows. Sections 2.2 is dedicated for Generalized Additive Independence Networks (GAI-nets). Section 2.3 presents a numerical extension of CP-nets entitled UCP-nets. Then, Section 2.4 present another utility-based graphical models called Marginal utility networks relying on a new independence relation. Section 2.5, explains the use of OCF networks for representing preferences. We end the chapter by a brief conclusion.

2.2 Generalized Additive Independence Networks (GAI-nets)

GAI-networks [Gonzales and Perny, 2005] are one of the first graphical quantitative preference models. They rely on generalized additive independence decomposition (GAI decomposition, for short) [Fishburn, 1970]. This independence allows to represent the preferences by a utility function separable into a sum of local functions. Each local function pertains to a subset of variables and represents a total ordering between their possible instantiations. Moreover, there may be some interactions between these local utilities since the subsets of variables pertaining to them can be non disjoint. Thus, these GAI-decompositions can express some general interactions between attributes while preserving some decomposability of the model. For $X \subset \mathcal{V}$, let $\omega[X]$ denote the restriction of ω to variables in X.

Definition 2.1 (GAI decomposition)**.** *Let $C_1, \ldots, C_k$ be subsets of $\mathcal{V}$ s.t. $\mathcal{V} = \bigcup_{j=1}^{k} C_j$. A utility function $u(\cdot)$ representing $\succeq$ over Ω is GAI-decomposable w.r.t. $C_1, \ldots, C_k$ iff $\forall\, j \in [1, k]$, there exists a function $u_j : D_{C_j} \mapsto \mathbb{R}$ s.t., $\forall\, \omega \in \Omega$:*

$$u(\omega) = \sum_{j=1}^{k} u_j(\omega[C_j]) \tag{2.1}$$

These GAI decompositions can be represented by graphical structures called GAI networks. They are undirected graphs where each clique consists of a subset of variables. Between two cliques having some variables in common there exists a path linking them. Each edge in the network is labeled by the intersection between the nodes.

Definition 2.2 (GAI-nets)**.** *A GAI network is an undirected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ where $\mathcal{C}$ denotes the set of cliques and $\mathcal{E}$ denotes the set of edges. $\mathcal{G}$ has two components:*

	$u_1(.)$
ab	0.7
a ¬b	1.2
¬ab	1.8
¬a ¬b	0.5

	$u_2(.)$		$u_2(.)$
ade	0	¬ade	2
ad ¬e	1	¬ad¬e	1.2
a¬de	0.5	¬a¬de	1.8
a¬d¬e	0	¬a¬d¬e	0.4

	$u_3(.)$
ac	1.5
a ¬c	1
¬ac	1.2
¬a ¬c	0.2

Figure 2.1: An example of GAI network

- *Graphical component: Each clique $C_j \in \mathcal{C}$, is a set of variables such that $C_j \subseteq \mathcal{V}$ and $\bigcup_{i=1}^{k} C_i = \mathcal{V}$; For each edge $(C_i, C_j) \in \mathcal{E}$, $C_i \cap C_j \neq \emptyset$. Each edge is labeled by $C_i \cap C_j$;*
- *Numerical component: To each clique C_j we associate a local utility function u_j that defines a complete preorder between the configurations in D_{C_j}.*

From a GAI decomposition, one can directly spot the cliques that should be created such that the common variables between those cliques correspond to the edges of the network. More precisely, each clique represents the variables of a sub-utility and the separators captures the dependencies between these sets. Note that when the separators are instantiated, one can get independent parts of the network that make the elicitation process much easier. This graphical structure of GAI-nets is similar to the notion of junction tree used for Bayesian networks [Jensen et al., 1990, Pearl, 1988]. Indeed, even for a GAI-net with a more general graph structure, we can always construct a tree-structured network based on the triangulation of the Markov network corresponding to it [Gonzales and Perny, 2004] The complexity of this transformation is NP-complete [Arnborg et al., 1987]).

Example 2.1. *Let $\mathcal{V} = \{A, B, C, D, E\}$ be a set of variables and $\{A, B\}$, $\{ADE\}$ and $\{AC\}$ be a utility decomposition. This decomposition can be represented by the GAI network of Figure 2.1 such that: $u(ABCDE) = u_1(AB) + u_2(ADE) + u_3(AC)$.*

Optimization queries look for the configurations having the maximal global utility value. A naive method for finding optimal configurations is to compute the utility of each configuration then choose the configurations with the highest utility (we can get more than one optimal configuration). This method is not realistic since it is impractical to compute the utility values of 2^N configurations (N is the number of variables). A standard algorithm for finding the optimal configurations has been proposed for tree structured GAI networks but, as mentioned above, this is not restrictive. It is essentially similar to the one used for Most Probable Explanation (MPE) in Bayesian networks [Dawid, 1992]. More precisely, optimization for GAI-nets corresponds to an adaptation of the belief propagation algorithm used in Bayesian networks. Searching for the best

	$u^*_1(.)$
a	1.2
¬a	1.8

	$u^*_3(.)$
a	1.5
¬a	1.2

	$u_2(.)$		$u_2(.)$
ade	2.7	¬ade	5
ad ¬e	3.7	¬ad¬e	6.2
a¬de	3.2	¬a¬de	7.8
a¬d¬e	2.7	¬a¬d¬e	3.4

Figure 2.2: Utility tables in the collection phase

configuration corresponds to solving the problem $\max_{A_1} \max_{A_2} \ldots \max_{A_N} u(A_1, A_2, \ldots, A_N)$. The procedure starts with external nodes by computing each time the maximal utility value such that only the variables appearing in the external nodes are maximized. Result is then transferred to the neighbouring nodes until all the cliques are visited. At the end of this collection step, we have the utility value of the best configuration but not its variables instantiations. Comes, therefore, the step of instantiation. It consists of propagating in the opposite sense the instantiations found, until all variables are assigned. This can better be described by the following example:

Example 2.2. *Let us reconsider the GAI-net represented by Figure 2.1. To find the best configuration, one should first start with maximizing the external cliques which are* $\{A, B\}$ *and* $\{AC\}$ *then the internal clique* $\{A, D, E\}$ *such that:*

- *For the clique* $\{A, B\}$*, we compute* $u_1^*(a) = \max_{b \in D_B} u_1(a, b)$ *for all* $a \in D_A$*.*
- *For the clique* $\{A, C\}$*, we compute* $u_3^*(a) = \max_{c \in D_C} u_1(a, c)$ *for all* $a \in D_A$*.*
- *For* $\{A, D, E\}$*, we should substitute* $u_2(a, b, e)$ *by* $u_2(abe) + u_1^*(a) + u_3^*(a)$ *for each instantiation* $a,\ b,\ e$ *of* A, B, E*.*
- *Then, we compute* $\max_{a,d,e} u_2(a, d, e)$ *corresponding to the maximal utility of the network.*

Utilities representing these computations are presented in Figure 2.2. We can check in u_2 *that the instantiation having the best utility is* $\neg a \neg d e$*. After the collection phase, we proceed to the instantiation phase where we assign to each variable each best instantiation with regards to the optimal utility. Starting from the clique* $\{ADE\}$*, the best utility* 7.8 *corresponds to the partial configuration* $\neg a \neg d e$*. Therefore,* $u_1^*(\neg a) = 1.8$ *corresponds to* $u_1(\neg a b) = 1.8$ *therefore* B *is instantiated to* b *and* $u_3^*(\neg a) = 1.2$ *corresponds to* $u_3(\neg a c) = 1.2$ *therefore* C *is instantiated to* c*. Thus, we can deduce that the best configuration is* $\neg a b c \neg d e$*.*

The complexity of optimization query is equal to the sum of sizes of the cliques in the network such that the size of a clique corresponds to the product of the variables domain sizes that are in the clique.

To compare two configurations ω and ω' by a GAI-net, we compute their corresponding utilities and compare them. We can say that $\mathcal{G} \models \omega \succ \omega'$ (resp. $\mathcal{G} \models \omega \sim \omega'$) if and only if $u_{\mathcal{G}}(\omega) > u_{\mathcal{G}}(\omega')$ (resp. $u_{\mathcal{G}}(\omega) = u_{\mathcal{G}}(\omega')$). Thus, the dominance test for GAI is linear in the number of cliques which is considered as an advantage compared to the other models.

Example 2.3. *Let $\omega_1 = abcde$ and $\omega_2 = a\neg bc\neg d\neg e$. From the GAI-network $\mathcal{G}$ of Fig. 2.1, we can compute the utilities of the configurations: $u_{\mathcal{G}}(\omega_1) = u_1(ab)+u_3(ac)+u_2(ade)= 0.7+1.5+0 = 2.2$, $u_{\mathcal{G}}(\omega_2) = u_1(a\neg b) + u_3(ac) + u_2(a\neg d\neg e)= 1.2 + 1.5 + 0 = 2.7$. Thus $u_{\mathcal{G}}(\omega_2) > u_{\mathcal{G}}(\omega_1)$, and $\omega_2 \succ_{GAI} \omega_1$.*

Bacchus and Grove [Bacchus and Grove, 1995] were first to develop a quantitative graphical model based on Conditional Additive Independence (CAI) structure. In particular, they establish that the CAI condition has a perfect map, a graph with attribute nodes $\mathcal{V}$ such that node separation reflects exactly the set of CAI conditions on V. More specifically, for any two sets of nodes $X, Y \subseteq \mathcal{V}$, $CAI(X, Y, X\bar{Y})$ holds if and only if there is no direct edge between a node in X and a node in Y. Then, they go on to show that the utility function has a GAI-decomposition over the set of maximal cliques of the CAI map. Such that a clique is a set of nodes in which each pair is connected by an edge, and a maximal clique is a clique that is not contained within a larger one. This result provide an alternative representation (perfect-map) to GAI net, with the significant advantage that the GAI condition can be detected incrementally based on a set of CAI conditions. However, it was noted in [Bacchus and Grove, 1995], that sometimes GAI conditions do not correspond to a collection of CAI conditions. Therefore, we can deduce that every CAI-net can be represented under the form of a GAI network. However, it is not always possible to construct a CAI-net from a GAI-decomposition. Note also, that no process to detect or verify the GAI condition directly has been proposed in the literature. In contrast, CAI conditions are much simpler to detect where it is sufficient to identify pairs of disjoint subsets that are independent, given that the rest of the attributes are fixed. Therefore, it is much more conceivable to identify GAI structures that rise from the graphical model of CAI. Those procedures verifying CAI formally are in [Keeney and Raiffa, 1993].

GAI-nets rely on a weak form of symmetric independence which make the model flexible enough to be applied to many situations. GAI-nets are not limited to the expression of *Ceteris Paribus* preferences as CP-nets, TCP-nets, or their numerical counterpart, UCP-nets. Still there are cases of numerical preferences that are not representable by a GAI-net [Engel and Wellman, 2008]. With regard to elicitation, there is no method to construct the GAI decompositions. In practice it is always assumed that an expert should provide the GAI decomposition and only the utilities are elicited. One may take advantage of the GAI structure for designing an elicitation method based on "local" utility queries rather than "global" queries over full configurations [Braziunas and Boutilier, 2005]. In fact, [Braziunas and Boutilier, 2005] proposed a new elicitation approach that exploits the graphical structure of GAI-nets to restrict attention to almost exclusively queries over local outcomes. They extended the key advantage of additive utility models, where each clique is totally independent from the others, to the generalized additive utility case.

In a larger scope, these generalized additive utilities where used to assess GAI decomposable utilities in the context of decision under risk [Gonzales and Perny, 2004]. It consists mainly on inquiring a sequence of questions to attest the basic features of the decision maker behavior under risk. In fact, in GAI networks it is possible to compare configurations where to each one is attached a lottery corresponding to a tuple of pairs utility and probability. Moreover these GAI utilities where used to endow CP-nets with utility functions both in a certain framework (as we shall see in the next section) and an uncertain one.

Another graphical model for numerical preferences, called CUI-nets, based on conditional utility independence (CUI) was proposed in [Engel and Wellman, 2008]. It is motivated by the use of a weaker *asymmetric* independence relation. This independence is not additive and may represent preferences that cannot be factored using strong additive independence conditions [Engel and Wellman, 2008]. However, this kind of independence does not lead to decompositions that are as easy to handle as those given under additive independence.

2.3 Utility CP-nets (UCP-nets)

Utility CP-nets (UCP-nets), introduced in [Boutilier et al., 2001], are an extension of CP-nets that replaces the ordinal preference relations of CP-nets by utility factors. In fact, UCP-nets combine the aspects of two preference models, namely, CP-nets and GAI-nets. Like GAI-nets, utility is obtained from the sum of functions associated to groups of variables, defined here by a variable and its parents. Similarly to CP-nets, arcs in UCP-nets reflect the *Ceteris Paribus* independence.

Definition 2.3 (UCP-nets)**.** *A UCP-net is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the graphical component is the same as for CP-nets and the conditional preference tables are replaced by a set of numerical factors $f_i(a_i, p(A_i))$, for all $a_i \in D_{A_i}$ and parents instantiation $p(A_i)$, such that the global utility of a configuration is defined by:*

$$u_{\mathcal{G}}(a_1, ..., a_N) = \sum_{i=1}^{N} f_i(a_i, p(A_i)) \tag{2.2}$$

Example 2.4. *The UCP-net $\mathcal{G}$ presented in Figure 2.3 has 3 variables $\mathcal{V} = \{A, B, C\}$. For instance, we can check that the configuration $a\neg b\neg c$ is preferred to abc since $u_{\mathcal{G}}(abc) = 5 + 2 + 2 = 9 < u_{\mathcal{G}}(a\neg b\neg c) = 5 + 10 + 6 = 21$.*

The UCP-net formalism has a number of computational advantages. In particular, dominance queries can be answered trivially since they amount to computing the global utilities and compare them, as in the above example. This can be done in linear time in the number of variables (this contrasts with CP-nets where dominance testing is computationally difficult).

Optimization queries can also be answered directly, taking linear time in the network size, where each node is instantiated to its maximal value given the instantiation of it parents. This procedure, inherited from CP-nets, exploits the considerable power of *Ceteris Paribus* semantics.

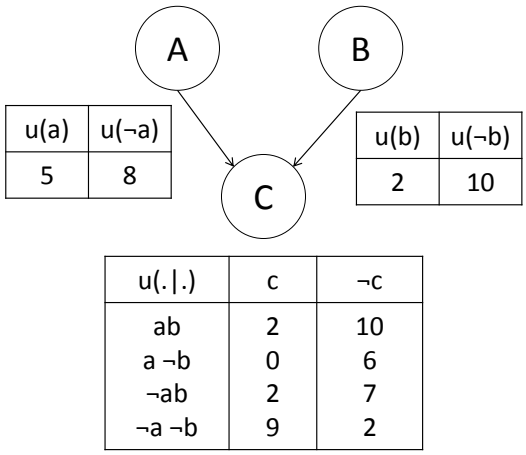


Figure 2.3: An example of a UCP-net

Example 2.5. *Let us consider the UCP-net of Figure 2.3, to find the optimal configuration we should traverse the network from roots to leaves. Thus, we start from A and B. it is easy to check that $A = \neg a$ and $B =_{\neg} a$ are the preferred values. Then, having $\neg a \neg b$ as the parent instatiation, we find that c is the best instantiation. Therefore, the optimal configuration is $\neg a \neg bc$ with a utility value equal to $u_{\mathcal{G}}(\neg a \neg bc) = 8 + 10 + 9 = 27$.*

In this context, CP-nets are endowed with quantitative utility information, and then the expressive power is enhanced and dominance queries become computationally efficient. Moreover, when introducing directionality and the *Ceteris Paribus* semantics to GAI relations, we allow utility functions to be expressed more naturally and optimization queries to be answered more easily.

This model is intuitive to assess since, as CP-nets, it captures preference statements that are naturally expressed by the user. However, in order to remain consistent with CP-nets, utilities should be subject to constraints expressing the priority of father nodes over child nodes. More precisely, let A be a variable with parents $\mathcal{P}(A)$ and children $\mathcal{Y}(A) = \{Y_1, \ldots, Y_n\}$ and let $\mathcal{Z}_i$ be the subset of parents of Y_i excluding A and any of its parents in $\mathcal{P}(A)$. Let $\mathcal{Z} = \bigcup \mathcal{Z}_i$ and P_i be the subset of variables in $\mathcal{P}(A)$ that are parents of Y_i and where p_i is an instantiation of P_i. The fact that the node corresponding to variable A dominates its children given any instantiation u of $\mathcal{P}(A)$ is expressed by the requirement $\forall\ a_1, a_2 \in D_A$ such that $f_A(a_1, u) \geq f_A(a_2, u)$, we should have $\forall\ z$ an instantiation of $\mathcal{Z}$ and $\forall y_i$ an instantiation of $\mathcal{Y}(A)$, $f_A(a_1, u) - f_A(a_2, u) \geq \sum_i f_{Y_i}(y_i, (a_2, p_i, z_i)) - f_{Y_i}(y_i, (a_1, p_i, z_i))$. This expresses that for any variable A, given an instantiation of its parents, the utility gain in choosing a_1 rather than a_2 in this context, should be more important than the maximum value of the sum of the possible utility loss for its children over all possible instantiations of the other related variables. This means that not every GAI decomposition can be represented by a UCP-net. Verifying if a quantified network is a UCP-net needs a case by a case testing of the constraint expressing the priority for each extended family i.e. the variable, its parents, its children's parents. This needs a number of tests exponential in the size

of the extended families. Some stronger sufficient conditions were proposed in [Boutilier et al., 2001].

Beside the difficulty encountered for learning utilities, added constraints should be taken into account in order to remain consistent with the *Ceteris Paribus* principle.

2.4 Marginal Utility Networks

With the aim to define preference networks that resemble Bayesian networks, [Brafman and Engel, 2009, Brafman and Engel, 2010] introduced a notion of conditional independence (denoted CDI_r) using an arbitrarily fixed reference instantiation ω^r. Indeed utility functions differ from probability distributions in the fact there is no obvious analogue of marginalization for utility; to cope with this difficulty, the authors propose to use a reference instantiation for fixing the values of the independent variables. Variables A_i and A_j are CDI_r if any difference in values among instantiations to A_i does not depend on the current instantiation of A_j, for any possible instantiation of the rest of the variables.

Definition 2.4 (Reference configuration and the reference utility)**.** *Let $\omega^r = a_1^r, \ldots, a_N^r \in \Omega$ be a predetermined configuration and, X and Y be subsets of $\mathcal{V}$. The reference utility function u_r is defined by $u_r(x) = u(x\bar{x^r})$, s.t. $\bar{X} = \mathcal{V} \backslash X$ is fixed on the values of the reference configuration ω^r. Its conditional form is defined by $u_r(X|Y) = u_r(XY) - u_r(Y)$.*

Definition 2.5 (Difference utility independence)**.** *Let Z and W be two subsets of $\mathcal{V}$, s.t. $Z \cap W = \emptyset$. Z and W are CDI_r given $X \subseteq \mathcal{V} \setminus (Z \cup W)$, denoted by $CDI_r(Z, W|X)$, if for all assignments x, z', z'', w', w'' we have: $u_r(z'w') - u_r(z''w') = u_r(z'w'') - u_r(z''w'')$.*

Then, the utility function satisfies additive analogues of the Bayes and chain rules of Bayesian networks. Indeed, when exponentiating both sides of the rules they regain the standard multiplicative Bayes and chain rules. These additive definitions are recalled as follows:

Definition 2.6 (Additive analogue of Bayes rule)**.** *Let X and Y be two subsets of $\mathcal{V}$, s.t. $X \cap Z = \emptyset$. For all assignments x, y:*

$$u_r(x|y) = u_r(y|x) + u_r(x) - u_r(y)$$

Definition 2.7 (The additive chain rule)**.**

$$u(A_1, \ldots, A_N) = u_r(A_1) + \sum_{i=2}^{N} u_r(A_i|A_1, \ldots, A_{i-1})$$

This leads to a preference representation by directed graphs. $Dn(A_i)$ denotes its descendants and $Co(A_i) = V \setminus (Dn(A_i) \cup Pa(A_i) \cup A_i)$ denotes the set of non-descendants.

Definition 2.8 (Marginal utility network)**.** *A marginal utility network is a directed graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *where* $\mathcal{V}$ *is the set of nodes and* $\mathcal{E}$ *is the set of edges.* $\mathcal{G}$ *has two components:*

- *Graphical component: A node for each variable and edges correspond to conditional (in)dependencies between variables such that, given a fixed configuration* $\omega^r \in \Omega$*, for any* $A_i \in \mathcal{V}$*,* $CDI_r(A_i, Co(A_i) \,|\mathcal{P}(A_i))$*.*
- *Numerical component: Each node* A_i *is associated to a conditional utility table (CUT) corresponding to the function* $u_r(a_i|p_i)$ *such that* $p(A_i)$ *is an instantiation of the parents* $\mathcal{P}(A_i)$ *of* A_i*. containing* $\forall a_i \in D_{A_i}$*,* $\forall p(A_i)$*,* $u_r(a_i \mid p(A_i))$*.*

The utility of a configuration is then computed as $u_{\mathcal{G}}(a_1, ..., a_N) = \sum_{i=1}^{N} u_r(a_i|p(A_i)$ where $p(A_i)$ is an instantiation of $\mathcal{P}(A_i)$. This is now exemplified.

Example 2.6. *Let us consider preferences over four binary variables* A*,* B*,* C *and* D *represented by the marginal utility network of Figure 2.4. Assume that* $\omega_r = abc\neg d$ *is the reference configuration. Then,* $u_r(abc) - u_r(a\neg bc) = u_r(ab\neg c) - u_r(a\neg b\neg c)$*. In fact,* $(4{+}8{+}3{+}4) - (4{+}11{+}3{+}4) = (4 + 8 + 9 + 4) - (4 + 11 + 9 + 4)$*. Thus,* $CDI_r(B, D|A)$*. The utility of a configuration is the summation of all the local utilities. For instance,* $u_{\mathcal{G}}(abcd) = u_r(a) + u_r(b|a) + u_r(c|a) + u_r(d|c) = 4 + 8 + 3 + 2 = 17$ *and* $u_{\mathcal{G}}(a\neg b\neg c\neg d) = 4 + 11 + 9 + 2 = 26$*. Therefore, we have* $abcd \prec_{MU} a\neg b\neg c\neg d$ *since* $u_{\mathcal{G}}(abcd) < u_{\mathcal{G}}(a\neg b\neg c\neg d)$*.*

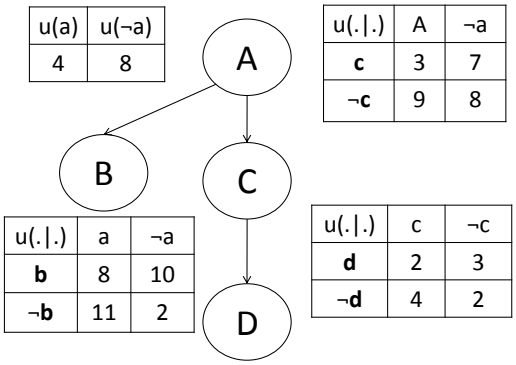


Figure 2.4: An example of a marginal utility net

The type of independence (CDI_r), underlain by marginal utility networks, satisfies the properties of graphoids [Brafman and Engel, 2010]. These properties are as follows:

Definition 2.9 (Graphoid properties)**.** *The axiomatic characterization of Markov independence is as follows:*

- Symmetry *:* $CDI_r(X, Y|Z) \Rightarrow CDI_r(Y, X|Z)$
  This relation asserts that if X *is conditionally independent of* Y *given* Z*, then* X *tells us nothing about the preferences over* Y*, then* Y *tells us nothing about the preferences of* Z*.*

- Decomposition*:* $CDI_r(X, Y|Z)$
  $\Leftrightarrow u_r(X, Y|Z) = u_r(X|Z) + u_r(Y|Z)$
  $\Leftrightarrow u_r(X, Z|Y) = u_r(X|Z) + u_r(Z|Y)$
  $\Leftrightarrow u_r(X, Z, Y) = u_r(X|Z) + u_r(Y, Z)$.

- Union*:* $CDI_r(X, Y \cup W|Z) \Rightarrow CDI_r(X, Y|W \cup Z)$
  This relation asserts that having the preference of W*, cannot change the independence of* Y *from* X*.*

- Contraction*:* $CDI_r(X, Y|Z \cup W)$ & $CDI_r(X, W|Z) \Rightarrow CDI_r(X, Y \cup W|Z)$
  This relation asserts that if Y *is irrelevant to* X *after receiving an independent preference information* $Z \cup W$*, and* X *is independent from* W *in the context* $W \cup Z$*, then* $Y \cup W$ *should be also independent from* X *knowing* Z*. Together, the union and this property state that independent preference information do not affect the preferences of the variables in question.*

- Intersection*:* $CDI_r(X, Y|Z \cup W$ & $CDI_r(X, Z|Y \cup W) \Rightarrow CDI_r(X, Y \cup Z|W)$
  This relation states that if Y *is independent from* X *when* W *is known and if* W *is independent from* X *when* Y *is known, then neither* W*, nor* Y*, nor their combination depend on* X*.*

Knowing all that, it is clear that marginal utility networks model the dependence relations via the concept of d-separation such that each variable is independent from its non descendants in the context of its parents as for Bayesian nets.

Thanks to the strong similarity between Bayesian nets and marginal utility nets, adaptations of algorithms are possible. The authors in [Brafman and Engel, 2010] briefly mention two of them. First, an optimization query for finding the optimal configuration that corresponds to finding the most probable explanation which they called utility maximization. Mainly, the algorithm is defined by the propagation of a message from children to parents until the root is reached. The message contains information about the node and its children. At the end of the propagation we get the maximizing configuration (i.e. the configuration that has the maximal utility value) and its utility value. Let A^- denote a shorthand for $Dn(A)$, defined as the set of descendants of A. Prime signs indicate a specific instantiation, for instance a' is an instantiation of A.

- Each child C_i computes the message $\lambda_{C_i}(a')$ it sends to A, for each $a' \in D_A$:
  $\lambda_{C_i}(a') = \max_{c_i \in D_{C_i}}(u_r(c_i|a') + \max_{c \in D_{C_i^-}} u_r(c|c_i))$

- A computes its own information based on the messages from its children:
  $\max_{a^- \in D_{A^-}} u_r(a^-|a') = \sum_{i=1}^{k} \lambda_{C_i}(a')$ such that k is the number of children of A. If A is a leaf then $\max_{a^- \in D_{A^-}} u_r(a^-|a') = 0$.

The proposed adaptation is illustrated by the following example:

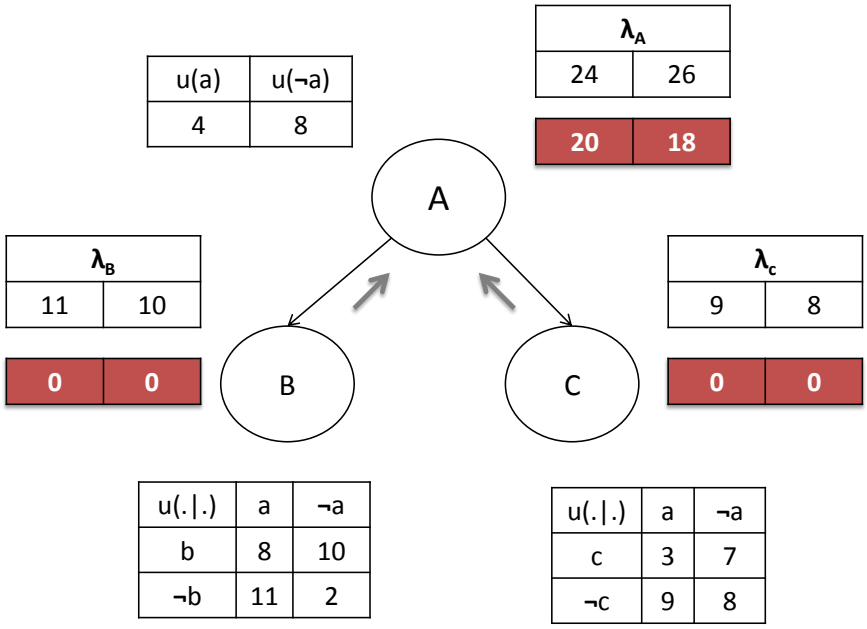


Figure 2.5: An example of utility maximization

Example 2.7. *Let us consider a simple marginal utility network over 3 variables* $\mathcal{V} = \{A, B, C\}$ *represented by Figure 2.5. Nodes* B *and* C *have no children (leaves) i.e., they do not have any information about children, which is explained by the empty local information. Then, each node computes the message sent to their parent* A*. For instance, the node* C *computes the messages* $\lambda_C(a)$ *and* $\lambda_C(\neg a)$ *such that* $\lambda_C(a) = \max(u(c|a), u(\neg c|a)) = 9$ *and* $\lambda_C(\neg a) = \max(u(c|\neg a), u(\neg c|\neg a)) = 8$*. After receiving all information about its children, the node* A *computes its local information such that* $\max_{a^- \in D_{A^-}} u_r(a^-|a) = \lambda_C(a) + \lambda_B(a) = 11 + 9$ *and* $\max_{a^- \in D_{A^-}} u_r(a^-|\neg a) = \lambda_C(\neg a) + \lambda_B(\neg a) = 10 + 8$*. At the end, we have the maximal utility and the maximal configuration that was constructed is the propagation process where* $u(\neg ab\neg c) = 26$*.*

One may consider a variant of optimization query, defined by finding the best configuration when particular combinations between the variables are impossible. This can be compared to constraint belief propagation. Regarding marginal utility networks, for any forbidden combination of values a_1 and a_2, one should create a dummy node $\hat{a}$ as their children such that $u(\hat{a}|a_1a_2) = -\infty$. Then, belief propagation is processed normally. No method to answer dominance queries has been proposed, however the algorithm used in GAI nets seems to be applicable in this case. Elicitation may be inspired from Bayesian nets [Brafman and Engel, 2009].

Besides, note that UCP-nets can be viewed as particular cases of marginal utility nets where constraints should be added in order to make them consistent with *Ceteris Paribus*.

2.5 Ordinal Conditional Functions networks (OCF-nets)

Ordinal Conditional Functions (OCF) [Spohn, 1988] are an uncertainty representation framework very close to possibility theory [Dubois and Prade, 2016] that have been recently used for preference modeling [Eichhorn et al., 2016].. They are functions $\kappa : \Omega \rightarrow \mathbb{N}$ such that $\kappa(\omega) = 0$ for some configuration ω, which means, in the preference setting that ω is not at all rejected, while the greater $\kappa(\omega)$, the less satisfactory it is. Besides it is assumed that $\kappa(U \cup V) = \min(\kappa(U), \kappa(V))$.

Definition 2.10 (Ordinal Conditional Function)**.** *Let Ω be the set of configurations ω. An ordinal conditional function is a mapping $\kappa : \Omega \rightarrow N_0 \cup \{+\infty\}$, where $\kappa(\omega)$ is the degree of implausibility of state ω, and there exists ω with $\kappa(\omega) = 0$ (plausible state).*

Following also the idea of keeping close to Bayesian nets, it has been recently proposed to use *Ordinal Conditional Function networks* (they are like Bayesian nets with infinitesimal probabilities: the value $\kappa(\omega) = n$ of the OCF is like the probability $P(\omega) = 10^{-n}$) for describing preferences [Eichhorn et al., 2016].

In OCF networks, each variable is annotated with a conditional ranking table. They introduce a rank onto the possible values in each context, such that higher the rank, less preferable the value is. Indeed, they have the same structure and carry the same conditional independence namely, each node is independent from its descendant in the context of its parents. This strong resemblance raises the question of a possible transformation between OCF-nets and π-pref nets as we shall see in Chapter 4. Formally,

Definition 2.11. *(Ordinal Conditional Functions networks) An OCF-net κG has two components:*

- *A graphical component, a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges representing the preferential dependencies;*
- *A quantitative component: each variable $A_i \in \mathcal{V}$ is associated to a normalized[1] conditional rank, a non-negative integer $\kappa(A_i|p(A_i))$, where $p(A_i)$ is an instantiation of the parents $\mathcal{P}(A_i)$ of A_i.*

OCF-nets satisfy the local directed independence property of Markov networks. They obey an additive chain rule. Precisely, the OCF relative to a configuration ω, denoted by $\kappa(\omega)$ is the sum of the elementary ranks of the conditional rank tables such that:

$$\kappa(A_1, A_2, \ldots, A_N) = \sum_{i=1}^{N} \kappa(A_i|p(A_i)) \tag{2.3}$$

Example 2.8. *Let us consider the OCF-net of Figure 2.6 over two variables A and B. The cost distribution is as follows: $\kappa(\neg a \neg b) = 0 < \kappa(\neg a b) = 2 < \kappa(ab) = 3 < \kappa(a \neg b) = 4$. We can check that $\neg a \neg b$ is the optimal configuration.*

[1] $\forall p(A_i)$ an instantiation of $\mathcal{P}(A_i)$, $\exists j$ such that $\kappa(a_j|p(A_i)) = 0$

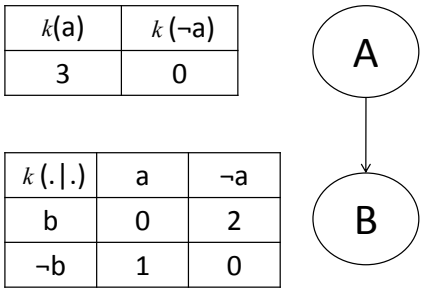


Figure 2.6: An example of a OCF-net

It is clear that the optimization query is linear in the size of the network. Indeed, it consists on a graph traversal where for each node we choose the value that have a cost value equal to 0. This means that the optimal configurations have dominance query is also relatively easy as it consists of computing the cost values of the two configurations then comparing them. From elicitation standpoint we can say that even if the model is claimed to be semi-qualitative, eliciting the cost is not always obvious since they are numerical.

[Eichhorn et al., 2016] proved that numerical OCF-nets can refine CP-net orderings. Precisely, OCF-nets will lead always to total orderings that are compatible with CP-nets. To do so they use a set of particular constraints to be imposed on their integer weights.

2.6 Conclusion

In this chapter we reviewed the best known graphical quantitative models. Each model relies on a type of independence that helps the handling of preferences. All these models offer accurate information about preferences since they lead to total orderings between configurations. However, one major drawback of such numerical models is the elicitation step which is often intractable and presents many flaws.

Therefore, the choice of a representation model for preferences is crucial in any process since some trade-off between the elicitation burden and the simplicity of queries should be made. In particular, one can adopt either a quantitative model (utility-based or OCF-based models), or a qualitative model (i.e. Ceteris Paribus based models). We believe that the main advantage of qualitative models is the fact that they do not need precise numerical values which can considerably reduce the preference elicitation burden. In contrast, we can see that quantitative model offer a good basis for computation where queries are generally easily executed.

The next chapter introduces a new graphical preference model based on possibilistic networks. We believe that this model can take advantage from both qualitative and quantitative models.

CHAPTER 3

Possibilistic Preference Networks

Contents

3.1 Introduction

Chapters 1 and 2 reviewed the best known graphical models for preferences. They show that the choice between qualitative and quantitative models is influenced by the trade-off between the easiness of elicitation and complexity of handling queries. In this chapter we propose a new graphical model, based on possibility theory, that can be positioned in between qualitative and quantitative models. We believe that such a model can take advantage from both classes.

Possibility theory [Dubois and Prade, 1988, Zadeh, 1978] offers a valuable setting for preference representation [Dubois et al., 2006], and enables us to reason with them. A remarkable feature of possibility theory is the existence of several representation formats. Precisely, possibilistic bases (weighted propositional formulas), comparative bases (a set of strict comparative statements) and possibilistic networks [Benferhat et al., 2001a].

Possibilistic logic [Dubois et al., 1994] has been advocated for preference representation. Beside its capability to express knowledge efficiently and reason with it, this logic is as well, very efficient to deal with preferences. It induces a total pre-order thanks to its semantics in terms of possibility distributions. Symbolic possibilistic bases [Hadjali et al., 2008], a variant of possibilistic logic, consists on attaching to each formula a symbolic (non-instantiated) weight. These weights define a partial ordering between the alternative choices.

The main idea in this Chapter is to advocate the interest in preference modeling in the same way as other graphical quantitative preference representations presented in Chapter 2. In this Chapter, we present another reading of possibilistic networks [Ben Amor et al., 2003, Ben Amor and Benferhat, 2005]. In fact, these possibilistic counterpart of Bayesian networks were used to handle knowledge. The proposed symbolic quantitative model uses possibilistic networks to express preferences in a graphical manner.

This Chapter is organized as follows. Section 3.2 provides a brief background on possibilistic networks, while Section 3.3 introduces possibilistic networks with symbolic weights as a way of representing preferences. Section 3.4 defines the different possible orderings we may think of for comparing vectors with symbolic components, and establishes that the product-based and the symmetric Pareto orderings always coincide in the presence of non-zero symbolic weights. Section 3.5 presents a thorough comparison of different possible orderings between symbolic vectors, including the case of additional constraints between the symbolic weights. Then, Section 3.6 presents the optimization and dominance queries, their algorithms and their respective complexities.

3.2 Background on possibility theory

Before describing π-pref nets in detail, we recall basic notions of possibility theory that will be useful in the sequel. Possibility theory relies on the use of a possibility distribution π [Zadeh, 1978], which is a mapping from a universe of discourse Ω to the unit interval $[0, 1]$, or to any bounded totally ordered scale such that $\pi(\omega) = 1$ for some element of Ω ($\exists\, \omega_i \in \Omega$ s.t. $\pi(\omega_i) = 1$). Then, the possibility distribution π is said to be normalized. This possibilistic scale can be the unit interval when possibility values are the result of a clear measurement procedure, or an ordinal scale when values only reflect a total preorder between the different elements of Ω. When used to represent uncertainty about some variable x taking values on Ω, the assignment $\pi(\omega_i) = 0$ means that ω_i is fully impossible, while $\pi(\omega_i) = 1$ means that ω_i is fully possible, i.e. non-surprising

We can describe the uncertainty about the occurrence of an event $F \subseteq \Omega$ via a possibility measure $\Pi(F) = \sup_{\omega_i \in F} \pi(\omega_i)$ measuring plausibility and its dual necessity measure $N(F) = 1 - \Pi(\bar{F}) = 1 - \sup_{\omega_i \notin F} \pi(\omega_i)$ expressing certainty. More precisely, the possibility degree $\Pi(F)$ evaluates to which extent F is *consistent* with the knowledge represented by π while the necessity degree $N(F)$ evaluates to which level F is certainly implied by π. See [Dubois and Prade, 1988] for an introduction to possibility theory.

Conditioning in possibility theory is defined from the Bayesian-like equation

$$\Pi(F \cap G) = \Pi(F|G) \otimes \Pi(G),$$

where $\otimes$ is associative, monotonic in the wide sense and 1 represents the identity element such that $1 \otimes \alpha = \alpha$. In this paper, $\otimes$ stands for the product in a quantitative setting (numerical), or for the minimum in a qualitative setting (ordinal). Namely,

- if $\otimes$ is the product, we get a straightforward counterpart of conditional probability:

$$\Pi(F|G) = \frac{\Pi(F \cap G)}{\Pi(G)} \text{ provided that } \Pi(G) > 0;$$

- if $\otimes$ is the minimum, we get a qualitative version of conditioning, that makes sense on a finite possibility scale:

$$\Pi(F|G) = \begin{cases} \Pi(F \cap G) \text{ if } \Pi(G) > \Pi(F \cap G); \\ 1 \text{ if } \Pi(G) = \Pi(F \cap G) > 0. \end{cases}$$

The two definitions of possibilistic conditioning lead to two variants of possibilistic networks: in the numerical context, we can express product-based networks, while in the qualitative context, we only have min-based networks (also known as qualitative possibilistic networks) [Benferhat et al., 2002a].

A possibilistic network has a definition similar to the one of a Bayesian network.

Definition 3.1 (Possibilistic networks)**.** *[Benferhat et al., 2002a, Ben Amor et al., 2003] A possibilistic network over a set of variables $\mathcal{V}$ is characterized by two components:*

(i) *a* graphical component *which is a Directed Acyclic Graph (DAG) $\mathcal{G}$= $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of nodes representing variables and $\mathcal{E}$ a set of directed edges $A_i \rightarrow A_j$ encoding conditional (in)dependencies between variables.*

(ii) *a* valued component *associating a local normalized conditional possibility distribution $\pi(A_i|\ p(A_i))$ to each variable $A_i \in \mathcal{V}$ in the context of each instantiation $p(A_i)$ of its parents $\mathcal{P}(A_i) = \{A_j : A_j \rightarrow A_i \in \mathcal{E}\}$.*

We assume that $\pi(A_i|\ p(A_i)) > 0$ in order to avoid conditioning on a value of possibility 0. It also comes down to assuming that all configurations are somewhat possible. This assumption will be innocuous in the modeling of preferences, if the scale of possibility contains at least 3 values.

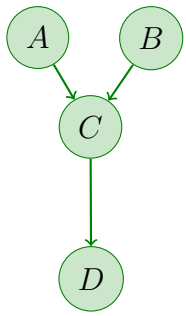
Given a possibilistic network, we can compute a joint possibility distribution using the following chain rule:

$$\pi(A_1, \ldots, A_N) = \otimes_{i=1..N}\ \Pi(A_i \mid \mathcal{P}(A_i)). \tag{3.1}$$

When $\otimes$ is the product, and no configuration is impossible, the conditional tables can be retrieved from the joint possibility distribution obtained by the chain rule, using the same ordering of variables as in the original network. However this is no longer the case if $\otimes$ is the minimum, as some conditional possibility values may be lost when computing the minimum in the chain rule.

Example 3.1. *Let us consider the possibilistic network over 4 binary variables $\mathcal{V} = \{A, B, C, D\}$ of Figure 3.1.*

$\pi(a)$	$\pi(\neg a)$
1	0.2



$\pi(b)$	$\pi(\neg a)$
1	0.8

$\pi(.\|.)$	ab	$a\neg b$	$\neg ab$	$\neg a\neg b$
c	1	1	1	1
$\neg c$	0.7	1	1	0.6

$\pi(.\|.)$	c	$\neg c$
d	1	0.2
$\neg d$	0.3	1

Figure 3.1: Standard possibilistic network of Example 3.1

If we consider the minimum-based conditioning for this networks then the possibility degree of the configuration $\omega = \neg ab\neg cd$ is $\pi(\neg ab\neg cd) = min(\pi(\neg a), \pi(b), \pi(c|\neg ab), \pi(d|\neg c)) = min(0.2, 1, 1, 0.2) = 0.2$.
Now, if we use the product-based conditioning then the possibility degree would be equal to $\pi(\neg ab\neg cd) = (\pi(\neg a) * \pi(b) * \pi(c|\neg ab) * \pi(d|\neg c)) = 0.04$.

3.3 Possibilistic preference networks

Originally, possibilistic networks were meant to model uncertainty and to compute the impact of observations assigning values to some variables so as to predict the values of other variables of the network. In this chapter, we advocate their interest in preference modeling rather than uncertainty management. Thus, let $\pi(\omega)$ define the level of satisfaction of choosing configuration ω. For a set of configurations F, $\Pi(F)$ evaluates to what extent satisfying a constraint modeled by F is satisfactory and $N(F)$ evaluates to what extent this constraint is imperative. As we shall see, beyond their graphical appeal, conditional preference possibilistic networks can be shown to provide a natural encoding of preferences. In the following, we first define the kind of preference

information needed to construct π-pref nets. Then, we present the definition of π-pref nets and their representational power.

3.3.1 Conditional preference statements

In qualitative preference models, users are supposed to express their preferences under the form of pairwise comparison statements between variable instantiations, conditioned by some other instantiated variables. For instance, in the particular case of Boolean variables, we deal with preferences of the form: "I prefer a to $\neg a$" if the preference is unconditional, and for conditional statements, of the form "in the context where c is true, I prefer a to $\neg a$", where c corresponds to the instantiation of several other variables. More generally,

Definition 3.2. A preference statement $(A_i, p(A_i), \succeq)$ *is a preference relation between values* $a_{ik} \in D_{A_i}$ *of a variable* A_i*, conditioned by the instantiation* $p(A_i)$ *of a set* $\mathcal{P}(A_i)$ *of other variables, in the form of a complete preorder* $\succeq$*:* $\forall a_{ik}, a_{im} \in D_{A_i}$*, we have*

i) either $p(A_i) : a_{ik} \succ a_{im}$*, i.e., in the context* $p(A_i)$*,* a_{ik} *is preferred to* a_{im}*,*

ii) or $p(A_i) : a_{ik} \sim a_{im}$*, i.e., in the context* $p(A_i)$*, one is indifferent between* a_{im} *and* a_{ik}*,*

where $\succ$ *is the strict part of* $\succeq$*, and* $\sim$ *is the indifference part of* $\succeq$*. If* $\mathcal{P}(A_i) = \emptyset$*, then the preference statement about* A_i *is unconditional.*

Note that we do not allow incomplete preference local specifications of the form $a_{ik} \succeq a_{im}$. The user must choose between $a_{ik} \succ a_{im}, a_{im} \succ a_{im}$ and $a_{ik} \sim a_{im}$. The running Example 3.2, inspired from [Boutilier et al., 2004a] illustrates such preference statements.

Example 3.2. *Consider a preference specification about an evening dress over 3 decision variables* $V = \{J, P, S\}$ *standing for jacket, pants and shirt respectively, with values in* $D_J = \{Red\ (j_r),\ Black\ (j_b)\}$*,* $D_P = \{White\ (p_w),\ Black\ (p_b)\}$ *and* $D_S = \{Black\ (s_b),\ Red\ (s_r),\ White\ (s_w)\}$*. The conditional preferences are given in Table 3.1. Preference statements* (s_1) *and* (s_2) *are unconditional. Note that the user is indifferent between the values of the color of the shirt if his jacket is black and his pants are white (in the context* $j_b p_w$*) which is not the case if he wears a red jacket and black pants. Indeed, he prefers red shirt to a black one (in the context* $j_r p_b$*).*

3.3.2 Introducing π-pref nets

Representing the preference statements in a graphical way means that each node in the graph represents a decision variable A_i which is associated to a set of local conditional preference statements, conditional to the values of variables that are its parent nodes in the graph. A (conditional) *preference network* can be defined as follows:

(s_1) $j_b \succ j_r$
(s_2) $p_b \succ p_w$
(s_3) $j_b p_b$: $s_b \succ s_r \succ s_w$
(s_4) $j_b p_w$: $s_w \succ s_b \succ s_r$
(s_5) $j_r p_b$: $s_r \succ s_b \succ s_w$
(s_6) $j_r p_w$: $s_b \sim s_r \sim s_w$

Table 3.1: Conditional preference specification of Example 3.2

Definition 3.3. *A preference network is a DAG $(\mathcal{E}, \mathcal{V})$ with nodes $A_i, A_j \in \mathcal{V}$, s.t. each arc from A_j to$A_i \in \mathcal{E}$ expresses that the preference about A_i depends on A_j. Each node A_i is associated with a preference table CPT_i that associates preference statements $(A_i, p(A_i), \succeq)$ between the values of A_i, conditional to each possible instantiation $p(A_i)$ of the parents $\mathcal{P}(A_i)$ of A_i (if any).*

In a possibilistic preference network, for each particular instantiation $p(A_i)$ of $\mathcal{P}(A_i)$, the preference order between the values of A_i stated by the user will be encoded by a local conditional possibility distribution *expressed by symbolic weights*. By a symbolic weight, we mean a symbol representing a *strictly* positive real number in $(0, 1]$ whose value is unspecified. We rely on symbolic weights in the absence of available numerical values.

Definition 3.4 (Conditional Preference Possibilistic network (π-pref net))**.** *A* possibilistic preference network based on operation $\otimes$ ($\otimes$-π-pref net) ΠG *over a set $\mathcal{V} = \{A_1, \ldots, A_N\}$ of decision variables is a preference network where each local preference relation at node A_i is associated with a* symbolic *conditional possibility distribution (π_i-table for short), encoding the ordering between the values of A_i such that:*

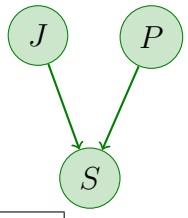
(i) If $p(A_i) : a_i \prec a'_i$ then $\pi(a_i|p(A_i)) = \alpha, \pi(a'_i|p(A_i)) = \beta$ where α and β are symbolic weights, and $0 < \alpha < \beta \leq 1$;

(ii) If $p(A_i) : a_i \sim a'_i$ then $\pi(a_i|p(A_i)) = \pi(a'_i|p(A_i)) = \alpha > 0$ where α is a symbolic weight such that $\alpha \leq 1$;

(iii) For each instantiation $p(A_i)$ of $\mathcal{P}(A_i)$, $\exists\, a_i \in D_{A_i}$ such that $\pi(a_i|p(A_i)) = 1$.

(iv) A symbolic degree of possibility is assigned to each configuration ω using the chain rule (3.1) based on $\otimes$.

Let $\mathcal{C}_0$ be the set storing the constraints between the symbolic possibility weights pertaining to each preference statement $(A_i, p(A_i), \succeq)$, encoding the complete preordering $\succeq$. In addition to these preferences encoded by a π-pref net, additional constraints can be taken into account. Such constraints, forming a set denoted by $\mathcal{C}_1$, may express that some weights pertaining to one



$\pi(j_b)$	1
$\pi(j_r)$	α

$\pi(p_b)$	1
$\pi(p_w)$	β

$\pi(.\|.)$	$j_b p_b$	$j_b p_w$	$j_r p_b$	$j_r p_w$
s_b	1	θ_1	λ_1	1
s_r	δ_1	θ_2	1	1
s_w	δ_2	1	λ_2	1

Figure 3.2: Possibilistic preference network of Example 3.3

preference statement is equal or greater than a weight pertaining to another preference statement. Let $\mathcal{C} = \mathcal{C}_0 \bigcup \mathcal{C}_1$ be the set of all constraints.

Note that in the case one can neither directly find in $\mathcal{C}$, nor infer by transitivity from $\mathcal{C}$, any relation between two weights α and β, we consider them as *incomparable*.

Example 3.3. *Consider the preference specification about an evening dress of Example 3.2. Its corresponding π-pref net and the conditional possibility weights are given by Figure 3.2. The graph is built based on Definition 3.3. In fact since $\mathcal{P}(J) = \mathcal{P}(P) = \emptyset$ the two variables J and P are independent, S depends on J and P since the preference statements associated to S are conditioned by $\mathcal{P}(S) = \{J, P\}$. The constraints between symbolic weights inherent from the preference specification are represented by the set $\mathcal{C}_0$ such that $\mathcal{C}_0 = \{(\delta_1 > \delta_2), (\theta_1 > \theta_2), (\lambda_1 > \lambda_2)\}$.*

A set of conditional preference tables encoded as a π-pref net determines a partial order among configurations. Indeed, each configuration has a satisfaction level encoded by a possibility degree computed from the possibilistic chain rule (3.1). This leads to the following definition of the induced preference ordering on configurations.

Definition 3.5 (Preference ordering)**.** *Let ΠG be a symbolic possibilistic preference network and $\mathcal{C}$ be a set of constraints between the symbolic weights. Let ω_i and ω_j be two configurations in Ω, and $\pi_{\Pi G}(\omega_i)$ (resp. $\pi_{\Pi G}(\omega_j)$) be the symbolic possibility degree of ω_i (resp. ω_j) computed by (3.1). Then, configuration ω_i is preferred to ω_i in the wide sense, denoted by $\omega_i \succeq_{\otimes} \omega_j$, iff $\pi_{\Pi G}(\omega_i) \geq \pi_{\Pi G}(\omega_j)$.*

In the definition, $\pi_{\Pi G}(\omega_i)$ is a combination of symbolic weights using $\otimes$. So, $\pi_{\Pi G}(\omega_i) \geq \pi_{\Pi G}(\omega_j)$ (resp. $\pi_{\Pi G}(\omega_i) > \pi_{\Pi G}(\omega_j)$, $\pi_{\Pi G}(\omega_i) = \pi_{\Pi G}(\omega_j)$) should be understood as follows: this inequality (resp. strict inequality, equality) holds whatever the numerical instantiations of the weights involved in the possibility values, in agreement with constraints in $\mathcal{C}$. This is respectively

	Symbolic vectors			Products
Configurations	J	P	S	
$j_b p_b s_b$	1	1	1	1
$j_b p_b s_r$	1	1	δ_1	δ_1
$j_b p_b s_w$	1	1	δ_2	δ_2
$j_b p_w s_b$	1	β	θ_1	$\beta \times \theta_1$
$j_b p_w s_r$	1	β	θ_2	$\beta \times \theta_2$
$j_b p_w s_w$	1	β	1	β
$j_r p_b s_b$	α	1	λ_1	$\alpha \times \lambda_1$
$j_r p_b s_r$	α	1	1	α
$j_r p_b s_w$	α	1	λ_2	$\alpha \times \lambda_2$
$j_r p_w s_b$	α	β	1	$\alpha \times \beta$
$j_r p_w s_r$	α	β	1	$\alpha \times \beta$
$j_r p_w s_w$	α	β	1	$\alpha \times \beta$

Table 3.2: Symbolic vectors associated to each configuration of Example 3.3

denoted by $\omega_i \succeq_{\otimes} \omega_j$, $\omega_j \succ_{\otimes} \omega_i$ and $\omega_i \sim_{\otimes} \omega_j$. When it is not possible to prove an inequality between $\pi_{\Pi G}(\omega_i)$ and $\pi_{\Pi G}(\omega_j)$, because it is possible to have strict inequalities in both directions by substituting distinct numerical values, we interpret this situation in terms of incomparability as already said, and this is denoted by $\omega_i \pm_{\otimes} \omega_j$.

Since we use symbolic weights, a definite preference between all configurations cannot be established (as long as we do not instantiate all symbolic weights). To each configuration $\omega = a_1 \dots a_N$ can also be associated with a vector $\vec{\omega} = (\alpha_1, \dots, \alpha_N)$, where $\alpha_i = \pi(a_i|p(A_i))$ and $p(A_i) = \omega[\mathcal{P}(A_i)]$, where $\omega[\mathcal{P}(A_i)]$ is the instantiation of the parents of A_i in the configuration ω. For instance, vectors associated to the preference possibilistic network of Example 3.3 are represented by Table 3.2. Thus, comparing configurations amounts to comparing vectors of symbolic weights attached to configurations, and the use of the chain rule is just one way of comparing such vectors, among other ones as discussed in the next section. However, note that symbolic weights attached to a variable depend on the instantiations of its parents.

3.4 Symbolic weights

In Section 2, we have shown how to encode the preference specifications in a possibilistic network format. In this section we will present a number of partial order relations with the purpose to use them to generate a particular ordering over configurations.

Vectors of these weights, $\vec{\omega} = (\alpha_1, \dots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \dots, \beta_N)$ for instance, can be compared using different ordering procedures namely, Product, symmetric Pareto, Minimum or

Leximin orderings. These procedures can be defined for partially ordered symbolic weights. Defined as follows:

Definition 3.6 (Product). $\vec{\omega} \succeq_{prod} \vec{\omega'}$ *iff* $prod(\vec{\omega}) \geq prod(\vec{\omega'})$ *where* $prod(\vec{\omega}) = \prod_{i=1}^{N} \alpha_i$.

Definition 3.7 (Minimum). $\vec{\omega} \succeq_{\min} \vec{\omega'}$ *iff* $\min(\vec{\omega}) \geq \min(\vec{\omega'})$ *where* $\min(\vec{\omega}) = \min_{i=1}^{N} \alpha_i$.

Definition 3.8 (Pareto). $\vec{\omega} \succeq_{Pareto} \vec{\omega'}$ *iff* $\forall\, k,\ \alpha_k \geq \beta_k$.

Definition 3.9 (Symmetric Pareto). $\vec{\omega} \succeq_{SP} \vec{\omega'}$ *iff there exists a permutation* σ *of the components of* $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$, *yielding a vector* $\vec{\omega_\sigma} = (\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(N)})$, *s.t.* $\vec{\omega_\sigma} \succeq_{Pareto} \vec{\omega'}$.

Definition 3.10 (Discrimin). *First, delete all pairs* (α_i, β_i) *from* $\vec{\omega}$ *and* $\vec{\omega'}$ *such that* $\alpha_i = \beta_i$. *Let* D *be the set of indices of components not deleted. Then,* $\vec{\omega} \succ_{discrimin} \vec{\omega'}$ *iff* $\min_{i \in D} \alpha_i > \min_{i \in D} \beta_i$.

Definition 3.11 (Leximin). *Let* $\vec{\omega_\sigma}$ *be the reordered vector* $\vec{\omega}$ *following permutation* σ *of its components.* $\vec{\omega} \succ_{leximin} \vec{\omega'}$ *iff* $\exists\, \sigma, \vec{\omega_\sigma} \succ_{discrimin} \vec{\omega'}$.

In the standard case of a totally ordered scale, the leximin order is defined by first reordering the vectors in an increasing way, and then applying the min order to the sub-parts of the reordered vectors without consideration of identical components. However, here we deal with symbolic possibility degrees between which the ordering can be unknown. In the extreme case, we may just know that $\alpha < 1$ for a weight α. Thus, reordering the vectors increasingly is no longer possible, and leximin must be defined as proposed above.

In this section we study possible refinement relations between the different ordering procedures. Thus, we need to define the concept of refinement of a strict preference relation.

Definition 3.12 (Refinement). *[Dubois et al., 1996]. Let* $\succ$ *and* $\succ$ *be any two strict order relations on* Ω. *Then, we say that* $\succ'$ *refines* $\succ$ *iff:*

$$\forall\, \omega,\ \omega' \in \Omega,\ \omega \succ \omega' \Rightarrow \omega \succ' \omega'.$$

As shown in [Dubois et al., 1996], in the instantiated case, leximin is a refinement of the symmetric ordering of Pareto. As well, the discrimin ordering refines the ordering induced by the minimum as well as the Pareto ordering, and is itself refined by leximin. Leximin refinements of min-based orderings can be very discriminant, as they would solve cases left pending by minimum order. Moreover, product-based orderings refine symmetric Pareto orderings of vectors containing no zero components. As for product-based orderings, they can strongly differ (including preference reversal) from the min-based orderings. These refinement relationships are illustrated by Figure 3.3 (where each arrow $a \rightarrow b$ expresses that a refines b).

The orderings do not behave in the same manner in the numerical case and in the symbolic case, as exemplified in the following.

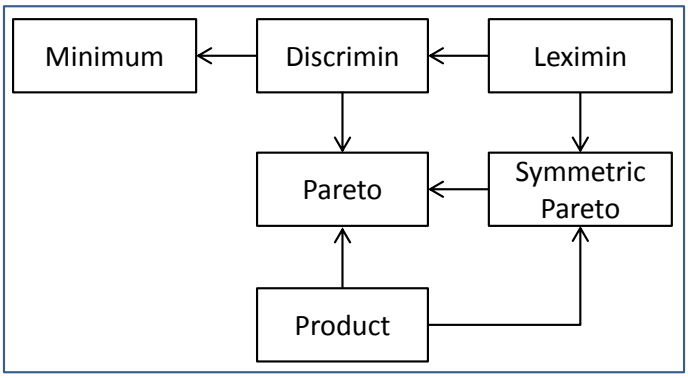


Figure 3.3: Refinements between orderings in numerical setting

Example 3.4. *Consider the vectors* (α, β) *and* (γ, δ) *where* $\alpha < \beta$ *and* $\gamma < \delta$*. If* $\alpha < \gamma$ *then* $(\alpha, \beta) < (\gamma, \delta)$ *for leximin and min, while according to symmetric Pareto these vectors are still incomparable. If* $\alpha = \gamma < \delta < \beta$ *then min considers* (α, β) *and* (γ, δ) *as equal, while we have* $(\alpha, \beta) > (\gamma, \delta)$ *with the product but the Leximin is unable to compare them. However, if* $\alpha < \gamma < \delta < \beta$ *then if we use the min operator we have* $(\alpha, \beta) < (\gamma, \delta)$ *while the product operator still fails to order them.*

In the symbolic framework, it has been suggested in [Ben Amor et al., 2015] that, when there is no constraint between symbolic weights within the vectors, the ordering induced by the product-based chain rule corresponds exactly to the a symmetric Pareto ordering. This result actually holds even in the presence of inequality constraints between symbolic weights.

Proposition 3.1. *Given any set of constraints* $\mathcal{C}$ *of the form* $\alpha_i \geq \beta_j$ *or* $\alpha_i > \beta_j$ *between symbolic weights:*
(i) $\vec{\omega} \succ_{SP} \vec{\omega'}$ *iff* $\vec{\omega} \succ_{prod} \vec{\omega'}$*.*
(ii) $\vec{\omega} \succeq_{SP} \vec{\omega'}$ *iff* $\vec{\omega} \succeq_{prod} \vec{\omega'}$*.*

Proof. The proof is not straightforward. We proceed in several steps. First notice that the implications:
$\vec{\omega} \succ_{SP} \vec{\omega'}$ implies $\vec{\omega} \succ_{prod} \vec{\omega'}$
and $\vec{\omega} \succeq_{SP} \vec{\omega'}$ implies $\vec{\omega} \succeq_{prod} \vec{\omega'}$
are obvious since the product is symmetric and monotonically increasing. For the converse, we must basically show that if $\vec{\omega}$ is SP-incomparable[1] with $\vec{\omega'}$ then they are also incomparable w.r.t. the product ordering. We use several lemmas.

Lemma 3.1. *If Proposition 3.1 holds for a set of constraints* $\mathcal{C}$*, it holds a fortiori for any subset of constraints in* $\mathcal{C}$*.*

[1] $\vec{\omega} \pm_{SP} \vec{\omega'}$

Proof. Indeed taking away constraints from $\mathcal{C}$ yields more freedom to the choice of values for the coefficients, which favours the non comparability of the symbolic product expressions associated to each vector. □

As a consequence of this lemma, the result should be established with the maximal amount of constraints, namely assuming a (non-trivial) complete pre-ordering of the symbolic coefficients appearing in the two vectors.

Lemma 3.2 ([Cayrol et al., 2014]). *Consider two symbolic vectors $\vec{\omega} = (\alpha_1, \dots \alpha_N)$ such that $\mathcal{C}$ enforces $\alpha_1 \leq \dots \leq \alpha_N$ and $\vec{\omega'} = (\beta_1, \dots \beta_N)$. Let σ be a permutation of the components of $\vec{\omega'}$ such that $\beta_{\sigma(1)} \leq \dots \leq \beta_{\sigma(N)}$ and $\vec{\omega'_\sigma}$ the corresponding reordered vector. Then $\vec{\omega} \succ_{SP} \vec{\omega'}$ if and only if $\vec{\omega} \succ_P \vec{\omega'_\sigma}$.*

In the totally ordered setting it gives a constructive way of expressing the SP ordering by applying the Pareto ordering to the increasingly reordered vectors.

Without loss of generality, due to Lemma 3.2, we can assume that vectors are increasingly ordered. Now we can try to prove that if $\vec{\omega}$ and $\vec{\omega'}$ are SP-incomparable then they are so for product. If they are SP-incomparable then there are $i \neq j$ such that $\alpha_i > \beta_i$ and $\alpha_j < \beta_j$. The most constrained case is when there is one constraint of the form $\alpha_i > \beta_i$, while all the other ones are of the form $\alpha_j < \beta_j$. In that case $\prod_{j \neq i} \beta_j > \prod_{j \neq i} \alpha_j$ and denoting by $\vec{\alpha_{-i}}$ the vector $\vec{\alpha}$ deprived of component i, we also have $\vec{\omega_{-i}} \succ_{SP} \vec{\omega'_{-i}}$.

Let us show that this strong prerequisite does not enforce an inequality between $\prod_{j=1}^N \beta_i$ and $\prod_{j=1}^N \alpha_j$. First, if α_i and β_j are very close, then $\prod_{j=1}^N \beta_j > \prod_{j=1}^N \alpha_j$. Now, for the reversed inequality, replace α_j by α_1 for $j < i$, and by α_i for $j > i$, β_j by β_i for $j < i$ and by β_n for $j > i$. The inequality $(\alpha_1)^{i-1} \cdot (\alpha_i)^{N-i+1} > (\beta_i)^i \cdot (\beta_N)^{N-i}$ is more demanding than the inequality $\prod_{j=1}^N \alpha_j > \prod_{j=1}^N \beta_j$. Let us show we can satisfy the former because $\alpha_i > \beta_i$, even if $\alpha_1 < \alpha_i, \beta_i < \beta_N$. To see it, we can write $\alpha_1 = \alpha_i/p$ and $\beta_n = q\beta_i$ with $p, q > 1$. It is easy to see that the inequality now writes $\frac{\alpha_i}{\beta_i} > p^{\frac{i-1}{N}} q^{\frac{N-i}{N}} > 1$. It is clear that we can set $p, q > 1$ and find $\alpha_i > \beta_i \in [0, 1]$ that verifies this inequality. □

The consequence of this result is that the use of product of symbolic values in the approach is just one way of implementing the SP-ordering, whose essence is qualitative. In particular the compensatory effect, usually present in product of numbers (whereby, e.g. 0.5×0.5 is the same as 0.25×1) is absent from the SP-ordering, which creates cases for incomparability.

3.5 On various ways of ordering configurations induced by conditional preference networks

We pursue the comparison of the different orderings defined in the previous section, first in the absence, and then in the presence of additional constraint on symbolic weights.

3.5.1 Comparison of orderings without additional constraints on symbolic weights

In this section we will study the possible relations between the different orderings in the particular case where the constraints known between the symbolic weights are only the ones relative to the expression of conditional preferences, as allowed by Definition 3.4. Thus, a constraint of this kind focuses only on a unique vector component, and we have $\mathcal{C}_1 = \emptyset$.

Under this assumption, Pareto ordering and symmetric Pareto yield the same ordering. Indeed, for two vectors $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \ldots, \beta_N)$ each symbolic weight $\alpha_i \neq 1$ of $\vec{\omega}$ can be only compared to the symbolic weight $\beta_i \neq 1$ of $\vec{\omega'}$. Thus, there is no need to permute components as the result would definitely be non comparable with another component weight since $\mathcal{C}_1 = \emptyset$. This is as well true for leximin and discrimin orderings since they coincide in this case. In fact, with this hypothesis, the difference between leximin and discrimin is that leximin deletes some components with value 1 which cannot affect the result of the final application of the min.

We first compare the different orderings induced by the use of product or minimum, depending on the chain rule applied to the possibilistic network. We will evaluate how well each option uses the information given to rank-order alternative solutions. We keep in mind that the product-based ordering and the symmetric Pareto ordering are the same.

Example 3.5 presents the different orderings induced by min-based and product-based chain rule for Example 3.3.

Example 3.5. *Let us consider the possibilistic preference network of Example 3.3. Using the chain rule, we obtain the symbolic vectors presented in Table 3.2. The product-based induced ordering without additional inequality constraint is represented by Figure 3.4.*

Now, if we use the min-based chain rule, we will not be able to compare many configurations as long as no other constraint is added. In fact, the only strict ordering information we can get at that stage is that $j_b p_b s_b > j_b p_b s_r > j_b p_b s_w$, $j_b p_b s_b > j_b p_w s_w$ *and* $j_b p_b s_b > j_r p_b s_b$. *Otherwise, we only get at best weak inequalities ; for example* $j_b p_w s_w$ *and* $j_r p_w s_b$, *since* $\pi_{\min}(j_b p_w s_b) = \min(\alpha, \beta) \leq \pi_{\min}(j_r p_w s_w) = \beta$. *Figure 3.5 depicts this min-based ordering.*

Symmetric Pareto on symbolic vectors may have a discriminating power greater than the one

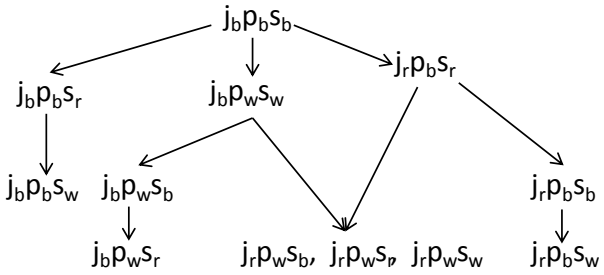


Figure 3.4: Possibilistic product-based order relative to Example 3.3

of the minimum operator, in the sense that $\alpha \cdot \beta < \alpha$, while we only have $\min(\alpha, \beta) \leq \alpha$. Clearly, when dealing with instantiated numerical values both product and minimum lead to total orders that can contradict each other: for instance $0.1 \cdot 0.9 > 0.2 \cdot 0.2$, while with the min we get $\min(0.1, 0.9) < \min(0.2, 0.2)$.

Proposition 3.2. *Let ω and ω' be two configurations, then $\omega \sim_{SP} \omega' \Leftrightarrow \omega \sim_{min} \omega'$.*

Proof. Trivial as it compares the same sets of weights. □

Indeed, if equalities are found between every pair of the same index then the two vectors contain the same symbolic weights. For instance, we have $j_r p_w s_b \sim_{SP} j_r p_w s_r$ (resp . $j_r p_w s_b \sim_{\min} j_r p_w s_r$) where $j_r \vec{p_w} s_b = j_r \vec{p_w} s_r = (\alpha, \beta, 1)$.

Proposition 3.3. *Let ω and ω' be two configurations, then $\omega \pm_{SP} \omega' \Leftrightarrow \omega \pm_{\min} \omega'$.*

Proof. Let us consider two configurations ω and ω' such that $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \ldots, \beta_N)$. If $\omega \pm_{SP} \omega'$ which means $\omega \pm_{Pareto} \omega'$, then we are in one of the following cases: either $\exists\ i$, s.t. $\alpha_i \pm \beta_i$ or $\exists\ i$, s.t. $\alpha_i < \beta_i$ and $\exists\ j$, s.t. $\alpha_j > \beta_j$. Besides, the only case where minimum is able to compare is when $\forall\ i$, $\alpha_i \geq \beta_i$. It is not the case here, then $\omega \pm_{SP} \omega' \Rightarrow \omega \pm_{\min} \omega'$. For the converse, if $\vec{\omega} \pm_{\min} \vec{\omega}$ then obviously $\vec{\omega} \pm_{SP} \vec{\omega}$ since we would not have $\forall\ i$, $\alpha_i \geq \beta_i$.

□

Many cases can be identified on Example 3.3 and the min-based (resp. product-based) ordering presented by Figure 3.5 (resp. Figure 3.4). For instance, we have $j_r p_b s_r \pm_{SP} j_b p_w s_w$ (resp . $j_r p_b s_r \pm_{\min} j_b p_w s_w$). Moreover, using symbolic weights, symmetric Pareto and minimum provide consistent orderings in the sense that:

Proposition 3.4. *If $\omega \succ_{SP} \omega' \Rightarrow \omega \succeq_{min} \omega'$.*

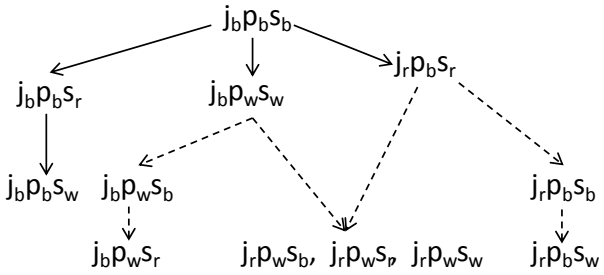


Figure 3.5: Possibilistic minimum-based order relative to Example 3.3

Proof. Let us consider two solutions ω and ω' such that $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \ldots, \beta_N)$. If $\omega \succ_{SP} \omega'$ then $\vec{\omega} \succ_{Pareto} \vec{\omega'}$ more precisely $\forall\ i,\ \alpha_i \geq \beta_i$ and $\exists\ j,\ \alpha_j > \beta_j$ where $i, j \in [1, \ldots, N]$. From Proposition 3.3 we can deduce that it is impossible to have $\vec{\omega} \pm_{min} \vec{\omega'}$ since $\forall i\ \alpha_i \geq \beta_i$, $\omega \geq_{\min} \omega'$ follows. The fact that the inequality is not strict is because α_j or β_j can be drowned among other coefficients for some instantiations of α_i, β_i.

□

Example 3.6. *Let us consider one strict preference pattern is $j_b p_b s_b > j_b p_b s_r > j_b p_b s_w$ induced by the min chain rule on Example 3.3. We can check that it is found on Figure 3.5 that depicts the preference relation induced by the product chain rule. The rest of preferences are preferences in the wide sense ($\succeq$), for example, $j_r p_b s_r \succeq j_r p_b s_b$.*

Proposition 3.5. *Let ω and ω' be two configurations, then $\omega \succ_{\min} \omega' \Rightarrow \omega \succ_{SP} \omega'$.*

Proof. Let us consider two configurations ω and ω' such that $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \ldots, \beta_N)$. If $\omega \succ_{\min} \omega'$ then $\forall\ i$ such that $\alpha_i \neq 1$ or $\beta_i \neq 1$, we have $\alpha_i > \beta_i$. Thus clearly $\omega \succ_{SP} \omega'$. □

This indicates that $\succ_{\min}$ is a strong form of Pareto, namely, $\omega \succ_{\min} \omega' \Leftrightarrow \forall i$, either $\beta_i \neq 1$ and $\alpha_i > \beta_i$ or $\alpha_i = \beta_i = 1$. Thus, the symmetric Pareto is a refinement of the minimum-based ordering.

Example 3.7. *In Example 3.3 we can see that $j_b p_w s_b \succeq_{\min} j_b p_w s_r$, while we have a strict order with the symmetric Pareto (equivalently, the product-based) ordering $j_b p_w s_b \succ_{SP} j_b p_w s_r$ and we have $j_b p_b s_r \succ_{SP|\min} j_b p_b s_w$.*

Proposition 3.6. *Let ω and ω' be two configurations, then $\omega \succeq_{\min} \omega' \Rightarrow \omega \succeq_{SP} \omega'$.*

Proof. It immediately follows from $\omega \succ_{\min} \omega' \Rightarrow \omega \succ_{SP} \omega'$ (Prop. 3.5) and from $\omega \sim_{min} \omega' \Rightarrow \omega \sim_{SP} \omega'$ (by Proposition 3.2). □

When there is no additional constraints, i.e., $\mathcal{C}_1 = \emptyset$, Pareto and discrimin orderings yield the same ordering:

Proposition 3.7. *Pareto and discrimin coincide on vectors when $\mathcal{C}_1 = \emptyset$.*

Proof. Let us consider two vectors $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \ldots, \beta_N)$, where a symbolic weight α_i of $\vec{\omega}$ may only be compared to the corresponding symbolic weight β_i of $\vec{\omega'}$ (if there is a relevant constraint in $\mathcal{C}_0$). Three cases arise:

- $\vec{\omega} \succ_{Pareto} \vec{\omega'}$ iff $\vec{\omega} \succ_{discrimin} \vec{\omega'}$: ($\Rightarrow$) if $\vec{\omega} \succ_{Pareto} \vec{\omega'}$ then $\vec{\omega} \succeq_{\min} \vec{\omega'}$. This means that $\min(\vec{\omega}) \geq \min(\vec{\omega'})$. Since discrimin deletes all equalities $\alpha_i = \beta_i$, we will have $\forall i \in D$ $\min_{i \in D} \alpha_i > \min_{i \in D} \beta_i$ s.t. D is the set of component indexes not deleted. Therefore, $\vec{\omega} \succ_{discrimin} \vec{\omega'}$. ($\Leftarrow$) Since discrimin deletes only weights where $\alpha_i = \beta_i$ and never strict comparisons, then after the deletion process we only have constraints such that $\alpha_j > \beta_j$, which means that the strict order is the same as Pareto ordering.
- $\vec{\omega} \sim_{Pareto} \vec{\omega'}$ iff $\vec{\omega} \sim_{discrimin} \vec{\omega'}$. Obvious.
- $\vec{\omega} \pm_{Pareto} \vec{\omega'}$ iff $\vec{\omega} \pm_{discrimin} \vec{\omega'}$: ($\Rightarrow$) if $\vec{\omega} \pm_{Pareto} \vec{\omega'}$ we have $\min(\vec{\omega}) \pm \min(\vec{\omega'})$ (by Proposition 3.3), and discrimin can only delete equalities, then the vectors remain non comparable with discrimin. Thus $\vec{\omega} \pm_{discrimin} \vec{\omega'}$. ($\Leftarrow$) if $\vec{\omega} \pm_{discrimin} \vec{\omega'}$ then we have not $\forall i \in D$ $\alpha_i < \beta_i$ where D is the set of component indexes not deleted. Thus we have $\vec{\omega} \pm_{Pareto} \vec{\omega'}$.

□

Consequently from Proposition 3.7, we can derive that $\succ_{leximin} \Leftrightarrow \succ_{discrimin} \Leftrightarrow \succ_{Pareto} \Leftrightarrow \succ_{SP}$. Relations between the different orderings are depicted by Figure 3.6 (inside each box, relations are equivalent). This indicates a collapse of many notions when no additional constraints between symbolic weights applicable to different components exist.

3.5.2 Comparison of orderings with additional constraint on symbolic weights

As already mentioned, constraints between symbolic weights, beside those induced from the preference specification, can be added when available. In this section we will study the relations between the different ordering relations in the presence of such constraints. First, we will see that the refinement relations that exist in the case of numerical values remain valid.

Proposition 3.8. *Let ω and ω' be two configurations, then $\vec{\omega} \succ_{Pareto} \vec{\omega'} \Rightarrow \vec{\omega} \succ_{SP} \vec{\omega'}$.*

Proof. It suffices to consider the permutation σ as the identity. □

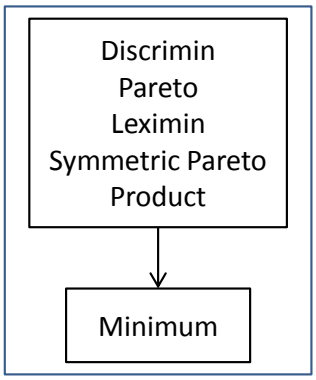


Figure 3.6: Refinements between orderings in symbolic setting with constraints

Minimum-based ordering suffer from a "drowning" effect. Ways to overcome this problem are the discrimin or leximin orderings. In the numerical case, the latter are refinements of the min-based ordering [Dubois et al., 1996]. We will prove that this is still the case in the symbolic framework. More formally:

Proposition 3.9. *Let ω and ω' be two configurations, then $\vec{\omega} \succ_{\min} \vec{\omega'} \Rightarrow \vec{\omega} \succ_{discrimin} \vec{\omega'}$ and, $\vec{\omega} \succ_{discrimin} \vec{\omega'} \Rightarrow \vec{\omega} \succ_{leximin} \vec{\omega'}$.*

Proof. Let us consider two configurations ω and ω' such that $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \ldots, \beta_N)$. If $\min(\vec{\omega'}) < \min(\vec{\omega})$, then $\exists\ \beta_i$ s.t. $\beta_i < \alpha_1, \ldots, \alpha_N$ and $\beta_i \neq \alpha_1, \ldots, \alpha_N$. This symbolic weight β_i cannot be eliminated in the deletion process of discrimin nor leximin. Thus, $\vec{\omega} \succ_{\min} \vec{\omega'} \Rightarrow \vec{\omega} \succ_{disrimin} \vec{\omega'}$ and $\vec{\omega} \succ_{\min} \vec{\omega'} \Rightarrow \vec{\omega} \succ_{leximin} \vec{\omega'}$. Besides, it is clear that leximin still refines discrimin since it suffices to consider the permutation σ processed by leximin as the identity. □

Example 3.8. *Let us consider the possibilistic preference network of Example 3.3. Let us consider some additional constraints such that $\mathcal{C}_1$ includes $(\alpha < \beta)$, $(\beta = \lambda_1)$, $(\lambda_1 < \theta_1)$. Thus, $\mathcal{C} = \{(\delta_1 > \delta_2), (\theta_1 > \theta_2), (\lambda_1 > \lambda_2), (\alpha < \beta), (\beta = \lambda_1), (\lambda_1 < \theta_1)\}$. Let us take the two configurations $j_b p_w s_b$ and $j_r p_b s_b$ such that $j_b \vec{p_w} s_b = (\beta, \theta_1)$ and $j_r \vec{p_b} s_b = (\alpha, \lambda_1)$. We can see that $j_b \vec{p_w} s_b \succ_{\min |discrimin|leximin} j_r \vec{p_b} s_b$.*

If we consider only partially ordered symbolic weights, leximin may lead to non comparability when discrimin or minimum considers two configurations equal. Thus, leximin ordering will sometimes lead to a partial ordering. This can be illustrated by the following example:

Example 3.9. *Let us consider the same two vectors of example 3.8, $j_b \vec{p_w} s_b = (\beta, \theta_1)$ and $j_r \vec{p_b} s_b = (\alpha, \lambda_1)$. We assume the set of constraints $\mathcal{C} = \{(\delta_1 > \delta_2), (\theta_1 > \theta_2), (\lambda_1 > \lambda_2)(\lambda_1 < \alpha), (\beta = \lambda_1), (\beta < \theta_1)\}$, then $j_b \vec{p_w} s_b \sim_{\min |discrimin} j_r \vec{p_b} s_b$ while $j_b \vec{p_w} s_b \pm_{leximin} j_r \vec{p_b} s_b$. Now if we suppose that $j_b \vec{p_w} s_b = (\beta, \theta_1)$, then $j_b \vec{p_w} s_b \sim_{\min} j_r \vec{p_b} s_b$ while $j_b \vec{p_w} s_b \pm_{discrimin|leximin} j_r \vec{p_b} s_b$.*

Let us now compare the discrimin and the Pareto orderings. Then the leximin and the symmetric Pareto orderings will be in a similar relation. Besides, there is no relation between the discrimin and the symmetric Pareto orderings when $\mathcal{C}_1 \neq \emptyset$. Indeed there are situations where discrimin can compare two vectors and the symmetric Pareto cannot (e.g., if we only know that the component of one vector is smaller than all the other components of the two vectors), and situations where symmetric Pareto can compare and discrimin cannot (e.g., $\vec{\omega} = (\alpha_1, \alpha_2, \alpha_3)$, $\vec{\omega'} = (\beta_1, \beta_2, \beta_N)$ and $\mathcal{C} = \{\alpha_1 > \beta_1, \alpha_2 > \beta_3, \alpha_3 > \beta_2\}$.

Proposition 3.10. *Let ω and ω' be two configurations, then $\vec{\omega} \succ_{Pareto} \vec{\omega'} \Rightarrow \vec{\omega} \succ_{discrimin} \vec{\omega'}$.*

Proof. Let us consider two solutions ω and ω' such that $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \ldots, \beta_N)$. By definition, if $\vec{\omega} \succ_{Pareto} \vec{\omega'}$ then $\forall i,\ \alpha_i \geq \beta_i$ and $\exists j, \alpha_i > \beta_i$. Let $\vec{\omega_*}$ (resp. $\vec{\omega'_*}$) denote the vector induced after deleting all vector components such that $\alpha_i = \beta_i$, $\forall i \in N$. Then, $\forall i \in D$, such that D is the set of the remaining vector component indexes, we have $\alpha_i > \beta_i$. This means that $\exists \beta_j \in \vec{\omega'_*}$ such that $\beta_j < \min(\vec{\omega_*})$. Therefore, $\vec{\omega} \succ_{\min} \vec{\omega'}$. Since discrimin refines minimum (Proposition 3.9), hence Proposition 3.10 holds.

□

From Proposition 3.10 we can derive that symmetric Pareto and leximin lead to consistent orderings. Moreover, each time when symmetric Pareto succeeds to order two configurations, discrimin will induce, if not the same ranking, at most non comparability.

Let us compare the minimum based-ordering and the product-based ordering (equivalently, SP). It is clear that we have:

Proposition 3.11. *Let ω and ω' be two configurations, then $\omega \sim_{SP} \omega' \Rightarrow \omega \sim_{min} \omega'$.*

Proof. Assume two solutions ω and ω' such that $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \ldots, \beta_N)$. If $\omega \sim_{SP} \omega'$ then $\omega \sim_{Pareto} \omega'_\sigma$. Thus, $\forall\ i, \alpha_i = \beta_{i\sigma}$, where $i \in [1 \ldots N]$. Therefore, $\min(\beta_1, \ldots, \beta_N) = \min(\alpha_1, \ldots, \alpha_N)$. Hence the symmetric Pareto ordering equalities are always found in min-based ordering. □

Equalities between solutions in product-based ordering may appear when one assumes equalities between symbolic weights associated to the same nodes and the same context or to symbolic weights of different nodes. This is unlike min-based ordering where it always considers the most important constraint violated, more precisely, having the smallest symbolic weight. Hence, in min-based orderings equalities appear when two solutions violate the same preference with the highest priority compared to the set of the other violated preferences.

Proposition 3.12 shows that symmetric Pareto is a special kind of refinement of the min-based ordering. Indeed:

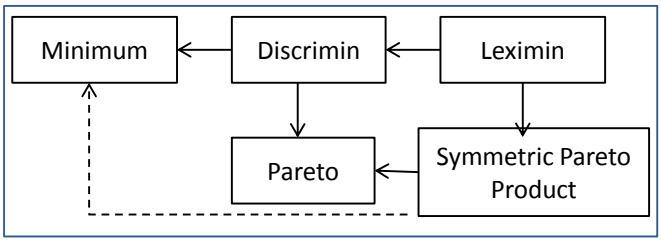


Figure 3.7: Refinements between orderings in symbolic setting without constraints

Proposition 3.12. *Let ω and ω' be two configurations, if $\omega \succ_{\min} \omega'$ we may either have $\omega \pm_{SP} \omega'$ or $\omega \succ_{SP} \omega'$.*

Proof. Let us consider two solutions ω and ω' such that $\vec{\omega} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{\omega'} = (\beta_1, \ldots, \beta_N)$. Indeed, from Proposition 3.11, if $\omega \sim_{SP} \omega'$ then $\omega \sim_{\min} \omega'$. Moreover, if $\omega \prec_{SP} \omega'$ then by the definition we have $\forall i,\ \alpha_i \leq \beta_i$, thus $\min(\alpha_1, \ldots, \alpha_N, \beta_i, \ldots, \beta_N) \in \{\alpha_i, \ldots, \alpha_N\}$, this proves that we cannot have $\omega \succ_{\min} \omega'$ in this case. Hence a contradiction, and Proposition 3.12 follows. □

Relations between the different orderings can be illustrated by Figure 3.7. Leximin ordering refines symmetric Pareto ordering, which in its turn refines Pareto ordering. Moreover, leximin refines discrimin and both are refinements of minimum ordering. It is important to notice that, in contrast with the numerical setting, minimum and leximin orderings may lead to non comparability and thus to partial orderings. Besides, symmetric Pareto still refines the minimum ordering, but in a wider sense since symmetric Pareto may yield non comparability when minimum succeeds in comparing (this relation is represented in Figure 3.7 by a dotted line).

One extreme case is when assuming a total preorder between the symbolic weights. In that case, leximin and minimum orderings are total. However, in the presence of such constraints, symmetric Pareto may still lead to non comparability. Indeed, the only case, where symmetric Pareto leads to a total ordering is when there are constraints between *subsets* of symbolic weights (corresponding to the comparison of subproducts). Thus, the relationships between the different orderings are the same as in the numerical setting except for the product and symmetric Pareto orderings, which are the same as previously proved.

3.6 Optimization and dominance query

In a preference model, two types of queries are commonly used: namely, optimization queries for finding the optimal configuration(s) (i.e. those which are not dominated by others) and dominance

queries for comparing configurations.

3.6.1 Optimization

Since π-pref nets allow the user to express indifference, the optimization query may return more that one configuration. Clearly, the best configurations are those having a joint possibility degree equal to 1. Indeed, such a configuration always exists since the joint possibility distribution associated to the possibilistic network is normalized, thanks to the normalization of each conditional possibility table (i.e. for each variable A_i, each instantiation $p(A_i)$ of $\mathcal{P}(A_i)$: $\max(\pi(a_i \mid p(A_i)), \pi(\neg a_i \mid p(A_i))) = 1$ where $\{\neg a_i\} = D_{A_i}/\{a_i\}$ with $a_i \in D_{A_i}$). Thus, we can always find an optimal configuration, starting from the root nodes where we choose each time the most or one of the most preferred value(s) (i.e. with possibility equal to 1). Then, depending on the parents instantiation, each time we again choose an alternative with a conditional possibility equal to 1. At the end of the procedure, we get one or several completely instantiated configurations having a possibility equal to 1. Consequently, partial preference orders with incomparable maximal elements can not be represented by a π-pref net.

Example 3.10. *Let us reconsider Example 3.3 and its product-based joint possibility degree depicted by Figure 3.4. Then, $j_b p_b s_b$ is the preferred configuration since its joint possibility is equal to 1, and this is the only one.*

This procedure is linear in the size of the network (using a forward sweep algorithm). A possible variant of the optimization problem is to compute the M most possible configurations using a variant of the MPE [Nilsson, 1998]. This query can be interesting in π-pref nets even if the answer is not always obvious to obtain in presence of incomparable configurations.

3.6.2 Dominance

The comparison between the symbolic possibility degrees can be found using Algorithm 1 that takes as input the set of constraints $\mathcal{C}$ between the symbolic weights and two vectors. Let us consider two configurations ω_i and ω_j with simplified respective vectors $\vec{\omega_i^*} = (\alpha_1, \ldots, \alpha_k)$ and $\vec{\omega_j^*} = (\beta_1, \ldots, \beta_m)$ where the components equal to 1 have been deleted, with $k \leq m \leq n$. Then, the algorithm proceeds by first deleting all pairs of equal components between the vectors so to get totally different components. Second, it checks if there exists an injective function $\varphi : \{1, \ldots, k\} \to \{1, \ldots, m\}$ such that $\forall i = 1, \ldots, k, \alpha_i \geq \beta_{\varphi(i)}$ and $\exists \ell \in \{1, \ldots, k\}, \alpha_\ell > \beta_{\varphi(\ell)}$ (otherwise they remain incomparable).

Thus the algorithm is based on the sequential application of:
(1) The function $equality$ that deletes the common values between $\vec{\omega_i}$ and $\vec{\omega_j}$.
(2) The function $sort$ that returns $true$ if given $\alpha_c \in \vec{\omega_i}$, there exists a constraint $\alpha_c > \delta$ in $\mathcal{C}$ such

that $\delta \in \vec{\omega_j}$. Each component of $\vec{\omega_j}$ can be used only one time in the comparison process. (3) The function $empty$ that tests if a vector of weights $\vec{\omega}$ is empty or not.

Algorithm 1: Comparison between two joint possibility degrees

Data: $\vec{\omega_i}, \vec{\omega_j}, \mathcal{C}$
Result: R
begin
  $equality(\vec{\omega_i}, \vec{\omega_j}, \mathcal{C})$;
  if $(empty(\vec{\omega_i})$ *and* $empty(\vec{\omega_j}))$ **then** $R \leftarrow \vec{\omega_i} = \vec{\omega_j}$;
  else $s \leftarrow sort(\vec{\omega_i}, \vec{\omega_j}, \mathcal{C})$;
  if $s == true$ **then** $R \leftarrow \vec{\omega_i} \succ \vec{\omega_j}$;
  else $s \leftarrow sort(\vec{\omega_j}, \vec{\omega_i}, \mathcal{C})$;
  if $s == true$ **then** $R \leftarrow \vec{\omega_j} \succ \vec{\omega_i}$;
  else $R \leftarrow \vec{\omega_j} \pm \vec{\omega_i}$;
  return R

Note that this algorithm will be discussed in detail in Chapter 6.

Example 3.11. *Let us consider the π-pref net ΠG of Example 3.3. Using Algorithm 1, the ordering between the configurations is defined in Figure 3.4 such that a link from ω_i to ω_j means that ω_i is preferred to ω_j. For instance, consider $j_b\vec{p_w}s_r = (\beta,\ \delta_4)$ and $j_r\vec{p_w}s_r = (\alpha,\ \beta)$. First, we should delete common values, namely the symbolic weight β. Then, we should check if $\mathcal{C}$ entails $\alpha < \delta_4$ or the inverse. Here, α and δ_4 are not comparable. Thus, we have $j_b p_w s_r \pm j_r p_w s_r$.*

Clearly, for π-pref nets, the complexity is due to the comparison step in Algorithm 1 (since the computation of the possibility degrees is a simple matter of using the chain rule) and in particular to the $sort$ function where the matching between the two vectors needs the definition of different possible arrangements i.e. the algorithm is of time complexity O(N!).

3.7 Conclusion

This chapter has presented an approach to preference modeling based on joint possibility distributions obtained from a conditional preference network, albeit without using numerical possibility values to represent preference intensity. We have used uninstantiated symbols taking values on the unit interval and constraints between them to describe local relative preferences. Then we compute the product thereof to assign symbolic possibility values to complete configurations (solutions) using the product chain rule of possibilistic networks. The preference graph between configurations is then obtained by comparing composite symbolic possibility values.

Moreover, this model proves to be flexible enough to support different readings leading to different orderings of solutions, and establishes the main relationships between them. π-pref nets correctly reflect the elicited information in the sense that no further implicit priority is enforced like with CP-nets (e.g., in favor of parent nodes). They also offer a cautious way of modeling preferences without requiring numerical values, which should make them attractive for the same class of applications as CP-nets. In fact, precise numerical assessments are hard to get for conditional preferences that are qualitative in nature. Moreover, we have shown that symbolic possibilistic networks can handle additional qualitative information when available.

In the next chapter, we discuss the possible relations that may exist between π-pref nets and other preferential models. We also highlight the assets of each model and its drawbacks.

CHAPTER 4

π-Pref nets vs Other Preference Models

Contents

4.1 Introduction

In this chapter, we raise the problem of comparing π-pref nets (proposed in Chapter 3) to various preferential models. First, we will discuss possible logical representations of π-pref nets following the same idea proposed by [Benferhat et al., 2002a] in order to establish a bridge from π-pref nets to possibilistic logic. Such transformation allows to preserve the connection to a formal logical framework which is interesting for fusing heterogeneous information. We also show that in the same manner one can build a symbolic penalty logic base.

Second, we will study different relations between π-pref nets and existing preference models presented in Chapters 1 and 2. More precisely, we compare and discuss existing relationships between π-pref nets over binary variables and CP-nets, CP-theories, and OCF-nets. Indeed, CP-nets share the same preference specification and graphical structure as π-pref nets, CP-theories are

a generalization of CP-nets, while OCF-nets are based on an additive structure which is close to the one of π-pref nets.

This Chapter is organized as follows: Section 4.2 discusses two logical counterparts of π-pref nets, namely, possibilistic logic and penalty logic. Section 4.3 is devoted to the relation between π-pref nets and OCF-nets. Section 4.4 details the transformation between π-pref nets and CP-nets and explains the constraints applied to the symbolic weights. Section 4.5 outlines some similarities between π-pref nets and CP-theories. Section 4.6 offers an overall comparison of the various models presented in this thesis.

4.2 Logical counterparts of π-pref nets

In this section, we are interested by possible logical representations of π-pref nets. In the possibilistic framework, some attempts to build bridges between graphical and logical models were proposed. We can mention in particular the transformation between possibilistic network and *possibilistic logic base* [Benferhat et al., 2002a, Benferhat et al., 2001a] and the one between possibilistic weights and penalty weights [De Saint-Cyr et al., 1994]. The main motivation behind these transformations is to take advantage of different models at both representational and reasoning levels.

Following these transformations, we are interested by studying the logical counterparts of π-pref nets in terms of symbolic possibilistic logic base and symbolic penalty logic base.

4.2.1 From π-pref nets to symbolic logic bases

Possibilistic logic bases [Dubois et al., 1994] and possibilistic networks [Fonck, 1994, Ben Amor and Benferhat, 2005] are two formats of the possibilistic framework. The former ranks the pieces of goals (expressed by logical formulas) according to their level of importance, while the latter exhibits relationships between variables. The two types of representations are semantically equivalent when they lead to the same possibility distribution. Possibilistic logic provides a sound and complete machinery for handling qualitative preference with respect to a semantics expressed by means of possibility distributions while Bayesian-like networks have a clear appeal for preference acquisition since they exhibit explicitly independence relations between decision variables.

The goal of this section is to establish a bridge between π-pref nets and symbolic possibilistic logic. In fact, we wish to take advantage of the graphical representation while preserving the connection to a formal logical framework since there exists computational machineries of reasonable complexity [Dubois et al., 1994] in possibilistic logic. In order to apply the existing transformation methods [Benferhat et al., 2001a, Benferhat et al., 2002a] to the symbolic case, we first provide a remainder of the possibilistic logic base and the transformations of the numerical case.

4.2.1.1 Recall on possibilistic logic

Possibilistic logic base [Dubois et al., 1994] is a set of a finite propositional language f_i (first-order logic) denoted by Σ. Each formula has a weight c_i, belonging to the scale $[0, 1]$, which expresses, in the preference setting, to what extent the constraint is imperative with consideration to the incomplete available information. A N-possibilistic base is a possibilistic base where the weights are computed as necessity measures. The possibilistic base is under the form $\Sigma = \{(f_i, c_i) : i = \{1, \ldots, m\}\}$ such that f_i can be represented under a conjunctive or disjunctive form [Benferhat et al., 2001b].

In the qualitative setting, this base can be presented by a set of partitions where each one contains formulas with the same necessity degree. The order between the alternatives is given by a possibility distribution. In fact, from this knowledge base one can generate a unique possibility distribution by associating to each configuration the level of compatibility with the preferences:

$$\forall \omega \in \Omega, \pi_{\Sigma}(\omega) = \begin{cases} 1 & \text{if } \forall (f_i, c_i) \text{ in } \Sigma,\ \omega \models f_i \\ 1 - max\{c_i : (f_i, c_i)\} \in \Sigma \text{ and } \omega \not\models f_i & \text{otherwise} \end{cases} \quad (4.1)$$

Therefore, the configurations satisfying all the logical formulas will have the highest possibility degree equal to 1. We can distinguish two completion principles that respectively generate the largest and the smallest possibility distributions which satisfies the set of goals [Gérard et al., 2007, Kaci et al., 2006]. More precisely, the minimal specificity principle accords the most important degree to configurations. It does not enforce any preference between the criteria if not explicitly provided. Contrariwise the maximal of specificity gives the lowest possible degree to configurations. In fact, they can refer to two kinds of decision makers. Optimistic decision makers apply the maximum of specificity, because they tend to give the configurations the highest satisfactory degree 1, and cautious ones use the minimum of specificity, by giving the highest degree only to configurations that are never dominated by others. In both settings, all configurations are compared and there is no room for incomparability. Note that standard possibility theory uses the minimal of specificity.

Example 4.1. *Let* $\Omega = \{\omega_1 = abc,\ \omega_2 = ab\neg c,\ \omega_3 = a\neg bc,\ \omega_4 = a\neg b\neg c,\ \omega_5 = \neg abc,\ \omega_6 = \neg ab\neg c,\ \omega_7 = \neg a\neg bc,\ \omega_8 = \neg a\neg b\neg c\}$ *be the universe of discourse. We consider the following possibilistic logic base:* $\Sigma = \{(\neg a \vee b, 1/3), (\neg a \vee \neg c, 1/3), (a, 1/3), (a \vee \neg b, 2/3), (a \vee c, 2/3)\}$. *We can generate a unique possibility distribution:* $\pi(\omega_2) = 1, \pi(\omega_1) = \pi(\omega_3) = \pi(\omega_4) = \pi(\omega_7) = 2/3, \pi(\omega_5) = \pi(\omega_6) = \pi(\omega_8) = 1/3$.
For example, ω_7 *violates* $(a, 1/3)$. *The satisfaction degree of* ω_7 *is then* $\pi(\omega_7) = 1 - 1/3 = 2/3$.

4.2.1.2 Recall on transformations from possibilistic networks to possibilistic logic

The basic idea of the transformation from a possibilistic network to possibilistic logic base is to view the possibilistic base corresponding to a possibilistic network as the result of fusing elemen-

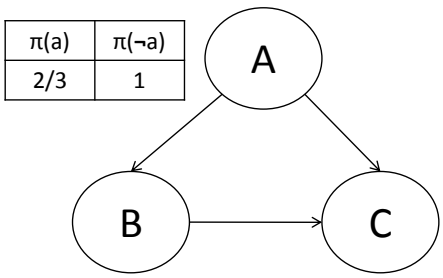
tary bases [Benferhat et al., 2001a]. Each elementary base is associated to a variable of the graph and considers all conditional possibilities different from 1 attached to the node. More precisely, the elementary base associated with the variable A_i is:

$$\Sigma_{A_i} = \{(\neg a_i \vee \neg p(A_i), 1 - \alpha_i) : \pi(a_i|p(A_i)) = \alpha_i \text{ and } \alpha_i \neq 1\}$$

To have the complete logic base associated with the possibilistic network, these local bases should be combined. More precisely, since we are mainly interested by possibilistic networks with product-based conditioning, applying the product-based chain rule (Equation 3.1) should give the same result as if we combine the possibility distributions associated to the elementary bases. Formally, the complete logic base is given by the following combination:

Definition 4.1. *Let Σ_1 and Σ_2 be two bases, π_1 and π_2 their associated distributions. The possibilistic base associated with $\pi_2 \times \pi_2$ is: $\Sigma_1 \cup \Sigma_2 \cup \{(\phi_i \vee \psi_j, \alpha_i + \beta_j - \alpha_i\beta_j) : (\phi_i, \alpha_i) \in \Sigma_1$ and $(\psi_j, \beta_i) \in \Sigma_2\}$*

Example 4.2. *Consider the possibilistic network ΠG of Figure 4.1. The set of variables is $\mathcal{V} = \{A, B, C\}$. From local possibility distributions we can compute the local possibilistic logic bases relative to each node. For instance the one relative to the node A is computed as follows: we consider the possibility degree different from 1 here we have $\pi(a) = 2/3$ and since A has no parents then when we apply $\Sigma_{A_i} = \{(\neg a_i \vee \neg p(A_i), 1 - \alpha_i)$ we get $\Sigma_A = \{(\neg a, 1 - 2/3)\} = \{(\neg a, 1/3)\}$. Following the same procedure we have these three local possibilistic logic bases: $\Sigma_A = \{(\neg a, 1/3)\}$, $\Sigma_B = \{(\neg a \vee \neg s, 1/2)\}$ and $\Sigma_C = \{(a \vee b \vee \neg c, 1/3), (a \vee \neg b \vee c, 1/3)\}$. We first compute the combination of Σ_A and Σ_B. We get $\Sigma_{AB} = \{(\neg a, 1/3), (\neg a \vee \neg b, 1/2), (\neg a \vee \neg b, 2/3)\}$. This base is equivalent to $\Sigma_{AB} = \{(\neg a, 1/3), (\neg a \vee \neg b, 2/3)\}$ since we consider formulas having the highest necessity degree if they are duplicated. Combining Σ_{AB} and Σ_C, we get: $\Sigma_{\Pi G} = \Sigma_{ABC} = \{(\neg a, 1/3), (\neg a \vee \neg b, 2/3), (a \vee b \vee \neg c, 1/3), (a \vee \neg b \vee c, 1/3)\}$.*



π(.|.)	a	¬a
b	1/3	1
¬b	1	1

π(.|.)	ab	a¬b	¬ab	¬a¬b
c	2/3	1	1	1
¬c	1	1	1/2	1

Figure 4.1: Possibilistic network of Example 4.2

Note that the converse transformation from a possibility distribution to possibilistic logic base is also possible. However, moving from a logical format to a graphical one is less straightforward since independence relations are not explicit in possibilistic logic [Benferhat et al., 2002a].

4.2.1.3 Transformation from π-pref nets to symbolic logic bases

As detailed earlier, π-pref nets represent a particular reading of possibilistic networks in the preference framework where weights are non-instantiated. This, naturally, leads us to apply the aforementioned transformation to the symbolic case, since computing the logical counterpart of π-pref nets may be of interest for defining inconsistency and merging preference of several sources.

Similarly to the numerical case, we first define the local possibilistic logic bases associated each node of the π-pref net:

Definition 4.2. *The symbolic possibilistic base Σ_i associated to a Boolean variable A_i in a π-pref net ΠG is defined as follows:*
- For each preference statement $p(A_i): a_{i1} \succ a_{i2}$ between the two possible values of a variable A_i, $(\neg p(A_i) \vee a_{i1}, \beta) \in \Sigma_i$ where $\pi(a_{i2}|p(A_i)) = 1 - \beta < 1$ in ΠG.
- There is no formula induced by preference statements $p(A_i) : a_{i1} \sim a_{i2}$.

Proposition 4.1. *If π_i is the possibility distribution induced by Σ_i associated with node A_i, then $\pi_i(\omega[\{A_i\} \cup \mathcal{P}(A_i)]) = \pi(a_i|p(A_i))$ where $a_i = \omega[A_i]$, $p(A_i) = \omega[\mathcal{P}(A_i)]$.*

Proof. The proof is trivial. □

We may merge the local possibilistic bases generated by Definition 4.2 in 3 different ways:

1. The possibilistic base associated with a π-pref net can be obtained by fusing the elementary bases Σ_i $(i = 1, ..., N)$ associated to its nodes. Since we are in the product-based setting (π-pref nets), the combination of these possibilistic bases is defined iteratively as $Comb(\Sigma_1, \Sigma_2) = \Sigma_1 \cup \Sigma_2 \cup \{(p_i \vee q_j, \alpha_i + \beta_j - \alpha_i \times \beta_j) : i \in I,\ j \in J,\ p_i \vee\ q_j \neq \top\}$, where $\Sigma_1 = \{(p_i, \alpha_i) : i \in I\}$ and $\Sigma_2 = \{(q_j, \beta_j) : j \in J\}$. The base resulting from this product-based combination is a (possibly large) possibilistic base that encodes the same possibility distribution as $\pi_{\Pi G}$, see [Dubois and Prade, 2004]. Such a logical counterpart is a symbolic possibilistic base of the form $\Sigma = \{(f_1, c_1), \ldots, (f_m, c_m)\}$ which is a finite set of weighted formulas f_i where $c_i > 0$ is also understood as a lower bound of a necessity degree $N(f_i)$ but refers to a symbolic weight [Dubois and Prade, 2004]. Its semantics is a possibility distribution $\pi_\Sigma(\omega) = min_{i=1,n}\pi_{\{(f_i,c_i)\}}(\omega) = 1$ if $\omega \vDash f_i$ and $1 - c_i$ if $\omega \vDash \neg f_i$. This method leads to a symbolic base where one can apply inference rules and revision. Note that any distribution can be associated with a possibilistic logic base, and also equivalently represented by a possibilistic network [Benferhat et al., 2001a].

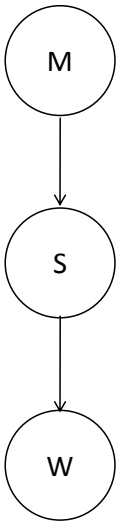
2. One can also consider the corresponding symbolic base as the union of these local symbolic bases. Then the possibility distribution would be computed via vector comparisons. To find the same ordering as π-pref net, we should use symmetric Pareto to compare vectors of configurations. This representation has no advantage compared to π-pref nets since we cannot apply inference rules on it. There were several attempts to represent CP-nets orderings using a symbolic possibilistic logic base. The construction of the local logic bases is exactly as presented here. However, to each node they associate only one symbolic weight. Therefore, to each node is associated at most one logic formula. Besides, they do not consider any indifference between variable values. See [Dubois et al., 2015] for a bibliography and a discussion. It was also observed that an exact logical representation of CP-nets was not possible when nodes in the network have several children even though good approximations can be considered. This is because additional constraints in that framework compare individual symbolic weights, not product thereof. In addition, they show that Symmetric Pareto and Leximin orderings respectively lower and upper bound the CP-net ordering. These results can be exploited with a π-pref-net since it represent its graphical counterpart [Dubois et al., 2013b, Dubois et al., 2013a].

3. A third method, is to construct a hybrid possibilistic network [Benferhat and Smaoui, 2007] where we conserve the graphical structure of π-pref net and we associate to each node a local possibilistic logic base. This hybrid representation generalizes the two representation frameworks: possibilistic logic and possibilistic networks. Graphical representation is used to take advantage of independence relations, and logic-based representation is used to have compact representation of possibility distributions. More precisely, local preferences are no longer represented by conditional possibility distributions but by possibilistic logic bases. From each local logic base Σ_i we can compute the local possibility distribution π_i using Proposition 4.1. Since with π-pref nets we deal with product (symmetric Pareto) then the possibility distribution π associated to the preference networks is computed such that $\pi = \times_{i=1}^{N} \pi_i$. Hybrid possibilistic networks were proposed for the aim of improving standard junction tree propagation algorithms [Benferhat and Smaoui, 2007].

Example 4.3. *Let us consider a preference specification for dinner over* 3 *variables* $\mathcal{V} = \{M, S, W\}$*, standing for the main course, the soup and wine respectively s.t.*
$D_M = \{$ *meat course* (M_{mc}), *fish course* $(M_{fc})\}$, $D_S = \{$ *fish soup* (S_f), *vegetable soup* $(S_v)\}$,
$D_W = \{$ *white wine* (W_w), *red wine* $(W_r)\}$.
The preference conditional set is:

The user prefers a meat course (M_{mc}) to a fish course (M_{fc})
If the main course is meat, he prefers to have a fish soup (S_f) to a vegetable one (S_v)
If the main course is fish, he prefers to have a vegetable soup to a fish soup
If the is served a vegetable soup, he prefers to have red wine (W_r) to white one (W_w)
If the is served a fish soup, he prefers to have white wine to red one

Based on these preference statements, we can construct the possibilistic network of Figure 4.2.



$\pi(M_{mc})$	1
$\pi(M_{fc})$	α

$\pi(.\|.)$	M_{mc}	M_{fc}
S_f	1	β_1
S_v	β_2	1

$\pi(.\|.)$	S_f	S_v
W_w	1	δ_1
W_r	δ_2	1

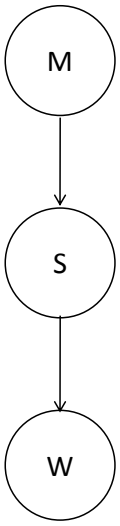
Figure 4.2: The possibilistic network of Example 4.3

- *The possibilistic base associated with the product of the different local possibility distributions* $\pi_1 * \pi_2$ *is* $\Sigma_{prod} = \Sigma_1 \cup \Sigma_2 \cup \{\phi \vee \psi, \alpha_i + \beta_j - \alpha_i\beta_j\}$*:* $(\phi_i, \alpha_i) \in \Sigma_1$ *and* $(\psi_j, \beta_j) \in \Sigma_2$ *(the first merging method). Therefore, to find the base associated with the graph we should apply successively the previous formula on each node logic base. Let us assume:*
$\alpha' = 1 - \alpha$, $\beta_1' = 1 - \beta_1$, $\beta_2' = 1 - \beta_2$, $\delta_1' = 1 - \delta_1$, $\delta_2' = 1 - \epsilon_2$.
The combination of the Σ_M *and* Σ_S *leads to:*
$\Sigma_{MS} = \Sigma_M \cup \Sigma_S \cup \{(M_{mc} \vee S_v, \alpha' + \beta' - \alpha' * \beta')\}$. *Then, combining the resulting base with the base associated with the node* W *we get the final base:*
$\Sigma_{MSW} = \Sigma_M \cup \Sigma_S \cup \{(M_{mc} \vee S_v, \alpha' + \beta' - \alpha' * \beta')\} \cup \Sigma_w \cup \{(M_{mc} \vee S_v \vee W_w, \alpha' + \delta_1' - \alpha' * \delta_1'), (M_{mc} \vee S_f \vee W_w, \alpha' + \delta_2' - \alpha' * \delta_2'), (M_{fc} \vee S_f \vee W_w, \beta_1' + \delta_2' - \beta_1' * \delta_2'), (M_{mc} \vee S_v \vee W_w, \beta_2' + \delta_1' - \beta_2' * \delta_1'), (M_{mc} \vee S_v \vee W_w, \delta_1' + (\alpha' + \beta' - \alpha' * \beta') - \delta_1' * (\alpha' + \beta' - \alpha' * \beta')\}$.
Note that resulting base may contain some subsumed formulas that cannot be detected since the weights are not comparable.

- *The second merging methods corresponds to union the local logic bases and applying the symmetric Pareto to the resulting vectors.*

- *Figure 4.3 corresponds to the hybrid network associated to the* π*-pref net of Figure 4.2 which combines the knowledge bases associated to the nodes and the graphical structure (third method).*

This correspondence is useful if we have to combine pieces of information expressed in different formats, and to check their consistency. Each compact format has its interest for communication purposes, either for modelling expert knowledge, or for supplying information to the user. Besides, from an inference point of view, the logical and the graphical formats are the most



$$\Sigma_M = \{(M_{mc}, 1-\alpha)\}$$

$$\Sigma_S = \{(M_{fc} \vee S_f, 1-\beta_1), (M_{mc} \vee S_v, 1-\beta_2)\}$$

$$\Sigma_W = \{(S_v \vee W_w, 1-\delta_1), (S_f \vee W_r, 1-\delta_2)\}$$

Figure 4.3: The hybrid possibilistic network corresponding to the π-pref net of Example 4.3

interesting ones.

4.2.2 From a π-pref net to penalty logic

This subsection points out another logical counterpart of π-pref nets in terms of a penalty logic base Pκ [De Saint-Cyr et al., 1994], where weights are additive. This logic associates to each formula the cost (in $[0, +\infty)$) to pay if this formula is violated. The penalty $k_{PK}(\omega)$ relative to a configuration ω is the sum of the elementary penalties of the violated formulas. This contrasts with possibilistic logic, where weights are combined by an idempotent operation. The best solution has a cost equal to 0.

Definition 4.3. *A penalty knowledge base Pκ is a finite multi-set of pairs $\{f_i, c_i\}$ where f_i is a logic formula and c_i is the penalty associated to f_i; it represents intuitively what one should pay in order to get rid of f_i, if we pay the requested price we do not need any longer to satisfy f_i; so the larger c_i is, the more important f_i is. In particular, if $c_i = +\infty$ then it is forbidden to remove f_i from Pκ (f_i is inviolable).*

Since Pκ is a multi-set of pairs, it is possible for a pair (f_i, c_i) to appear many times in Pκ. For instance, P$\kappa_1 = \{(f_i, 1), (f_i, 1)\}$ is not equivalent to P$\kappa_2 = \{(f_i, 1), \}$ since using Pκ, it costs 2 to delete f_i while using Pκ_2, it costs only 1. However, in the case where a same formula appears several times in Pκ then we may replace all the occurrences of the formula f_i by only one occurrence annotated with the sum of the penalties associated to this formula in the first base.

The new knowledge base obtained is equivalent to the initial base. This contrasts with possibilistic logic where formulas having the same necessity degrees should appear only once in the logic base. This is explained by the use of the minimum function rather than sum.

The transformation from a π-pref net (based on the product) to a (symbolic) possibilistic logic (based on the minimum of specificity) leads to a complex logic base. A way to consider a simpler logic is to consider penalty logic. This interest is explained by the fact that the combination between penalty logic bases corresponds to the union of them. The aim is to find a transformation function that enables us to move from a possibility distribution to a distribution of costs. Let us consider a set of configurations $\Omega = \{\omega_1 = ab,\ \omega_2 = a\neg b,\ \omega_3 = \neg ab,\ \omega_4 = \neg a\neg b\}$ and $f()$ a function that transforms possibilities to costs, such that: The function $f()$ should respect these

Configuration	Possibilistic distribution	Costs
ω_1	1	$f(1) = 0$
ω_2	β	$f(\beta)$
ω_3	α	$f(\alpha)$
ω_4	$\alpha * \beta$	$f(\alpha * \beta)$

Table 4.1: Transformation function from possibilities to penalty weights

properties:

- $f(1) = 0$, since in penalty logic the best configurations have a cost equal to 0.
- $f : \alpha \in [0, 1] \rightarrow [0, +\infty[$
- $f(\alpha) > 0$ where α is a symbolic possibility degree, since costs are always positive.
- $f(\alpha * \beta) = f(\alpha) + f(\beta)$, since when violating more than one formula the cost is the sum of each formula's cost.

The function $f()$ acts like the logarithm function, we have:

- $\log(1) = 0$.
- $\log((\alpha) * (\beta)) = \log(\alpha) + \log(\beta)$.

In order to have positive values, the corresponding function is $f(\alpha) = -\log(\alpha)$. Therefore, $\alpha = 1 - \exp(-a)$ such that a is a penalty cost.

Thus, this logic with a cost interpretation has a close relationship with product-based π-pref nets. Indeed, the cost of a solution induced by a penalty logic base corresponds actually to the logarithmic transformation of the possibility degree computed from a π-pref net. Namely, in each local possibilistic base Σ_i associated to a node A_i we can at most violate one formula. Thus, for

each possibilistic base $\Sigma_i = \{(f_{i1}, \alpha_{i1}), \ldots, (f_{ik}, \alpha_{ik})\}$ there exists a penalty logic base $PK_i = \{(f_{i1}, -\log(\alpha_{i1})), \ldots, (f_{ik}, -\log(\alpha_{ik}))\}$ such that the ordering induced by π_i is the same as the ordering induced by the cost function. This mirrors the fact that $\pi_{\Pi G}(\omega) = \alpha_1 \cdot \cdots \cdot \alpha_N \Leftrightarrow k_{PK}(\omega) = -(\log(\alpha_1) + \cdots + \log(\alpha_N))$. Contrarily to possibilistic bases, the combination between penalty bases is the union of all PK_i $(i = 1, ..., N)$. This yields the same ordering as π-pref nets. However, there is no proof system for penalty logic yet.

Example 4.4. *Considering the hybrid possibilistic network of Figure 4.2 the associated penalty base is:* $PK = \{(M_{mc}, \log(\alpha^{'})), (M_{fc} \vee S_f, \log(\beta_1^{'})), (M_{mc} \vee S_v, \log(\beta_2^{'})), (S_v \vee W_w, \log(\delta_1^{'})), (S_f \vee W_r, \log(\delta_2^{'}))\}$. *such that prime signs correspond to reversed scales, for instance,* $\alpha' = 1 - \alpha$.

	(M_{mc}, α')	$(M_{fc} \vee S_f, \beta_1')$	$(M_{mc} \vee S_v, \beta_2')$	$(S_v \vee W_w, \delta_1')$	$(S_f \vee W_r, \delta_2')$	Cost
ω_1	0	0	0	0	0	0
ω_2	0	0	0	$\log(\delta_1')$	0	$\log(\delta_1')$
ω_3	0	$\log(\beta_1')$	0	0	$\log(\delta_2')$	$\log(\beta_1') + \log(\delta_2')$
ω_4	0	$\log(\beta_1')$	0	0	0	$\log(\beta_1')$
ω_5	α'	0	$\log(\beta_2')$	0	0	$\log(\alpha') + \log(\beta_2')$
ω_6	$\log(\alpha')$	0	$\log(\beta_2')$	$\log(\delta_1')$	0	$\log(\alpha') + \log(\beta_2')$
ω_7	$\log(\alpha')$	0	0	0	$\log(\delta_2')$	$\log(\alpha') + \log(\delta_2')$
ω_8	$\log(\alpha')$	0	0	0	0	$\log(\alpha')$

Table 4.2: Transformation to penalty weights of Example 4.4

4.3 π-Pref nets vs OCF-nets

We have explained that the expression of ordinal ranks parallels the product-based possibilistic chain rule, where weights are combined by the product operator. It was also proved that OCF networks share the same independence relation as possibilistic networks, namely, Markov independence [Eichhorn and Kern-Isberner, 2015, Eichhorn et al., 2016]. Therefore, it is clear that OCF networks with a rank interpretation has a close relationship with product-based π-pref nets. Indeed, the cost of a configuration induced by an OCF-net corresponds actually to a transformation of the possibility degree computed from a π-pref net. As mentioned before, the set-function $\pi_\kappa(A_i) = 2^{-\kappa(A_i)}$ is a possibility measure [Dubois and Prade, 1991]. The converse holds to some extent insofar as if $\pi(A_i) = \alpha$, the values $\kappa(A_i) = -\log_2(\alpha)$ are integer rank weights. However, we can also extend the OCF framework to positive reals. Up to this proviso, the ordering induced by the product-based chain rule of π-pref nets is the same as the order induced by the corresponding rank function. In [Ben Amor et al., 2015], it was proposed to use this transformation at the symbolic level, yielding a symbolic additive counterpart to π-pref nets.

Clearly, for a π-pref ΠG and an OCF network κG, $\pi_{\Pi G}(\omega) = \alpha_1 \cdot \cdots \cdot \alpha_N \Rightarrow \kappa_{\kappa G}(\omega) = -(\log_2(\alpha_1) + \cdots + \log_2(\alpha_N))$ and $\kappa_{\kappa G}(\omega) = (\alpha_1 + \cdots + \alpha_N) \Rightarrow \pi_{\Pi G}(\omega) = 2^{-\alpha_1} \cdot \cdots \cdot 2^{-\alpha_N})$. Thus, after the logarithmic transformation, OCF-nets yield the same ordering on configurations as π-pref nets. Note that π-pref nets with products cannot always be turned into OCF-nets with integer values. However, OCF-nets can be turned into π-pref nets with products.

Example 4.5. *Let us consider the following conditional rank tables corresponding to an OCF-net of two binary variables A and B: $\kappa(a) = 3$, $\kappa(\neg a) = 0$, $\kappa(b|a) = 0$, $\kappa(b|\neg a) = 2$, $\kappa(\neg b|a) = 1$ and $\kappa(\neg b|\neg a) = 0$. This yields $\kappa(\neg a\neg b) = 0 < \kappa(\neg ab) = 2 < \kappa(ab) = 3 < \kappa(a\neg b) = 4$. Thus we have $\neg a\neg b \succ_{\kappa G} \neg ab \succ_{\kappa G} ab \succ_{\kappa G} a\neg b$. The transformation from this OCF-net to a numeric π-pref net leads to the following possibilistic conditional tables: $\pi(a) = 0.125$, $\pi(\neg a) = 1$, $\pi(b|a) = 1$, $\pi(b|\neg a) = 0.25$, $\pi(\neg b|a) = 0.5$ and $\pi(\neg b|\neg a) = 1$ which yields $\pi(\neg a\neg b) = 1 > \pi(\neg ab) = 0.25 > \pi(ab) = 0.125 > \pi(a\neg b) = 0.0625$. Clearly the two models lead to the same ordering after this transformation.*

Until now, OCF-nets have been used for dealing with numerical values only. However, the transformation of a *symbolic* possibilistic network leads to a *symbolic* OCF-net. Thus, the application of the different ordering relations defined in Chapter 3 leads exactly to the same orderings induced by possibilistic networks. In fact, summation and product are handled similarly when we work in a symbolic setting.

4.4 π-Pref nets vs CP-nets

Recently, numerical OCF-nets have been shown to "mimic" the CP-net ordering [Eichhorn et al., 2016]. The proposed generation of an OCF-net from a CP-net leads to a total ordering, which

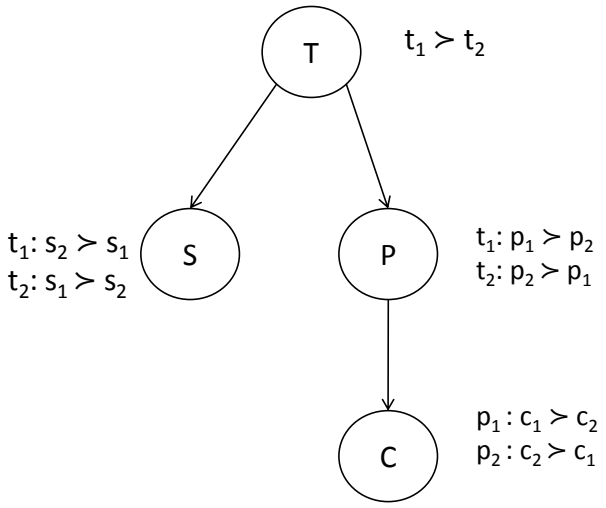


Figure 4.4: Preference network for Example 4.6

contrasts with CP-nets. However, they proved that such an ordering is always consistent with the one induced by the corresponding CP-net. They also showed that the CP-net formalism is able to represent only a subclass of OCF-nets, which proves that OCF-nets are more expressive than CP-nets. These remarks can be immediately applied to *numerical* π-pref nets as well.

Example 4.6. *Consider a preference specification about a holiday house in terms of* 4 *decision variables* $\mathcal{V} = \{T, S, P, C\}$ *standing for Type, Size, Place and Car park respectively, with values* $T \in \{flat\,(t_1),\ house\,(t_2)\}$, $S \in \{big\,(s_1),\ small\,(s_2)\}$, $P \in \{downtown\,(p_1),\ outskirt\,(p_2)\}$ *and* $C \in \{car\,(c_1),\ nocar\,(c_2)\}$. *Preference on* T *is unconditional, while all the other preferences are conditional as shown in Figure 4.4.*

These preference statements correspond to the CP-net of Figure 4.4. its induced worsening flip graph is on Figure 4.5. The preferences expressed by the CP-nets can be represented by a π*-pref net sharing the same graphical structure and where the conditional possibility distributions are as follows:* $\pi(t_1) = 1$, $\pi(t_2) = \alpha$, $\pi(p_1|t_1) = \pi(p_2|t_2) = 1$, $\pi(p_2|t_1) = \beta_1$, $\pi(p_1|t_2) = \beta_2$, $\pi(s_1|t_1) = \gamma_1$, $\pi(s_2|t_2) = \gamma_2$, $\pi(s_2|t_1) = \pi(s_1|t_2) = 1$, $\pi(c_1|p_1) = \pi(c_2|p_2) = 1$, $\pi(c_2|p_1) = \delta_1$ *and* $\pi(c_1|p_2) = \delta_2$. *Applying the product-based chain rule, we can compute the joint possibility distribution relative to T,P,C and S. Fig. 4.6 represents with thin arrows the configuration graph induced from the joint possibility distribution. Clearly, the configuration* $t_1p_1c_1s_2$ *is the root (since it is the unique one with degree* $\pi(t_1p_1c_1s_2) = 1$*).*

If we consider the preference statement at node P*, based on the preferential independence 1.6 and from the preference statement* t_1*:* $p_1 \succ p_2$*, we can deduce that* $t_1s_1p_1c_1 \succ_{CP} t_1s_1p_2c_1$, $t_1s_2p_1c_1 \succ_{CP} t_1s_2p_2c_1$, $t_1s_1p_1c_2 \succ_{CP} t_1s_1p_2c_2$ *and* $t_1s_2p_1c_2 \succ_{CP} t_1s_2p_2c_2$. *Indeed, we can deduce as many comparisons as the number of possible configurations of the variables other than*

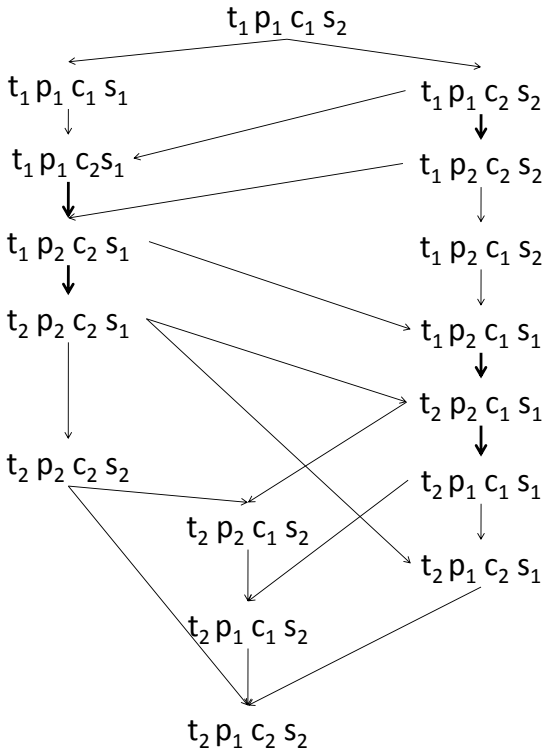


Figure 4.5: CP-net preferences for Example 4.4 up to transitive closure (5 bold arrows represent Ceteris Paribus preference relations that are not recovered by π-pref net, 8 one-flip comparisons over 32 can be recovered by transitivity, e.g. from $t_1p_1c_1s_2$ to $t_2p_1c_1s_2$).

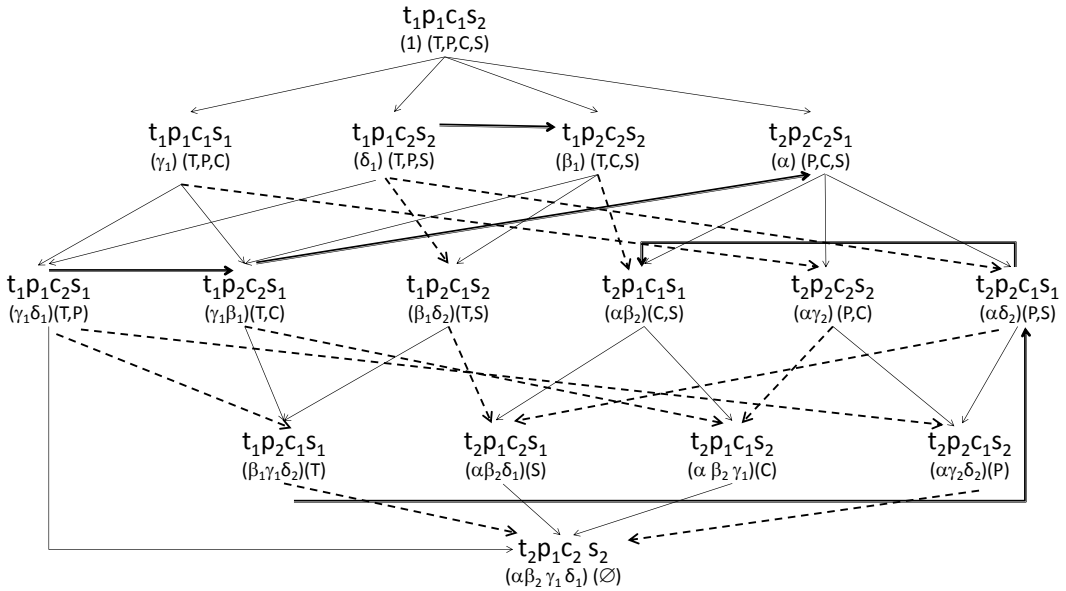


Figure 4.6: Configuration graph of Ex. 4.6. Thin arrows reflect $\succ_\pi$, dotted arrows compare sets $\mathcal{S}(\omega_i)$, and bold arrows reflect additional Ceteris Paribus comparisons recovered by the constraints, also in bold on Fig. 4.5

P and T, namely C and S. However, from the same statement, based on the possibilistic network, we can deduce that $t_1s_1p_1c_1 \succ_\pi t_1s_1p_2c_1$ and $t_1s_2p_1c_1 \succ_\pi t_1s_2p_2c_1$. This is due the fact that node S is independent of node P in the context of its parent T. Thus, the preference relation holds no matter the instantiation of S. Node C depends on D, thus, based on the context, we choose each time the best values for D.

From this simple example, we can see that π-pref nets and CP-nets do not share the same form of preference independence. Although both graphical networks are syntactically based on the same preference statements, they are semantically handled in different ways. More precisely, orderings in CP-nets are induced from *Ceteris Paribus* and transitivity, while orderings in π-pref nets are built using the chain rule and conditional preferences.

Indeed, if we consider the two preference (in)dependencies closely, we can notice that both have somehow contrasting properties. Let $Dn(A)$ be the set of descendants of the node A and let $Co(A) = V \setminus Dn(A) \setminus \mathcal{P}(A)$ be the set of A non descendants, their possible instantiations are denoted by d and n respectively. The conjunction of instantiations is denoted by xy such that $X \cap Y = \emptyset$ and $X, Y \subseteq V$. Let us consider a node A such that $D_A = \{a_1, a_2\}$ with this preference statement: $p(A) : a_1 \succ a_2$ where $p(A)$ an instantiation of $\mathcal{P}(A)$. In CP-nets setting and based on the *Ceteris Paribus* independence, we can deduce that $p(A)a_1dn \succ p(A)a_2dn$ (aside the instantiations of A, the rest of the variables have the same instantiation). In contrast, the same preference statement is handled differently by possibilistic networks and means that $\pi(a_1|p(A)) >$

$\pi(a_2|p(A))$. Therefore, we have $\pi(a_1|p(A)n) > \pi(a_2|p(A)n')$ thanks to the Markov properties of possibilistic networks, namely, each node is independent from its non-descendants in the context of it parents. Thus, in contrast with CP-nets, the preference is still preserved even if some variables, precisely, $Co(A)$ are configured differently. Moreover, we can see that based on *Ceteris Paribus* independence we have $Dn(A)$ instantiated similarly in both configurations, thus independently of A, which cannot be the case with possibilistic networks since $Co(A)$ depends on the instantiation of A.

In this section, we show that the configuration graph of any CP-net is consistent with the configuration graph of the π-pref net without local indifference, based on the same preference network, provided that some constraints on products of symbolic weights are added to the π-pref net, in order to restore the Ceteris Paribus priorities. Precisely, the added constraints reflect the higher importance of parent nodes with respect to their children. Under an additional property whose validity can be conjectured, π-pref net can capture CP-nets exactly.

4.4.1 Consistency between CP-nets and π-pref nets

In the following, we first recall that the ordering between configurations induced by a π-pref net corresponds to the Pareto ordering between the vectors $\vec{\omega} = (\theta_1(\omega), \ldots, \theta_N(\omega))$ where $\theta_i(\omega) = \pi(\omega_{A_i}|\omega_{\mathcal{P}(A_i)}), i = 1, \ldots, N$ (See Section 3.4). The Pareto ordering is defined by

$$\vec{\omega} \succ_{Pareto} \vec{\omega'} \text{ iff } \forall\, k,\ \theta_k(\omega) \geq \theta_k(\omega') \text{ and } \theta_i(\omega) > \theta_i(\omega') \text{ for some } i.$$

It is easy to see that $\theta_i(\omega) \in \{1, \alpha_{A_i|p(A_i)}\}$ where $\alpha_{A_i|p(A_i)}$ is the symbolic weight that appears in the preference table for variable A_i in the context $\omega_{\mathcal{P}(A_i)}$. It is easy to see that $\theta_k(\omega) > \theta_k(\omega')$ if and only if $\theta_k(\omega) = 1$ and $\theta_k(\omega')$ is a symbolic value. But it may be that $\theta_k(\omega)$ and $\theta_k(\omega')$ are distinct symbolic values, hence making ω and ω' incomparable. In particular, there are as many different symbolic weights $\alpha_{A|p(A)}$ pertaining to variable A as instantiations of parents of A. As symbolic weights are not comparable across variables, it is easy to see that the only way to have $\pi(\omega) \geq \pi(\omega')$ is to have $\theta_k(\omega) \geq \theta_k(\omega')$ in each component k of $\vec{\omega}$ and $\vec{\omega'}$. Otherwise the products will be incomparable due to the presence of distinct symbolic variables on each side. So,

$$\omega \succ_\pi \omega' \iff \vec{\omega} \succ_{Pareto} \vec{\omega'}$$

Given the ordinal nature of preference tables of CP-nets, it also makes sense to characterize the quality of ω using the set $\mathcal{S}(\omega) = \{A_i : \theta_i(\omega) = 1\}$ of satisfied preference statements (one per variable). It is then clear that the Pareto ordering between configurations induced by the preference tables is refined by comparing these satisfaction sets:

$$\vec{\omega} \succ_{Pareto} \vec{\omega'} \Rightarrow \mathcal{S}(\omega') \subset \mathcal{S}(\omega) \tag{4.2}$$

since if two configurations contain variables having bad assignments in the sense of the preference tables, the corresponding symbolic values may differ if the contexts for assigning a value to this variable differ.

Example 4.7. *To see that this inclusion-based ordering is stronger than the π-pref net ordering, consider Figure 4.6 where $\pi(t_1p_2c_1s_2) = \beta_1\delta_2$ with $\mathcal{S}(t_1p_2c_1s_2) = \{T, S\}$ and $\pi(t_2p_1c_2s_1) = \alpha\beta_2\delta_1$ with $\mathcal{S}(t_2p_1c_2s_1) = \{S\}$. We do have that $\mathcal{S}(t_1p_2c_1s_2) \supset \mathcal{S}(t_2p_1c_2s_1)$, but $\beta_1\delta_2$ is not comparable with $\alpha\beta_2\delta_1$. Dotted and thin arrows of Figure 4.6 represent the configuration graph induced by comparing sets $\mathcal{S}(\omega)$*

It is noticeable that if the weights $\alpha_{A_i|p(A_i)}$ reflecting the satisfaction level due to assigning the bad value to A_i in the context $p(A_i)$ do not depend on the context, then we have an equivalence in Equation (4.2):

Proposition 4.2. *If $\forall i = 1, \ldots, N, \alpha_{A_i|p(A_i)} = \alpha_i, \forall p(A_i) \in \mathcal{P}(A_i)$, then*

$$\vec{\omega} \succ_{Pareto} \vec{\omega'} \iff \mathcal{S}(\omega') \subset \mathcal{S}(\omega).$$

Proof. ($\Rightarrow$) This direction is proved by 4.2.
($\Leftarrow$) Suppose $\mathcal{S}(\omega') \subset \mathcal{S}(\omega)$ then if $A \in \mathcal{S}(\omega')$ we have $\theta_i(\omega) = \theta_i(\omega') = 1$; if $A \in \mathcal{S}(\omega) \setminus \mathcal{S}(\omega')$, then $\theta_i(\omega') = \alpha_i, \theta_i(\omega) = 1$ and $\theta_i(\omega') = \alpha_i = \theta_i(\omega)$ otherwise. This implies $\vec{\omega} \succ_{Pareto} \vec{\omega'}$. □

The inclusion-based ordering $\mathcal{S}(\omega') \subset \mathcal{S}(\omega)$ does not depend on the parent variables context but only on the fact that a variable has a good or a bad value. Similarly, when the symbolic weights no longer depend on parents instantiations, there is only one symbolic weight per variable. So, the above result is not surprising.

Example 4.8. *Using the same nodes as in Example 4.7, the unique weight assumption enforces $\beta_1 = \beta_2 = \beta$ and $\delta_1 = \delta_2 = \delta$, which yields $\pi(t_1p_2c_1s_2) = \beta\delta > \pi(t_2p_1c_2s_1) = \alpha\beta\delta$.*

In the following, we assume that the components of vector $\vec{\omega}$ are linearly ordered in agreement with the partial ordering of variables in the symbolic preference network, namely, if $i < j$ then A_i is not a descendant of A_j in the preference net (i.e. topological ordering). For instance in the preference net of Figure 4.4, we can use the ordering (T, P, C, S).

Let us first prove that, in the configuration graphs induced by a CP-net and the corresponding π-pref net, there cannot be any preference reversals between configurations. Let $\mathcal{C}h(A)$ denote the children set of $A \in \mathcal{V}$.

Lemma 4.1. *Let ω and ω' be two configurations such that $\omega \succ_{CP} \omega'$ and ω and ω' differ by one flip of a variable A_i then $\mathcal{S}(\omega) \subset \mathcal{S}(\omega')$ is not possible.*

Proof. Compare $\mathcal{S}(\omega)$ and $\mathcal{S}(\omega')$. It is clear that $A_i \notin \mathcal{S}(\omega')$ (otherwise the flip would not be improving) and $\mathcal{S}(\omega) = (\mathcal{S}(\omega') \cup \{A_i\} \cup \mathcal{C}h^+_-(A_i)) \setminus \mathcal{C}h^-_+(A_i)$, where $\mathcal{C}h^+_-(A_i)$ is the set of variables that switch from a bad to a good value when going from ω' to ω, and $\mathcal{C}h^-_+(A_i)$ is the set of variables that switch from a good to a bad value when going from ω' to ω. It is clear that it can never be the case that $\mathcal{S}(\omega) \subset \mathcal{S}(\omega')$, indeed A_i is in $\mathcal{S}(\omega)$ and not in $\mathcal{S}(\omega')$ by construction. But $\mathcal{S}(\omega')$ may contain variables not in $\mathcal{S}(\omega)$ (those in $\mathcal{C}h^-_+(A_i)$ if not empty). So either $\mathcal{S}(\omega') \subset \mathcal{S}(\omega)$ or the two configurations are not Pareto-comparable. □

In the following, given two configurations ω and ω', let $\mathcal{D}^{\omega,\omega'}$ be the set of variables which bear different values in ω and ω'.

Proposition 4.3. *If $\omega \succ_{CP} \omega'$ then $\mathcal{S}(\omega) \subset \mathcal{S}(\omega')$ is not possible.*

Proof. If $\omega \succ_{CP} \omega'$, then there is a chain of improving flips $\omega_0 = \omega' \prec_{CP} \omega_1 \prec_{CP} \cdots \prec_{CP} \omega_k = \omega$. Applying the above Lemma, $\mathcal{S}(\omega_i) = (\mathcal{S}(\omega_{i-1}) \cup \{V_{i-1}\} \cup \mathcal{C}h^+_-(V_{i-1})) \setminus (\mathcal{C}h^-_+(V_{i-1})$ for some variable $V_{i-1} = A_j$. By the above Lemma, we cannot have $\mathcal{S}(\omega_{i-1}) \subset \mathcal{S}(\omega_i)$. Suppose we choose the chain of improving flips by flipping at each step a top variable A_j in the preference net, among the ones to be flipped, i.e. $j = \min\{\ell : A_\ell \in \mathcal{D}^{\omega_{i-1},\omega}\}$. It means that when following the chain of improving flips, the status of each flipped variable will not be questioned by later flips, as no flipped variable will be a child of variables flipped later on. So $\mathcal{S}(\omega)$ will contain some variables not in $\mathcal{S}(\omega')$, so $\mathcal{S}(\omega) \subset \mathcal{S}(\omega')$ is not possible. □

The previous results show that it is impossible to have a preference reversal between CP-net ordering and the inclusion ordering, which implies that no preference reversal is possible between CP-net ordering and the π-pref net ordering. It suggests that we can try to add Ceteris Paribus constraints to a π-pref net so as to capture the preferences expressed by a CP-net.

As previously noticed, in CP-nets, parent preferences look more important than children ones. This property is not ensured by π-pref nets where all violations are considered having the same importance. Indeed, we can check from Figures 4.5 and 4.6 that the two configuration graphs built from the same preference statements of Example 4.6 are different. In the following, we highlight local constraints between each node and its children that enable Ceteris Paribus to be simulated. Let $D_{\mathcal{P}(A)} = \times_{A_i \in \mathcal{P}(A)} D_{A_i}$ denote the Cartesian product of domains of variables in $\mathcal{P}(A)$, $\alpha_{A|p(A)} = \pi(a^-|p(A))$ and $\gamma_{C|p(C)} = \pi(c^-|p(C))$.

Proposition 4.4. *Suppose a CP-net and a π-pref net built from the same preference statements. Let us add to the latter all constraints induced by the condition: $\forall\, A \in \mathcal{V}$ s.t. $\mathcal{C}h(A) \neq \emptyset$:*

$$\max_{p(A) \in D_{\mathcal{P}(A)}} \alpha_{A|p(A)} < \prod_{C \in \mathcal{C}h(A)} \min_{p(C) \in D_{\mathcal{P}(C)}} \gamma_{C|p(C)} \tag{4.3}$$

Let $\succ^+_\pi$ be the resulting preference ordering built from the preference tables and applying constraints between symbolic weights of the form of Equation 4.3, then, $\omega \succ_{CP} \omega' \Rightarrow \omega \succ^+_\pi \omega'$.

Proof. The relation $\succ_{CP}$ is determined by comparing configurations ω, ω' of the form $\omega = a^+ \wedge p(A) \wedge r$ and $\omega' = a^- \wedge p(A) \wedge r$ (where $R = \mathcal{V} \setminus (A \cup \mathcal{P}(A))$), that differ by one flip of variable A. So the local preference $p(A) : a^+ \succ a^-$ is equivalent to have $\omega \succ_{CP} \omega'$ under Ceteris Paribus assumption, and also equivalent to have $\pi(a^-|p(A)) < \pi(a^+|p(A)) = 1$ in the corresponding π-pref net based on the same preference tables.

Now let us show that $\forall A \in \mathcal{V}$, $\forall p(A) \in D_{\mathcal{P}(A)}$ and every instantiation r of the variables in $\mathcal{V} \setminus (\{A\} \cup \mathcal{P}(A))$, the local preference $\pi(a^-|p(A)) < \pi(a^+|p(A))$ implies $\pi(a^- \wedge p(A) \wedge r) < \pi(a^+ \wedge p(A) \wedge r)$ under the condition expressed by Equation (4.3). Consider the instantiation $ch(A) = \wedge_{C \in \mathcal{C}h(A)} \omega_C$, where $\omega_C \in \{c, \neg c\}$, of the children of A such that $ch(A) \wedge o = r$ (i.e. $O = \mathcal{V} \setminus (A \cup \mathcal{P}(\mathcal{A}) \cup \mathcal{C}h(A))$. The chain rule states (Equation (3.1)) :
$\pi(\omega') = \prod_{B \in \mathcal{V}} \pi(\omega'_B|\omega'_{\mathcal{P}(B)}) =$
$\pi(a^-|p(A)) \cdot \prod_{C \in \mathcal{C}h(A)} \pi(\omega'_C|p'(C)) \cdot \prod_{B \notin \{A\} \cup \mathcal{C}h(A)} \pi(\omega'_B|\omega'_{\mathcal{P}(B)})$.

Clearly the last term does not depend on A and is thus a constant β. So $\pi(\omega') = \beta \cdot \alpha_{A|p(A)} \cdot \prod_{C \in \mathcal{C}h(A)} \pi(\omega'_C|p'(C))$.
Likewise, since $\omega = a^+ \wedge p(A) \wedge ch(A) \wedge o$, we have
$\pi(\omega) = \beta \cdot \prod_{C \in \mathcal{C}h(A)} \pi(\omega_C|p(C))$, since $\pi(a^+|p(A)) = 1$. Note that while $p'(C)$ is of the form $a^- \wedge p_{-A}(C)$ where $\mathcal{P}_{-A}(C)$ is the set of parents of C but for A, $p(C)$ is of the form $a^+ \wedge p_{-A}(C)$.

So the inequality $\pi(\omega) > \pi(\omega')$, present in the CP-net, requires:

$$\prod_{C \in \mathcal{C}h(A)} \pi(\omega_C|p(C)) > \alpha_{A|p(A)} \cdot \prod_{C \in \mathcal{C}h(A)} \pi(\omega'_C|p'(C)).$$

Condition (4.3) implies $\alpha_{A|p(A)} < \prod_{C \in \mathcal{C}h(A)} \pi(\omega_C|p(C))$, which implies the above inequality. It proves that, under Condition (4.3), $\omega \succ_{CP} \omega'$ implies $\omega \succ^+_\pi \omega'$. □

This proposition ensures that the ordering induced by the joint possibility distribution of a π-pref net enhanced by constraints of the form (4.3) can refine the CP-net ordering having the same preference tables, provided that suitable constraints are added at each node $A \in \mathcal{V}$ between the local conditional possibility distribution at this node and the product of possibility degrees of the children of A. It comes down to constraints between each symbolic weight and a product of other ones. Indeed the less preferred value, $\min(\pi(a|p(A), \pi(\neg a|p(A))$, of A in the context of the parents $p(A)$ of A is a symbolic weight (non instantiated possibility degree). In other words, the inequality ensures that the less preferred value of each A given $p(A)$ is strictly less preferred than the product of the less preferred values of the children of A. This result is the symbolic counterpart of the one in [Eichhorn et al., 2016], using preference networks with numerical ranking functions.

Example 4.9. *In the graph of Figure 4.6 induced by the π-pref net of Example 4.6, Proposition 4.4 leads us to add conditions $\alpha < \min(\gamma_1, \gamma_2) \cdot \min(\beta_1, \beta_2)$ and $\max(\beta_1, \beta_2) < \min(\delta_1, \delta_2)$. Clearly these conditions are too strong here. First some of the products like $\gamma_1\beta_2$ never appear in Figure 4.6. Moreover, the reader can check that adding constraints $\beta_1\gamma_1 > \alpha$ and $\beta_i < \delta_i, (i = 1, 2)$ turns the configuration graph of Figure 4.6 into the CP-net-induced configuration graph of Figure 4.5.*

Instead of imposing priority of parents over children, we can also add the Ceteris Paribus constraints to the π-pref net directly, considering only worsening flips. Let ω, ω' differ by one flip, and such that none of $\omega \succ_\pi \omega'$, $\omega' \succ_\pi \omega$ holds, and moreover, $\omega \succ_{CP} \omega'$. We must enforce the condition $\pi(\omega) > \pi(\omega')$. Suppose the flipping variable is A. Clearly, $A \in \mathcal{S}(\omega)$, but $A \notin \mathcal{S}(\omega')$. Let α be the possibility degree of A when it takes the bad value in context $\omega_{p(A)}$ (it is 1 when it takes the good value). When flipping A from a good to a bad value, only the quality of the children variables $\mathcal{C}h(A)$ of A may change. $\mathcal{C}h(A)$ can be partitioned into at most 4 sets, $\mathcal{C}h^-_-(A)$ (respectively $\mathcal{C}h^-_+(A), \mathcal{C}h^+_-(A), \mathcal{C}h^+_+(A)$), which represents the set of children of A whose values remain bad (resp. change from good to bad, from bad to good, and stay good) when flipping A from a^+ to a^-. Strictly speaking these sets depend upon ω. Then it can be easily checked that:

$$\pi(\omega) = 1 \cdot \prod_{C_i \in \mathcal{C}h^+_-(A)} \gamma_i \cdot \prod_{C_j \in \mathcal{C}h^-_-(A)} \gamma_j \cdot \beta$$

$$\pi(\omega') = \alpha \cdot \prod_{C_k \in \mathcal{C}h^-_+(A)} \gamma_k \cdot \prod_{C_j \in \mathcal{C}h^-_-(A)} \gamma_j \cdot \beta$$

where β is a product of symbols, pertaining to nodes other than A and its children, that remain unchanged by the flip of A. Then the constraint $\pi(\omega) > \pi(\omega')$ comes down to the inequality:

$$\prod_{C_i \in \mathcal{C}h^+_-(A)} \gamma_i > \alpha \cdot \prod_{C_k \in \mathcal{C}h^-_+(A)} \gamma_k \tag{4.4}$$

where symbols appearing on one side do not appear on the other side. Such constraints are clearly weaker than Condition (4.3) but are sufficient to retrieve all the preferences of the CP-net. Note that the preferences $\omega \succ_\pi \omega'$ and $\omega \succ_{CP} \omega'$ conjointly hold in both approaches whenever A has no child node, and more generally whenever the worsening flip on A corresponds to no child variable moving from a bad to a good state, i.e. $\mathcal{C}h^+_-(A) = \emptyset$. In fact, condition (4.4) holds for all preference arcs in the configuration graph of the CP-net, whether this preference appears in the π-pref net or not. We get the following result.

Proposition 4.5. *Consider a CP-net and the preference relation $\succ^+_\pi$ on configurations built from the same preference tables by adding all constraints of the form (4.4) between configurations differing by one flip to the preferences of the form $\omega \succ_\pi \omega'$. Then:*

$$\omega \succ_{CP} \omega' \Rightarrow \omega \succ^+_\pi \omega'$$

Proof. Indeed, first the preferences according to $\succ_{CP}$ and $\succ_\pi$ do not contradict each other, per Proposition 4.3. Then we add constraints to the π-pref net for all CP-net worsening flips that are not captured by $\succ_\pi$, using constraints (4.4). So we have then captured the whole preference graph of the CP-net, plus possibly other preferences between configurations. □

In the transformation of a CP-net into a π-pref net, we keep the same graphical structure and the tables are filled directly from the preference statements of the CP-net. Besides, we must point out that, when mimicking CP-nets, constraints are not elicited from the user but computed directly from the graph structure.

Example 4.10. *The above constraints (4.4) that must be added to the π-pref configuration graph of Figure 4.6 are precisely those found to be necessary and sufficient in Example 4.9 to recover the CP-net ordering, i.e., $\alpha < \beta_1\gamma_1$, $\beta_1 < \delta_1$ and $\beta_2 < \delta_2$. Note that the number of of additional constraints to be added to capture the CP-net comparisons missed by the π-pref net is quite small. For instance, the number of constraints here is 4 against 120 potential comparisons.*

So, in the example, we exactly capture the preference graph of a CP-net using additional constraints between products of symbolic weights. The above considerations thus encourage us to study whether π-pref nets without constraints are *refined* by CP-nets, namely that the configuration graph of the former contains less strict preferences between configurations than the one of the latter, so that adding the constraints (4.4) are enough to simulate a CP-net by a π-pref net with constraints. Note that if it were not the case, it would mean that CP-nets do not respect Pareto-ordering.

4.4.2 Towards exact representations of CP-nets by π-pref nets

In this subsection, we consider the inclusion-ordering. One may wonder if there may exist some configurations that can be compared by the inclusion-based ordering, while they remain incomparable for CP-nets. This is not the case in our running example.

Example 4.11. *Consider the top configuration $\omega' = t_1p_1c_1s_2$ which inclusion-dominates $\omega = t_2p_2c_2s_1$ in the π-pref net configuration graph in Figure 4.6, since the former has good values for all variables and only the value t_2 is bad in the latter, i.e., $\mathcal{S}(\omega') = \{T, P, C, S\}$ and $\mathcal{S}(\omega) = \{P, C, S\}$. But the two configurations are far away in terms of flips since $\mathcal{D}^{\omega,\omega'} = \{T, P, C, S\}$. They can, however, be related by a chain of worsening flips. Namely, as $\mathcal{S}(\omega') \setminus \mathcal{S}(\omega) = \{T\}$, we must flip T first, and $\omega_1 = t_1p_2c_2s_1$, with $\mathcal{S}(\omega_1) = \{T, C\}$ so $\mathcal{S}(\omega') \setminus \mathcal{S}(\omega_1) = \{P, S\}$ and $\mathcal{D}^{\omega_1,\omega'} = \{P, C, S\}$. We now must flip P and get $\omega_2 = t_1p_1c_2s_1$ with $\mathcal{S}(\omega_2) = \{T, P\} = \mathcal{D}^{\omega_2,\omega'}$. As $\mathcal{S}(\omega') \setminus \mathcal{S}(\omega_2) = \{C, S\}$, we must flip C, and $\omega_3 = t_1p_1c_1s_1$, with $\mathcal{S}(\omega_3) = \{T, P, C\}$ so $\mathcal{S}(\omega') \setminus \mathcal{S}(\omega_3) = \{S\} = \mathcal{D}^{\omega_3,\omega'}$. We now must flip S and get $\omega_4 = t_1p_1c_1s_2 = \omega'$.*

The question whether the preference ordering of configurations induced by CP-nets is consistent with the ordering between the sets of variables that take good values in agreement with the preference tables seems to have been overlooked so far in the CP-net literature. The inclusion ordering between sets of variables with satisfactory values is intuitive in the sense that if a configuration ω violates all the preference statements violated by another configuration ω' plus some other(s), then ω' should indeed be strictly preferred to ω. The consistency of CP-nets with inclusion, namely the property

$$\mathcal{S}(\omega) \subset \mathcal{S}(\omega') \Rightarrow \omega' \succ_{CP} \omega \qquad (*)$$

can be naturally conjectured since the opposite case would cast a doubt on the rationality of such networks. Proposition 4.3 proves a weak consistency between them. However, at this stage providing a formal complete proof looks tricky and besides, is not directly related to the expressivity of

π-pref nets, the very topic of this section. The results in the following are conditioned by the truth of the conjecture, or are restricted to those CP-nets that agree with the inclusion-based orderings. Based on this assumption, Proposition 4.6 indicates that the CP-net ordering refines, hence is consistent with, the ordering induced by a π-pref net built from the same preference specification. This is because the inclusion-ordering refines the the Pareto (or π-pref net) ordering.

Proposition 4.6. *Consider a CP-net that refines the inclusion-based ordering and a π-pref net built from the same preference statements, we have:*

$$\omega' \succ_\pi \omega \Rightarrow \omega' \succ_{CP} \omega$$

Let us now prove that, if the conjecture (*) is valid, we are able to *exactly* induce the CP-net ordering from the π-pref net ordering by adding suitable constraints between symbolic weights or their products. First, we have seen that we can add to the π-pref net configuration graph all missing preference statements induced by the CP-net and not already present in the π-pref net configuration graph. These statements concern all pairs (ω, ω') that differ by one flip and such that $\pi(\omega)$ and $\pi(\omega')$ are not comparable. Note that adding such preference statements to the Pareto configuration graph in case of Pareto-incomparability yields the CP-net configuration graph (up to transitive closure).

The question remains whether we can express the latter in terms of additional constraints between symbolic weights or products thereof.

Proposition 4.7. *Consider a CP-net that refines the inclusion-based ordering and the preference relation $\succ_\pi^+$ on configurations built from the same preference tables by enforcing all constraints of the form (4.4) between configurations differing by one flip. Then:*

$$\omega \succ_{CP} \omega' \Leftrightarrow \omega \succ_\pi^+ \omega'$$

Proof. ($\Rightarrow$) This direction is proved by Proposition 4.5.
($\Leftarrow$) As $\omega \succ_\pi \omega' \Rightarrow \omega \succ_{CP} \omega'$ by assumption, adding Ceteris Paribus constraints corresponding to worsening flips to $\succ_\pi$ will not produce by transitivity any preference relation not in $\succ_{CP}$.

□

It is clear that, beside Ceteris Paribus constraints, other constraints could be added to a π-pref net, that cannot be expressed by a CP-net, and that account for different types of preference information. This fact suggests that π-pref nets with constraints have a better expressive power and are more flexible than CP-nets (known as a powerful qualitative model), and provide a general class of qualitative graphical models where the *Ceteris Paribus* ordering could be further refined without going numerical (i.e. unlike UCP-nets). It is clear therefore that the constraints added to refine this order should be consistent with *Ceteris Paribus*. Note, however, a π-pref net may represent distinct orderings when adding other constraints where similarly the consistency of these

constraints must, each time, be verified. Finally, π-pref nets are sometimes able to represent preference orderings when CP-nets fail to do it, as shown by the example below [Ben Amor et al., 2014].

Example 4.12. *Let us consider two binary variables A and B standing respectively for "vacations" and "good weather". Suppose that we have the following preference ordering: $ab \succ \neg a\neg b \succ a\neg b \succ \neg ab$. We observe that this complete preorder cannot be represented by a CP-net. In fact, given two variables we can define two possible structures: either A depends on B or conversely, both of them are unable to capture this order in the CP-net setting. This is due to the fact that in both structures we have a reversal of the Ceteris Paribus preferences. However, such preferences can be represented by a joint possibility distribution such that: $\pi(ab) > \pi(\neg a\neg b) > \pi(a\neg b) > \pi(\neg ab)$. Thus, we have $\top : a \succ \neg a$, $a : b \succ \neg b$ and $\neg a : \neg b \succ b$. It corresponds to a network with two nodes with their corresponding conditional possibility distributions: $\pi(a) = 1$, $\pi(\neg a) = \alpha$, $\pi(b|a) = 1$, $\pi(b|\neg a) = \gamma$, $\pi(\neg b|a) = \beta$ and $\pi(\neg b|\neg a) = 1$. This yields $\pi(ab) = 1 > \pi(\neg a\neg b) = \alpha > \pi(a\neg b) = \beta > \pi(\neg ab) = \alpha\gamma$ taking $\alpha > \beta$ and $\beta = \gamma$.*

Some extensions of CP-nets can be considered as akin to π-pref nets. TCP-nets [Brafman and Domshlak, 2002] also add priority constraints between variable nodes, that we can render in π-pref nets by inequalities between symbolic weights. Utility-enhanced CP-nets (UCP-nets) [Boutilier et al., 2001] add additive utility functions to CP-nets in order to encode total orderings consistent with the Ceteris Paribus assumption. To do so, linear constraints are added on utility values that are somewhat similar to constraints (4.4). They express that for any variable, given an instantiation of its parents, the utility gain in choosing the good value rather than the bad one in this context, should be more important than the maximum value of the sum of the possible utility loss for its children over all possible instantiations of the other related variables.

4.5 π-pref nets vs CP-theories

As mentioned in Chapter 1, CP-theories [Wilson, 2004, Wilson, 2011] is a set of formulas of the form $u : a \succ a'[W]$ where u is a an instantiation of a subset of variables $U \subset \mathcal{V}$, a and a' are different values of some variable A (not necessarily Boolean). From each formula of that form, one can deduce a preference relation between two sets of configurations. More precisely, each formula ϕ corresponds to some pairs of outcomes (ω_i, ω_j) such that each pair is intended to represent a preference for ω_i over ω_j. Informally, ϕ represents that given u and any z (i.e. $Z = \mathcal{V} \setminus (\{A\} \cup U \cup W)$), a is preferred to a', irrespective of the instantiations of W. Each preference formula allow locally partially ordered preferences. Indeed, it is not assumed to have a total order between the values of each variables. Moreover, more preference statement could be added downstream preference elicitation.

Clearly, CP-theories do not have any graphical structure but such model generalizes CP-nets and TCP-nets. Indeed, it represents preference statements that cannot be expressed by CP-nets

and TCP-nets. More precisely, a preference formula $u : a \succ a'[W]$ such that $W = \emptyset$ is CP-net preference statement, while if $|W| = 1$ then the formula represent an importance relation of A over W. Besides CP-theories are able to represent lexicographic orders when CP-nets and TCP-nets are proved to be unable to represent them [Wilson, 2011]. However, determining the consistency of CP-theories is a hard process. Testing consistency is based on search trees [Wilson, 2011] which consists on creating an upper approximation of ordering consistent with the (partial) order of CP-theories. the search trees used for this aim are strongly related to P-trees detailed in Chapter 1. Therefore, one can assume that these two preferential models are linked such that, built from a same preference specification, one can construct a total ordering by P-tree that refines the partial ordering of CP-theories provided that the CP-theories is consistent. To find the P-tree satisfying a particular CP-theories the following conditions should hold:

- For any preference formula $\phi : u : a \succ \neg a[W]$ in the CP-theories such that $D_A = \{a, \neg a\}$ and any configuration ω such that $\omega \models u$, each variable of U appears once before A on the path from the root to the implicit leaf ω and A appears necessarily before variables in W.
- For any node and any preference formula ϕ local preferences are consistent.

Note that these conditions are the same proposed for constructing the search trees [Wilson, 2004]

Example 4.13. *Let us consider a CP-theories on three variables $\mathcal{V} = \{A, B, C\}$. The preference formulas are: $a \succ \neg a[B, C]$, $a : \ c \succ \neg c[B]$, $a : b \succ \neg b[\emptyset]$, $\neg a : \ b \succ \neg b[C]$, $\neg a : \neg c \succ c[\emptyset]$. The corresponding P-tree is represented in Figure 4.7. Let us consider in particular the preference formula: $a : \ c \succ \neg c[B]$. For any configuration ω such that $\omega \models a$ we can check that node A precedes C which precedes B. This is confirmed for all the other formulas. Besides, we can check that local preferences are consistent. The total ordering induced by P-tree is: $abc \succ a\neg bc \succ ab\neg c \succ a\neg b\neg c \succ \neg ab\neg c \succ \neg abc \succ \neg a\neg b\neg c \succ \neg a\neg bc$*

At the best of our knowledge, the problem of dominance for CP-theories is not yet addressed.

π-pref nets can also be compared with CP-theories. The latter interpret conditional preference statements assuming they hold *irrespectively* of the values of other variables. It means that any configuration ω such that $\omega_A = a^+$ and $\omega_{\mathcal{P}(A)} = p(A)$ is preferred to any configuration ω such that $\omega_A = a^-$ and $\omega_{\mathcal{P}(A)} = p(A)$. In terms of possibility functions, it reads $\Delta(p(A) \wedge a^+) > \Pi(p(A) \wedge a^-)$, where $\Delta(\varphi) = \min_{\omega \models \varphi} \pi(\omega)$. In [Wilson, 2004] are studied hybrid nets where some variables are handled Ceteris Paribus, while the preference holds irrespectively of other variables. In π-pref nets preference statements are interpreted by $\pi(a^+|p(A)) > \pi(a^-|p(A))$ which is provably equivalent to $\Pi(p(A) \wedge a^+) > \Pi(p(A) \wedge a^-)$, i.e. comparing best configurations. It is clear that if $\omega \succ_{CP} \omega'$ holds, then $\omega \succ \omega'$ holds in a CP-theory, where conditional preference holds irrespectively of other variables, because the CP-theory generates more preference constraints between configurations, including the ones induced by the *Ceteris Paribus* assumption. Constraints induced by CP theories can thus be captured in π-pref nets by adding more constraints between products of symbolic weights.

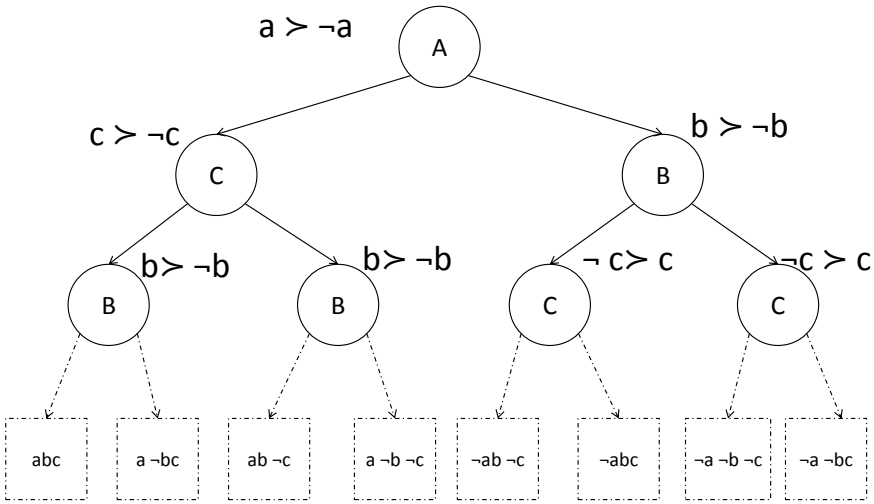


Figure 4.7: P-tree corresponding to the CP-theories of Example 4.13

4.6 π-Pref nets vs other models: General discussion

Figure 4.8 presents a classification of the preferential graphical models surveyed. Roughly speaking, there are two classes of graphical models: qualitative and quantitative models. However, one can distinguish models that are halfway namely OCF-networks and π-pref nets.

A summary of the main differences and similarities between models is given in Table 4.3.

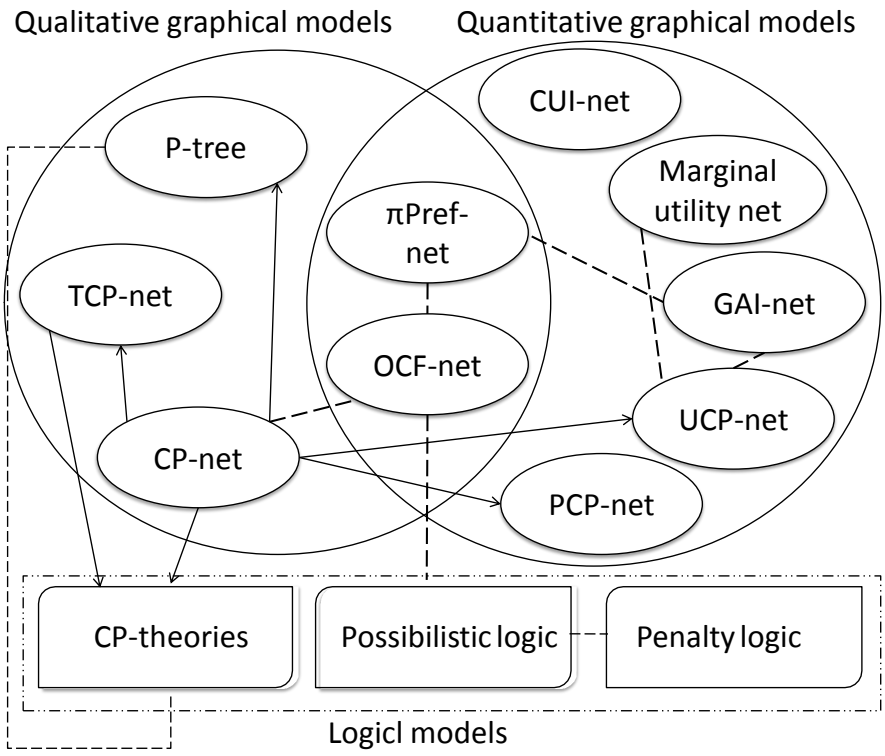


Figure 4.8: Classification of preferential graphical models (Continuous arrows point to extensions of CP-nets (including Probabilistic CP-nets (PCP-nets) and Multiple agents CP-nets (mCP-nets) not covered in this chapter since they enlarge the representation to other features, namely uncertainty or multiple agents. These models will be discussed in Chapter 5) and dashed lines corresponding to possible relations are discussed throughout the chapter)

Table 4.3: Summary table of the graphical preference representation models

Properties \ Model	**CP-nets**	**TCP-nets**	**GAI-nets**	**UCP-nets**	**Marginal utility nets**	**π Pref-nets**
Graphical component Node: Edges:	 Variable Directed	 Variable Directed	 Cliques Undirected	 Variable Directed	 Variable Directed	 Variable Directed
Preference table	Conditional pref. relation on variables	Cond. pref. relation + Importance relation	Utility functions	Conditional utility distribution	Conditional utility distribution	Conditional symbolic possibility distributions
Independence relation	*Ceteris Paribus*	*Ceteris Paribus*	Generalized Additive	*Ceteris Paribus* + GAI	Markovian	Markovian
Ordering	Partial	Partial	Total	Total	Total	Partial/ Total
Queries Complexity						
Optimization	Linear	Linear	Exponential	Linear	Unknown	Linear
Dominance	NP-complete to PSPACE	Unknown	Linear	Linear	Unknown	Linear to O(N!)

These models can further be compared in terms of the underlying independence relation (and expressiveness), and the ease of elicitation. Regarding the first issue, we distinguish three situations:

1. *Ceteris Paribus* independence is shared by CP-nets, and its extensions. Models based on it are unable to express all possible orderings between configurations. UCP-nets can represent some total orderings, at the expense of constraints added on utilities;
2. Generalized additive independence (GAI) used in GAI-nets, is a weaker form of independence leading to an improved expressive power;
3. Markov independence is used by π-pref nets, OCF-nets and marginal utility nets. In contrast with GAI, this kind of independence does not allow mutual dependencies between variables due to the acyclicity constraint.

Ceteris Paribus and Markov independence lead to different completion principles. With *Ceteris Paribus*, pairs of compared partial configurations are completed with the same instantiation of the rest of the variables, while with Markov-based nets, first we choose the best instantiation for all dependent variables, and next, instantiate the other variables in the same manner in all possible ways.

Regarding elicitation, although quantitative models are convenient for providing total orderings, they are not easy to assess (any difference in values may lead to different orderings). In contrast, eliciting qualitative models is easier since it suffices to provide contextual preference ordering. π-pref nets enable a progressive elicitation since we may add constraints between symbolic weights, or completely instantiate them.

Thanks to some resemblances between these models many transformations can be considered and are depicted by dashed lines in Figure 4.8. UCP-nets are a restriction of GAI-nets and a generalization of CP-nets. Indeed, a UCP-net structure can be transformed into a junction tree, using methods like [Huang and Darwiche, 1996], where to each clique we associate the sum of the local utilities of the variables belonging to it. However, due to the acyclic restriction of UCP-nets and the necessary commitment with *Ceteris Paribus*, not any GAl-net can be represented by a UCP-net.

The construction of junction tree from any DAG structure can be performed via the following three steps [Huang and Darwiche, 1996]:

- Moralization of the initial DAG $\mathcal{G}$: corresponding to creating an undirected graph $\mathcal{UG}$ from $\mathcal{G}$ by dropping the directions of the arcs. Then, creating $\mathcal{MG}$ from $\mathcal{UG}$ by connecting the parent set of each nodes (by adding edges to $\mathcal{UG}$).
- Triangulation of the moral graph: is defined by finding in each time the node with smallest set of adjacent nodes, adding edges and then constructing the clusters. Note that it is possible to have different triangulations of a moral graph. Finding the optimal triangulation is

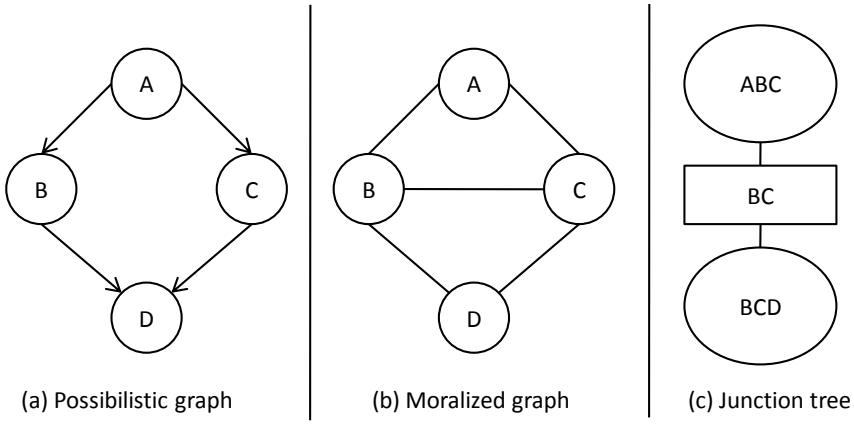


Figure 4.9: Example of transformation from a DAG to a junction tree

not obvious and needs greedy polynomial heuristic [Kjærulff, 1990]. The task of finding an optimal triangulation is stated as an *NP-complete* problem [Cooper, 1990].

- Building a junction tree from the triangulated moral graph comes down to connect the clusters created in the previous task with separators. These separators contain the variables in common between two adjacent clusters.

Note that steps 2 and 3 are *non-deterministic* which explains the fact that different junction trees can be built from the same DAG. These steps are illustrated by the following example.

Example 4.14. *Let us consider the directed acyclic graph of Figure 4.9(a) representing the graphical component of an UCP-nets. The construction of the moral graph of Figure 4.9(b) corresponds to dropping the direction, then relating the nodes B and C by an edge since they are parents of D. Triangulation yields two cliques ABC and BCD and one seperator BC represented by Figure 4.9(c). The utility table associated to ABC is computed as the sum of utilities in nodes A, B and C. More precisely, for any instantiations a, b and c of nodes A, B and C respectively, we have $u(abc) = u(a) + u(b|a) + u(c|a)$.*

Besides, when handled symbolically, π-pref nets and marginal utility nets lead to the same orderings. Indeed comparing configuration vectors using product or addition makes no difference on symbolic weights. Transformation from π-pref nets to GAI-nets might also be considered since, as for Bayesian nets, possibilistic nets can be translated into junction trees. However, an important difference between these two settings lies in the meaning of values. Both utilities and possibility degrees express levels of satisfaction, but the latter are bounded. In GAI-nets, what really matters is the difference between utilities. Thus, representing the same information in π-pref nets is not possible; one may only try to induce the same qualitative order between the configurations. The opposite transformation requires two steps. First, translating utilities to possibility degrees. Second, moving from a junction tree to a possibilistic network. Such a procedure has never been studied in the literature.

As can be seen, the advantages of the different models are a matter of trade-off. One may prefer one or another depending on the level of information available, the expressiveness needed for the situation at hand, and the time available for eliciting preferences. From a computational viewpoint, UCP-nets, instantiated π-pref nets and OCF-nets are the less demanding. On the other hand, elicitation and construction might be onerous for UCP-nets, GAI-nets and TCP-nets, while CP-nets and π-pref nets (with symbolic weights) are easy to elicit. Getting a total order may also be considered as important. Thus, one may prefer models such as GAI-nets, OCF-nets and instantiated π-pref nets in that respect.

4.7 Conclusion

In this chapter, we have explored the expressive power of π-pref nets and compared them to several preferential models. First, we have proposed to adapt the method proposed in [Benferhat et al., 2002a] for the transformation from possibilistic networks to logic bases in the symbolic case. Then, we have proved that the CP-net orderings cannot contradict those of the π-pref nets and we found suitable additional constraints to refine π-pref net orderings in order to capture Ceteris Paribus constraints of CP-nets. CP-nets would then be exactly captured by π-pref nets with constraints. This indicates that CP-nets potentially represent a subclass of π-pref nets with constraints. Besides, we have discussed possible transformations between preferential models namely, π-pref nets and OCF-nets and some others.

In many situations, one may need to aggregate users preferences in order to have a syntactic view of the preferences of a group of agents. Next chapter discusses an extension of π-pref nets to multiple agents preferences where each subset of agents is defined by particular profile.

CHAPTER 5

Graphical Representations of Multiple Agent Preferences

Contents

5.1 Introduction

Only a few graphical models have been proposed for modelling *multiple agent* preferences, they are based on different extensions of Conditional Preference networks (CP-nets) [Rossi et al., 2004, Bigot et al., 2013], or Generalized Additive Independence networks (GAI-nets) [Dubus et al., 2009]. Besides, a multiple agent logic [Belhadi et al., 2013], where formulas are pairs of the form (p, P) made of a proposition p and a subset of agents P, has been advocated for handling beliefs. In fact, (p, P) means '(at least) all agents in P believe that p is true'. But (p, P) may also have a preference reading ('(at least) all agents in P want p to be true').

The strong similarity of multiple agent logic with possibilistic logic and the existence of transformations between possibilistic logic and possibilistic networks [Benferhat et al., 2002a] suggest

to develop a graphical counterpart to multiple agent logic. When modelling preferences, multiple agent networks can be seen as a generalization of individual π-pref nets (when possibility degrees are binary valued). In the following we investigate the interest of multiple agent networks (and of their graded extension) for handling preferences.

This chapter is organized as follows. Section 5.2 defines conditioning in the case of Boolean possibilities. Section 5.3 introduces multiple agent logic, and its graphical counterpart in a preference perspective. Section 5.4 presents the main steps for transforming one format into another. Section 5.5 discusses queries evaluation for multiple agent networks. Section 5.6 presents an extension with priority levels of multiple agent logic and network. Section 5.7 reviews some related work.

5.2 Conditioning and possibilistic networks: Boolean case

Conditioning is a crucial notion when dealing with possibilistic networks. Here we consider the elementary situation of a single agent and of two-valued possibility distributions. Possibilistic networks [Benferhat et al., 2002a], when valued on $[0, 1]$, are usually defined for non-dogmatic possibility distributions, which means that the distribution is *never* equal to 0. However, in the two-valued case, the only non-dogmatic possibility distribution is the vacuous one with value 1 for all states. So we must use a definition of conditioning that makes sense for dogmatic possibility distributions.

Conditioning in this case is defined in the following way: having Ω the universe of discourse (set of all configurations). Then the configurations known as possible are restricted by a subset $E \neq \emptyset$, $E \subset \Omega$, and the considered possibility measure Π is such that $\Pi(T) = 1$ if $E \cap T \neq \emptyset$ and $\Pi(T) = 0$ otherwise (the possibility distribution being the characteristic function of E). Conditioning obeys the equation:

$$\Pi(T \cap S) = \Pi(T|S) \wedge \Pi(S) \tag{5.1}$$

where $\wedge$ is a Boolean conjunction. Then we define $\Pi(\cdot|S)$ is the possibility measure associated with the subset $E_S = S \cap E$ if $S \neq \emptyset$ and $E_S = S$ if $S \cap E = \emptyset$. E_S is the result of revising E by S, the minimally specific solution of the above equation under the success postulate $E_S \subseteq S$. Thus:

$$\begin{aligned}\Pi(T|S) = 1 \text{ if } &\begin{cases} T \cap E_S = T \cap S \cap E \neq \emptyset \ (\Pi(T \cap S) = \Pi(S) = 1) \\ T \cap E_S = T \cap S \neq \emptyset, \ S \cap E = \emptyset \ (\Pi(T \cap S) = \Pi(S) = 0) \end{cases} \\ = 0 \quad &\begin{cases} \text{otherwise } (\Pi(T \cap S) = 0, \Pi(S) = 1) \end{cases}\end{aligned}$$

A Boolean possibility distribution can be decomposed into a combination of conditional possibility distributions. This can be done by applying repeatedly the definition of conditioning. Indeed, taking an arbitrarily order of variables in set $\mathcal{V} = \{A_1, \ldots, A_N\}$: $\pi(A_1, \ldots, A_N) =$

ABC	$\pi(C\|B)$	$\pi(B\|A)$	$\pi(A)$	$\pi(ABC)$
$\neg a\neg b\neg c$	1	1	0	0
$\neg a\neg bc$	1	1	0	0
$\neg ab\neg c$	1	1	0	0
$\neg abc$	1	1	0	0
$a\neg b\neg c$	1	0	1	0
$a\neg bc$	1	0	1	0
$ab\neg c$	1	1	1	1
abc	1	1	1	1

Table 5.1: Recovering the original knowledge from conditional distributions of Example 5.1

$\pi(A_1|A_2,\ldots,A_N) \wedge \cdots \wedge \pi(A_N)$. This decomposition can be simplified when assuming some independence between variables. Graphically, it can be represented by a possibilistic network where each node represents a variable, edges represent the dependencies and conditional distributions define the associated tables.

Example 5.1. *Consider three Boolean variables A, B, C and the possibility distribution defined by the two configurations of $a \wedge b$. Let us construct a possibilistic network $\mathcal{G}_1$ associated to the ordering (A, B, C) corresponding to this possibility distribution:*

- *We compute $\pi(C|A,B)$: $\pi(c|ab) = \pi(\neg c|ab) = \pi(c|\neg ab) = \pi(\neg c|\neg ab) = \pi(c|a\neg c) = \pi(\neg c|a\neg b) = \pi(c|\neg a\neg b) = \pi(\neg c|\neg a\neg b) = 1$.*
- *We check $\pi(c|b) = \pi(\neg c|b) = \pi(c|\neg b) = \pi(\neg c|\neg b) = 1$, so $\pi(C|B,C) = \pi(C|B)$.*
- *We compute $\pi(B|A)$: $\pi(b|a) = 1, \pi(\neg b|c) = 0, \pi(b|\neg a) = \pi(\neg b|\neg a) = 1$, and $\pi(a) = 1,\ \pi(\neg a) = 0$.*

It corresponds to a a possibilistic network $A \to B \to C$, where C is independent from A given B.

The original knowledge ab can be recovered by the chain rule as follows (Table 5.1):

Let us consider another possibilistic network $\mathcal{G}_2$ such that $A \to B \to C$ with three binary variables, with their corresponding conditional possibility distributions: $\pi(a) = 1,\ \pi(\neg a) = 0$, and

$\pi(C\|B)$	b	$\neg b$
c	1	0
$\neg c$	1	1

$\pi(B\|A)$	a	$\neg a$
b	0	0
$\neg b$	1	1

We can check that $\pi(c|b) = 1$ *while* $\pi(bc) = \pi(b) = 1$ *and* $\pi(b|\neg a) = 1$ *while* $\pi(\neg ab) = \pi(\neg a) = 0$. *This illustrates the two first cases above in the definition of* $\Pi(T|S)$.

Let us reconstruct the possibility distribution using the chain rule (Equation 3.1):

ABC	$\pi(C\|B)$	$\pi(B\|A)$	$\pi(A)$	$\pi(ABC)$
$\neg a\neg b\neg c$	1	0	0	0
$\neg a\neg bc$	0	0	0	0
$\neg ab\neg c$	0	1	0	0
$\neg abc$	1	1	0	0
$a\neg b\neg c$	1	0	1	0
$a\neg bc$	0	0	1	0
$ab\neg c$	1	1	1	1
abc	1	1	1	1

We can make several observations from these examples

- The two networks $\mathcal{G}_1$ and $\mathcal{G}_2$ are different but correspond to the same possibility distribution.
- The first network $\mathcal{G}_1$ has conditional distributions less specific than the second one. This is not surprising as the conditioning rule is based on minimum specificity.
- In the second network $\mathcal{G}_2$, there is actually no link from A to B since $\pi(B|A)$ does not depend on A. So having fixed the ordering of variable, not only the conditional tables are not unique, but even the network topology is not unique. So we have that $\pi(A, B, C) = \min(\pi(C|B), \pi(B), \pi(A))$, with $\pi(b) = 1, \pi(\neg b) = 0$.

5.3 Multiple agent representations

Multiple agent logic has been discussed in details in [Belhadi et al., 2013]. Formulas in this logic are pairs of the form (p, P), made of a proposition p and a subset of agents P. In this section, we explain the use of this logic for modelling preferences and present its graphical counterpart. All will denote the set of all the agents and capital letters, e.g., $P_i, \cdots$ denote subsets of All. Let lowercases e.g. p, q, p_i denote propositional formulas of a finite language.

5.3.1 Multiple agent logic

As detailed in Section 4.2.1.2, a possibilistic logic formula [Dubois and Prade, 2004] of the form (p, α) is understood as $N(p) \geq \alpha$ (N is a necessity degree), where the higher α, the more imperative p. Multiple agent logic shares formal similarity with possibilistic logic in terms of inference

rules, axioms, possibilistic measures and possibility distribution [Belhadi et al., 2013]. However, a multiple agent formula (p, P) is understood at the semantic level as a constraint of the form $\boldsymbol{N}(p) \supseteq P$ where $\boldsymbol{N}$ is a *set-valued* mapping that returns the set of agents for whom satisfying p is imperative. Therefore, the formula (p, P) means that at least all the agents in P find p imperative. Set-valued possibility measure and necessity measure are related via duality. Indeed, $\boldsymbol{\Pi}(p) = \overline{\boldsymbol{N}(\neg p)}$, which corresponds to the maximal set of agents for whom the falsity of p is not imperative, which could be expressed as "the truth of p is acceptable". $\boldsymbol{\Pi}(p) \cap \boldsymbol{\Pi}(\neg p)$ represents the set of agents that are indifferent to the truth value of p, and $\boldsymbol{N}(p) \cap \boldsymbol{N}(\neg p)$ represents a set of inconsistent agents, which may be empty or not. It can be checked that the set of agents who think that the truth of p is imperative is a subset of the set of agents who think that its falsity is not imperative, namely, $\boldsymbol{N}(p) \subseteq \boldsymbol{\Pi}(p)$ provided there is no inconsistent agent. The semantics of such a logic is defined by a so-called ma-distribution from a universe of discourse Ω to subsets of agents, formally, $\boldsymbol{\pi} : \Omega \rightarrow 2^{All}$. Subsets are partially ordered, which contrasts with a possibilistic logic distribution that maps to a totally ordered scale. A multiple agent formula (p_i, P_i) leads to the following semantic representation by the ma-distribution

$$\boldsymbol{\pi}_{(p_i,P_i)}(\omega) = \begin{cases} All & if\ \omega \models p_i \\ \overline{P_i}\ (= All \setminus A_i) & otherwise. \end{cases} \tag{5.2}$$

This expression indicates that agents not in P_i are indifferent to p_i, but agents in P_i find $\neg p_i$ unacceptable More generally an ma-distribution should be interpreted as follows: $\boldsymbol{\pi}(\omega)$ is the set of all agents that do not find ω unacceptable

A ma-logic base $\Gamma = \{(p_i, P_i) | i = 1, m\}$ is associated to an ma-distribution, s.t. $\boldsymbol{\pi}_\Gamma(\omega)$ is the intersection of sets of agents $\overline{P_i}$ that find the configuration ω, for which all formulas p_i are false, acceptable.

$$\boldsymbol{\pi}_\Gamma(\omega) = \begin{cases} All & \text{if } \forall (p_i, P_i) \in \Gamma, \omega \models p_i \\ \bigcap \{\overline{P_i} : (p_i, P_i) \in \Gamma,\ \omega \models \neg p_i\} & otherwise. \end{cases} \tag{5.3}$$

Two types of normalization exist for $\boldsymbol{\pi}$:

- The *ma-normalization* where $\exists\, \omega \in \Omega$ s.t. $\boldsymbol{\pi}(\omega) = All$. Thus, all agents are altogether consistent and have at least one common not unacceptable configuration. This normalization entails the following one.
- The *i-normalization* where $\bigcup \{\boldsymbol{\pi}(\omega), \omega \in \Omega\} = All$. This means that each agent is consistent individually by having at least one configuration that is not rejected. Yet, there may exist some contradictions between subgroups of agents, for instance $\Gamma = \{(p, P), (\neg p, \overline{P})\}$.

Example 5.2. *Let us consider preferences of subsets of agents about drinks and their accompaniments. We consider that the agent population is described by two characteristics namely, being a woman* (W) *or a man* (M) *and being young* (Y) *or old* (O)*. The variables are* $Drink = \{Tea(t),\ Coffee(\neg t)\}$, $Sugar = \{Yes(s),\ No(\neg s)\}$, $Cake = \{Yes(c),\ No(\neg c)\}$.

Let us consider the following ma-base: $\Gamma = \{(\neg t, M), (t, \overline{M \cap Y}), (\neg s, O), (s, Y)\}$. *We can check that the ma-normalization is not verified. This is because* $\boldsymbol{N}(t) \supseteq M$ *and* $\boldsymbol{N}(\neg t) \supseteq \overline{M \cap Y}$, *hence* $\boldsymbol{N}(t) \cap \boldsymbol{N}(\neg t) \supseteq \overline{M \cap Y} \cap M = M \cap O$. *The old men demand tea and not tea.*

5.3.2 Graphical representation of multiple agent preferences

Possibilistic networks are the graphical counterpart of possibilistic logic and one may go from one format to another while preserving semantics [Benferhat et al., 2002a]. Likewise, given the close similarity between possibilistic and multiple agent logic, we propose a graphical reading of the latter. First, we introduce the multiple agent conditioning rule:

$$\boldsymbol{\pi}(p \wedge q) = \boldsymbol{\pi}(p|q) \cap \boldsymbol{\pi}(q) \tag{5.4}$$

This means that the set of all agents for whom the truth of $p \wedge q$ is not unacceptable is equal to the intersection between the set of all agents for whom the truth of q is not unacceptable and the set of all agents for whom the truth of p is not unacceptable when q is true. It generalizes the conditioning of Boolean possibilities to multiple agents. As in standard possibilistic networks, the decomposition of a possibility distribution consists in expressing a joint possibility distribution as a combination of conditional possibility distributions, a process that in the two-valued possibility case, does not yield a unique result, even fixing the ordering of the variables, as shown above.

Let E be a subset of $\mathcal{A} \times \Omega$ representing an ma-distribution $\boldsymbol{\pi}$, and let $E(a)$ the set of configurations that agent $a \in \mathcal{A}$ does not reject. The result of conditioning E by a set of configurations B will be again defined as the minimally specific revision of $E(a)$ by B that agrees with the definition of conditioning (5.4), for each agent $a \in \mathcal{A}$, namely $E_B(a) = E(a) \cap B$ if this intersection is not empty and B otherwise. Note that the result differs from $E \cap (\mathcal{A} \times B)$ even if this set is not empty. If B contains the set of models $[q]$ of q, then the characteristic function of E_B is denoted by $\boldsymbol{\pi}(\cdot|q)$. The solution to equation (5.4) is then as follows:

$$\boldsymbol{\pi}(p|q) = \begin{cases} All & \text{if } \boldsymbol{\pi}(p \wedge q) = \boldsymbol{\pi}(q) \\ \boldsymbol{\pi}(p \wedge q) & \text{otherwise.} \end{cases} \tag{5.5}$$

We can follow the same procedure for the multiple agent model. Let $\mathcal{V} = \{A_1, \ldots, A_N\}$ be a set of variables, each variable A_i has a value domain D_{A_i}. a_i denotes any value of A_i. In coherence with Equation 5.4, we can use the chain rule:

$$\boldsymbol{\pi}(A_1, ..., A_N) = \boldsymbol{\pi}(A_1|A_2, ..., A_N) \cap .. \cap \boldsymbol{\pi}(A_{N-1}|A_N) \tag{5.6}$$

to decompose a joint ma-distribution into a conjunction of conditional possibility distributions. Now, we introduce a new graphical model for representing multiple agent preferences, called ma-net for short. This model shares similar graphical component and independence relations as possibilistic networks [Benferhat et al., 2002a]. Formally,

$\boldsymbol{\pi}(t)$	$\boldsymbol{\pi}(\neg t)$
W	$M \cap Y$

$\boldsymbol{\pi}(s)$	$\boldsymbol{\pi}(\neg s)$
Y	O

$\boldsymbol{\pi}(.\|.)$	ts	$t\neg s$	$\neg ts$	$\neg t\neg s$
c	$M \cap Y$	O	$M \cap Y$	W
$\neg c$	$M \cap O$	W	W	$M \cap O$

Table 5.2: Conditional distributions corresponding to the network of Figure 5.1

Definition 5.1 (ma-net)**.** *A multiple agent network $\mathcal{G}$ over a set of variables $\mathcal{V}$ consists of two components:*

- *Graphical component composed of a directed acyclic graph (DAG).*
- *Numerical component associating to each node A_i a conditional multiple agent distribution for each the context $p(A_i)$ of its parents $\mathcal{P}(A_i)$.*

Example 5.3. *Using the same variables and sets of agents as in Example 5.2, the network of Figure 5.1 and its associated conditional distributions in Table 5.2 represent conditional preferences of agents. Using the chain rule (Equation 5.6), we have the following ma-distribution:* $\boldsymbol{\pi}(tsc) = \emptyset$, $\boldsymbol{\pi}(ts\neg c) = W \cap Y \cap O = \emptyset$, $\boldsymbol{\pi}(t\neg sc) = W \cap O$, $\boldsymbol{\pi}(t\neg s\neg c) = W \cap O$, $\boldsymbol{\pi}(\neg tsc) = M \cap Y$, $\boldsymbol{\pi}(\neg ts\neg c) = M \cap Y \cap W = \emptyset$, $\boldsymbol{\pi}(\neg t\neg sc) = M \cap Y \cap O \cap W = \emptyset$, $\boldsymbol{\pi}(\neg t\neg s\neg c) = M \cap Y \cap O \cap M \cap O = \emptyset$. *In ma-logic, we can encode it by the following base:* $\{(t \wedge \neg s) \vee (\neg t \wedge s \wedge c), All), (\neg t \vee s, M \cup Y), (t \vee \neg s \vee \neg c, W \cup O)\}$

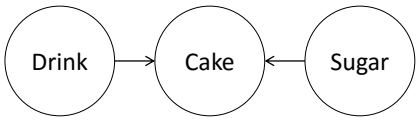


Figure 5.1: DAG of Example 5.3

Let us reconstruct the ma-conditional distributions $\boldsymbol{\pi}(C|T,S)$ and the marginals $\boldsymbol{\pi}(T), \boldsymbol{\pi}(S)$ using the conditioning rule:

- $\boldsymbol{\pi}(tsc) = \boldsymbol{\pi}(ts\neg c) = \boldsymbol{\pi}(ts) = \emptyset$ *so* $\boldsymbol{\pi}(c|ts) = \boldsymbol{\pi}(\neg c|ts) = All$.
- $\boldsymbol{\pi}(t\neg sc) = \boldsymbol{\pi}(t\neg s\neg c) = \boldsymbol{\pi}(t\neg s) = W \cap O$ *so* $\boldsymbol{\pi}(c|t\neg s) = \boldsymbol{\pi}(\neg c|t\neg s) = All$
- $\boldsymbol{\pi}(\neg tsc) = \boldsymbol{\pi}(\neg ts) = M \cap Y$ *so* $\boldsymbol{\pi}(c|\neg ts) = All$. *But* $\boldsymbol{\pi}(\neg ts\neg c) = \emptyset, \boldsymbol{\pi}(\neg ts) = M \cap Y$ *so* $\boldsymbol{\pi}(\neg c|\neg ts) = \emptyset$.
- $\boldsymbol{\pi}(\neg t\neg sc) = \boldsymbol{\pi}(\neg t\neg s\neg c) = \boldsymbol{\pi}(\neg t\neg s) = \emptyset$ *so* $\boldsymbol{\pi}(c|\neg t\neg s) = \boldsymbol{\pi}(\neg c|\neg t\neg s) = All$.
- *It can be checked that* $\boldsymbol{\pi}(s) = M \cap Y$, $\boldsymbol{\pi}(\neg s) = W \cap O$, $\boldsymbol{\pi}(t) = W \cap O$, $\boldsymbol{\pi}(\neg t) = M \cap Y$.

$\boldsymbol{\pi}(t)$	$\boldsymbol{\pi}(\neg t)$
$W \cap O$	$M \cap Y$

$\boldsymbol{\pi}(s)$	$\boldsymbol{\pi}(\neg s)$
$M \cap Y$	$W \cap O$

$\boldsymbol{\pi}(.\|.)$	ts	$t\neg s$	$\neg ts$	$\neg t\neg s$
c	All	All	All	All
$\neg c$	All	All	$\emptyset$	All

Table 5.3: Recovered conditional distributions corresponding the network of Figure 5.1

TSC	$\pi(C\|TS)$	$\pi(T)$	$\pi(S)$	$\pi(TSC)$
111	All	$W \cap O$	$M \cap Y$	$\emptyset$
110	All	$W \cap O$	$M \cap Y$	$\emptyset$
101	All	$W \cap O$	$W \cap O$	$W \cap O$
100	All	$W \cap O$	$W \cap O$	$W \cap O$
011	All	$M \cap Y$	$M \cap Y$	$M \cap Y$
010	$\emptyset$	$M \cap Y$	$M \cap Y$	$\emptyset$
001	All	$M \cap Y$	$W \cap O$	$\emptyset$
000	All	$M \cap Y$	$W \cap O$	$\emptyset$

Table 5.4: New conditional distributions yielding the same possibility distribution

This network has different conditional tables from the ones of Table 5.2, but it yields again the same possibility distribution. We can check that the chain rule applied to above network gives back the same ma-distribution (Table 5.4) as the one of Table 5.2.

5.4 Bridging logical and graphical multiple agent representations

As proposed in Section 4.2.1 for π-pref nets, transformations between possibilistic graphical and logical representations [Benferhat et al., 2002a] can be also adapted to multiple agent representations.

5.4.1 Logical encoding of a multiple agent network

The main idea of the logical encoding of ma-nets consists in considering the ma-net $\mathcal{G}$ as a combination of local multiple agent logic bases. Each node $A_i \in \mathcal{V}$ is associated to a logic base as follows:

$$\Gamma_{A_i} = \{(\neg a_i \vee \neg p(A_i), \overline{P}) \mid \boldsymbol{\pi}(a_i|p(A_i)) = P \text{ in the tables of } \mathcal{G} \text{ and } P \neq All\}$$

where a_i is an instantiation of A_i and $p(A_i)$ an instantiation of $\mathcal{P}(A_i)$. Each (conditional) possibility is viewed as a necessity formula expressing the material counterpart of the condition. Indeed, for a single agent $N(\neg p|q) = 1 - \Pi(p|q) = 1 - \Pi(p \wedge q) = N(\neg q \vee \neg p) = 1$ provided that $\Pi(p|q) = 0$. So, in the multiagent case we can replace $\mathbf{\Pi}(a_i|p(A_i))$ by the clause $\neg a_i \vee \neg p(A_i)$ when $P \neq All$. When considered separately, we can see that the conditional possibilities can be recovered from the local possibility distribution such that $\mathbf{\Pi}(a_i) = \bigcup_{\omega \models a_i} \boldsymbol{\pi}(\omega)$ since from $\mathbf{\Pi}(a_i \wedge p(A_i)) = P$ and $\mathbf{\Pi}(p(A_i)) = All$ we get $\boldsymbol{\pi}(a_i|p(A_i)) = P$ (by solving Equation 5.3).

A multiple agent network is rarely normalized due to conflicting preferences (which contrasts with standard possibilistic networks), thus each conditional possibility distribution is represented by more that one formula. Combined together, it is clear that the resulting logic base is inconsistent with a degree equal to the intersection of all necessity values associated to formulas. Then, the multiple agent base associated with the ma-net $\mathcal{G}$ is $\Gamma_{\mathcal{G}} = \Gamma_{A_1} \bigcup \cdots \bigcup \Gamma_{A_N}$, $\forall A_i \in \mathcal{V}$. The joint possibility distribution computed from the multiple agent network $\mathcal{G}$ by the chain rule is the intersection of the possibility distributions associated to each node. The possibility distribution associated to $\Gamma_{\mathcal{G}}$ is also an intersection of distributions associated to the formula(s) corresponding to each node. This explains why the two representations are represented by the same ma-distribution. This is the counterpart of the fact that the union of possibilistic logic bases corresponds to the min-based aggregation of their distributions [Benferhat et al., 1998].

Example 5.4. *Let us continue Example 5.3; its ma-net can be rewritten as the ma-logic base of Example 5.2, both being semantically associated with the same ma-distribution. The ma-net with conditional tables in Table 5.3 can be expressed by the following ma-base:* $\{(t \vee \neg s \vee c, All), (\neg t, M \vee Y), (t, W \vee O), (\neg s, W \vee O), (s, M \vee Y)\}$.

5.4.2 Transformation of a multiple agent logic into a graphical structure

This converse transformation is more complex. Indeed, the independencies represented by the network are not explicit in logic bases. The transformation consists of two steps: (i) Constructing the network, thus detecting the dependencies, (ii) Computing the conditional possibilities. First, the logic base should be put into a special form, where tautologies are removed (by removing subsumed formulas) each formula should represent a disjunction of a variable value and an instance of all it parents. An algorithm performing this type of transformation is given in [Benferhat et al., 2002a]. To adapt this algorithm the following definitions are useful:

Definition 5.2. *Let* (p, P) *be a formula in* Γ. *Then* (p, P) *is said to be subsumed by* Γ *if* $\Gamma_{\supseteq P} \vdash p$, *where* $\Gamma_{\supseteq P}$ *is composed of classical formulas that appear in* Γ *in association with sets of agents that include* P *or are equal to* P.

Removing subsumed formulas does not change the possibility distribution. This means that

several syntactically different multiple agent logic bases may have the same possibility distribution as their semantic counterpart. For instance, $(x \vee y, A \cap B)$ is subsumed by (x, B), therefore $\Gamma = \{(x \vee y, A \cap B), (x, B)\} = \{(x, B)\}$.

Definition 5.3. *Let Γ be a multiple agent logic base in a clausal form, where* all *clauses involve an instance of a variable A. Let $\mathcal{Z}$ be the set of other variables appearing in the clauses of Γ. A* clausal completion *of Γ with respect to variable A, denoted by $E(\Gamma)$, is the set of clauses of the form $(a \vee \neg \mathbf{z}, P)$ where a is an instance of A, $\mathbf{z}$ is an instance of all variables in $\mathcal{Z}$, and $P = \bigcup\{P_i : (a \vee p_i, P_i) \in \Gamma, \mathbf{z} \models \neg p_i\}$, with $\bigcup(\emptyset) = \emptyset$.*

Proposition 5.1. *The two bases Γ and $E(\Gamma)$ are equivalent.*

Proof. Variables involved in Γ and $E(\Gamma)$ are node A and its parents $\mathcal{P}(A)$. Thus, we restrict to configurations built on these variables without any loss of generality. Let a be an instance of A, $p(A)$ an instance of $\mathcal{P}(A)$. We have,

$$\boldsymbol{\pi}_{E(\Gamma)}(a \wedge p(A)) = \begin{cases} \overline{P} & \text{if } (\neg a \vee \neg p(A), P) \in E(\Gamma) \\ All & \text{otherwise} \end{cases}$$

$$\Leftrightarrow \boldsymbol{\pi}_{E(\Gamma)}(a \wedge p(A)) = \begin{cases} All \setminus \bigcup\{P_i : (\neg a \vee p_i, P_i) \in \Gamma,\ p_i \models \neg p(A)\} \\ All \text{ otherwise} \end{cases}$$

$$\Leftrightarrow \boldsymbol{\pi}_{E(\Gamma)}(a \wedge p(A)) = \begin{cases} All \setminus \bigcup\{P_i : (\neg a \vee p_i, P_i) \in \Gamma,\ p(A) \models \neg p_i\} \\ All \text{ otherwise} \end{cases}$$

$$\Leftrightarrow \boldsymbol{\pi}_{E(\Gamma)}(a \wedge p(A)) = \begin{cases} All \setminus \bigcup\{P_i : (\neg a \vee p_i, P_i) \in \Gamma,\ a \wedge p(A) \models a \wedge \neg p_i\} \\ All \text{ otherwise} \end{cases}$$

$$\Leftrightarrow \boldsymbol{\pi}_{E(\Gamma)}(a \wedge p(A)) = \boldsymbol{\pi}_{\Gamma}(a \wedge p(A))$$

□

The notions of subsumption and clausal completion are instrumental in the procedure (similar to the one in [Benferhat et al., 2002a]) for finding the dependence graph from the multiple agent logic base. More precisely, for each A_i in $\mathcal{V}$ we execute these steps:

- Determination of the local base for A_i: Let $(a_i \vee p, P)$ be a clause of Γ s.t. a_i is an instance of A_i, and p is only built from $A_{i+1}, \ldots A_N$. If $(a_i \vee p, P)$ is subsumed, then remove it from Γ. If $\Gamma \vDash (p, P)$, then replace $(a_i \vee p, P)$ by (p, P). Let K_i be the set of clauses $(a_i \vee p, P)$ in Γ s.t. p is only built from $A_{i+1}, \ldots, A_N$

- The parents of the variable A_i are $\mathcal{P}(A_i) = \{A_j : \exists c \in K_i \; s.t. \; c$ contains an instance of $A_j\}$
- Compute the clausal completion of K_i: Replace in Γ, K_i by its clausal completion $E(K_i)$
- Remove incoherent data: For each $(a_i \vee p, P)$ of Γ (where p is built from $A_{i+1}, \ldots, A_N$ s.t. $\Gamma \vDash (p, P)$) replace $(a_i \vee p, P)$ by (p, P).
- Produce Γ_i: Let Γ_i be the set of clauses $(a_i \vee p, P)$ in Γ s.t. p is only built from $A_{i+1}, \ldots, A_N$.

At the end of the procedure, each node A_i of the constructed graph is associated to a local multiple agent base $\Gamma_{A_i} = \{(a_i \vee p(A_i)) | a_i \in D_{A_i}$ and $p(A_i)$ an instatiation of $\mathcal{P}(A_i)\}$ containing only an instantiation of the node and its parents. These local bases are useful to compute the conditional possibilities such that:

$$\boldsymbol{\pi}_P(a_i|p(A_i)) = \begin{cases} \overline{P} & \text{if } (\neg a_i \vee \neg p(A_i), P) \in \Gamma \\ All & otherwise. \end{cases} \tag{5.7}$$

For instance if $\Gamma = \{(x \vee y, A), (x \vee t, B)\}$, this base is equivalent to $\{(x \vee y \vee t, A \cup B), (x \vee \neg y \vee t, B), (x \vee y \vee \neg t, A)\}$, so, $\boldsymbol{\pi}(\neg x | \neg y \wedge \neg t) = \overline{A} \cap \overline{B}$, $\boldsymbol{\pi}(\neg x | \neg y \wedge t) = \overline{A}$, $\boldsymbol{\pi}(\neg x | y \wedge \neg t) = \overline{B}$, $\boldsymbol{\pi}(\neg x | y \wedge t) = All$.

5.5 Specializing representations and queries

Before handling queries, we discuss two types of specializations performed equivalently on ma-nets and ma-logic bases, w.r.t. a subset of agents.

5.5.1 Sections and restrictions of networks and logic bases

In some cases, one may need to display preferences that are only related to a subset of agents. we propose two possible queries.

First, one may ask about *common* preferences expressed by a subset of agents P, i.e. preferences approved by each element in P. The network obtained by such a *section* has the same structure (with possible deletion of nodes or edges) as the original ma-net and its conditional possibilities are computed such that: $\boldsymbol{\pi}^{\forall}_P(a_i|p(A_i)) = A$ if $P \subseteq \boldsymbol{\pi}(a_i|p(A_i))$ and $\boldsymbol{\pi}^{\forall}_P(a_i|p(A_i)) = \emptyset$ otherwise. Its logical counterpart Γ_P is a propositional logic base where only formulas weighted by P_i, such that $P \subseteq P_i$, are retained. This network can be represented by a Boolean one where for each agent in P such that $\boldsymbol{\pi}^{\forall}_P(a_i|p(A_i)) = P$, we have $\pi(a_i|p(A_i)) = 1$ and 0 otherwise (as in Section 5.2). Its joint possibility distribution has two layers namely, configurations that are preferred and those totally rejected. the section Γ_P is inconsistent, then, all the configurations have a possibility degree equal to 0.

Second, one may focus on *all* the preferences expressed by each subset of agents in P. Thus, we forget about preferences of agents out of P. In this kind of *restriction*, there may exist two agents in P that may express two opposite preferences. The corresponding network can be constructed as: $\boldsymbol{\pi}^{\downarrow}_{P}(a_i|p(A_i)) = \boldsymbol{\pi}(a_i|p(A_i)) \cap P$. Its logical reading corresponds to a multiple agent logic base containing multiple agent formulas of the form $(\neg a_i \vee \neg p(A_i), P_i \cap P)$ s.t. $P_i \cap P \neq \emptyset$.

Example 5.5. *In the previous example 5.2, the logic base corresponding to the common preferences of the subset $W \cap O$ is $\Gamma_{W\cap O} = \{t, \neg s\}$. However, the restriction of the multiple agents base to the subset $W \cap O$ correspond to $\Gamma^{\downarrow}_{W\cap O} = \{(t, W \cap O), (\neg s, W \cap O)\}$*

5.5.2 Optimization, dominance and other queries

In the case of non-normalized conditional distributions finding the optimal configurations for a set of agents P does not always make sense. Indeed, such configurations exist if the set of preferences of group of agents P is consistent, precisely, if for each node and depending on the parents instantiation, the set of agents represented by the conditional possibility is a superset of P. The procedure is explained by Algorithm 2 such that the function $Upgrade\_Opt\_set(config)$ updates the set of optimal configurations. Finding this configuration is straightforward and linear w.r.t. the number of variables. Starting from the root nodes, we choose each time the value(s) a_i s.t. $P \subseteq \boldsymbol{\pi}(a_i)$. Then, depending on the parents instantiation, each time we again choose a value with a conditional possibility that includes or equals P. In case no subset $\boldsymbol{\pi}(a_i)$ for some i is a superset of P then the algorithm stops and the set of agents P have inconsistent preferences. Note that under the ma-normalization, one is always sure to have at least one preferred configuration no matter the set P.

In this bi-valued setting, dominance queries just amount to testing if the configurations belong to the same layer or not; it suffices to check if both configurations are preferred or rejected. The procedure is described by Algorithm 3. Another possible query, is to search for the maximal set of agents that prefer a given configuration. The answer can be obtained by sweeping through the ma-net starting from the roots with the set of agents initialized to All, performing, at each node, the intersection of the current evaluation with the ma-possibility corresponding to the value of the node variable for the given configuration.

5.6 Extension to graded possibilistic networks

We now recall basic notions of multi-agent possibilistic logic that extends both possibilistic logic and multi-agent logic. Then, we propose its graphical counterpart.

Algorithm 2: Finding the optimal configurations for a set of agents

```
Result: Set of optimal configurations config
1  begin
2      order ← topological_sort(ma-net.dag), n ← ma-net.node_sizes
3      N ← length(n), config ← [], out←0, i ← 1
4      while (i <= N) and (out == 0) do
5          ps=parents(pnet.dag, order(i)), CPT ← Get_CPT(pnet.CPT{order(i)})
6          if isempty(ps) then
7              if find(P, CPT))== true then
8                  config ← Upgrade_Opt_set(config)
9              else
10                 out← 1
11         else
12             out← 1
13             for j← 1 to size(config) do
14                 Actual=config(j,:)
15                 for each instantiation m of the node order(i) parents in Actual do
16                     if find(P, CPT,m))== true then
17                         config ← Upgrade_Opt_set(config), out← 0
18         i← i+1
19     return config
```

Algorithm 3: Comparing two configurations

Data: A multiple agent network $ma\text{-}net$, A set of agents P, two configurations ω_1 and ω_2

Result: The set of preferred configuration(s) $result$

```
1  begin
2      order ← topological_sort(ma-net.dag)
3      n ← ma-net.node_sizes, N ← length(n), result← ∅
4      for i← 1 to 2 do
5          out←0, j ← 1, CPT = Get_CPT(Mprefnet.CPTorder(i))
6          while (j <= N) and (out == 0) do
7              Degree← the instantiation of the node order(j) in ω_i according the
                parents instantiation in ω_i
8              if The set of agents A ⊈ Degree then
9                  out← 1
10             else
11                 j← j+1
12         if (out==0) then
13             result← result ∪ω_i
14     return result
```

5.6.1 Possibilistic multiple agent logic

This logic describes the graded preferences of agents using fuzzy set-valued counterparts of the notions of possibility distribution, possibility measure, and necessity measure. Formulas in this logic are of the form $(p, \alpha/P)$ (where α is a necessity measure and P is a subset of agents) expressing that at least all the agents in P think that it is imperative to satisfy p with a minimal priority degree equal to α. Asserting $(p, \alpha/P)$ means that P is the maximal set of agents that tolerate the falsity of p with level at most $1-\alpha$, while the agents in $\overline{P}$ are indifferent to the truth or falsity of p, finding both tolerable at level 1. By duality, $\mathbf{\Pi}(p)$ is the fuzzy set of agents who do not require the truth of $\neg p$ imperatively. Each possibilistic ma-logic base Γ is associated to an ma-π distribution $\boldsymbol{\pi}_\Gamma$.

$$\boldsymbol{\pi}_\Gamma(\omega) = \begin{cases} 1/All & \text{if } \forall (p_i, \alpha_i/P_i) \in \Gamma, \omega \models p_i \\ \bigcap\{(1-\alpha_i)/P_i \cup 1/\overline{P_i} \mid (p_i, \alpha_i/P_i) \in \Gamma,\ \omega \models \neg p_i\} & \text{otherwise.} \end{cases} \tag{5.8}$$

where $\boldsymbol{\pi}_\Gamma(\omega) = \alpha/P$ means that at most all the agents in A find ω acceptable with a maximal satisfaction degree equal to α. The ma-normalization and the i-normalization continue to be defined as above. Precisely, ma-normalization is still related to the consistency of the propositional logic base and means that $\exists\, \omega \in \Omega,\ \boldsymbol{\pi}(\omega) = 1/All$, where $1/All$ is clearly the same as All. Moreover, the i-normalization is still defined by $\mathbf{\Pi}(\Omega) = \bigcup_{\omega\in\Omega} \boldsymbol{\pi}(\omega) = All$, and means that all the agents are individually consistent.

Example 5.6. *Let us consider a multiple agent possibilistic logic corresponding to the preferences over the variable Drink= $\{t, \neg t\}$: $\Gamma = \{(\neg t, 0.9/\overline{W}), (t, 0.3/\overline{M \cap Y})\}$. The possibility distribution (computed by 5.8) corresponding to this base is:*
$\boldsymbol{\pi}(t) = ((1 - 0.9/\overline{W}) \cup (1/All \setminus \overline{W})) \cap (1/All) = (0.1/\overline{W}) \cup (1/W) = ((0.1/\overline{W}) \cup (1/W)) \cap (1/All) = (0.1/\overline{W}) \cup (1/W)$,
$\boldsymbol{\pi}(\neg t) = (0.7/\overline{M \cap Y}) \cup (1/M \cap Y) \cap (1/All) = 0.7/\overline{M \cap Y}) \cup (1/M \cap Y)$.
The value of $\boldsymbol{\pi}(t)$ means that women find a cup of tea fully acceptable and men find it tolerable at best at a very low level 0.1. *The possibilistic logic base representing the preferences of women is $\Gamma_W = \{(t, 0.3)\}$ and the one representing the preferences of men is $\Gamma_W = \{(\neg t, 0.9)\}$.*

5.6.2 Multi-agent possibilistic graphical representation

As for ma-logic (with Boolean possibility), we have a graphical counterpart to ma-π logic. This graphical representation uses fuzzy set-valued degrees as in ma-π logic.

5.6.2.1 Multi-agent possibilistic networks.

Based on the same conditioning (Eq. 5.4) and the same chain rule (Eq. 5.6) where intersection is extended to fuzzy sets, we can define multi-agent possibilistic networks (ma-π nets for short)

that are a counterpart of ma-π logic. ma-π nets are an extension of the above-defined graphical counterpart of ma-logic, and have the same structure as ma-nets.

Example 5.7. *Let us consider the ma-π tables of Table 5.5, associated to the network of Figure 5.1 and corresponding to an extension of the network defined in Figure 5.1 and Table 5.2. We can see that the local possibility distribution associated to node 'Drink' corresponds to the logic base of Example 5.6. It is clear that the network is not ma-normalized and this can be verified on its associated possibility distribution. For instance,* $\boldsymbol{\pi}(t\neg sc) = (1/W \cup 0.1/\overline{W}) \cap (0.2/Y \cup 1/O) \cap (1/O \cup 0.1/Y) = (1/W \cup 0.1/\overline{W}) \cap (0.1/Y \cup 1/O)$. *It is clear that* $W \cap O$ *agents prefer* $t\neg sc$ *with a satisfaction degree equal to* 1.

$\boldsymbol{\pi}(t)$	$\boldsymbol{\pi}(\neg t)$
$1/W \cup 0.1/\overline{W}$	$1/M \cap Y \cup 0.7/\overline{M \cap Y}$

$\boldsymbol{\pi}(s)$	$\boldsymbol{\pi}(\neg s)$
$1/Y \cup 0.6/O$	$1/O \cup 0.2/Y$

$\boldsymbol{\pi}(.\|.)$	ts	$t\neg s$
c	$(1/Y) \cup (0.3/O)$	$(1/O) \cup (0.1/Y)$
$\neg c$	$(1/O) \cup (0.5/Y)$	$(1/W) \cup (0.6/M)$

$\boldsymbol{\pi}(.\|.)$	$\neg ts$	$\neg t\neg s$
c	$(1/M \cap Y) \cup (0.2/\overline{M \cap Y})$	$(1/W) \cup (0.9/M)$
$\neg c$	$(1/W) \cup (0.1/M)$	$(1/M \cap O) \cup (0.7/\overline{M \cap O})$

Table 5.5: Conditional tables of an ma-π net

5.6.2.2 From an ma-π net to an instantiated π-pref net

In contrast with ma-nets, ma-π nets enable us to express levels of preference. Indeed, preferences are no longer all or nothing. Then, the network pertaining to the preferences of a set P of agents, induced as a section of the ma-π net, corresponds to a possibilistic preference network (π-pref net) with instantiated weights. Its structure is similar to the ma-π net and the local possibility distributions associated to P are defined by: $\pi_P(a_i|p(A_i)) = \alpha$, $\forall P \subseteq B$ s.t. $\boldsymbol{\pi}_\Gamma(a_i|p(A_i)) \subseteq \alpha/B$. Note that the induced net is not always normalized due to the possible lack of normalization of the ma-π net. Clearly, normalization states that the preferences of the set P of agents are consistent and at least one configuration has a possibility degree equal to 1 for agents in P.

Example 5.8. *Let us consider the ma-π net defined by Figure 5.1 and Table 5.5. The induced possibilistic tables representing the preferences of the set* $W \cap O$ *are:* $\pi(t) = 1$, $\pi(\neg t) = 0.7$, $\pi(s) = 0.6$, $\pi(\neg s) = 1$, $\pi(c|ts) = 0.3$, $\pi(c|t\neg s) = 1$, $\pi(c|\neg ts) = 0.2$, $\pi(c|\neg t\neg s) = 1$, $\pi(\neg c|ts) = \pi(\neg c|t\neg s) = \pi(\neg c|\neg ts) = 1$, *and* $\pi(\neg c|\neg t\neg s) = 0.7$. *The resulting network is normalized. Then, the possibility distribution is normalized and can be computed using the standard product-based chain rule (3.1). We can check that* $t\neg sc$ *and* $t\neg s\neg c$ *are the best configurations* ($\pi(t\neg sc) = \pi(t\neg s\neg c) = 1$).

ABC	$\pi(C\|TS)$	$\pi(S)$	$\pi(T)$	$\pi(TSC)$
tsc	0.3	1	1	0.3
$ts\neg c$	1	1	1	1
$t\neg sc$	1	0.6	1	0.6
$t\neg s\neg c$	1	0.6	1	0.6
$\neg tsc$	0.2	1	0.7	0.14
$\neg t\neg s\neg c$	1	1	0.7	0.7
$\neg t\neg sc$	1	0.6	0.7	0.42
$\neg ts\neg c$	0.7	0.6	0.7	0.94

Table 5.6: Possibility distribution corresponding to the set $W \cap O$

5.7 Related work

Few models exist for representing multiple agent preferences. First, multi-agent CP-nets (mCP-nets) [Rossi et al., 2004] are an extension of CP-nets in a multiple agent setting. They are made of several partial CP-nets representing the preferences of each agent, where a partial CP-net is a CP-net where some variables may not be ranked when the agent is indifferent about the values of these variables. Graphically, the network is obtained by combining the partial CP-nets. We can reason about an mCP-net by querying each partial CP-net, and then deduce the answer using different voting concepts like Pareto optimality, lexicographic ordering, and quantitative ranking. Second, probabilistic CP-nets (PCP-nets) [Bigot et al., 2013] enable a compact representation of a probability distribution over several CP-nets and stand for a summary of collective preferences. A PCP-net has the same graphical component as a CP-net. Lastly, generalized additive independence (GAI) nets [Dubus et al., 2009] are quantitative graphical models where preferences of agents are expressed by utilities. In a multiple agent framework, each node is characterized by a utility vector where each of its elements represents the utility of the node given by an agent. An aggregation procedure is then applied to these utilities to find the optimal solution.

As shown here, ma-nets represent the collective preferences of agents with a single network, similarly to PCP-nets and GAI nets (and in contrast with mCP-nets), which facilitates the handling of preferences. Besides, it may handle the indifference and non consistency of some agents, and can deal with the agents based on their profiles and not only in terms of proportions contrarily to GAI nets and PCP-nets. The model can be extended to describe preference intensities by adding priorities, unlike mCP-nets and PCP-nets.

5.8 Conclusion

In this chapter, we have proposed a multiple agent graphical representations as an extension of π-pref nets. The proposed models represent the graphical counterparts of multiple agent logic. First, we have proposed a graphical model where preferences of agents are all or nothing; each set of agents either totally accepts the choice or totally rejects it. Second, this preference representation has been extended to handle gradual preferences. We have discussed the basic properties of the proposed models and proposed their associated possible queries. Moreover, we have shown how from these networks we can construct individual networks pertaining to subsets of agents.

This chapter calls for several future lines. Indeed, we may think about the symmetrical form of formulas (p_1, P_1), where the roles played by p and P should be studied, particularly, for inducing preferences of a subset of agents such that, $P_2 \subset P_1$, or $P_1 \cap P_2 = \emptyset$. In fact, based on such symmetrical form, we may think of inducing another type of networks associated to each configuration. Computational issues are also of interest. We may think of proposing algorithms to answer ordering queries and other possible variants in a *local manner*. For instance, we may think of identifying non consistent agents *directly* from multi-agents (possibilistic) networks. One may also think of several extensions, where the set of agents are described under the form of logical formulas which would allow us to consider the disjunction and the negation between the characteristics of each set of agents. Finally, the full strength of the representation power of π-pref nets comes from a symbolic handling of the priorities still to be introduced in the multiple agent case. Other perspectives are detailed in the general conclusion.

CHAPTER 6

Implementation: A toolbox for Preference Possibilistic Networks

Contents

6.1 Introduction

This chapter proposes a Matlab toolbox dedicated to π-pref nets, their queries and their main extension. The purpose of this implementation is to offer an extensible standard tool allowing the exploitation of new models developed in this thesis.

This Chapter is organized as follows: Section 6.2 presents the toolbox implemented to construct π-pref net and to execute optimization and dominance queries. Section 6.3 is dedicated to multiple agent networks. It presents the methods to construct the network and execute its possible queries.

6.2 A toolbox for π-pref nets

In this section, we propose a π-pref nets toolbox implemented with Matlab 7.10 in order to handle the preference queries namely, dominance and optimization. The implementation uses some adaptations of the Possibilistic Networks Toolbox (PNT) [Ben Amor, 2012] implemented in order to handle several propagation algorithms in possibilistic networks. The software offers 4 main functionalities:

1. Defining the preference network structure;
2. Defining the associated preference tables;
3. Executing the optimization query;
4. Executing the dominance query.

Figure 6.1 presents the main window of the implemented toolbox.

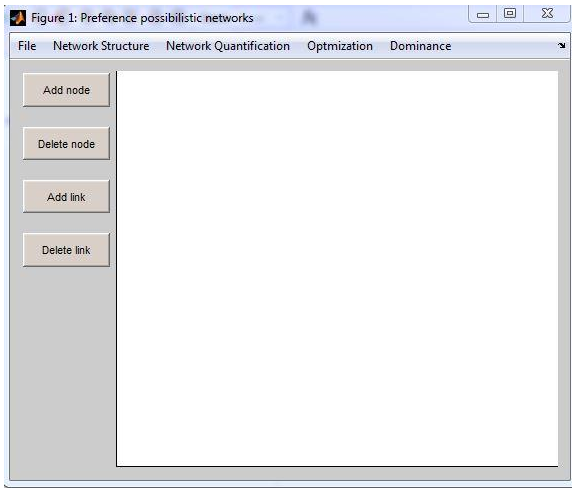


Figure 6.1: The main window of the π-pref net toolbox

6.2.1 Definition of the network structure

This functionality allows to define the set of nodes and the dependency structure. The software enables the user to define the name of the node, its position, the set of its parents and its cardinality. Figure 6.2 presents the step of creating a node. Note that the program enables the user to delete a node such that deleting a node leads to deleting all incoming and outgoing arcs. Figures 6.2 and

6.3 represent examples of creating a node and defining the parents (thus defining the structure), respectively. The preference possibilistic network is represented by an object called **prefnet** defined as follows:

Algorithm 4: Constructing the π-pref net

Result: prefnet

begin

- **nodes**: 1-by-N matrix,
row vector containing nodes in a topological order (ancestors before descendants);
- **node_sizes**: 1-by-N matrix
node_sizes(i) is the number of values node i can take on (its arity);
- **dag**: N-by-N matrix
dag(i, j) = 1 if and only if i is parent of j;
- **CPD**: 1-by-N cell array of matrices
Each cell CPD{i} contains a *tabular_cpd object* defined by:
$*$ CPD{i}.self : Node i
$*$ CPD{i}.CPT : A vector containing the initial Conditional Possibility Distribution of node i in the context of its parents;
-**symbols**: 1-by-C matrix (C is the number of symbolic weights) symbols(i) is a symbolic weight
-**consts**: C-by-C matrix
consts(i,j)=1 if and only if the symbolic weight i is bigger that j

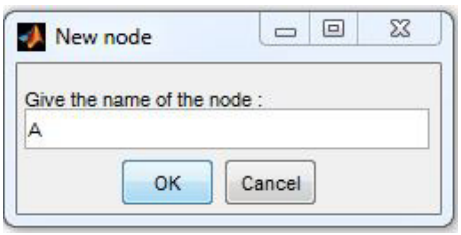


Figure 6.2: Creating a node

As a first step, Function **mk_prefnet** (Algorithm 5) prepares the possibilistic network structure (i.e. *prefnet*) using the initial dag (i.e. *dag*), the node sizes (i.e. *node_sizes*), the list of nodes (i.e. *nodes*, the list of symbols (i.e. *symbols*) and the matrix of constraints (i.e. *consts*).

Example 6.1. *Let us consider a π-pref net over a set of variables $\mathcal{V} = \{A, B, C, D, E, F, G, H\}$. Figure 6.4 represents the graph structure constructed by the software. Clearly, A is the unique root and H and G are the leaves.*

Algorithm 5: mk_prefnet

```
Data: dag, node_sizes, nodes, symbols, consts
Result: prefnet
begin
    N ← length(dag);
    prefnet.dag ← dag;
    prefnet.node_sizes ← node_sizes(:)';
    prefnet.nodes ← nodes;
    prefnet.symbols ← symbols;
    prefnet.consts ← consts;
```

Figure 6.3: Selection of parents

6.2.2 Network preference tables and constraints

After constructing the network structure, one should enter the possibility distributions. Figure 6.5 presents an example of defining a conditional possibility distribution of a node A in the context of B.

Example 6.2. *To encode the preference possibilistic network of Example 6.1, we should first define the topological order (i.e. ancestors before descendants) which can be* $[A\ B\ C D\ E\ F\ G\ H]$ *for instance (note that one can define another topological order provided that the parents appear before children). In the following we consider that each node is associated to an integer number such that* $A = 1, B = 2, C = 3$*,* $D = 4$*,* $F = 5$*,* $E = 6$*,* $G = 7$ *and* $H = 8$*. Then, the* ***prefnet*** *object takes the following values:*

- ***prefnet.nodes****= [1 2 3 4 5 6 7 8]*
- ***prefnet.node_sizes****= [2 2 2 2 2 2 2 3]*

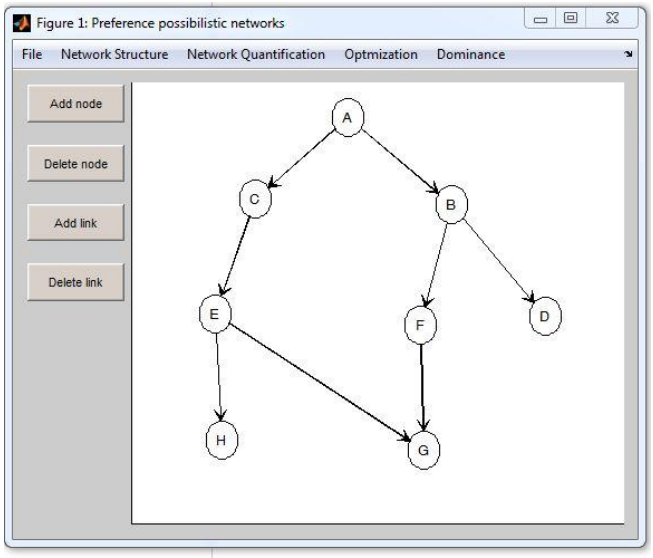


Figure 6.4: An example of a π-pref net with 8 nodes

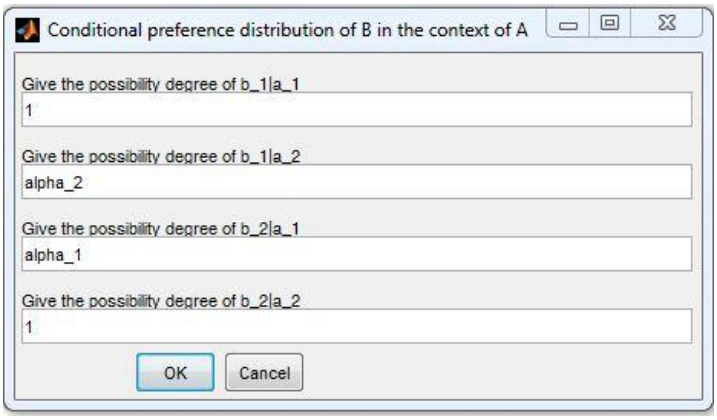


Figure 6.5: An example of defining a preference distribution

- ***prefnet.dag*** $= \begin{pmatrix} 0 & \boldsymbol{1} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

For instance the value 1 in the line 1 column 2 means that the variable 1 (A) is a parent of the variable 2 (B).

- *The conditional possibility distributions are stored as multidimentional arrays (prefnet.CPD) where the variables are arranged s.t the low numbered parents come before the high numbered one:*

  ***prefnet.CPD**= [1x1 tabular_cpd] [1x1 tabular_cpd] [1 x1 tabular_cpd] [1x 1 tabular_cpd][1x 1 tabular_cpd][1x 1 tabular_cpd][1x 1 tabular_cpd][1x 1 tabular_cpd]*

  - *prefnet.CPD{1}.self = 1*
    prefnet.CPD{1}.CPT = $\begin{pmatrix} 1 \\ \alpha \end{pmatrix}$
    The order of instances in prefnet.CPD{1}.CPT is as follows: $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$
    The value 1 (resp. 2) corresponds to the first (resp. second) instance of the variable 1 (A).
  - *prefnet.CPD{2}.self = 2*
    prefnet.CPD{2}.CPT = $\begin{pmatrix} \beta_1 & 1 \\ 1 & \beta_2 \end{pmatrix}$
    The order of instances in prefnet.CPD{2}.CPT is as follows: $\begin{pmatrix} 11 & 12 \\ 21 & 22 \end{pmatrix}$
    The value 11 (resp. 21, 12, 22) corresponds to the first (resp. second, first, second) instance of the the variable 1 (A) and the first (resp. first, second, second) instance of the variable 2 (B).
  - *prefnet.CPD{3}.self = 3*
    prefnet.CPD{3}.CPT = $\begin{pmatrix} 1 & \gamma_2 \\ \gamma_1 & 1 \end{pmatrix}$
    The order of instances in prefnet.CPD{3}.CPT is as follows: $\begin{pmatrix} 11 & 12 \\ 21 & 22 \end{pmatrix}$
    The value 11 (resp. 21, 12, 22) corresponds to the first (resp. second, first, second) instance of the the variable 1 (A) and the first (resp. first, second, second) instance of the variable 3 (C).
  - *prefnet.CPD{4}.self = 4*
    prefnet.CPD{4}.CPT = $\begin{pmatrix} \delta_1 & \delta_2 \\ 1 & 1 \end{pmatrix}$

- *prefnet.CPD{5}.self = 5*
  prefnet.CPD{5}.CPT $= \begin{pmatrix} 1 & \epsilon_2 \\ \epsilon_1 & 1 \end{pmatrix}$

- *prefnet.CPD{6}.self = 6*
  prefnet.CPD{6}.CPT $= \begin{pmatrix} \theta_1 & 1 \\ 1 & \theta_2 \end{pmatrix}$

- *prefnet.CPD{7}.self = 7*
  prefnet.CPD{7}.CPT $= \begin{pmatrix} 1 & \lambda_2 & \lambda_3 & 1 \\ \lambda_1 & 1 & 1 & 1 \end{pmatrix}$

  The order of instances in prefnet.CPD{4}.CPT is as follows: $\begin{pmatrix} 111 & 121 & 112 & 122 \\ 211 & 221 & 212 & 222 \end{pmatrix}$

  The value 111 (resp. 211, 121, 221, 112, 212, 122, 222) corresponds to the first (resp. second, first, second, first, second, first, second) instance of the the variable 2 (B), the first (resp. first, second, second, first, first, second, second) instance of the variable 3 (C) and to the first (resp. first, first, first, second, second, second, second) instance of the variable 4 (D).

- *prefnet.CPD{8}.self = 8*
  prefnet.CPD{8}.CPT$= \begin{pmatrix} 1 & \phi_3 \\ \phi_1 & 1 \\ \phi_2 & \phi_4 \end{pmatrix}$

The set of constraints is described by a matrix C-by-C where C is the number of constraints. Figure 6.7 represents an example of setting the constraints between symbolic weights. Since comparisons between symbolic weights are made between every two weights, the transitive closure should be computed in order to find all relations between the weights.

This transitive closure is computed by Algorithm 6 which is based on:

- The function *sparse*: it converts a full matrix into a sparse form by squeezing out any zero elements. If a matrix contains many zeros, converting the matrix to sparse storage saves memory.
- The function *graphallshortestpaths*: its finds the shortest paths between every pair of nodes in the graph represented by a matrix.
- The function *numel*: it returns the number of elements in an array.

Indeed, the matrix *consts* is considered as a graph where each symbolic weight is represented by a node such that two nodes are related by an arc if there exists a preference relation between

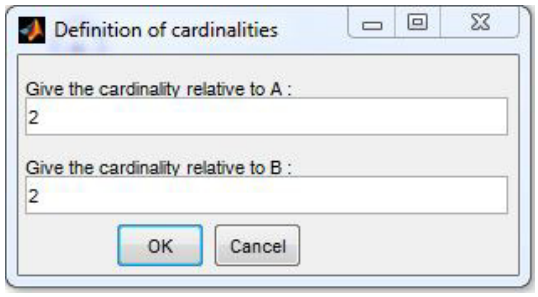


Figure 6.6: An example of defining cardinalities

Algorithm 6: Transitive closure of constraints (Function Complete_consts)

```
   Data: consts, symbols
   Result: consts
1  begin
2      m←sparse(consts);
3      allShortest← graphallshortestpaths(m);
4      for i← 1 to numel(allShortest) do
5          if allShortest(i)==Inf then
6              allShortest(i)←0;
7      consts←logical(allShortest);
8      return consts
```

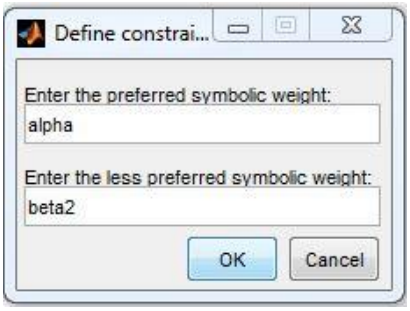


Figure 6.7: A example of defining a constraint

the two symbolic weights. The algorithm searches if there is a path between two symbolic weights not directly related by an arc. If such a path exists then the two symbols can be compared. This can be illustrated by the following example.

Example 6.3. *Let us consider the preference network of Example 6.1 with the preference tables of Example 6.2. The constraints between the symbolic weights is represented by the matrix **consts** and is as follows:*

prefnet.symbols$=\{\alpha; \beta_1; \beta_2; \gamma_1; \gamma_2; \delta_1; \delta_2; \epsilon_1; \epsilon_2; \theta_1; \theta_2; \lambda_1; \lambda_2; \lambda_3; \phi_1; \phi_2; \phi_3; \phi_4\}$

$$\textbf{\textit{prefnet.consts}}= \begin{pmatrix}
0 & \boldsymbol{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

For instance, the value 1 in the line one column 2 means that β_1 *is bigger than* α*, and the value 1 in the line two column three means that* β_2 *is bigger than* β_1*. By the transitive closure one can deduce that* β_2 *is bigger than* α*.*

6.2.3 Optimization query

As mentioned before, optimization query is defined by traversing the network from roots to leaves while instantiating the variables to values where the conditional possibility degrees are equal to 1. Algorithm 7 is dedicated for optimization and uses the following functions:

- The function *topological_sort*: it returns the nodes in a topological order (parents before children).
- The function *parents*: it returns the list of a node parents.

- The function *find*: it returns a vector containing the linear indices of each nonzero element in an array.
- The function *subv2ind*: it returns an equivalent single index corresponding to a subscript vector.

Example 6.4. *Let us continue the previous example. As we already noted the topological ordering between the nodes is:* $[ABCDEFGH]$. *The algorithm proceeds by the node* A. *It tests as a first step if the node has parents. Then, it instantiates the node to its value* a_1 *since* $\pi(a_1) = 1$. *The algorithm processes in the same way till instantiating all the nodes. The result of the algorithm is given by Figure 6.8 that displays the best configuration which is* $a_1b_2c_1d_2e_1f_1g_1h_1$.

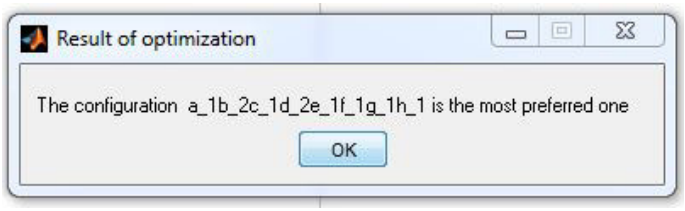


Figure 6.8: A example of the optimization query result

6.2.4 Dominance query

Regarding dominance query, comparing configurations amounts to comparing vectors of weights. Indeed, the software implemented proceeds by asking the user to select two configurations to be compared then compares their corresponding vectors of weights. Figure 6.9 presents a result of dominance query displayed by the toolbox. The main steps are as follows:

- Find the vectors of weights by traversing the network once. Algorithm 8 explains the main steps. It takes as input a configuration *interp*, the *prefnet* and the table storing the symbols *symbols*. The algorithm uses the following functions:
  - The function *isempty*: It returns a logical 1 (true) if the array is empty and a logical 0 (false) otherwise.
  - The function *strcmp*: It compares two strings and returns 1 (true) if the two are identical and 0 (false) otherwise.
- Permute one vector while it tests each time if each element of the vector of weights can be compared to its corresponding element in the other vector. Algorithm 9 uses the following function:

Algorithm 7: Optimization query

```
   Data: prefnet
   Result: config
1  begin
2      order=topological_sort(dag);
3      n=prefnet.node_sizes;
4      config=zeros(1,N);
5      for i←1 to N do
6          ps=parents(prefnet.dag,order(i));
7          if isempty(ps) then
8              CPT ← CPD_to_CPT(pnet.CPD{order(i)});
9              f←find(CPT==1);
10             dup←length(f);
11             if dup==1 then
12                 config(:,order(i))=f(1);
13                 temp←config
14             else
15                 for j←1 to dup do
16                     B←config;
17                     B(:,order(i))←f(j);
18                     if j==1 then temp←B; else temp←[temp; B] ;
19         else
20             s←size(config);
21             CPT ← CPD_to_CPT(pnet.CPD{order(i)});
22             for j←1 to s(1) do
23                 actual←config(j,:);
24                 in←actual(ps);
25                 for m← 1 to n(order(i)) do
26                     actual←config(j,:);
27                     subv2ind(size(CPT), [in m]);
28                     if CPT(subv2ind(size(CPT), [in m]))==1 then
29                         actual(order(i))←m;
30                         if isempty(temp) then temp←actual;
31                         else temp←[temp; actual] ;
32         config←temp; temp←[];
33     return config
```

– The function *perms*: It returns a matrix containing all permutations of the elements of a vector. Each row of the matrix contains a different permutation of the n elements in the array. The matrix has the same data type as the array, and it has n! rows and n columns (such that n is the number of symbolic weights in the longest vector).

Algorithm 8: Defining the vector of weights

```
Data: interp, prefnet, symbols
Result: vect
begin
    vect←[];
    for i←1 to length(prefnet.dag) do
        ps←parents(prefnet.dag,i);
        CPT ← CPD_to_CPT(prefnet.CPD{i});
        if isempty(ps) then
            if isempty(vect) then
                interp(i);
                vect←[CPT(interp(i))];
            else
                vect←[vect CPT(interp(i))];
        else
            in←interp(ps);
            vect←[vect CPTsubv2ind(size(CPT), [in interp(i)])];
    temp←[];
    for i←1 to length(vect) do
        if not(vect{i}==1) then
            z←find(strcmp(symbols, vect(i)));
            if i==1 then
                temp←z;
            else
                temp←[temp z];
    vect←temp;
    return vect
```

Example 6.5. *Let us continue the running example. We consider that the user selected the two configurations* $a_1b_2c_1d_2e_1f_1g_1h_1$ *and* $a_2b_1c_2d_2e_2f_2g_1h_1$ *to be compared. After the execution of Algorithm 8 we get the two vectors of symbolic weights such that* $a_1b_2c_1d_2e_1f_1g_1h_1$ *is associated to [1 1 1 1 1 1 1 1], and* $a_2b_1c_2d_2e_2f_2g_1h_1$ *is associated to* $[\alpha\ 1\ 1\ 1\ 1\ 1\ 1\ \phi_3]$*. Clearly, Algorithm 9 displays the second configuration since* $1 > \alpha * \phi_3$ *(note that as seen in Example 6.4,*

Algorithm 9: Comparing vectors of weights

```
Data: consts, vect1, vect2
Result: dominant, res
begin
    if isempty(vect1) then
        if isempty(vect2) then dominant←[]; res←0; else dominant←2; res←1 ;
    else
        if isempty(vect2) then
            dominant←1; res←1;
        else
            change←0;
            if length(vect2)>length(vect1) then
                temp←vect1; vect1←vect2; vect2←temp; change←1;
            vect11←vect1; nb_elim←0;
            for i←1 to length(vect1) do
                z←find(vect2==vect1(i));
                if not(isempty(z)) then
                    vect11(i-nb_elim)←[]; vect2(z)←[]; nb_elim←nb_elim+1;
            vect1←vect11; permutation←perms(vect1); s←size(permutation);
             res←0; i←1;
            while (i<=s(1))&&(res==0) do
                good←1; j←1; v1←0; v2←0;
                while (j<=length(vect2))&&(good==1) do
                    if consts(permutation(i,j),vect2(j))==1 then
                        v2←1;
                    else
                        if consts(vect2(j),permutation(i,j))==1 then v1←1; else
                        good←0 ;
                    if (v1&&v2) then good←0 ;
                    j←j+1;
                if good==1 then res=1;
                i←i+1;
            if not(res==1) then
                dominant=[];
            else
                dominant←1;
                if ((v1==1)$$(change =0))||((v1 =1)$$(change==0)) then
                    dominant←2;
    return vect
```

$a_1b_2c_1d_2e_1f_1g_1h_1$ is the best configuration). The result of this dominance query is represented by Figure 6.9.

Let us now consider two other configurations $a_1b_2c_2d_2e_1f_2g_1h_1$ and $a_2b_1c_2d_1e_2f_2g_1h_2$. The corresponding vectors of weights are [1 1 γ_1 1 ϵ_2 θ_2 λ_3 1] and [α 1 1 δ_1 1 1 1 1] respectively. Algorithm 9 returns an empty variable dominant and the logical variable res is equal to false (logical 0). This means that the two configurations are incomparable. The result of this dominance query is represented by Figure 6.10.

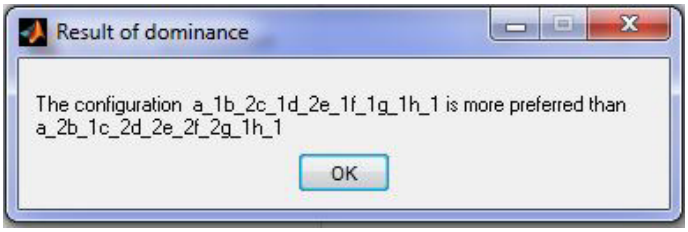


Figure 6.9: An example of a dominance query result

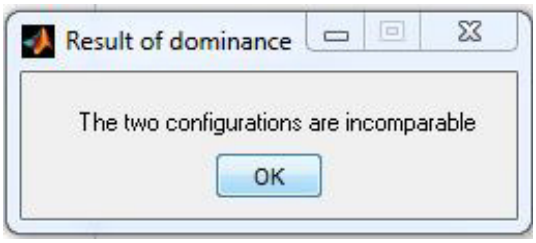


Figure 6.10: An example of a dominance query result (incomparable configurations)

6.3 An extension for ma-nets

This section extends the toolbox for π-pref nets to the multiple agent framework and more specifically to ma-nets (developed in Chapter 5) in order to handle all or nothing preferences.

6.3.1 The construction of ma-nets

Similarly to π-pref nets, the multiple agent network is represented by an object called **Mprefnet** and is defined by Algorithm 10.

Algorithm 10: Constructing the ma-net

Result: Mprefnet

begin

- **nodes**: 1-by-N matrix,
row vector containing nodes in a topological order (ancestors before descendants);
- **node_sizes**: 1-by-N matrix
node_sizes(i) is the number of values node i can take on (its arity);
- **dag**: N-by-N matrix
dag(i, j) = 1 if and only if i is parent of j;
- **CPD**: 1-by-N cell array of matrices
Each cell CPD{i} contains a *tabular_cpd object* defined by:
∗ CPD{i}.self : Node i
∗ CPD{i}.CPT : A vector containing the initial Conditional Possibility Distribution of node i in the context of its parents;
- **Lib_characteristic**: 1-by-N matrix
Lib_characteristic(i) represents a characteristic of a set of agents.

The function **mk_Mprefnet** makes the possibilistic network structure (i.e. *Mprefnet*) using the initial dag (i.e. *dag*), the node sizes (i.e. *node_sizes*), the list of nodes (i.e. *nodes*, the list of characteristics (i.e. *Lib_characteristic*).

Algorithm 11: mk_Mprefnet

Data: dag, node_sizes, nodes, Lib_characteristic

Result: Mprefnet

begin

N ← length(dag);
Mprefnet.dag ← dag;
Mprefnet.node_sizes ← node_sizes(:)';
Mprefnet.nodes ← nodes;
Mprefnet.Lib_characteristic ← Lib_characteristic;

Example 6.6. *Let us reconsider the same example of Chapter 5 Example 5.2. We suppose that the user entered two characteristic, the age and the gender such that age=[1 2 3] where* 1 *corresponds to All,* 2 *corresponds to 'women' and* 3 *corresponds to 'men'. Similarly gender=[1 4 5] where* 1 *corresponds to All,* 4 *to 'young and* 5 *to old. To encode this multiple agent, the* ***Mprefnet*** *object takes the following values:*

- ***Mprefnet.nodes****= [1 2 3]*

- ***Mprefnet.node_sizes***= *[2 2 2]*
- ***Mprefnet.Lib_characteristic***= *['All'; 'Men', 'Women'; 'Young';'Old']*
- ***Mprefnet.dag***= $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$
- *The conditional possibility distributions are stored as multidimentional arrays (Mprefnet.CPD) where the variables are arranged s.t the low numbered parents come before the high numbered one:*

  Mprefnet.CPD= *[1x1 tabular_cpd] [1x1 tabular_cpd] [1 x1 tabular_cpd]*

  - *prefnet.CPD{1}.self = 1*

    prefnet.CPD{1}.CPT = $\begin{pmatrix} 31 \\ 24 \end{pmatrix}$

    The value 31 (resp. 24) corresponds to the set of agents women W *(resp. Young Men* $M \cap Y$*). Indeed, in the value 31, the number three corresponds to the element of index 3 in the table* $Lib\_characteristic$ *and the number* 1 *corresponds to the element of index* 1 *in* $Lib\_characteristic$ *.*
  - *prefnet.CPD{2}.self = 2*

    prefnet.CPD{2}.CPT = $\begin{pmatrix} 14 \\ 15 \end{pmatrix}$

    The value 14 (resp. 15) corresponds to the set of agents Young Y *(resp. Old* O*).*
  - *prefnet.CPD{3}.self = 3*

    prefnet.CPD{3}.CPT = $\begin{pmatrix} 24 & 15 & 24 & 31 \\ 25 & 31 & 31 & 25 \end{pmatrix}$

6.3.2 Queries for ma-nets

The software executes different types of queries:

- Searching for the most preferred configurations. Algorithm 12 presents the main steps to find the best configurations. The algorithm needs the Mprefnet object (*Mprefnet*), the number characteristics (*Critere*) and the set of agents of interest selected by the user (*set*). The algorithm outputs a matrix M-by-N where N is the number of nodes and M is the number of the best configurations for the selected set of agents. Note that the selected set of agents can be All where if the resulting matrix is empty then the ma-net is not ma-normalized.

- Finding the maximal set of agents which prefers a given configuration. This query is executed by Algorithm 13.
- Verifying if a given set of agents prefers a given configuration. Algorithm 14 corresponds to this query. It returns $out = 1$ if the entered configuration is preferred by the selected set of agents and $out = 0$ otherwise. Algorithm 14 is a variant of Algorithm 13 where the user should also specify the set of agents.
- Comparing two configurations given a set of agents. The algorithm 15 executes the function *select_set* (Algorithm 14) for the chosen set of agents. Then if the set of agents prefers both configurations Algorithm 15 outputs a matrix containing the two configurations, if it only prefers one of them then variable *dominant* will contain the preferred one, otherwise we get $domiant = \emptyset$.

Example 6.7. *Let us continue Example 6.6. We consider Algorithm 12 for optimization query and assume that the user selected:*

- *Young Men* $M \cap Y$*, then config=*$\begin{pmatrix} 2 & 1 & 1 \end{pmatrix}$
  Which can be read as the best configuration for young men is $\neg tsc$*.*
- *Old Women* $W \cap O$*, then config=*$\begin{pmatrix} 1 & 2 & 1 \\ 1 & 2 & 2 \end{pmatrix}$
  Which can be read as the best configurations for old women are $t\neg sc$ *and* $t\neg s\neg c$*.*
- *Men* M*, then config=*$\emptyset$ *Which means that the set of agents Men is not i-normalized (not consistent).*
- All*: Clearly the ma-net is not ma-normalized (*All *is inconsistent). config is then an empty matrix.*

Algorithm 12: Optimization query by ma-net (function *Moptimization*)

```
Data: Mprefnet, set, Critere
Result: config
begin
    order←topological_sort(Mprefnet.dag); n←Mprefnet.node_sizes; N←length(n);
     config←zeros(1,N); out←0; i←1;
    while (i<=N) && (out==0) do
        ps←parents(Mprefnet.dag,order(i));
        if isempty(ps) then
            CPT ← CPD_to_CPT(Mprefnet.CPD{order(i)}); f←[];
            for z=1:length(CPT) do
                sortir←0; degree←int2str(CPT{z}); y←1;
                while (y<=Critere) && (sortir==0) do
                    if (strcmp(degree(y), int2str(set(y)))==0) && (strcmp(degree(y),
                     '1')==0) then sortir←1; else y←y+1;
                if (sortir==0) then
                    if isempty(f) then f(1)←z; else f←[f z];
            dup←length(f);
            if dup==1 then config(:,order(i))←f(1);
            else for j←1 to dup do
                B←config; B(:,order(i))←f(j);
                if j==1 then temp←B; else temp←[temp; B];
            if dup==0 then out←1; config←[]; else config←temp;

        else
            if isempty(config) then out←1;
            else s←size(config); CPT ←CPD_to_CPT(Mprefnet.CPD{order(i)});
            for j←1 to s(1) do
                delete←0; for m←1 to n(order(i)) do
                    actual←config(j,:); in←actual(ps);
                     degree←int2str(CPT{subv2ind(size(CPT), [in m])}); y←1;
                     sortir←0;
                    while (y<=Critere) && (sortir==0) do
                        if (strcmp(degree(y), int2str(set(y)))==0) && (strcmp(degree(y),
                         '1')==0) then sortir←1; else y←y+1;
                    if (sortir==1) then
                        delete←delete+1;
                    else
                        actual(order(i))←m;
                        if isempty(temp) then temp←actual;
                        else temp←[temp; actual];
                config←temp;

        temp←[]; i←i+1
    return config
```

Algorithm 13: Finding the maximal set of agents which prefers a given configuration (function *accept*)

```
   Data: Mprefnet, interp, Critere
   Result: set
1  begin
2      order←topological_sort(Mprefnet.dag); n←Mprefnet.node_sizes; N←length(n)
         set←ones(1,Critere); out←0; i←1;
3      while (i<=N) && (out==0) do
4          ps←parents(Mprefnet.dag,order(i)); y←1;
5          CPT ← CPD_to_CPT(Mprefnet.CPD{order(i)});
6          if isempty(ps) then
7              degree←int2str(CPT{interp(order(i))});
8              while (y<=Critere) && (out==0) do
9                  if (strcmp(degree(y), '1')==0) then
10                     if (set(y)==1) then
11                         set(y)←str2num(degree(y));
12                     else
13                         if (strcmp(degree(y), int2str(set(y)))==0) then
14                             out←1; set←[];
15                 y←y+1;
16         in←interp(ps); m←interp(order(i)); degree←int2str(CPT{subv2ind(size(CPT), [in
             m])});
17         while (y<=Critere) && (out==0) do
18             if (strcmp(degree(y), '1')==0) then
19                 if (set(y)==1) then
20                     set(y)←str2num(degree(y));
21                 else
22                     if (strcmp(degree(y), int2str(set(y)))==0) then
23                         out=1; set=[];
24             y=y+1;
25         i=i+1 ;
26     return set
```

Example 6.8. *Now, let us execute Algorithm 13. We assume that the user selected:*

- $t\neg sc$*: the result of the procedure is set=*$[35]$ *which corresponds to old women.*
- tsc*: set=*$\emptyset$ *which means that nobody prefers this configuration.*

Algorithm 14: Verifying if a given set of agents prefers a given configuration

```
   Data: Mprefnet,interp, Critere, set
   Result: out
1  begin
2      order←topological_sort(Mprefnet.dag); n←Mprefnet.node_sizes; N←length(n);
        out←1; i←1; while (i<=N) && (out==1) do
3          ps←parents(Mprefnet.dag,order(i)); y←1;
4          CPT ← CPD_to_CPT(Mprefnet.CPD{order(i)}); if isempty(ps) then
5              degree←int2str(CPT{interp(order(i))});
6              while (y<=Critere) && (out==1) do
7                  if (strcmp(degree(y), '1')==0) &&(strcmp(degree(y), int2str(set(y)))==0)
8                   then out←0;
9
10                 y←y+1;
11         in←interp(ps); m←interp(order(i)); degree←int2str(CPT{subv2ind(size(CPT), [in
            m])}); while (y<=Critere) && (out==1) do
12             if (strcmp(degree(y), '1')==0) &&(strcmp(degree(y), int2str(set(y)))==0)
                then out←0;
13
14             y←y+1;
15         i←i+1 ;
16     return out
```

Example 6.9. *Let us execute Algorithm 14 on Example 6.6, we assume that:*

- *interp=[1 2 1] (corresponding to $t\neg sc$) and a set of agents set=[3 5] (corresponding to old women $W \cap O$). The algorithm outputs out=1. This means that the set $W \cap O$ prefers the configuration $t\neg sc$.*
- *interp=[2 1 1] (corresponding to $\neg tsc$) and a set of agents set=[3 1] (corresponding to women W). The algorithm outputs out=0. This means that the set W does not prefer the configuration $\neg tsc$*

Algorithm 15: Comparing two configurations given a set of agents

```
Data: set, Mprefnet, Critere, interp1, interp2
Result: dominant
1  begin
2      dominant←[];
3      out1 ← accept_set(Mprefnet,interp1, Critere, set);
4      out2 ← accept_set(Mprefnet,interp2, Critere, set);
5      if (out1==out2) then
6          if (out1==1) then dominant←[interp1; interp2] ;
7      else
8          if (out1==1) then
9              dominant←interp1;
10             if (out2==1) then dominant←interp2 ;
11     return dominant
```

Example 6.10. *Let us continue Example 6.6, dominance query of the set $W \cap O$ and the two configurations $t\neg sc$ and $t\neg s\neg c$ executed by Algorithm 15 outputs a matrix such that:*

$$dominant = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 2 & 2 \end{pmatrix}$$

This means that the two configurations are both equally preferred.

6.4 Conclusion

This chapter has illustrated our implementation concerning the toolbox *Prefnet* and *MPrefnet*. We have shown the use of the *Prefnet* software and the main interfaces. Besides, we have presented algorithms for dominance and optimization queries. Moreover, we also extended the toolbox to the multiple agent case. We presented the four important queries.

The multiple agent framework, proposed in this implementation, imposes the preferences of the agents to be all or nothing. As a continuity of this work, it is important to consider the extension of the toolbox for multiple agent network where preferences are graded.

CONCLUSION

Representing preferences into a compact structure aavhas become an important research topic. Graphical models are of special interest. Indeed, they facilitate elicitation, exhibit some form of independence, and serve as a basis for solving optimization and dominance queries about choices. The expressiveness of the representation setting and the complexity of answering queries are then central issues for each approach. This thesis has proposed an extensive overview of the main graphical models for preference representation and has provided a comparative survey by emphasizing their main characteristics.

This thesis has discussed the use of product-based possibilistic networks for representing conditional preference statements on discrete variables. The approach uses non-instantiated possibility weights to define conditional preference tables. Moreover, additional information about the relative strengths of symbolic weights can be taken into account. Therefore, it is an interesting compromise between the two types of models, namely, qualitative models and quantitative models. It yields a partial preference order among possible choices corresponding to a symmetric form of Pareto ordering. In the case of Boolean variables, this partial ordering coincides with the inclusion between the sets of preference statements that are violated.

The fact that at best we have some information about the relative values of these weights acknowledges the qualitative nature of preference specification. These conditional preference tables give birth to vectors of symbolic weights that reflect the preferences that are satisfied and those that are violated in a considered situation. The comparison of such vectors may rely on different orderings: the ones induced by the product-based, or the minimum-based chain rule underlying the possibilistic network, the discrimin, or leximin refinements of the minimum-based ordering, as well as Pareto ordering, and the symmetric Pareto ordering that refines it. A thorough study of the relations between these orderings in presence of vector components that are symbolic rather numerical has been presented. In particular, we have established that the product-based ordering and the symmetric Pareto ordering coincide in presence of constraints comparing pairs of symbolic weights. This ordering agrees in the Boolean case with the inclusion between the sets of preference statements that are violated. The symmetric Pareto ordering may be itself refined by

the leximin ordering. We have highlighted the merits of product-based possibilistic networks for representing preferences and the flexibility and the representational power of the approach.

The main purpose of any preference model is to support answering various queries about the preference of the decision maker. These queries might be summed up into two main tasks namely optimization and dominance. In this thesis, we have investigated the general procedures of both these tasks with respect to possibilistic symbolic preference networks. We have proved that optimisation query can be answered easily by traversing the graph from roots to leaves and choosing each time the best value depending on the context. Thus, this query is linear with the number of nodes. Besides, we have showed that dominance queries correspond to comparing vectors of weights. This product-based comparison is defined by reorganizing the vectors given the inequality constraints between the symbolic weights. The complexity of this query is $O(N!)$ where N corresponds to the number of nodes.

Furthermore, this graphical model has two logical counterparts in terms of possibilistic logic and penalty logic. These logical counterparts are of use for reasoning tasks such as consistency checking and up dating. Transformation to possibilistic logic base is provided as well as an intermediate format corresponding to hybrid π-pref nets where we associate to each node a local symbolic possibilistic logic base instead of a conditional possibility table. Besides, transformation to penalty logic can be performed by a logarithmic transformation of the possibility degrees and the union of local penalty logic bases.

Among several graphical models for preferences, CP-nets are often used for learning and representation purposes. They rely on a simple preference independence property known as the *ceteris paribus independence*. In this thesis we have proved that π-pref nets induce a preference ordering on configurations consistent with the ordering induced by CP-nets. Ceteris paribus preferences in the latter can be retrieved by adding suitable constraints between products of symbolic weights. This connection between possibilistic networks and CP-nets allows for an extension of the expressive power of the latter while maintaining its qualitative nature. Elicitation complexity is thus kept stable, while the complexity of dominance and optimization queries is cut down.

A multiple-agent logic, which associates subsets of agents to logical formulas, has been recently proposed. We have presented a graphical counterpart of this logic, based on multiple agent version of possibilistic conditioning, and applied it to preference modeling. First, preferences of agents are supposed to be all or nothing. We have discussed how one can move from the network to the logic representation and vice-versa. The new representation enables us to focus on networks associated to subsets of agents, and to identify inconsistent agents, or conflicting subsets of agents. The question of optimization and dominance queries is discussed.

Besides, we have proposed an extension of multiple agents networks where gradual preferences are handled. In this extended graphical network, conditional weights are then associated with both sets of agents and preference levels. A degree α/A of a configuration ω then expresses that 'at most all the agents in A find the configuration acceptable with a maximal satisfaction degree equal to α'. The semantics is then given in terms of fuzzy sets of agents that find a configuration

more or less satisfactory.

Finally, we have ended this thesis by an implementation of a toolbox for the two main graphical models proposed in this thesis namely, π-Pref nets and ma-nets. We have presented the most important algorithms to handle the different queries proposed all along the chapters.

The research done in this dissertation opens up some interesting lines of future research, some of theoretical nature, and others on practical issues. They concern either π-Pref nets or their extension to multiple agents preference handling.

- Various extensions of this new framework seem to be interesting. In particular, we might think on including a bipolar representation of preferences [Benferhat et al., 2002b]. Indeed, preferences in everyday life come in two forms: positive preferences that describe what is satisfactory and negative preferences that refer to what is unacceptable. The framework of possibility theory allows to represent such preferences, but without graphic counterpart (by possibilistic networks) to the positive part. Thus, our aim will be to offer a compact representation of preferences in terms of generalized possibilistic networks.

- Uncertainty is something worth investigating in conjunction with the preferences, it concerns those of an agent or agent-variability. In the field of preference representation, some models have been proposed to deal with uncertainty, including probabilistic CP-nets [Bigot et al., 2013]. It is natural to ask the question of the possible interest of possibilistic CP-nets, and extend the scope of representation of preferences by possibilistic networks taking account of uncertainty.

- Another problem, as part of the continuity of the previous, concerns the learning of the proposed models. Learning preferences in the form of probabilistic CP-net has already been studied [da Silva and de Amo, 2011]. This learning is based on Bayesian networks. We should consider the development of methods and algorithms for learning possibilistic networks for preferences.

- For multiple agents preference model, We may think about the symmetrical form of formulas (a, A), where the roles played by a and A should be studied, particularly, for inducing preferences of a subset of agents such that, $B \subset A$, or $A \cap B = \emptyset$. In fact, based on such symmetrical form, we may think of inducing another type of networks associated to each interpretation.

- Computational issues are also of interest. We may think of proposing algorithms to answer ordering queries and other possible variants in a local manner. For instance, we may think of identifying non consistent agents directly from multi-agents (possibilistic) networks.

- One may also think of several extensions, where the set of agents are described under the form of logical formulas which would allow us to consider the disjunction and the negation between the characteristics of each set of agents.

Bibliography

[Allen, 2013] Allen, T. (2013). CP-nets with indifference. In *Proc. Communication, Control, and Computing (Allerton)*, pages 1488–1495.

[Allen, 2014] Allen, T. E. (2014). Making CP-nets (more) useful. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI'14)*, pages 3057–3058.

[Allen et al., 2015] Allen, T. E., Chen, M., Goldsmith, J., Mattei, N., Popova, A., Regenwetter, M., Rossi, F., and Zwilling, C. (2015). Beyond theory and data in preference modeling: Bringing humans into the loop. In *Proc. International Conference on Algorithmic Decision Theory (ADT'15)*, pages 3–18.

[Allen et al., 2016] Allen, T. E., Goldsmith, J., Justice, H. E., Mattei, N., and Raines, K. (2016). Generating CP-nets uniformly at random. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI'16)*, pages 872–878.

[Allen et al., 2014] Allen, T. E., Goldsmith, J., and Mattei, N. (2014). Counting, ranking, and randomly generating CP-nets. In *Proc. Multidisciplinary Workshop on Advances in Preference Handling (MPREF'14)*.

[Arnborg et al., 1987] Arnborg, S., Corneil, D. G., and Proskurowski, A. (1987). Complexity of finding embeddings in ak-tree. *Journal on Algebraic Discrete Methods*, 8(2):277–284.

[Bacchus and Grove, 1995] Bacchus, F. and Grove, A. (1995). Graphical models for preference and utility. In *Proc. Uncertainty in Artificial Intelligence (UAI'95)*, pages 3–10.

[Belhadi et al., 2013] Belhadi, A., Dubois, D., Khellaf-Haned, F., and Prade, H. (2013). Multiple agent possibilistic logic. *Journal of Applied Non-Classical Logics*, 23:299–320.

[Ben Amor, 2012] Ben Amor, N. (2012). *Qualitative Possibilitic Graphical Models: From Independence to Propagation Algorithms*. PhD thesis, Institut Supérieur de Gestion Tunis, Tunis, Tunisie.

[Ben Amor and Benferhat, 2005] Ben Amor, N. and Benferhat, S. (2005). Graphoid properties of qualitative possibilistic independence relations. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 13(1):59–96.

[Ben Amor et al., 2003] Ben Amor, N., Benferhat, S., and Mellouli, K. (2003). Anytime propagation algorithm for min-based possibilistic graphs. *Soft Computing*, 8(2):150–161.

[Ben Amor et al., 2014] Ben Amor, N., Dubois, D., Gouider, H., and Prade, H. (2014). Possibilistic networks: A new setting for modeling preferences. In *Proc. Scalable Uncertainty Management (SUM'14)*, pages 1–7.

[Ben Amor et al., 2015] Ben Amor, N., Dubois, D., Gouider, H., and Prade, H. (2015). Possibilistic conditional preference networks. In *Proc. European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQUARU'15)*, pages 36–46.

[Ben Amor et al., 2016a] Ben Amor, N., Dubois, D., Gouider, H., and Prade, H. (2016a). Graphical models for preference representation: An overview. In *Proc. Scalable Uncertainty Management (SUM'16)*, pages 96–111.

[Ben Amor et al., 2016b] Ben Amor, N., Dubois, D., Gouider, H., and Prade, H. (2016b). Preference modeling with possibilistic networks and symbolic weights: A theoretical study. In *Proc. European Conference on Artificial Intelligence (ECAI'16)*, pages 1203–1211.

[Ben Amor et al., 2017a] Ben Amor, N., Dubois, D., Gouider, H., and Prade, H. (2017a). Expressivity of possibilistic preference networks with constraints. In *Proc. Scalable Uncertainty Management - 11th International Conference, SUM 2017*, pages 163–177.

[Ben Amor et al., 2017b] Ben Amor, N., Dubois, D., Gouider, H., and Prade, H. (2017b). Graphical representations of multiple agent preferences. In *Proc. Advances in Artificial Intelligence: From Theory to Practice - 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*, pages 142–153.

[Benferhat et al., 2002a] Benferhat, S., Dubois, D., Garcia, L., and Prade, H. (2002a). On the transformation between possibilistic logic bases and possibilistic causal networks. *International Journal of Approximate Reasoning*, 29(2):135–173.

[Benferhat et al., 2001a] Benferhat, S., Dubois, D., Kaci, S., and Prade, H. (2001a). Bridging logical, comparative, and graphical possibilistic representation frameworks. In *Proc. European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'01)*, pages 422–431.

[Benferhat et al., 2002b] Benferhat, S., Dubois, D., Kaci, S., and Prade, H. (2002b). Bipolar representation and fusion of preferences on the possibilistic logic framework. volume 2, pages 421–432.

[Benferhat et al., 1997] Benferhat, S., Dubois, D., and Prade, H. (1997). Nonmonotonic reasoning, conditional objects and possibility theory. *Artificial Intelligence*, 92(1):259–276.

[Benferhat et al., 1998] Benferhat, S., Dubois, D., and Prade, H. (1998). From semantic to syntactic approaches to information combination in possibilistic logic. In *Aggregation and Fusion of Imperfect Information*, pages 141–161. Springer.

[Benferhat et al., 2001b] Benferhat, S., Dubois, D., and Prade, H. (2001b). Towards a possibilistic logic handling of preferences. *Applied Intelligence*, 14(3):303–317.

[Benferhat and Smaoui, 2007] Benferhat, S. and Smaoui, S. (2007). Hybrid possibilistic networks. *International journal of approximate reasoning*, 44(3):224–243.

[Bigot et al., 2013] Bigot, D., Zanuttini, B., Fargier, H., and Mengin, J. (2013). Probabilistic conditional preference networks. In *Proc. Uncertainty in Artificial Intelligence*.

[Boutilier et al., 2001] Boutilier, C., Bacchus, F., and Brafman, R. I. (2001). UCP-networks: A directed graphical representation of conditional utilities. In *Proc. Association for Uncertainty in Artificial Intelligence (UAI'01)*, pages 56–64.

[Boutilier et al., 1999] Boutilier, C., Brafman, Ronen, I., Hoos, H., and Poole, D. (1999). Reasoning with conditional ceteris paribus preference statements. In *Proc. Association for Uncertainty in Artificial Intelligence (UAI'99)*, pages 71–80.

[Boutilier et al., 2004a] Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., and Poole, D. (2004a). CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research (JAIR)*, 21:135–191.

[Boutilier et al., 2004b] Boutilier, G., Brafman, R., Domshlak, C., Hoos, H., and Poole, D. (2004b). Preference-based constrained optimization with CP-nets. *Computational Intelligence*, 20(2):137–157.

[Bouveret et al., 2009] Bouveret, S., Endriss, U., Lang, J., et al. (2009). Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI'09)*.

[Bouyssou and Vincke, 2010] Bouyssou, D. and Vincke, P. (2010). Binary relations and preference modeling. *Decision-making Process: Concepts and Methods*, pages 49–84.

[Brafman and Dimopoulos, 2004] Brafman, R. I. and Dimopoulos, Y. (2004). Extended semantics and optimization algorithms for CP-networks. *Computational Intelligence*, 20(2):218–245.

[Brafman and Domshlak, 2002] Brafman, R. I. and Domshlak, C. (2002). TCP-nets for preference-based product configuration. In *Proc. Workshop on Configuration (ECAI'02)*, pages 101–106.

[Brafman et al., 2006] Brafman, R. I., Domshlak, C., and Shimonyl, S. (2006). On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research (JAIR)*, pages 389–424.

[Brafman and Engel, 2009] Brafman, R. I. and Engel, Y. (2009). Directional decomposition of multiattribute utility functions. In *Proc. Algorithmic Decision Theory (ADT'09)*, pages 192–202.

[Brafman and Engel, 2010] Brafman, R. I. and Engel, Y. (2010). Decomposed utility functions and graphical models for reasoning about preferences. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI'10)*, pages 267–272.

[Braziunas and Boutilier, 2005] Braziunas, D. and Boutilier, C. (2005). Local utility elicitation in GAI models. In *Proc. Uncertainty in Artificial Intelligence (UAI'05)*, pages 42–49.

[Cayrol et al., 2014] Cayrol, C., Dubois, D., and Touazi, F. (2014). Ordres partiels entre sous-ensembles d'un ensemble partiellement ordonné. Research report RR–2014-02–FR, IRIT, Université Paul Sabatier, Toulouse.

[Cooper, 1990] Cooper, G. F. (1990). The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2-3):393–405.

[da Silva and de Amo, 2011] da Silva, F. and de Amo, S. (2011). CPrefMiner: A bayesian miner of conditional preferences. *Journal of Information and Data Management (JIDM)*, 2(1):35–42.

[Dawid, 1992] Dawid, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2(1):25–36.

[De Saint-Cyr et al., 1994] De Saint-Cyr, F. D., Lang, J., and Schiex, T. (1994). Penalty logic and its link with Dempster-Shafer theory. In *Proc. Uncertainty in Artificial Intelligence (UAI'94)*, pages 204–211.

[Debreu, 1959] Debreu, G. (1959). *Theory of value: An axiomatic analysis of economic equilibrium*. Number 17. Yale University Press.

[Dimopoulos et al., 2009] Dimopoulos, Y., Michael, L., and Athienitou, F. (2009). Ceteris paribus preference elicitation with predictive guarantees. In *Proc. Joint Conference on Artificial Intelligence (IJCAI'09)*, volume 9, pages 1–6.

[Domshlak, 2008] Domshlak, C. (2008). A snapshot on reasoning with qualitative preference statements in ai. In *Proc. Preferences and similarities*, pages 265–282.

[Domshlak et al., 2001] Domshlak, C., Brafman, R. I., and Shimony, S. E. (2001). Preference-based configuration of web page content. In *Proc. international Joint Conference on Artificial Intelligence (IJCAI'01)*, volume 17, pages 1451–1456. Citeseer.

[Domshlak et al., 2011] Domshlak, C., Hüllermeier, E., Kaci, S., and Prade, H. (2011). Preferences in AI: An overview. *Artificial Intelligence*, 175(7-8):1037–1052.

[Dubois et al., 1996] Dubois, D., Fargier, H., and Prade, H. (1996). Refinements of the maximin approach to decision-making in fuzzy environment. *Fuzzy Sets and Systems*, 81:103–122.

[Dubois et al., 2015] Dubois, D., Hadjali, A., Prade, H., and Touazi, F. (2015). Erratum to: Database preference queries - a possibilistic logic approach with symbolic priorities. *Annals of Mathematics and Artificial Intelligence*, 73(3-4):359–363.

[Dubois et al., 2006] Dubois, D., Kaci, S., and Prade, H. (2006). Approximation of conditional preferences networks "CP-nets" in possibilistic logic. In *Proc. IEEE International Conference on Fuzzy Systems*, pages 2337–2342.

[Dubois et al., 1994] Dubois, D., Lang, J., and Prade, H. (1994). Possibilistic logic. In Gabbay, D., Hogger, C., Robinson, J., and Nute, D., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3*, pages 439–513. Oxford University Press.

[Dubois and Prade, 1988] Dubois, D. and Prade, H. (1988). *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press.

[Dubois and Prade, 1991] Dubois, D. and Prade, H. (1991). Epistemic entrenchment and possibilistic logic. *Artificiel intelligence*, 50(2):223–239.

[Dubois and Prade, 1995] Dubois, D. and Prade, H. (1995). Possibility theory as a basis for qualitative decision theory. In *Proc. International Joint Conference on Artificial Intelligence,*, volume 95, pages 1924–1930.

[Dubois and Prade, 2004] Dubois, D. and Prade, H. (2004). Possibilistic logic: a retrospective and prospective view. *Fuzzy Sets and Systems*, 144(1):3–23.

[Dubois and Prade, 2016] Dubois, D. and Prade, H. (2016). Qualitative and semi-quanlitative modeling of uncertain knowledge-a discussion. In *Proc. Computational Models of Rationality Essays Dedicated to Gabriele Kern-Isberner on the occasion of her 60th Birthday*, pages 280–292. College Publications.

[Dubois et al., 2013a] Dubois, D., Prade, H., and Touazi, F. (2013a). Conditional preference nets and possibilistic logic. In *Proc. European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'13)*, pages 181–193.

[Dubois et al., 2013b] Dubois, D., Prade, H., and Touazi, F. (2013b). Conditional Preference-nets, possibilistic logic, and the transitivity of priorities. In *Proc. SGAI International Conference of Artificial Intelligence*, pages 175–184.

[Dubus et al., 2009] Dubus, J., Gonzales, C., and Perny, P. (2009). Choquet optimization using GAI networks for multiagent/multicriteria decision-making. In *Algorithmic Decision Theory (ADT'09))*, pages 377–389.

[Eichhorn et al., 2016] Eichhorn, C., Fey, M., and Kern-Isberner, G. (2016). CP-and OCF-networks–a comparison. *Fuzzy Sets and Systems*, 298:109–127.

[Eichhorn and Kern-Isberner, 2015] Eichhorn, C. and Kern-Isberner, G. (2015). Using inductive reasoning for completing ocf-networks. *Journal of Applied Logic*, 13(4):605–627.

[Engel and Wellman, 2008] Engel, Y. and Wellman, M. P. (2008). CUI networks: A graphical representation for conditional utility independence. *Journal of Artificial Intelligence Research (JAIR)*, 31:83–112.

[Fargier et al., 2012] Fargier, H., Lang, J., Mengin, J., and Schmidt, N. (2012). Issue-by-issue voting: an experimental evaluation. *Proc. Workshop on Advances in Preference Handling (MPREF'12)*, 2012.

[Fishburn, 1970] Fishburn, P. C. (1970). Utility theory for decision making. Technical report, DTIC Document.

[Fonck, 1994] Fonck, P. (1994). Conditional independence in possibility theory. In *Proc. Uncertainty in Artificial Intelligence (UAI'94)*, pages 221–226.

[Gérard et al., 2007] Gérard, R., Kaci, S., and Prade, H. (2007). Ranking alternatives on the basis of generic constraints and examples-a possibilistic approach. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 393–398.

[Goldsmith et al., 2008] Goldsmith, J., Lang, J., Truszczynski, M., and Wilson, N. (2008). The computational complexity of dominance and consistency in CP-nets. *Journal of Artificial Intelligence Research (JAIR)*, pages 403–432.

[Gonzales and Perny, 2004] Gonzales, C. and Perny, P. (2004). GAI networks for utility elicitation. In *Proc. Conference on Principles of Knowledge Representation and Reasoning*, pages 224–234.

[Gonzales and Perny, 2005] Gonzales, C. and Perny, P. (2005). GAI networks for decision making under certainty. In *Proc. IJCAI–Workshop on Advances in Preference Handling*.

[Guerin et al., 2013] Guerin, J. T., Allen, T. E., and Goldsmith, J. (2013). Learning cp-net preferences online from user queries. In *International Conference on Algorithmic Decision Theory (ADT'13)*, pages 208–220.

[Hadjali et al., 2008] Hadjali, A., Kaci, S., and Prade, H. (2008). Database preferences queries–a possibilistic logic approach with symbolic priorities. volume 4932, pages 291–310.

[Huang and Darwiche, 1996] Huang, C. and Darwiche, A. (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263.

[Jensen et al., 1990] Jensen, F. V., Lauritzen, S. L., and Olesen, K. G. (1990). Bayesian updating in causal probabilistic networks by local computations. *Proc. Comput. Stat. Quarterly*, 4:269–282.

[Kaci et al., 2006] Kaci, S., Dubois, D., and Prade, H. (2006). Representing preferences in the possibilistic setting. In *Proc. agstuhl Seminar*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[Kaci et al., 2014] Kaci, S., Lang, J., and Perny, P. (2014). Représentation des préférences. In *Panorama de l'Intelligence Artificielle*, volume 1, pages 181–214. Cépaduès éditions.

[Keeney and Raiffa, 1993] Keeney, R. L. and Raiffa, H. (1993). *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press.

[Kjærulff, 1990] Kjærulff, U. (1990). Triangulation of graphs–algorithms giving small total state space.

[Koriche and Zanuttini, 2010] Koriche, F. and Zanuttini, B. (2010). Learning conditional preference networks. *Artificial Intelligence*, 174(11):685–703.

[Kushmerick et al., 1995] Kushmerick, N., Hanks, S., and Weld, D. S. (1995). An algorithm for probabilistic planning. *Artificial Intelligence*, 76(1):239–286.

[Lang and Mengin, 2009] Lang, J. and Mengin, J. (2009). The complexity of learning separable ceteris paribus preferences. In *Proc. international Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 848–853.

[Lang et al., 2012] Lang, J., Mengin, J., and Xia, L. (2012). Aggregating conditionally lexicographic preferences on multi-issue domains. In *Proc. Principles and Practice of Constraint Programming*, pages 973–987.

[Liu and Liao, 2015] Liu, J. and Liao, S. (2015). Expressive efficiency of two kinds of specific CP-nets. *Information Sciences*, 295:379–394.

[Liu, 2016] Liu, X. (2016). *Modeling, learning and reasoning about preference trees over combinatorial domains*. University of Kentucky.

[Liu and Truszczynski, 2013] Liu, X. and Truszczynski, M. (2013). Aggregating conditionally lexicographic preferences using answer set programming solvers. In *Proc. International Conference on Algorithmic Decision Theory (ADT'13)*, pages 244–258.

[Liu and Truszczynski, 2014] Liu, X. and Truszczynski, M. (2014). Preference trees: a language for representing and reasoning about qualitative preferences. In *Proc. Multidisciplinary Workshop on Advances in Preference Handling (MPREF)*, pages 55–60.

[Liu and Truszczynski, 2015] Liu, X. and Truszczynski, M. (2015). Learning partial lexicographic preference trees over combinatorial domains. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI'15)*, volume 15, pages 1539–1545.

[Nilsson, 1998] Nilsson, D. (1998). An efficient algorithm for finding the m most probable configurations in probabilistic expert systems. *Statistics and Computing*, 8(2):159–173.

[Orlin et al., 2010] Orlin, J. B., Madduri, K., Subramani, K., and Williamson, M. (2010). A faster algorithm for the single source shortest path problem with few distinct positive lengths. *Journal of Discrete Algorithms*, 8(2):189–198.

[Patel, 2016] Patel, N. (2016). *Preference Handling in Decision-Making Problems. (Mise en œuvre de préférences dans les problèmes de décision)*. PhD thesis, University of Montpellier, France.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[Rossi et al., 2004] Rossi, F., Venable, K., and Walsh, T. (2004). mCP-nets: representing and reasoning with preferences of multiple agents. In *Proc. 19th Association for the Advancement of Artificial Intelligence*, pages 729–734.

[Santhanam et al., 2016] Santhanam, G. R., Basu, S., and Honavar, V. (2016). Representing and reasoning with qualitative preferences: Tools and applications. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(1):1–154.

[Spohn, 1988] Spohn, W. (1988). Ordinal conditional functions: a dynamic theory of epistemic states. In Harper, W. L. and Skyrms, B., editors, *Causation in Decision, Belief Change, and Statistics*, volume 2, pages 105–134. D. Reidel.

[Tversky and Kahneman, 1975] Tversky, A. and Kahneman, D. (1975). Judgment under uncertainty: Heuristics and biases. In *Proc. Utility, probability, and human decision making*, pages 141–162.

[Wilson, 2004] Wilson, N. (2004). Extending CP-nets with stronger conditional preference statements. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI'04)*, volume 4, pages 735–741.

[Wilson, 2011] Wilson, N. (2011). Computational techniques for a simple theory of conditional preferences. *Artificial Intelligence*, 175(7-8):1053–1091.

[Xin and Liu, 2016] Xin, G. and Liu, J. (2016). Exact dominance querying algorithm on cp-nets. *International Journal of Database Theory and Application*, 9(5):21–36.

[Yaman and Desjardins, 2008] Yaman, F. and Desjardins, M. (2008). More-or-less cp-networks. *Association for Uncertainty in Artificial Intelligence (UAI'08)*.

[Zadeh, 1978] Zadeh, L. A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28.

APPENDIX A

Notations

Notion	Notation
Basic notions	
A set of variables	$\mathcal{V} = \{A_1, A_2, ..., A_N\}$
Subsets of variables	$W,\ Z,\ Y,\ X$
Finite domain associated with variable A_i	D_{A_i}
The universe of discourse, which is the Cartesian product of all variable domains in $\mathcal{V}$	$\Omega = \times_{A_i \in \mathcal{V}} D_{A_i}$
A configuration	ω
The restriction of ω to variables in X	$\omega[X]$
A good quality of the choice for a variable A	a^+
A bad quality of the choice for a variable A	a^-
Operations over sets	$\cap,\ \cup,\ \subset\ \ \setminus$
Complement of set X referring to variables not in X	$\overline{X}$
Propositional formulas	$\phi, \psi, \cdots$
Inference	$\models$
Logical operators	$\wedge, \vee, \neg, \rightarrow$
Tautology	$\top$
Contradiction	$\bot$
Sets of nodes	
A node	A_i
The set of parents of A_i	$\mathcal{P}(A_i)$
Instantiation of $\mathcal{P}(A_i)$	$p(A_i)$
The Set of descendants of A_i	$Dn(A_i)$
The set of non descendants of A_i	$Co(A_i) = \mathcal{V} \setminus (Dn(A_i) \cup \mathcal{P}(A_i) \cup A_i)$
The children set of A_i	$\mathcal{C}h(A_i)$
Binary relations	

Indifference	$\sim$
Incomparability	$\pm$
Strictly preferred	$\succ$
Importance relation	$\triangleright$
π-Pref net	
Constraints between symbolic weights pertaining to preference statements	$\mathcal{C}_0$
Additional constraints added by the user	$\mathcal{C}_1$
Set of all the constraints	$\mathcal{C}$
Relations d'ordres sur des vecteurs	
Product dominance relation	$\succ_{prod}$
$\min$ dominance relation	$\succ_{\min}$
Discrimin dominance relation	$\succ_{discrimin}$
Leximin dominance relation	$\succ_{leximin}$
Pareto dominance relation	$\succ_{Pareto}$
Symmetric Pareto dominance relation	$\succ_{SP}$
Possibilistic logic	
Possibilistic Formula	(f_i, c_i)
Possibilistic logic base	Σ
Possibility degree of a configuration ω	$\pi(\omega)$
Possibility measure of an event $F \subseteq \Omega$	$\Pi(F)$
Necessity measure of an event $F \subseteq \Omega$	$N(F)$
Multiple agent logic	
A set of agents	P_i
A propositional formula	p_i
Possibilistic Formula	(p_i, P_i)
The set of all the agents	All
Multiple agent logic base	Γ
Possibility degree of a configuration ω	$\boldsymbol{\pi}(\omega)$
Possibility measure of an event $F \subseteq \Omega$	$\boldsymbol{\Pi}(F)$
Necessity measure of an event $F \subseteq \Omega$	$\boldsymbol{N}(F)$

Table A.1: Notations

APPENDIX B

List of publications

National conferences :

- Nahla Ben Amor, Didier Dubois, Héla Gouider, Henri Prade. Représentation graphique de préférences multi-agents. Rencontres Francophones sur la Logique Floue et ses Applications, LFA 2016, La Rochelle, France, Cépaduès Editions, November 2016.

International conferences :

- Nahla Ben Amor, Didier Dubois, Héla Gouider, Henri Prade. Expressivity of possibilistic preference networks with constraints. In Proceedings 11th International Conference on Scalable Uncertainty Management, Granada, Spain, October 4-6, 2017.
- Nahla Ben Amor, Didier Dubois, Héla Gouider, Henri Prade. Graphical Representations of Multiple Agent Preferences. In Proceedings 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017, Arras, France, June 27-30, 2017.
- Nahla Ben Amor, Didier Dubois, Héla Gouider, Henri Prade. Graphical Models for Preference Representation: An Overview. In Proceedings 10th International Conference in Scalable Uncertainty Management, SUM 2016, Nice, France, September 21-23, 2016.
- Nahla Ben Amor, Didier Dubois, Héla Gouider, Henri Prade. Preference Modeling with Possibilistic Networks and Symbolic Weights: A Theoretical Study. In Proceedings 22nd European Conference on Artificial Intelligence, ECAI 2016, The Hague, The Netherlands, 29 August-2 September 2016.
- Nahla Ben Amor, Didier Dubois, Héla Gouider, Henri Prade. Possibilistic Conditional Preference Networks. In Proceedings 13th European Conference in Symbolic

and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2015, Compiègne, France, July 15-17, 2015.

Journals :

- Nahla Ben Amor, Didier Dubois, Héla Gouider, Henri Prade. Possibilistic preference networks. Information Sciences, 2017.