Pontifícia Universidade Católica
do Rio de Janeiro

Ingrid Oliveira de Nunes

User-centric Preference-based Decision Making

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Doutor em Informática.

Advisor: Prof. Carlos José Pereira de Lucena

Rio de Janeiro
September 2012

Pontifícia Universidade Católica
do Rio de Janeiro

Ingrid Oliveira de Nunes

User-centric Preference-based Decision Making

Thesis presented to the Programa de Pós-Graduação em Informática, of the Departamento de Informática do Centro Técnico Científico da PUC-Rio, as partial fulfillment of the requirements for the degree of Doutor.

Prof. Carlos José Pereira de Lucena
Advisor
Departamento de Informática — PUC-Rio

Prof. Simone Diniz Junqueira Barbosa
Departamento de Informática – PUC-Rio

Prof. Hugo Fuks
Departamento de Informática – PUC-Rio

Prof. Rafael Heitor Bordini
PUC/RS

Prof. Jaime Simão Sichman
USP

Prof. José Eugenio Leal
Coordinator of the Centro Técnico Científico da PUC-Rio

Rio de Janeiro, September 20, 2012

Ingrid Oliveira de Nunes

She has obtained the Master degree in Informatics at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), in 2009, Rio de Janeiro, Brazil, and received the degree of Bachelor in Computer Science at the Federal University of Rio Grande do Sul (UFRGS), in 2006, Porto Alegre, Brazil. During her Ph.D., she visited the University of Waterloo (Canada), with two three-month research visits, and King's College London (UK) for one year, as part of the CNPq Sandwich Ph.D. Programme. She also worked as a software developer at the e-Core Desenvolvimento de Software company from 2005 to 2007. Her main research interests are multi-agent systems, decision making, preference reasoning and agent-oriented software engineering.

Bibliographic data

Nunes, Ingrid Oliveira de

User-centric Preference-based Decision Making / Ingrid Oliveira de Nunes; orientador: Carlos José Pereira de Lucena. — 2012.

298 f.: il. ; 30 cm

1. Tese (doutorado) — Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2012.

Inclui bibliografia

1. Informática – Teses. 2. Tomada de Decisão. 3. Representação de Preferências. 4. Raciocínio sobre Preferências. 5. Explicação a Usuários. 6. Raciocínio Humano. 7. Sistemas de Suporte à Decisão. I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To my parents, Daltro and Suzana.
To my brothers, Gustavo and Matthias.
I love you all.

Acknowledgments

I must first express my gratitude towards my supervisor, Prof. Carlos Lucena, who is one of the leading Computer Science researchers in Brazil. It was a privilege to have the opportunity to work with him during my master and Ph.D. His guidance was essential to complete this thesis and make me an independent researcher. I also give my sincere thanks to Prof. Michael Luck and Dr. Simon Miles, my supervisors during the sandwich Ph.D. period at King's College London. I am grateful for their insightful comments both in my work and in this thesis, for their support, and for many motivating discussions.

I would like to thank the other members of my examining committee, Prof. Rafael Bordini, Prof. Jaime Sichman, Prof. Simone Barbosa and Prof. Hugo Fuks, for taking the time of reading my thesis and giving me constructive feedbacks. Prof. Simone also contributed to this thesis with fruitful discussions, and had an important role in the work that was the seed of this thesis. Thanks to Prof. Don Cowan for his feedback and making my research visits to University of Waterloo possible.

I would also like to thank all my friends and colleagues from PUC-Rio and King's College London for their help, support and friendship. I will mention some of them, and I apologise for those not explicitly mentioned. Many thanks to Elder Cirilo for our engaging joint work, discussions, and friendship. The research visits to UW would not have been so productive and fun without him. Thanks to Camila Nunes, Francisco Dantas and Isela Macia, who started their Ph.D. at the same time as me, and became wonderful friends and colleagues. A special thanks also goes to Vera Menezes for her friendship and taking care of me during this journey in Rio. Daniele Nantes, Luis Fernando Dalla Santa, and Lina Barakat, thank you for making my period in London so nice. My friends from Porto Alegre, Ana Maria Souza, Daniela Malvasio, Graziela Becker and Paula Kruger, thank you for always being there. I also thank Prof. Carlos Lisbôa and Prof. Maria Lúcia Lisbôa for their friendship and for having encouraged me in some of my important achievements.

I am and will always be extremely grateful to my beloved parents, Daltro Nunes and Suzana Nunes, who raised me and taught me to give priority in my life to the quest for knowledge. Thanks for their love and support throughout my studies and work, and being examples of the kind of person I want to be. I also thank my brothers, Gustavo Nunes and Matthias Nunes, for being so supportive.

In conclusion, I recognise that this research would not have been possible without the financial assistance of CNPq, FAPERJ and CAPES, and infrastructure of the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) and King's College London, and express my gratitude to those agencies and universities.

Resumo

Nunes, Ingrid Oliveira de; Lucena, Carlos José Pereira de. **Tomada de Decisão baseada em Preferências e centrada no Usuário**. Rio de Janeiro, 2012. 298p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A escolha de uma entre um conjunto de opções disponível normalmente requer a resolução de *trade-offs*, contudo esperar que as pessoas avaliem cada uma das opções de um grande conjunto pode ser inviável devido ao tempo e ao esforço cognitivo necessários para realizar tal análise, fazendo com que elas fiquem freqüentemente insatisfeitas com suas escolhas. Sistemas de software podem dar suporte à tomada de decisão humana ou mesmo automatizar esse processo, entretanto existem muitos desafios que estão associados com o oferecimento de tal suporte. Esta tese lida, em particular, com três destes desafios: (i) como representar preferências dos usuários; (ii) como raciocinar sobre estas preferências e tomar decisões; e (iii) como justificar tais decisões. Diferentes abordagens têm sido propostas para a representação e raciocínio sobre preferências qualitativas, mas estas abordagens lidam com um conjunto restrito de tipos de preferências, e portanto não são capazes de processar preferências fornecidas por usuários em muitos cenários realistas. Nesta tese, apresentam-se três principais contribuições. A primeira delas consiste de um novo metamodelo de preferências, o qual foi desenvolvido de acordo com um estudo sobre a expressão de preferências, permitindo a representação de preferências em alto-nível. Segundo, uma nova técnica de tomada de decisão automatizada é proposta, a qual escolhe uma opção de um conjunto de opções disponível baseada em preferências expressas em uma linguagem construída de acordo com o metamodelo proposto, explorando termos da linguagem natural, tais como atos de fala expressivos. A técnica vai além das preferências fornecidas para tomar a decisão através da incorporação de princípios da psicologia, que focam como os humanos tomam decisões, já que as preferências fornecidas tipicamente não são suficientes para resolver *trade-offs* entre as opções disponíveis. Terceiro, apresenta-se uma técnica de geração de explicação, que utiliza modelos construídos pela técnica de tomada de decisão para justificar escolhas, e segue diretrizes e padrões que foram derivados de um estudo sobre explicações a respeito de escolhas, também realizado no contexto desta tese. Um estudo com usuários foi feito para avaliar a abordagem, o qual mostra que (i) a linguagem de preferências é adequada para usuários expressarem suas preferências, que (ii) a técnica de tomada de decisão faz escolhas que os usuários consideram de alta qualidade, e que (iii) as explicações fornecidas permitem que usuários entendam por que a escolha foi feita, bem como melhora a confiança na decisão tomada.

Palavras-chave

Tomada de Decisão. Representação de Preferências. Raciocínio sobre Preferências. Explicação a Usuários. Raciocínio Humano. Sistemas de Suporte à Decisão.

Abstract

Nunes, Ingrid Oliveira de; Lucena, Carlos José Pereira de. **User-centric Preference-based Decision Making**. Rio de Janeiro, 2012. 298p. DSc Thesis — Departmento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Choosing from a set of available options often requires resolution of trade-offs but it can be unfeasible for humans to carefully evaluate each option of a large set due to the required time and cognitive effort. Consequently, they are often unsatisfied with their choices. Software systems can support human decision making or even automate this process, but there are many challenges associated with the provision of such support. In this thesis we deal in particular with three of them: (i) how to represent user preferences; (ii) how to reason about preferences and make decisions; and (iii) how to justify such decisions. Different approaches have been proposed for representing and reasoning about qualitative preferences, but they address a restricted set of preference types, and therefore are not able to process preferences provided by users in many realistic scenarios. This thesis provides three main contributions. First, we introduce a new preference metamodel founded on a study of how humans express preferences, allowing the representation of high-level preferences. Second, we propose an automated decision making technique, which chooses an option from a set available based on preferences expressed in a language based on our metamodel, exploiting natural-language terms. Our technique goes beyond the provided preferences to make a decision with the incorporation of psychology principles, which concern how humans make decisions, as the provided preferences are typically not enough to resolve trade-offs among available options. Third, we present an explanation generation technique, which uses models built by our decision making technique to justify choices, and follows guidelines and patterns that we derived from a study of choice explanation. A user study was performed to evaluate our approach, which shows that (i) our preference language is adequate for users to express their preferences, (ii) our decision making technique makes choices that users consider as having good quality, and (iii) the provided explanations allow users to understand why the choice was made and improves the confidence in the decision.

Keywords

Decision making. Preference Representation. Preference Reasoning. User Explanations. Human Reasoning. Decision Support Systems.

Table of Contents

List of Figures

List of Tables

Abbreviation and Acronym List

AI artificial intelligence

AMVF additive multi-attribute value function

AVPO attribute value partial order

BMO Best-Matches-Only

CSP Constraint Satisfaction Problem

CIT conditional importance table

CPT conditional preference table

DAG directed acyclic graph

EAF Extended Argumentation Framework

EBNF Extended Backus Naur Form

ES Expert System

GA generalised additive

GAI generalised additive independence

GEA Generator of Evaluative Arguments

GQM goal-question-metric

IVA Interpretive Value Analysis

MAUT Multi-Attribute Utility Theory

OAPM Options-Attribute Preference Model

OVF optimal value function

PSM Preference Satisfaction Model

SCSP Soft Constraint Satisfaction Problem

SE Software Engineering

SLO Soft-constraint Lexicographic Ordering

SVM Support Vector Machines

UML Unified Modeling Language

UF utility function

1
Introduction

Many tasks in everyday life involve making decisions over a large number of options (Schwartz 2005): we must decide which clothes to wear, what to eat, where to go for fun. Both these frequent decisions and infrequent ones, such as shopping for expensive goods or planning vacations, demand an effort that can be reduced by delegating decision-making to intelligent agents. For agents to appropriately perform tasks on our behalf, however, they must be aware of individual user preferences and the available options.

Our vision is for agents to make decisions on behalf of users so that their choices match those of users themselves, given adequate time and knowledge. However, it is important to understand that humans do not make decisions in isolation, and agents acting on their behalf should not do so either. Where the option chosen for one user may affect that of another (e.g. in deciding at which hotel to stay, we both prefer to stay at the same hotel), agents need to coordinate their actions. Such coordination between users reflects just one among the many interacting preferences that agents may need to consider. We argue that, by reflecting how users themselves decide, there is a *rationale* for choices that is *convincing* to users. Nevertheless, before we can make decisions appropriate to multiple users, we must first have agent reasoning appropriate to a single user, which is addressed in this thesis, consisting of a first step towards the broader vision.

Choosing from a set of available options often requires resolution of trade-offs, but it can be unfeasible for humans to carefully evaluate each option of a large set due to the required amount of time and cognitive effort, so that they are often unsatisfied with their choices (Schwartz 2005). As understanding human decision making and how to support them in making choices is important from many perspectives, such as understanding consumer behaviour and aiding managers in making high-impact business decisions, decision making has been extensively studied in a wide range of areas, including *economics* (Keeney and Raiffa 1976), *marketing* (Wierenga 2008), and *psychology* (Tversky 1996), for many decades. Moreover, receiving support or delegating a decision to a software system that is aware of users' preferences and is able to make decisions like them without effort restrictions and with adequate time can be very helpful. Therefore, decision making

has also received much attention in *computer science*, including in the areas of artificial intelligence, databases, and semantic web. Our goal in particular is to automate the decision making process and, in order to do so, we deal with three widely investigated issues: (i) how to capture and represent user preferences; (ii) how to reason about preferences and make decisions; and (iii) how to justify to users the decisions made.

According to Lichtenstein and Slovic (Lichtenstein and Slovic 2006), humans have a set of preferences that they are aware of — referred to as *known* preferences — which guide the decision making process. These preferences are expressed by individuals in different forms, but existing work on preference reasoning is only able to handle a restricted set of preference types, thus constraining users in expressing their preferences. One way to deal with that is to capture preferences through elicitation processes, but these can be tedious, discouraging users from using such processes. Moreover, these processes not only capture known preferences but also those needed to resolve trade-offs because the choices available often present a conflict among known preferences, so that trade-offs must be made. However, these additional preferences are constructed (as opposed to revealed) during the decision making process, as "*people do not maximise a precomputed preference order but construct their choices in the light of available options*" (Tversky 1996), and trade-off resolution requires cognitive effort of users (Schwartz 2005), thus compromising the acceptance of decision support systems that use elicitation processes.

Many approaches to reasoning about preferences are based on Multi-Attribute Utility Theory (MAUT) (Keeney and Raiffa 1976), which is designed to handle trade-offs among multiple objectives assuming a set of axioms (von Neumann and Morgenstern 1944) for preferences and utilities. Three of these axioms are (i) *description invariance*: preference order is always the same no matter how options are presented; (ii) *procedure invariance*: preferences do not depend on the elicitation process; and (iii) *context independence*: the addition of options does not impact preferences. A decision-maker whose preferences satisfy these axioms is considered rational, from an economic perspective, and consequently has a utility function (UF) that quantitatively represents preferences. However, as Tversky (Tversky 1996) has observed, human preferences often do not satisfy these axioms and, considering human irrationality, can we say that human preferences over a set of options are wrong? Moreover, should they be changed, in order to be considered rational? We assume that human preferences are not wrong, and if a decision model is not consistent with them, the model has to change. Consequently, preferences represented in the form of UF are not only hard to elicit but may also be inadequately represented.

Furthermore, *explanations* play an essential role in decision making. Humans often make and accept decisions made by others when they are able to identify the reasons for accepting and rejecting choices (Shafir et al. 1998), so that there are plausible arguments that justify the decision. Therefore, providing users with explanations that justify automated decision making is as important as providing adequate preference representation and reasoning.

Given this context, we present the problem we address in this thesis and the limitations of existing work in Section 1.1. We next describe our proposed solution and provide an overview of the contributions in Section 1.2, and then detail the structure of the remainder of this thesis in Section 1.3.

1.1 Problem Statement and Limitations of Existing Work

As introduced earlier, our research aims to automate decision making by tackling problems in three directions: (i) how to represent user preferences at a high level of abstraction (end-user level); (ii) how to make a choice from a set of available options using such preferences as input; and (iii) how to provide users with acceptable explanations that justify the decision. Based on these three issues, we state our primary research question below.

Research Question.

How can an automatic mechanism aware of a user's preferences choose one option from a set of available options and explain that choice, such that the user would be convinced of the adequacy of the choice?

Limitations of existing work, which are associated with this research question, are listed as follows.

There is no in-depth investigation of how humans express preferences nor is there a model that represents them. Many preference representation models have been proposed, to capture user preferences for decision making processes. However, such models are able to *represent* only a restricted set of preferences, constraining users in expressing their preferences, and creating the need for tedious interactive elicitation methods. Moreover, existing preference models are not justified by research studies, but are built in an *ad-hoc* way, based on intuition.

Existing approaches to reasoning about preferences are able to handle a restricted set of preference types and are limited to the identification

of non-dominated options. Even if there were models to represent high-level preferences, existing approaches to preference *reasoning* cannot handle all the constructions typically adopted by humans to express preferences. Therefore, it is important not only to represent different types of preferences, but also to be able to use them to make decisions.

Furthermore, preferences that users are able to specify without the aid of elicitation processes, i.e. their known preferences, are not enough to resolve trade-offs that emerge during the decision making process and, as a consequence, a decision making technique must provide a way to resolve trade-offs. Existing techniques are limited to selecting options that can be considered better according to provided (known) preferences. But, as these preferences often conflict — for example, maximising quality and minimising price — they are not enough to choose *one* option, but allow only the selection of a *subset* of options that have both pros and cons with respect to each other. The difficult step of the decision making process, namely trade-off resolution, remains for the user to perform.

Finally, most existing approaches rely on classical decision theory, which does not match how humans make decisions. Therefore, in order to make decisions like humans do, there is a need to take into account *human* decision making.

There is no consensus on what constitutes a good explanation to justify choice. The main goal of research into decision support and recommender systems has been to improve their *accuracy* (typically measuring the mean squared error of predicted ratings), associating this measure with the quality of the choice or recommendation. However, as argued by McNee et al. (McNee et al. 2006), the most accurate systems (based on standard metrics) may not be those that provide the most useful choices to users. Other aspects, such as *trust* and *transparency*, have also been considered, and many of these can be improved by providing users with *explanations* (Tintarev and Masthoff 2007).

There are different existing approaches to generating explanations (Klein and Shortliffe 1994, Labreuche 2011), from exposing the rationale of the underlying recommendation technique to selecting the essential attributes on which the decision is based. However, there is no consensus on what constitutes a *good* explanation, and what kinds of information must be presented to users in such explanations. Even though existing work (Tintarev and Masthoff 2007) provides qualitative arguments that characterise good explanations, there is only limited research on the kinds of explanation that users expect and need to understand recommendations or decisions made on their behalf. Where work does exist in this context, it is particular to a specific system.

1.2 Proposed Solution and Contributions Overview

In order to provide the kind of support we aim to give users — i.e. making choices that are consistent with high-level user preferences but going beyond them to resolve trade-offs — we propose an approach that involves preference representation, decision making and explanation generation, and these are founded on work in psychology and studies we performed with humans.

With the goal of providing a deeper understanding of how users express their preferences, we performed a study that involves the investigation of seven research questions, including how knowledge about a domain influences the expression of preferences and how users change their preferences after being exposed to decision-making situations. This study allowed us to identify the kinds of support users need to better express their preferences so that a system can make choices on their behalf. Given this study, we propose a preference metamodel that captures different kinds of preferences that humans adopt.

In order to tackle the problems associated with decision making, we propose a novel technique for making choices based on preferences and available options, whose main contributions are the following. First, it is able to handle qualitative preferences expressed in a high-level language, allowing users to express their preferences in a similar way to natural language, thus requiring less user effort than using a restricted preference language. Second, it incorporates psychological principles concerning how humans resolve trade-offs, as the provided user preferences are often not enough for making a decision. Our technique thus chooses one option from a finite set available, based on user preferences that have natural-language-like expressions, such as expressive speech acts (e.g. *like*, *accept* or *need*) — which are part of our preference metamodel. The decision making process is inspired by research work on human decision making (Shafir et al. 1998, Tversky 1972).

With regard to the identification of explanations that users expect to receive to justify choices, we present a study from which we extract types of explanation that humans use to justify a choice from a set of available options. As, based on the design of the study, we can assume that the explanations provided by study participants are those that the users would expect to receive, we derive a set of *guidelines* and *patterns*, which are a basis for generating explanations for users as to why particular options are chosen by decision support systems. Considering these identified explanation patterns, we also propose an explanation generation technique in order to produce appropriate and convincing explanations. The input for generating explanations is decision models generated during the decision making process of our user-centric preference-based technique. We include

algorithms to choose the appropriate explanation pattern in a given instance, and derive the parameters required to complete the explanation.

In summary, the main contributions of this thesis are:

(i) a **study of how humans express preferences**, which identifies preference constructions adopted by humans in natural language;

(ii) a **preference metamodel**, which allows the modelling of preferences at a high level;

(iii) an **automated decision making technique**, which chooses one option from a set available, based on high-level preferences;

(iv) a **study of how explanations can justify choices**, which identifies guidelines and patterns to generate explanations for users;

(v) an **explanation generation technique**, which justifies a choice made based on models produced by our decision making technique; and

(vi) a **user study**, which evaluates different aspects of our approach (preference metamodel, decision making technique and explanations), and also compares existing explanation generation approaches.

1.3 Outline

The remainder of this thesis is organised in four parts. Part I is related to the representation of preferences. Chapter 2 presents a study of how humans express preferences, which allows us to identify expressions that humans adopt to state their preferences. Based on the results of this study, Chapter 3 describes a preference metamodel that provides the different forms that are used to express preferences. Chapter 4 then discusses existing preference representation models, and compares them with our metamodel.

After discussing preference representation, Part II focuses on preference reasoning. First, a systematic review of preference reasoning approaches is presented in Chapter 5, introducing the work proposed in different areas of computer science. Chapter 6 details our novel decision making technique, which is able to handle different types of preference and incorporates user-centric principles. This chapter also compares our technique with existing work and presents an empirical evaluation.

Part III is concerned with another important issue in automated decision making: user explanations. Chapter 7 first presents a literature review of explanations, followed by Chapter 8, which describes a study in which we

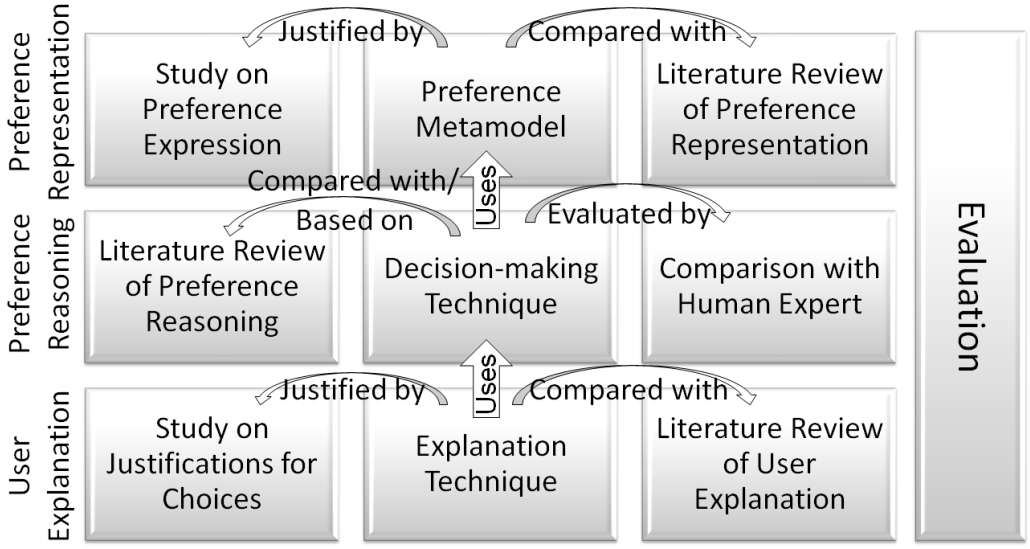


Figure 1.1: Thesis components and their relationship.

investigate explanations given by humans to justify a choice, from which we derive patterns and guidelines to be adopted by explanation approaches. Based on this result and our decision making technique, an explanation generation technique is detailed in Chapter 9.

Part IV connects all the previous parts and concludes this thesis. Chapter 10 describes a user study performed to evaluate all aspects of our approach: preference metamodel, decision making technique and explanation generation technique. Finally, conclusions and future work are presented in Chapter 11. We summarise in Figure 1.1 the different parts that will be presented in this thesis, and how they are related to each other.

Part I

Preference Representation

In this part, we address the problem of representing natural preference statements provided by users. The main goal of capturing and representing (user) preferences is to use them as input for a reasoning process that makes decisions on behalf of users. Examples of questions to be answered taking into account preferences are: (i) given a set of options, which is the most preferred? or (ii) how options are ranked according to user preferences? As preference representation models are used as input for algorithms that answer these questions, representation and algorithms are tightly coupled, because little can be done with preferences without having the ability to reason about them. On the other hand, considering that users have to state their preferences, there is no point in defining a preference representation model, which can be reasoned about, but users are not able to express preferences with constructions provided by this representation.

In fact, this issue was pointed out by Domshlak, who defines it as a paradoxical deadlock situation suggesting the "chicken-and-egg" metaphor. "*On the one hand, it is only natural to assume that reasoning about user's preference expressions is useful in many applicative domains (e.g., in online catalog systems). On the other hand, to our knowledge, no application these days allows its users to express any but trivial (e.g., "bag-of-word") preference expressions. It seems that the real-world players wait for the research community to come up with a concrete suggestion on how natural-language style preference expressions should be treated, while the research community waits for the real-world to provide it with the data essential to make the former decision. It is clear that this deadlock situation should somehow be resolved, and we believe that now this should be a primary goal for both sides.*" (Domshlak 2008)

In this context, our focus here is to understand how people explicitly express their preferences (without the aid of elicitation mechanisms), without being concerned with how to reason about stated preferences and make decisions based on them. Our goal is to define a preference representation model that serves as a reference for what should be ideally used as input for a preference reasoning algorithm, i.e. we aim at providing the essential data mentioned by Domshlak. We first present a study of how humans express preferences about a particular domain, from which we extracted patterns and expressions used by people to express their preferences (Chapter 2). Based on the analysis of preferences of our study, we propose a preference metamodel (Chapter 3). Finally, we present research work on preference representation models in different research areas from computer science (e.g. artificial intelligence and databases) and analyse their expressivity by identifying which kind of statements they are able to represent explicitly, which comprise only a subset of the statements identified in our study (Chapter 4).

2
Understanding User Ability to Express Preferences

Given that our approach requires users to express their preferences in a high-level language, we performed an exploratory study, presented in this chapter, to evaluate the feasibility of this requirement, and also to investigate how people express their preferences about a domain, including the common expressions they use. This study also evaluates the impact of experiencing a concrete decision-making situation and of the knowledge about the domain, while people state their preferences. We thus focus on answering two main research questions: (i) *are users able to express their preferences in such a way that a domain specialist is able to make an adequate choice in this domain on their behalf?* and (ii) *do users need to be exposed to a concrete decision-making situation, i.e. being aware of the available options, to be able to express their preferences about a familiar domain?* Other issues are also analysed, such as which kinds of changes users make after being exposed to a decision-making situation; and how the domain knowledge or other relevant aspects (age, gender, etc.) impact the users' expression of their preferences. If we conclude that the preference specifications given prior to a decision-making situation are not enough for making a decision on behalf of users — question (i) — it is essential to identify which kind of support can be provided for each user category in order for users to better express their preferences, but still without having to go through the entire decision process. Our study allowed us to identify kinds of support users need to better express their preferences and relevant concepts that should be part of an end-user preference language, which can be used by users to express their preferences in a way similar to natural language, or to correct and refine preferences initially acquired implicitly.

Our study consists of applying a questionnaire for participants of different profiles (knowledge about the target domain, age, working area, gender, and so on) to collect preference specifications expressed in natural language before and after experiencing a concrete decision-making situation. Later a domain specialist uses the initial specification to make recommendations according to each specification. Different measures are extracted from the collected data, both qualitative (e.g. type of preference specification and type of change) and quantitative (e.g. the amount of time and number of steps to make a decision). The domain chosen for our study is

Definition element	Our experiment goal
Motivation	To understand how people express their preferences,
Purpose	characterise and evaluate
Object	preference specifications
Perspective	from a perspective of the researcher
Domain:user	as people with different knowledge about a domain express their preferences
Scope	in the context of the researcher's social network.

Table 2.1: Goal Definition (GQM template).

laptop purchasing. This choice was made because choosing laptops represents the kind of task we are addressing in our work: users are aware of a set of preferences over laptops, as buying laptops is a task that is potentially performed repetitively (every x years), but each time it is different as the available options and features evolve over time. In addition, users have different levels of knowledge about this domain and we had a domain specialist available to participate in the experiment. The main goal of this study is to give foundation to our work on automated decision making.

The remainder of the chapter is organised as follows. Sections 2.1 and 2.2 describe the design and results of our study, respectively. Section 2.3 discusses the results, followed by Section 2.4, which concludes.

2.1 Study Description

In this section we detail the design of our exploratory study, as well other relevant information, including the research questions we aim to answer and the participants involved.

In order to design our study, we have followed the framework proposed by Basili et al. (Basili et al. 1986), which provides guidelines to elaborate experimental studies in Software Engineering (SE). The first phase of the framework is the definition of the experiment adopting the goal-question-metric (GQM) template (Basili and Rombach 1988), which establishes experiment goals that are used for defining research questions associated with it. Then metrics are defined or selected for answering those questions. Following this template, the experiment goal is presented in Table 2.1. After that, the phases of planning, operation and interpretation are executed. Both (Basili et al. 1986) and (Basili and Rombach 1988) provide guidance for performing experimental studies in the context of SE, but they are generic enough to be used in our study. Their adoption was due to our previous experiences with SE studies.

Our study consisted of applying a web-based questionnaire for profiling the participants, capturing their preferences on a domain before a decision-making situation, engaging participants in concrete decision making in that domain, and then asking them to review their previous preference specification. Next, a domain expert made recommendations for participants based on their initial preference specifications and the same available options, and all the collected data was then analysed. The domain chosen for our study is laptops. This decision was made due to the availability of domain experts, and also because this domain illustrates a scenario in which participants might have experienced a similar decision-making situation, but always with different available options, as laptops evolve over time, and new features are introduced. More details of the study procedure are given in Section 2.1.2.

2.1.1 Research Questions

The main goal of this study is to evaluate how users would typically express their preferences about a domain, and how useful the provided preference specification is to make a decision on their behalf. In addition, we aim to investigate the impact of experiencing a concrete decision-making situation on this specification. This evaluation was performed in different directions, which are associated with seven research questions addressed in the study, as presented in Table 2.2(a).

With these research questions, we aim to acquire a deeper knowledge on user preference expression. This information enables us to make statements about users, and it is helpful and necessary for developing approaches in the context of preference-based decision making. In particular for our approach, it allows us to verify whether it makes sense to provide an end-user language for users to express or adjust their preferences so they can delegate tasks to systems provided with automated decision making. If users are unable to specify their preferences in natural language in such a way that a (human) expert in that domain is able to make an appropriate choice on their behalf, as it is the case in our study, it is unlikely that it will work with a restricted language and software systems. This issue is addressed by *RQ1*. In addition, we also investigate whether users need to experience a concrete decision-making situation, i.e. they need to know the available options, in order to adequately express their preferences (*RQ2*).

Moreover, we also address other issues with the elaboration and execution of our study, which can help in different aspects of the development of preference-based approaches. With *RQ3*, we aim to identify which problems (wrong, missing or outdated preferences) occur when users specify their

RQ1. Are users able to express their preferences about a familiar domain in such a way that a (human) domain specialist is able to make an adequate choice in this domain on their behalf?
RQ2. Do users need to be exposed to a concrete decision-making situation to be able to express their preferences about a familiar domain?
RQ3. Which type(s) of preferences users usually forget or incorrectly specify before being exposed to a concrete decision-making situation?
RQ4. How different are specifications provided by users with a high degree of knowledge about a domain from the ones provided by those with a low degree of knowledge?
RQ5. Which user profiles take less time to express their preferences?
RQ6. When users make a choice, which ones select fewer options from among the offered ones? In other words, which user profiles are more confident in which is the right choice for them?
RQ7. Which user profiles take less steps (filtering, comparing, analysing, ...) in the decision-making process (choosing among available options)?

(a) Research Questions.

EA1. Comparison between laptops selected by participants and the ones recommended by the domain specialist based on their specification.
EA2. Analysis of the differences between the initial preference specification and the reviewed version of it.
EA3. Analysis of the most common types of preferences that appeared only in the preferences review.
EA4. Comparison between the preference specifications provided by participants with high degree of knowledge about the domain and by those with low degree of knowledge about it.
EA5. Comparison of how long participants classified in different categories (domain knowledge, gender, ...) take to specify their preferences.
EA6. Comparison of how many laptop options were chosen by participants in different categories (domain knowledge, gender, ...).
EA7. Comparison of how many steps (filtering, looking details, comparing,...) participants in different categories (domain knowledge, gender, ...) took to define their laptop options.

(b) Evaluation Approaches.

Table 2.2: Research questions and their evaluation approach.

preferences, so that we can provide mechanisms to prevent them. With *RQ4*, we investigate whether and how users with different knowledge about the domain express their preferences in different forms; if so, languages with different vocabularies might be needed for different user profiles. As the effort that users spend to perform a task is directly related to how much they are willing to do it, they might not want to do it. Therefore, in order to evaluate the feasibility of expecting users to provide preferences in a high-level language, we evaluate in *RQ5* whether users (or some of them) take too long to express their preferences. Confidence on the choices made is relevant to investigate (*RQ6*), as it can indicate that different options may satisfy users' needs or that trade-off situations were not resolved, and different approaches may be adopted by automated decision making systems to include users in the decision making process, according to their confidence on the decision. Finally, *RQ7* helps to understand the user decision-making process, and how it differs when the user has deep knowledge about the domain. This is also

related to the elaboration of approaches that make decisions on behalf of users.

2.1.2 Procedure

The study we planned to answer our research questions is mainly based on a web-based questionnaire applied to a wide spectrum of users (see next section for details). The domain selected for performing our study is shopping for products; in particular, we chose the laptop as the target product. This decision was made due to the reasons introduced before.

In a nutshell, the idea of the questionnaire is to first ask users to specify their preferences for someone who is going to buy a laptop for them. Later, they are asked to navigate on a laptop catalog and select from one to five laptops. Finally, we give users a chance to modify their preference specification.

The applied questionnaire, which can be seen in Appendix A, consists of four parts, each of which is explained next.

- *User Information Data.* The questionnaire is anonymous, but we collect relevant information related to the study from the participant: (i) age; (ii) location (city and country); (iii) working/studying field; (iv) how many laptops the participant has already had (current one included); (v) from these, how many were chosen by the participant herself; and (vi) how she rates her knowledge about the domain. These last three items are used to evaluate the participants' knowledge about the domain.
- *Preference Specification.* The study participant is requested to imagine a situation in which she is going to ask someone to buy a laptop for her. Therefore, she is requested to specify all her preferences and restrictions. An example in the flight domain is provided in order to give some instructions for participants, as they are not assisted while answering the questionnaire. An example in this domain was adopted because we did not want to influence the participants by providing an example in the same domain of the study, and also because the process of choosing seats has similarities with the process of choosing laptops: available seat locations vary each time a decision is made. Besides storing the provided preference specification, we logged the current state of the specification every 15 seconds and the time the participant took in this part of the study.
- *Choosing Product.* Next, the participant is requested to analyse a set of different computers and say which one she would have bought. We ask her to rank her favourite ones, up to five laptops. We used the Best Buy[1]

[1] http://www.bestbuy.com/

catalog, which had 144 laptops by the time we imported it (at the same day in which the survey was released). We recorded each step (comparing, filtering, detailing, ...) the participant performed, as well as the time taken for choosing the laptops.

– *Preference Specification (review).* Finally, after analysing the available computers, the participant is given a chance to review her preferences and modify them, in case she realised that something was missing or wrong in her specification. We have notified participants in the third part that they would have this reviewing chance. We also asked the participant's comments on what changed on her specification. The additional logs are the same as in the second part of the questionnaire.

After collecting all the data, a domain expert was involved in the study. The domain expert's responsibility was to analyse the first version of the preference specifications provided by the participants, and to rank up to five laptops he would have recommended for each one. We are aware that involving more than one domain expert to make recommendations would significantly improve the results of our study, but we did not find other experts willing to participate in it.

The study participants and the domain expert were allowed to choose up to five laptops, which is a limit we established. We chose this number because we wanted to provide flexibility for participants and for the expert to choose more than one laptop, and we assumed that five options were enough. In order to confirm that five is a good number as well as to evaluate our web interface, we executed a pilot study with few individuals, and they approved both the interface and the limit of five options. In addition, as it can be seen in the results, several participants selected less than five laptops. With regard to the domain expert recommendations, we asked him whether he wanted to recommend more than five options and he answered that five was a good number.

Based on the questionnaires and the recommendations made by the domain expert, we analysed the data according to two main aspects, related to the research questions 1 and 2: (i) were the participants able to express their preferences in such a way the domain expert could make adequate recommendations for them? and (ii) did the participants change their preference specifications after experiencing the process of choosing a computer? Furthermore, we have also analysed other relevant aspects in order to answer the additional research questions, from 3 to 7. In Table 2.2(b), we detail how we analysed the survey data to answer each research question.

The evaluation approach presented in Table 2.2(b) shows we have performed mainly a subjective but also an objective analysis of the data to answer all our research questions. Table 2.3 summarises all collected data, both qualitative and

Subjective	Objective
• Similarity score between chosen laptops and those recommended by the domain expert • Types of preference changes • Types of preference specifications • Characteristics of preference specifications	• Time taken to specify preferences • Number of chosen options • Steps taken to choose options • Number of participants that changed their specifications

Table 2.3: Qualitative and quantitative data collected.

quantitative, from our study. Some of the measures, e.g. similarity score and types of preference changes, are described later, when they are used to analyse results.

2.1.3 Participants

Our study involved a total of 192 participants, who answered our questionnaire, and one domain expert, who indicated laptops for each participant according to their initial preference specification. Due to the effort needed to analyse a large number of questionnaires, only one domain expert agreed to participate in our study.

The questionnaire was available online from May 20 to July 13, 2010 (almost two months). For selecting the participants, we used convenience sampling, based on the social network of the researchers involved in this study. First, invitations for participating in the study were sent by e-mail and people were asked to forward the invitation for other people in a snowball approach. In addition, a call for participation in the study was published in different Orkut[2] communities.

As result, we collected a database with 451 surveys that were initiated, from which 192 were completed (42.6%) — incomplete surveys were discarded. As the researcher that performed this study is Brazilian, most of the participants are from this country (86.98%), and the remaining ones (13.02%) are from four other countries. The same situation happens with the working area (63.54% participants work with a background in informatics-related areas). In our analysis, we did not detail other working areas (despite having this information available in our database) because our focus was to identify participants with a higher knowledge about our study domain (laptop purchasing). The description of the demographic characteristics of our study participants is detailed in Table 2.4.

The domain expert that was involved in our study has an M.Sc. degree in Computer Science. Moreover, his work involves giving technical support to the Software Engineering Laboratory at PUC-Rio as well as specifying and

[2] http://www.orkut.com

Working Area	Informatics 122 (63.54%)	Non-informatics 70 (36.41%)	**Gender**	Male 134 (69.79%)	Female 58 (30.21%)
Domain Knowledge	No Knowledge 5 (2.60%)	Beginner 16 (8.33%)	Intermediate 40 (20.83%)	Advanced 83 (43.23%)	Expert 48 (25.00%)
Country	Brazil 167 (86.98%)	Germany 10 (5.21%)	Canada 10 (5.21%)	United States 4 (2.08%)	Peru 1 (0.52%)
Age	16-25 years 60 (31.25%)	26-35 years 83 (43.23%)	36-45 years 21 (10.94%)	>45 years 28 (14.58%)	

Table 2.4: Demographic Characteristics of Participants.

recommending new computers and laptops for the laboratory and its individual members. Therefore this expert is used to listening to clients specifying their preferences and to recommending computers and laptops for them.

2.2 Results and Analysis

In this section we provide the results we collected from the execution of our study as well as interpretations for those results. We have made a qualitative analysis of the preference specifications (initial and revised versions) given by the study participants and a quantitative analysis of part of the collected data, such as time taken to answer each part of the questionnaire.

The first analysis that we made was how to measure the participants' knowledge about laptops. The fields (iii) to (vi) in the *User Information Data* part of the questionnaire were used for that. Based on this data, we make the following observations.

- Participants that work in the computer science area have at least an INTERMEDIATE[3] level of domain knowledge. In other fields, participants are mostly INTERMEDIATE.
- Most of the participants who have had several laptops have at least an ADVANCED knowledge; the more laptops participants have had, the higher their knowledge.
- Almost all of the participants chose their laptops; only the ones who had several laptops had some laptops chosen for them (possibly because they get laptops from their companies).
- Not having had a laptop does not indicate a low knowledge about the domain — some participants chose not to have a laptop.

The relationship between the other fields and the domain knowledge provided by the participants presented the behaviour we expected: participants with a background in informatics-related areas or those that had many laptops (and chose

[3]Values for evaluating the domain knowledge: NO_KNOWLEDGE, BEGINNER, INTERMEDIATE, ADVANCED, EXPERT.

them) rated themselves with expertise on laptops. Therefore, when analysing the collected data, we adopted the domain knowledge that the participants themselves provided as a resource to determine their knowledge about the domain.

RQ1. Are users able to express their preferences about a familiar domain in such a way that a (human) domain specialist is able to make an adequate choice in this domain on their behalf? Based on the initial preference specifications of all participants, and the domain specialist recommendation for each of them, we have compared the laptops recommended and the ones the participants chose. The goal was to investigate the users' ability to express their preferences and how their knowledge about the domain influences this ability. Some participants, instead of providing a specification, stated that they would never delegate this task to a person, or provided templates using variables for referring to attribute values of the laptops, e.g. *"I would like laptops with processor X."* Without a specification, the domain expert is not able to make a recommendation, therefore nine of the surveys were discarded for this research question. From 183 surveys, 53 (28.96%) had at least one of the specialist's recommendations that matched at least one of the participants' choices. Besides analysing exact recommendation matches, we further calculated the similarity between recommendations and participants' choices. As we have up to five laptop choices for both the domain expert and participants, we did not calculate this similarity by simply comparing one selection with another, such as calculating the mean square error of individual laptop features. We thus have elaborated a function — shown below — to calculate this similarity score (SS), which takes into account the positions matched to calculate a weighted average.

$$SS = \frac{\sum_{i=0}^{size(CL)}(5-i) * \left(\frac{\sum_{j=0}^{size(SR)}(5-|i-j|)*(sim(CL[i],SR[j]))}{\sum_{j=0}^{size(SR)} 5-|i-j|}\right)}{\sum_{i=0}^{size(CL)} 5-i} \quad (2\text{-}1)$$

where CL is the chosen laptops (by the participant), SR is the specialist recommendation (for the participant), $size(v)$ returns the size of a vector v and $sim(x, y)$ is the function that calculates the similarity between two laptops. If they are equal, its value is 100, otherwise it is the average of each feature compared. If the feature has a numeric domain, the feature comparison is $|featureValue1 - featureValue2| / (upper\ bound - lower\ bound)$, where the domain boundaries are given by the highest and lowest values for the feature considering all laptops. Otherwise, the feature comparison is 100 for equal values, 50 for unspecified values, and 0 for different values. Table 2.5 presents the values found in our study, which ranged from 47.87 to 100.00. The column *matches* is the number of surveys in which at least one of the laptops matched, and the columns *SS(M)* and

	Matches	SS**(M)**	SS**(Median)**	SS**(SD)**
Domain Knowledge				
NO_KNOWLEDGE	3 (60.00%)	68.58	63.76	18.45
BEGINNER	2 (13.33%)	59.14	58.90	4.37
INTERMEDIATE	9 (23.08%)	60.93	59.39	6.31
ADVANCED	27 (33.33%)	62.82	60.30	8.38
EXPERT	12 (27.91%)	62.73	58.96	9.41
Gender				
FEMALE	17 (29.82%)	61.65	59.64	7.62
MALE	36 (28.57%)	62.52	59.81	8.79
Age				
16-25 years	14 (24.14%)	61.51	59.16	8.74
26-35 years	27 (35.06%)	63.28	60.27	9.10
36-45 years	8 (38.10%)	63.70	62.05	7.99
>45 years	4 (14.81%)	59.78	59.41	5.17
Total	**53 (28.96%)**	**62.25**	**59.76**	**8.43**

Table 2.5: Domain Specialist Recommendation — Matches per Group of Participants.

SS(SD) are the average and standard deviation of the similarity score, respectively. In Table 2.5, we show not only these numbers of the SS for all participants, but also for different group categories.

Considering the SS values, it can be seen that the domain expert was not able to match laptops very closely to participant choices — only 19 participants had $SS > 70$, 11 had $SS > 80$, 4 had $SS > 90$, and 28.96% matches — however he managed to recommend laptops that have at least half of the characteristics selected by participants. This result indicates that specifications explicitly provided by users are valuable sources for providing recommendations for them, as in some cases they allow the specialist to make adequate decisions. However, for making decisions on their behalf, users must be instructed on how to provide better specifications or additional support should be provided for improving their specifications. Research question 4 indicates some problems identified in the preference specifications.

Table 2.6 presents the number of matches according to each rank position of the laptops chosen by participants. For some participants, more than one position matched. It can be seen that the number of matches is higher in the first positions. This result corresponds to the quality of the specification: when the specification provides good details of what users want, it is more likely that the exact laptop they want is matched.

Position Matched	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
#Matches	30	17	11	8	2

Table 2.6: Domain Specialist Recommendation — Position Matched.

Table 2.5 also shows that the number of matches is higher for participants with a higher knowledge about the domain (ADVANCED and EXPERT). This can

Domain Knowledge	Who Changed (%)	#Changes (M)
NO_KNOWLEDGE	0 of 5 (0.00%)	0.00
BEGINNER	4 of 16 (25.00%)	2.50
INTERMEDIATE	14 of 40 (35.00%)	2.29
ADVANCED	28 of 83 (33.73%)	3.04
EXPERT	16 of 48 (33.33%)	2.13
Gender	Who Changed (%)	#Changes (M)
MALE	47 of 134 (35.07%)	2.53
FEMALE	15 of 58 (25.86%)	2.80
Age	Who Changed (%)	#Changes (M)
16-25 years	27 of 60 (45.00%)	2.07
26-35 years	25 of 83 (30.12%)	3.12
36-45 years	6 of 21 (28.57%)	2.33
>45 years	4 of 28 (14.29%)	3.25

Table 2.7: Preference Changes.

also be seen in the similarity score. Nevertheless, the highest number of matches (in percentage) were in the group of NO_KNOWLEDGE participants. We observed that these specifications, even though they do not contain specific details of the laptop, provide key information about the purpose for which the laptop will be used. But it is important to highlight that we are not aware of which criteria the participants used to choose the laptops, as they do not have knowledge about the domain.

In order to test whether the difference among the matches for the groups with different domain knowledge is statistically significant, we used the Kruskal-Wallis test. The recommendations *did not differ* significantly across the five levels of domain knowledge, $H(4) = 3.755$, $p = 0.4402$.

RQ2. Do users need to be exposed to a concrete decision-making situation to be able to express their preferences about a familiar domain? From the 192 participants, only 62 (32.29%) modified their preference specification after experiencing choosing laptops and navigating through the catalog. This result shows that even though the preference construction for available laptops, i.e. establishing an order for the available options, was made when participants had to choose one (or some) of the laptops, they did not change their preferences for individual attributes, and any method that they may used for resolving trade-off among attributes was not reported as a preference. Table 2.7 shows the participants that changed their preferences according to the domain knowledge, gender and age. In addition, it presents the average number of changes (we explain how we counted it in the next research question).

From these 62 participants, no one had NO_KNOWLEDGE about the domain. Even after searching laptops and seeing their features, NO_KNOWLEDGE participants were unable to describe preferences in terms of the laptop features — they did not know (and maybe did not want to know) what these features mean. A

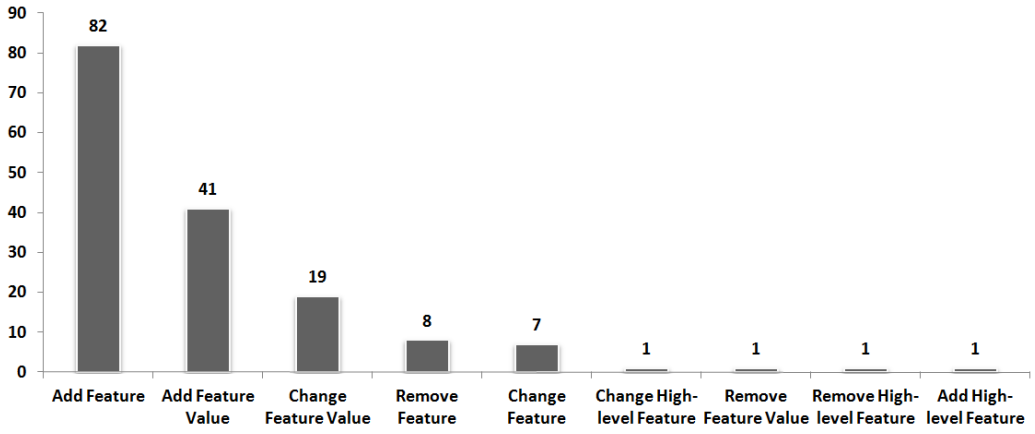


Figure 2.1: Nature of Preference Changes.

few BEGINNERS changed their specification, but they added high-level features such as "modern design" and "installed software," and particular features learned from the catalog did not influence their specification. Approximately one third of the three remaining categories changed their specification. They understand the domain (some of them better), but not necessarily know the latest news (this was the main reason for the changes made by EXPERTS). When they see new and updated features, or features they forgot to mention, they provide further details on their specification.

Analysing changes and ages, the older the participant is, the fewer changes she made. Older people provided less *detailed* specifications (see research question 4), but still did not change them after going through the process of decision-making. However, when they changed their specification, they made more changes.

RQ3. Which type(s) of preferences users usually forget or incorrectly specify before being exposed to a concrete decision-making situation? We analysed all specifications that changed when they were revised, and classified each change with a target and a type. There are three kinds of types: *add*, *remove*, or *change*. Also, there are three kinds of targets: (i) *Feature*: it describes a characteristic of the laptop, e.g. "HDMI;" (ii) *Feature value*: it describes the value of a feature, e.g. "Processor i5" changed to "Processor i5 or i7;" (iii) *High-level Feature*: it describes a high-level characteristic of the laptop, e.g. "Mobility." When a participant added a feature *and* its value in the preferences review, it was considered as a feature, because the feature would not make sense without a value. But if the participant only added a value to an existing feature, it was considered as add feature value. Figure 2.1 shows the occurrence of preference changes according to their nature (target and type).

As it can be clearly seen, the three most common types of preference change

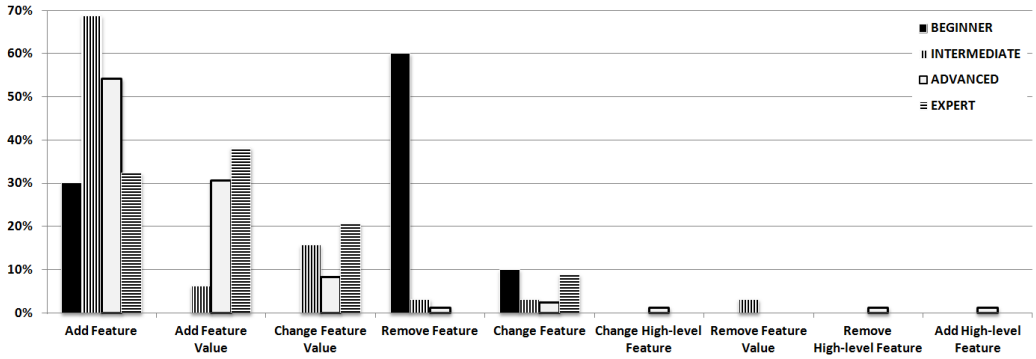


Figure 2.2: Preference Changes x Domain Knowledge (Percentage).

that participants made in the preference review are: (a) Add Feature (50%); (b) Add Feature Value (25%); and (c) Change Feature Value (12%). What happened was that users forgot to specify some characteristics that are important for them, or there is a new characteristic that they did not know about. At the moment they saw them in the laptop catalog, they remembered to specify them.

Moreover, some of the users were not aware of the current average or top values (price, processor, etc.), and as they know this by searching an up-to-date catalog, they realise that the value is different from what they thought (it is mainly related to feature values). However, some participants specified feature values in terms of relative values ("second best value"), instead of absolute ones ("4GB"). Using this kind of specification makes the preference specification reusable in different occasions.

Figure 2.2 presents how preference changes occurred distributed across the different domain knowledge categories, where it can be seen the three most common types of preference change presented above is the same for almost all categories. The only exception is in the BEGINNER category, in which 60% of the changes are of the type remove feature. However, it happened because a single participant changed the way she provided her specification, and therefore she removed the previously provided features and added a different kind of information (provided a specific laptop model).

RQ4. How different are specifications provided by users with a high degree of knowledge about a domain from those provided by users with a low degree of knowledge? The goal of this research question is to investigate how users with different knowledge about the domain express their preferences. As users that do not know much about laptops are not aware of their features, they tend to use an alternative vocabulary in comparison with domain experts.

We analysed each preference specification and classified it in four different types, which take into account only the laptop specific features. In addition, we have

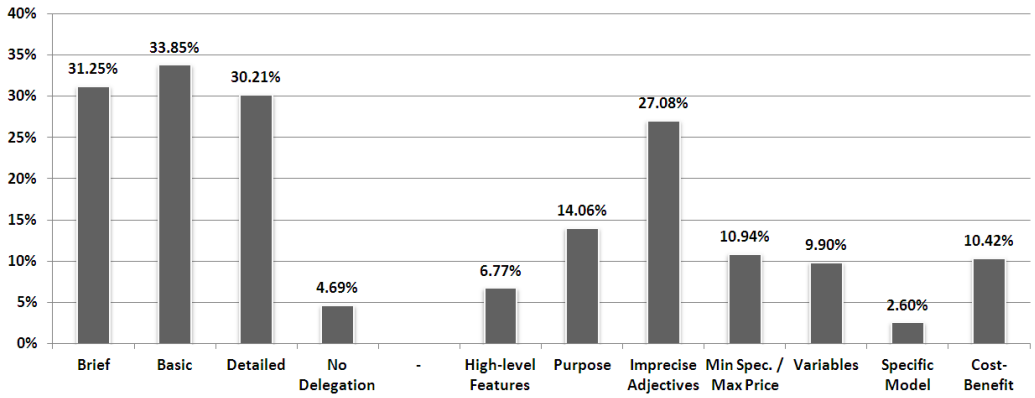


Figure 2.3: Preference Specification Analysis.

identified particular characteristics and common patterns. Figures 2.3, 2.4 and 2.5 present the charts that show the data collected (percentage) from the preference specifications from our study. Figure 2.3 shows the results related to the whole group of participants, while Figures 2.4 and 2.5 show two perspectives from the specifications, their type and their characteristics, classified according to three different categorisations: (i) domain knowledge — Figures 2.4(a) and 2.5(a); (ii) gender — Figures 2.4(b) and 2.5(b); and (iii) age — Figures 2.4(c) and 2.5(c). The four types of preference specification are presented below.

– **Basic** specifications mention characteristics for features that are part of every laptop (processor, RAM memory, hard drive, screen size). Characteristics can be specific values, or adjectives, such as “good” and “big.” Example:
Brand HP or Dell
Processor Core 2 Duo or better
RAM Memory 4 GB or more
HD 300 GB or more
Monitor of 15” or more

– **Brief** specifications do not cover laptop basic features (they mention none or few of them). Usually other kind of specification is provided, such as for what the participant will use the laptop. Example:
I want one that is light
- Screen of 14 inches
- fast
*- beautiful^^*

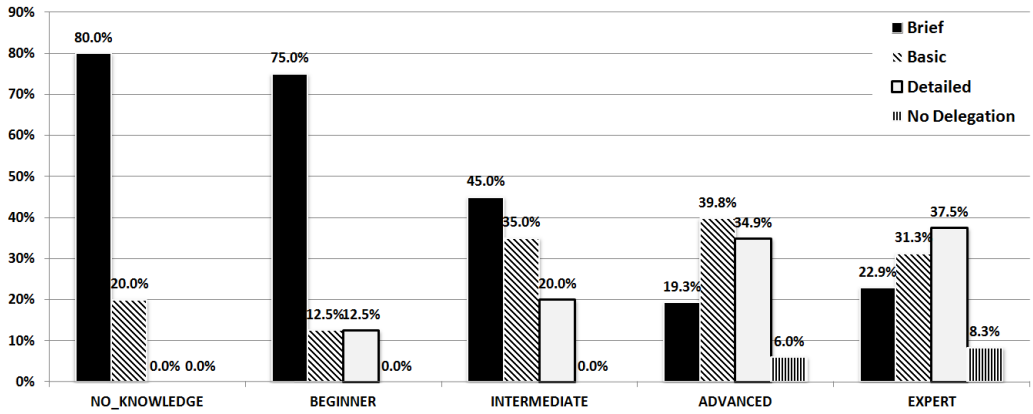
– **Detailed** specifications give further details about laptops than the basic specifications, i.e. they are more specific, tending to narrow the laptops search space. We added to this category brief and basic specifications of *Apple* laptops, namely **Brief but Enough**, because participants who want laptops

of this manufacturer, by describing only a few features, already indicate a specific laptop. Example of a detailed specification:
1 - The laptop must be of a traditional brand, preferentially Sony, Dell or Apple.
2 - Hardware configuration must be up-to-date (processor, RAM memory, HD, video card and video output, etc.).
3 - The operating system must be Windows 7 or Leopard Snow.
4 - The screen should preferably have 13" (it can have 14" if the laptop is very good and the price is attractive).
5 - I don't care about buying an "open box" if the computer is good and the price attractive.
6 - The keyboard must be comfortable.
7 - The looks and design of the laptop must be sober and of a good taste.
8 - At least 2 USB ports and there must be an integrated microphone and camera.
9 - The audio output for earphone must be at the side of the computer (it cannot be in front of it). It is important also that it has volume control on the keyboard or in the chassis.
10 - The laptop must have a competitive price.

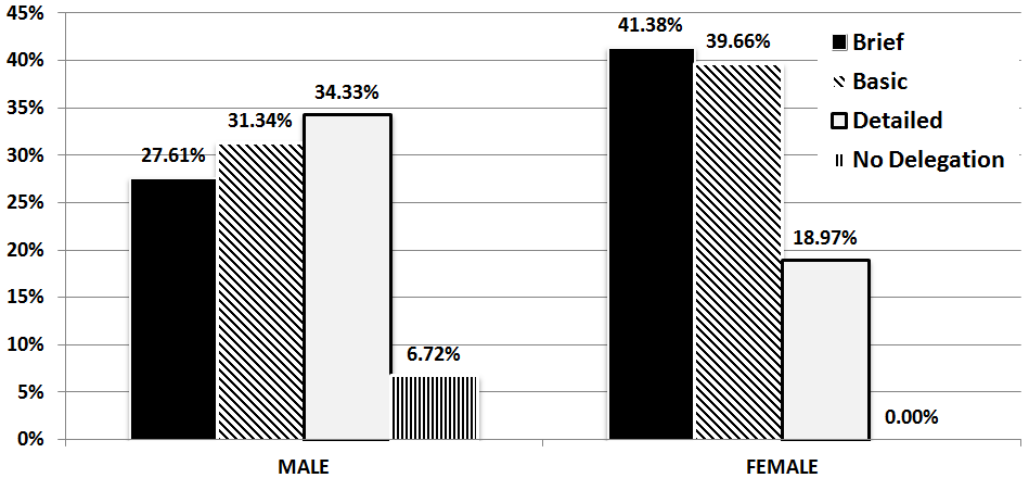
– **No Delegation.** Few (nine) participants did not provide a descriptive specification but informed the specific model they wanted, or stated that they would never delegate the decision for another person. Example:
Buy the Sony VPC F1190X, with the following configurations:
- Proc: Core i7 820QM 1.73GHz
- OS: Win7 64bits
- HD: 500GB 7200RPM
- Memory: 8GM DDR3 (1333MHZ)
- Drive Blu-Ray
- Monitor 16.4"
- VGA: GeForce GT 330M (1 GB of video)
- No additional software.

This categorisation emerged from the qualitative analysis of collected specifications, which is supported by principles of grounded theory (Glaser 1992), which is a systematic methodology for generating a "theory" from collected data. By observing the preference specifications, we identified patterns among them, and noticed that they could be classified in this way.

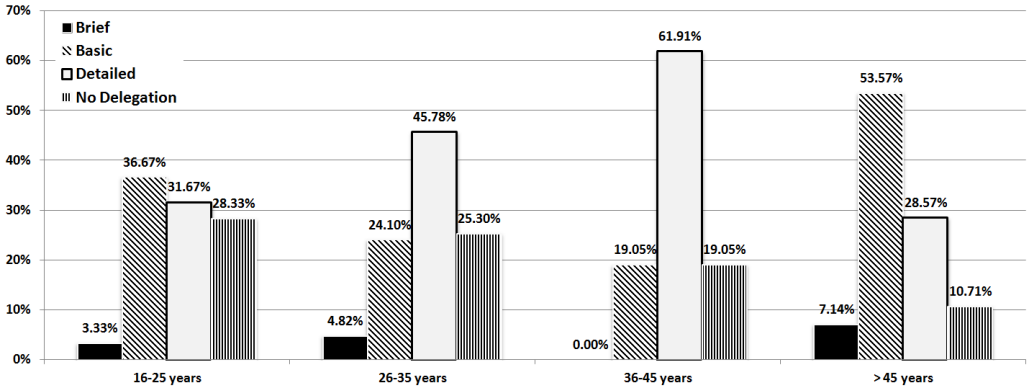
As stated previously, the presented categorisation has taken into account only how laptop specific features were described. We also observed other characteristics of the specifications, which are described next.



2.4(a): Domain Knowledge.

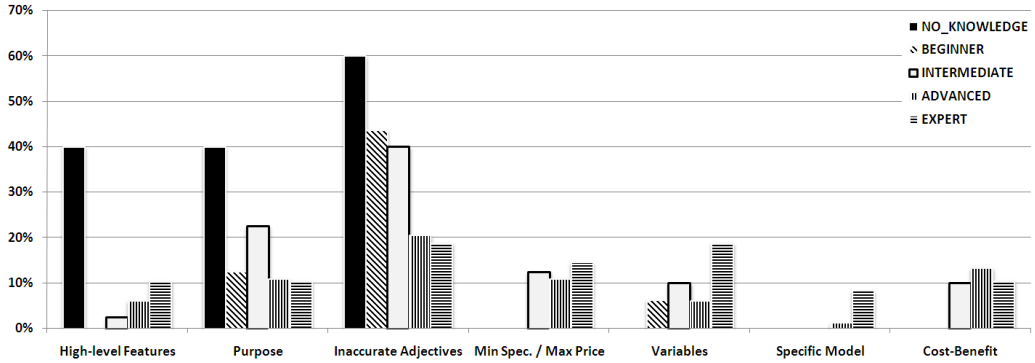


2.4(b): Gender.

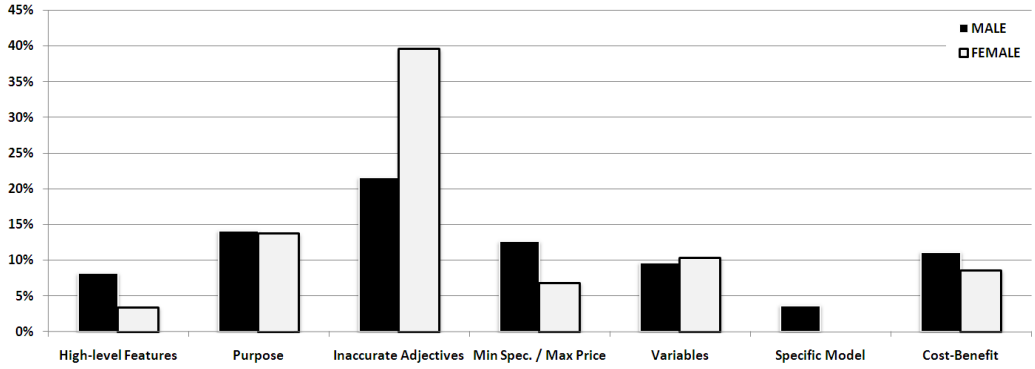


2.4(c): Age.

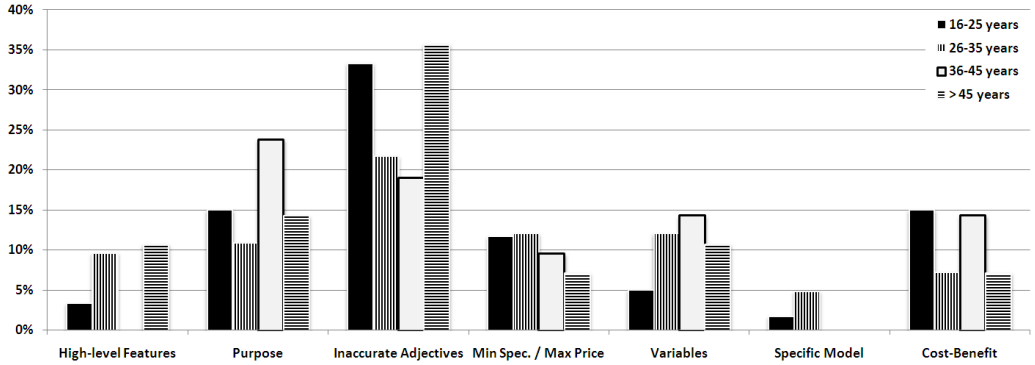
Figure 2.4: Preference Specification Types (percentage).



2.5(a): Domain Knowledge.



2.5(b): Gender.



2.5(c): Age.

Figure 2.5: Preference Specification Characteristics (percentage).

- Presence of **High-level Features**, which describe the consequences of having a value for a (set of) specific laptop feature, e.g. mobility, readability, performance. Example: *"Portability is an important factor — I'm not looking for a desktop replacement, but rather something that I can take with me when I travel."*
- Description of **Purpose** — specifications that contain the purpose for which the participant wants the laptop, for instance playing games. Example: *"I like playing not so hardware demanding games, however with good quality"* and *"My notebook is a desktop replacement."*
- Presence of **Imprecise Adjectives**, which are adjectives whose meanings depend on the point of view of the participant, e.g. "good," "modern design," "beautiful." Example: *"3. Not too heavy."*
- **Minimum Specification/Maximum Price** — pattern of specification that specifies a minimum specification for the laptop features and establishes a maximum price that the participant is willing to pay.
- Presence of **Variables**, which is when the participants used variables for feature values on their specification. Example: *"I would like laptops with processor X or Y with a screen size Z with optical drive K with hd of size L and memory of size A and video card F."*
- **Specific Model** — specifications that do not describe laptops but indicate the specific model the participant wishes. Example: *"Apple MacBook Air With 13.3 Display Aluminum."*
- **Cost-benefit** — participants that mentioned this characteristic on their specifications. Example: *"5. Look for the best cost vs. benefit among the laptops that fill the above specifications."*

We observed that participants with high degree of knowledge about the domain express themselves with fine-grained features (e.g. laptop-specific features) and participants with low degree of knowledge tend to refer to high-level features, as it can be seen in Figure 2.4(a), in which the level of detail of the preference specifications increases as the knowledge about the domain grows. Participants with a lower degree of knowledge specify their preferences without detailing too much the specific features of the laptop. They use high-level features to describe what they want and for what they need a laptop. Some of them mention that they would ask a friend who understands the domain for receiving a recommendation — these are not even interested in learning about the domain. On the other hand, participants with a higher degree of knowledge were much more specific, stating the exact values (or a range) for most of the laptop specific features. The level of precision in defining

	SS**(M)**	SS**(Median)**	SS**(SD)**
Brief	60.79	59.06	7.40
Basic	61.48	59.79	7.07
Detailed	64.14	61.73	9.71
No description	75.53	75.53	22.39
Total	**62.25**	**59.76**	**8.43**

Table 2.8: Preference Types vs. Domain Specialist Recommendation.

feature values decreases as the domain knowledge increases, but even EXPERTS use inaccurate adjectives (18.75%).

Even though this specification is supposed to instruct an individual to execute a task for the participant, there is a certain degree of autonomy — choosing the laptop. Some participants (6.02% ADVANCED and 8.33% EXPERT) did not provide a specification but gave the exact model they want. One of the participants stated *"I would never delegate such a decision to someone else."* This shows a group of people that do not trust other parties to decide on their behalf (at least for certain tasks). However, there are still other kinds of support that could be provided, such as making recommendations from which users could choose and make the final decision or checking prices in different stores, as stated by one of the participants: *"SEARCH other stores, before buying it."*

We have analysed the similarity score of the domain specialist recommendation used previously with respect to these preference types, in order to know how useful they were to make choices on the participants' behalf — the results are summarised in Table 2.8. We then used the Kruskal-Wallis test, performing the post-hoc tests of Nemenyi-Damico-Wolfe-Dunn. The results show that the similarity scores significantly differs across the different specification types, $H(3) = 7.882$, $p = 0.0485$. And the posthoc analysis shows us that the difference is due to the difference between the **Brief** and **Detailed** specifications (p-value= 0.0492).

We have also analysed the effect of age and gender on the preference specifications. However, the data obtained does not allow us to conclude significant difference among these different groups. An initial investigation indicates that differences of specifications provided by participants of different ages or genders are related to their levels of domain knowledge. Further investigations about it are outside the scope of this study.

RQ5. Which user profiles take less time to express their preferences? Table 2.9 shows how long (average, median and standard deviation) participants took for providing their initial preference specification, according to their domain knowledge, gender and age. We have observed different task times among participants with different domain knowledge.

		Specification Time		
		M	Median	SD
Domain Knowledge	NO_KNOWLEDGE	04:30	02:42	05:38
	BEGINNER	04:57	03:40	04:05
	INTERMEDIATE	06:35	05:12	05:11
	ADVANCED	06:45	05:49	05:28
	EXPERT	06:14	04:44	07:57
Gender	MALE	05:33	04:12	04:47
	FEMALE	06:44	05:21	06:27
Age	16-25	06:09	05:14	04:37
	26-35	06:32	05:17	05:53
	36-45	05:49	05:11	03:03
	>45	06:49	04:03	09:48

Table 2.9: Time Taken for Specifying Initial Preferences.

Participants with NO_KNOWLEDGE or BEGINNER domain knowledge took less time for building their specifications. One reason for this is that their specifications are smaller than the others'. Second, their specifications contain details about the purpose for which they need the laptop or high-level specifications, which are details that may be easier to remember. The participants who took longer specifying what they wanted were those with INTERMEDIATE or ADVANCED knowledge. Their specifications are more detailed, but they did not promptly remember what they wanted (we observed that in the specification logs). Sometimes they went backward and changed or added details to their specifications. Finally, EXPERT participants also provided detailed specifications, but as they are more familiar with the domain, their preferences have come easier to their mind.

RQ6. When users make a choice, which ones select fewer options from among the offered ones? In other words, which user profiles are more confident in which is the right choice for them? When users know exactly what they want — they are confident that there is one option that is best for them — they choose one option from the available laptops. Therefore, by observing the number of options chosen by participants, we can have an idea of their confidence while making the choice. So, regarding the number of laptops chosen by participants, we can observe that no group has an average or a median of less than three (see Table 2.10). It indicates that even when an individual knows the domain very well, there are different options that satisfy her needs. In addition, BEGINNER and INTERMEDIATE participants have a slightly higher average and median than the other categories of domain knowledge. Possibly, they do not care about minor details of the laptops, as ADVANCED and EXPERT participants do.

There is an existing framework (Chen and Pu 2010) that considers the *objective* accuracy as one of the criteria for evaluating recommender systems, which compares the final option found with the decision tool to the target (best) option that

		Options		
		M	Median	SD
Domain Knowledge	NO_KNOWLEDGE	3.40	3.00	1.67
	BEGINNER	3.88	5.00	1.54
	INTERMEDIATE	4.10	5.00	1.30
	ADVANCED	3.67	4.00	1.44
	EXPERT	3.21	3.00	1.53
Gender	MALE	3.74	4.50	1.49
	FEMALE	3.62	4.00	1.46
Age	16-25	3.60	4.00	1.55
	26-35	3.75	4.00	1.41
	36-45	3.76	4.00	1.37
	>45	3.43	3.50	1.55

Table 2.10: Number of Chosen Laptops.

users find after reviewing all available options in an offline setting. However, as our participants did not chose only one laptop, it might lead to the conclusion that such "best option" does not exist. In the field of marketing, it is more common to talk about "client satisfaction," which is more related to the *perceived* accuracy criteria of the framework.

RQ7. Which user profiles take less steps (filtering, comparing, analysing, ...) in the process of decision-making (choosing among available options)?
Besides storing the laptops chosen by participants, we have also logged their actions each time they executed one of these actions to analyse the steps participants take in the decision process. Table 2.11 shows the data we have collected.

The catalog we presented for participants initially presented all laptops, with a short description and a small picture of each one. Additionally, the following actions can be performed in the catalog: (i) *Sort*: laptops can be ordered according to the selected value (price, name, etc.); (ii) *Filter*: different filters (price range, brand, ...) can be added or removed, when the filter links are clicked; (iii) *Show laptop details*: by clicking on the laptop name, a new window opens with the specification of the selected laptop; and (iv) *Compare laptops*: two or three selected laptops can be compared (a table is displayed with laptop features side-by-side).

Table 2.11 shows that the standard deviation of each group is high. It means that, within a group, there are participants that took many more steps to choose laptops than others. Observing the mean value, we see that the participants with lower knowledge levels took more actions to choose their options. When users have little knowledge about the domain, they need to search the catalog to learn about it.

Participants with NO_KNOWLEDGE executed random actions in the catalog, indicating that they had little idea about how to choose the laptop. BEGINNER and INTERMEDIATE participants explored much more to detail laptops in the catalog, showing their exploration of the domain. And ADVANCED and EXPERT

		Steps		
		M	Median	SD
Domain Knowledge	NO_KNOWLEDGE	7.40	1.00	13.35
	BEGINNER	5.56	2.00	9.67
	INTERMEDIATE	3.15	1.00	4.91
	ADVANCED	3.75	2.00	4.46
	EXPERT	4.04	1.00	7.20
Gender	MALE	3.86	1.00	6.41
	FEMALE	3.98	2.00	6.08
Age	16-25	3.57	1.00	6.38
	26-35	4.05	1.00	5.97
	36-45	3.62	3.00	3.71
	>45	4.68	2.00	7.77

Table 2.11: Number of Steps Taken to Choose Laptops.

participants made an extensive use of filters. As they have a more precise idea of what they wanted, they reduced the search space in order to look only at the laptops they were interested in. In case of applications that aid users on the decision process, it is essential to give a personalised assistance that considers their domain knowledge.

In these last three research questions, we do not make any statistical analysis as the standard deviation of some values are very high, for instance, the number of steps taken to choose laptops as seen in Table 2.11. Therefore, we limit ourselves to the qualitative discussion presented above.

2.3 Discussion

In this section, we discuss conclusions we derived from results of our study and present lessons learned that could be used as directions for works that aim at capturing preference specifications from users.

Based on the preference specifications provided by participants, the domain expert was not able to make the choice they made for most of them. However, some participants managed to provide adequate specifications so that the expert made the right choices on their behalf, and in the cases in which this happened, the best choice (first position) was recommended. Therefore, when users provide “good” preference specifications, it is possible to make a decision on their behalf equal to what they would have decided on their own. With the analysis of the specifications, we observed the “good” specifications tend to give the expert orientation about the attributes that matter and preferences over each of them. Nevertheless no information about their interaction (trade-off) has to be given. In general, people use heuristics and principles to resolve trade-offs (Payne et al. 1988), and as the expert tends to use the same principles, in many cases the decision converges into the same choices, made by either the specialist or by the participants. Preferences provided

by participants could be given in a short time, regardless of their knowledge about the domain; therefore, it is not unrealistic to expect users to provide preference specifications using a language that is close to natural language.

It is important to highlight one group of participants that made a particular type of specification: no delegation. This group indicated a particular model or explicitly emphasised that they would never delegate such decision to another person (or a system). Therefore, for systems that aim to automate user tasks, it is essential to consider the autonomy degree being achieved and let users control it. Without allowing the user to make the final decision in the decision making process, this group of users will never accept a decision support system.

By analysing changes that participants made after being aware of available options, we observed that most of them did not change their specifications. This indicates that, even though the preference order over available options is constructed when they are actually seen, preferences over individual attributes do not change. When they change, it is because some characteristics were forgotten or they evolved over time and the participant was not aware of it.

As we gave participants the ability to choose more than one option, we could analyse if, after going through the decision making process, they would decide for one and only one option. This was not the case, as participants chose three or four options — and not five, even though it was possible. It shows that people can reduce their search space of options to a very low number, but deciding among them is the most difficult part. In addition, it also shows that there is more than one option that people can accept as an adequate choice.

Finally, it is important to note how participants made the decision. Most of existing work about preferences relies on principles of multi-attribute decision theory (Keeney and Raiffa 1976), and psychology work indicates that people do not follow them (Tversky 1996). If we aim to automate users' tasks, it is important to consider human behaviour, and not only what "rational" decision makers should do, according the definitions of economy. The analysis of the steps that participants with ADVANCED or EXPERT domain knowledge took to make the decision showed that the approach was similar to the one proposed by Tversky (Tversky 1972), namely elimination by aspects, in which people apply cut-off values for attributes according to their relevance.

As discussed above, there is a group of users who are able to express their preferences in such a way that someone can make an appropriate decision on their behalf; and other users need help to specify their preferences. Based on these two groups, we identify different kinds of support for each of them: (a) help for users of the second group to better express their preferences ; and (b) a language that is expressive enough for users of the first group to state their preferences. Next, we

present a discussion related to these issues.

2.3.1 Supporting the Preference Expression

Research work on preference elicitation has reported different techniques for it. The kind of support we are looking at is not to elicit preferences from scratch, but to identify issues in preferences specified by users and to help them be more precise. According to the preferences changes of our study, we identified that users do not provide wrong information, but incomplete or out-of-date information in case of values that change over time. In such situations, information about the domain should be provided, such as features left unmentioned, new features and updated values. However, this must take into account the domain knowledge of users so as not to annoy them with things they are aware of. Moreover, some of our participants provided templates of how they specify preferences about laptops, with variables for features that change over time. This can be adopted for providing guidance for users with a starting point for their specification, thus reducing the effort necessary to accomplish this task.

Our study also showed that users typically adopt imprecise adjectives in their preferences statements, even when they are domain experts. A *good* video card has a different meaning for a user who plays games and another who watches movies. Therefore, these adjectives should be identified and scales should be shown to users so they can rate what is "good" or "fast." But the point is to let users express themselves to obtain better specifications later. The same situation happened frequently with the term "cost-benefit." Only one of the participants provided an accurate specification for that: *"5 - the secondary laptops should only be chosen if and only if the price for the same configuration differs around 500 reais, or is 20% or 30% lower than the highest price."* A common issue is also dealing with subjective characteristics, e.g. "modern design," "beautiful." In these cases, samples of groups of items could be shown to users so we could understand what they meant. Naturally, we are not excluding the help of learning algorithms as a complementary approach.

2.3.2 Providing Different Forms of Expressing Preferences

The second point focuses on identifying preference expressions that should be part of a domain-neutral metamodel to represent user preferences, which can be instantiated for different application areas. By analysing the preference specifications of our study, we have concluded that they are significantly different when provided by users with different degrees of domain knowledge. Yet, even

I prefer $\langle target \rangle$	$\langle target \rangle$ is attractive
I (don't) need $\langle target \rangle$	$\langle target \rangle$ is interesting
It is desirable $\langle target \rangle$	I want $\langle target \rangle$
Avoid $\langle target \rangle$	I prioritise $\langle target \rangle$
I (don't) like $\langle target \rangle$	Observe (attribute) $\langle target \rangle$
$\langle target \rangle$ should (not) be A	$\langle target \rangle$ is (not) required
I (don't) want $\langle target \rangle$	I don't care too much about $\langle target \rangle$
$\langle attribute \rangle$ can be $\langle target \rangle$	I make a decision based on $\langle target \rangle$
It is nice to have $\langle target \rangle$	

Table 2.12: Expressive speech acts adopted by participants in assessment statements.

though specifications are indeed different, there is no significant difference in the domain specialist matches among the groups with different knowledge. Therefore, different forms of preference expression must be provided to users, and they are equally important.

We next present common patterns and expressions that we identified in the preference specifications given by the study participants. We group these observations into two main groups, the first associated with preference statements; and the second, preference targets. Preference statements consist of language constructions that indicate preferences over a domain, while targets are the kinds of objects that are referred to by statements. Moreover, we also point out other observed issues, which are related to perlocutionary acts (Back 2006) and trade-offs.

Preference Statements

Five main types of preference statements were identified in the provided preference specifications, which involve *monadic* and *dyadic* preferences. Monadic (classificatory) statements (Hansson 2001) evaluate a single referent, as opposed to dyadic statements, which refer to two referents. The identified types are presented next, and they are associated with one or more targets, which are discussed in next section.

Assessment. In assessment statements, users evaluate a target with a rate or an expressive speech act. We observed that participants, and more generally people, widely use **expressive speech acts**, which include want, need and desire, to express how much they want a certain preference to be satisfied, and we compiled these expressive speech acts in Table 2.12. Although assessment statements refer to a single target, they implicitly establish an order relation among elements, with the information of how much an element is preferred (or equivalent) to another.

Example: *I rate* $< target >$ *with the value* $< rate >$.

Reference Value. Reference value statements enable users to indicate one or more preferred values for an element. These preferred values can also be specified as an **interval**. For instance, there are many occurrences of the specification of an interval for screen size, which is limited by a lower bound, an upper bound, or both.
Example: *I prefer* $< target >$ *as close as possible to* $< reference\ value >$.

Goal. A goal indicates that the user preference is to minimise or maximise a certain element.
Example: *I prefer to maximise* $< target >$.

Order. Order statements establish an order relation between two elements, stating that one element is preferred (strictly or not) to another. A set of instances of the order preferences comprises a partial order.
Example: *I prefer* $< target_1 >$ *to* $< target_2 >$.

Indifference. Indifference statements consist of the indication of a set of elements that are equally (un)important to the user.
Example: I am indifferent to $< target_1 >$ *and* $< target_2 >$.

Many approaches use questions and answers to derive numeric values for user preferences, so that these numbers can be used to calculate a recommendation or make a choice on behalf of the user. However, instead of making users go through a tedious questionnaire, approaches can provide users with the ability of expressing statements such as those presented in Table 2.12. By capturing expressive speech acts used by people, and adopting an interpretation for them, such approaches can detect: (i) which preferences are user requirements (hard constraints); (ii) which attribute values are not the best, but acceptable; and (iii) which attribute values users would appreciate, but are not essential.

With respect to the relative importance among preferences and features, we observed that participants explicitly compared the importance of two features (e.g. “the performance of the laptop is more important to me than its price”) and among feature values (e.g. “I prefer brands A, B and C, in this order”), but several participants also **ordered preference statements**. These participants ranked the provided statements in their specifications, indicating the statements relevance. In addition, the relevance was also expressed by using different expressive speech acts as presented previously. Together, this information is an indication of how much people want to satisfy a particular preference. Moreover, some users explicitly stated which preferences are hard constraints: “*if my preferences are not satisfied, don’t buy it.*” Finally, participants also indicated features that they do not care about.

Preference Targets

Now that we have shown different constructions of preference statements, we focus on their targets. In the literature, often preferences are over a *class* of options (e.g. laptop), which is composed of attributes (or features), and each of which is associated with a domain of values. Given this way of describing a domain, we identified three main preference targets: (i) **class:** I prefer *laptops* to *desktops*; (ii) **attributes:** I don't care about the laptop *colour*; and (iii) **attribute values:** I don't like *laptops* whose *colour* is *pink*. An attribute value is commonly a value of the respective attribute domain, but for some attributes, participants also used subjective values, such as the examples following.

- *Speed*: very slow, slow, normal, fast, very fast.
- *Size*: tiny, small, normal, big, huge.
- *Weight*: very light, light, normal, heavy, very heavy.

Therefore, in many cases, **more than one domain** can be associated with attributes. Moreover, besides grouping attribute values using categories, participants also adopted **adjectives to qualify these attributes**, giving an indication of which values are preferred for an attribute but not being specific, as there is no obvious metrics associated with these adjectives. Examples are: *"comfortable keyboard," "light laptop"* and *"fast laptop."* These adjectives can also be categorically quantified, e.g. *"I prefer a laptop with a screen, whose visibility is **good**."* Furthermore, some of these adjectives are **subjective**, their perception is different for each individual, such as those related to design, quality, reputation and fragility.

Participants can also add new **high-level attributes** (or high-level features) to describe an option, such as a laptop. Participants, mainly those that are not domain experts, tend to express their preferences in terms of high-level attributes about options, being used as a proxy for a set of attributes. For instance, *performance* is related to the processor speed, RAM memory, and so on. The notion of high-level attributes matches the concept of *value*, discussed by Keeney (Keeney 1944). *"Values are what we care about. As such, values should be the driving force for our decision-making. They should be the basis for the time and effort we spend thinking about decisions."* It describes preferences not related to characteristics of the object but the value it brings. Finally, in many situations, participants used a reference option (**prototype**), and express their preferences with respect to it, for instance: *"(second) best model," "most up-to-date," "top configuration"* and *"check the most expensive processor, choose one 10% or 15% cheaper."* This is an interesting way to capture preferences about domains that evolve over time: even though new features of laptops appear constantly, the process of looking for them and stating features

relative to a reference point can be used as a pattern for future executions of the task.

Perlocutionary acts

Perlocutionary acts (Austin 1975, Searle 1969) refer to what one achieves by saying something, i.e. the acts that are consequence of saying something. In many cases, when people say something, they mean something else — and this is the situations we report in this section.

First, there are laptops that, generally, are chosen for particular **user profiles**. Therefore, some participants provide their characteristics as an indication of what type of options would be better for them, such as *"I like playing games"* and *"all the places I usually go to are supplied with energy."* Second, participants also indicate which kind of laptop they want by showing for what **purpose** the laptop is going to be used, e.g. *"I want to be able to watch videos on YouTube"* and *"my notebook is a desktop replacement."* Finally, there are cases of **implicit preferences**, as some preferences are common sense, and some participants express them in an implicit way. For instance, there were preference specifications that mentioned: "5. Price," which explicitly indicates that price is relevant. However, it is implicit that the preference is to *minimise* the price.

Trade-offs

Users typically face trade-off situations, such as choosing between a small laptop, with low weight and size, and a big one, with a big screen, but heavy. As the participants did not have available options when they provided their preferences in our study, they did not indicate concrete laptop examples showing the trade-off resolution. The most common expression adopted by participants to indicate how someone should resolve trade-off situations on their behalf is: *"choose a laptop with a good (or the best) cost-benefit relationship,"*, or its variant — *"optimise cost-benefit relationship."* Additionally, few participants provided further details about this relationship, as shown below.

- *"Minimise price"* together with *"I prefer property X, even if that implies a higher price."*
- *"A better brand justifies a higher price at most in 25%."*
- *"5 - the secondary laptops should only be chosen if and only if the price for the same configuration differs around 500 reais, or is 20% or 30% lower than the highest price."*
- *"Not too big a screen because it means a laptop that is too heavy."*

2.4 Final Remarks

This chapter presented a study whose focus is to provide a deeper understanding about user preference specification. We investigated user's ability in expressing their preferences about domains with which they might be familiar with or not, without having experienced a prior decision-making process in such domain with the same options, i.e. users are not aware of the available options. We have targeted the identification of the characteristics of preference specifications provided by users with different degrees of domain knowledge, and how effective they were in order for a domain specialist to use those specifications to make decisions on the users' behalf.

Seven research questions were analysed individually. Our main findings were that users with different knowledge about our study domain, laptops, provide different types of specifications — they are significantly different, according to four types of specifications that we identified as patterns. Users with lower degree of knowledge mainly give high-level preferences and personal information, such as for what purpose the laptop will be used. On the other hand, expert users provide information about fine-grained features. Despite these differences, domain specialists are able to provide recommendations of the same quality for all groups. Therefore, it is essential to provide a rich vocabulary for users to express their preferences, including coarse- and fine-grained preferences. Moreover, we observed users typically provide the right information about their preferences, but they might be incomplete or outdated for preferences whose values evolve over time. This made the domain specialist make a choice on behalf of participants that has at least half of the characteristics of choices made by participants. In addition, mechanisms to help to remember about features to be mentioned and eliminate subjectivity in specifications must be adopted. Finally, we discussed relevant issues to be addressed by preference languages to be used by end-users, such issues include common patterns and expressions of preferences that are typically adopted by people.

It is important to highlight that a limitation of our study is that our findings are based solely on a single domain and the opinion of one specialist, which was due to the lack of other specialists willing to collaborate with our study. However, the knowledge extracted from this study already provides valuable information to propose a domain-neutral metamodel to represent user preferences, which is presented in next chapter. This metamodel takes into account all the identified preference patterns and expressions.

3
Preference Metamodel

Chapter 2 presented a user study in which a set of preference specifications in natural language was collected. These specifications allowed us to analyse patterns and common expressions that people typically adopt to express their preferences. In this chapter, we describe a preference metamodel developed based on our previous study. The goal is to provide constructions to model user preferences in a way independent from the target application area and to allow users to use a language as close as possible to natural language. Besides using our user study as a source to build our model, we also took into account existing preference representation models in computer science (which will be discussed in Chapter 4) and other areas, such as psychology and philosophy.

As preferences are expressed in terms of entities of specific application areas, we first introduce an ontology metamodel, before detailing the preference metamodel. The ontology metamodel defines how these entities are structured so that they can be referred to in preference specifications.

All diagrams presented in this chapter are modelled with Unified Modeling Language (UML). Entities of our metamodel are highlighted in **boldface**, and examples of preferences and other entities are in *italics*. UML was adopted to introduce our metamodel, because it is simple, widely known and used. However, as UML lack a formal semantics, we also present a specification of our preference metamodel using the Z notation (Wordsworth 1992) (Appendix B), which is a formal specification language used for describing and modelling software. This formal specification models constraints and other aspects of our metamodel that are described informally in this chapter and cannot be represented in UML.

3.1
Ontology Metamodel

Most of the existing work in the context of preferences and decision making defines a decision problem by considering a set of alternatives (or outcomes), which are structured in terms of features $X_1, ..., X_n$, each of which associated with a particular domain $Dom(X_i) = x_1^i, ..., x_{n_i}^i$. The set of all possible alternatives is therefore $Dom(X_1) \times ... \times Dom(X_n)$. The subset of possible alternatives is called

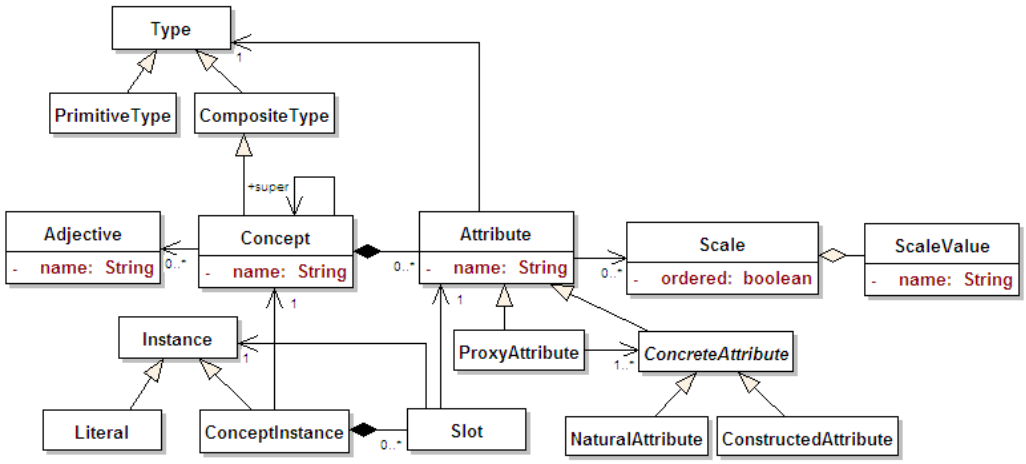


Figure 3.1: Ontology metamodel.

feasible outcomes.

Our experimental study has shown that users adopt a richer vocabulary than attributes and their (single) associated domain to refer to entities of the application area, thus not only making statements about the specific features of entities. Based on the analysis of the specifications collected in our study, we have defined an ontology metamodel, depicted in Figure 3.1, which structures entities of an application area. It is also built on the foundation of decision making research work (Keeney 1944).

Our model defines a main coarse-grained entity, namely concepts. A **concept** represents a class of elements that is composed of attributes, and is denoted by a name. As it is a **type** composed of finer-grained entities, it is a **composite type**. A concept typically represents a class of concrete entities of the world, such as a *laptop*.

Attributes, which are also denoted by a name, have a type, which defines the domain of values that can be assigned to a particular attribute. Based on our study, we identified three kinds of attributes: (i) objective attributes (e.g. *size of the RAM memory*); (ii) attributes that are built subjectively by individuals (e.g. *quality*); and (iii) attributes that represent a collection of other attributes, serving as a proxy to them (e.g. *laptop size*, for which the real attributes are the dimensions of the laptop). These three kinds of attributes are referred to as **natural attributes**, **constructed attributes** and **proxy attributes**, respectively. The first two kinds are collectively referred to as **concrete attributes**. We adopted these terms because they match similar concepts proposed by Keeney (Keeney 1944), used to measure the achievement of objectives.

The type of an attribute can be either composite, already introduced, or primitive. **Primitive types** are the basic building blocks to create composite types. These types, presented on the left hand side of Figure 3.2, can be a single character

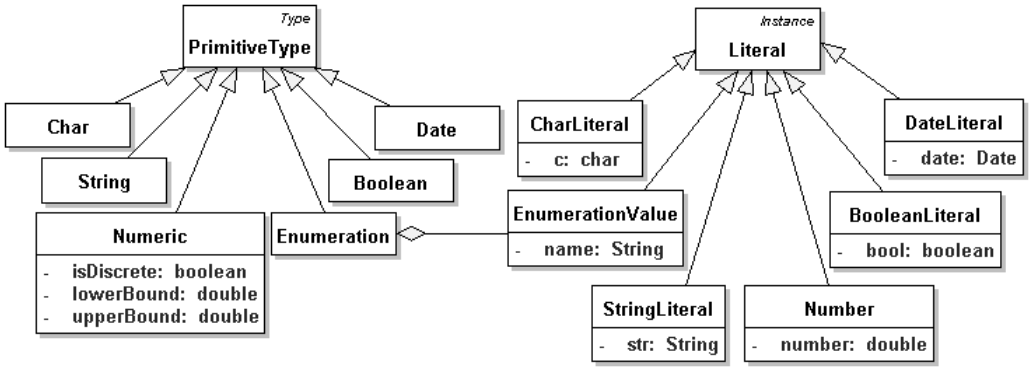


Figure 3.2: Ontology primitive types.

(**char**), a **string** (a sequence of characters), **numeric** (discrete or continuous, optionally with lower and upper bounds), **boolean** (true and false), a **date** and an **enumeration** (a set of elements, possibly having a natural order).

Even though instances of a concept always have a particular value assigned to each of its attributes, users typically use fuzzy domains to express preferences about attributes values, e.g. "*I want a big screen*." Therefore, we associate attributes not only with their type but also with a set of **scales**, which are composed of **scale values**. An example could be a scale *size*, whose values can be *tiny*, *small*, *normal*, *big* and *huge*.

Another common construction adopted by our study participants is adjectives. This is illustrated by the expression "*fast laptop*," which can be translated to a laptop whose processor speed is high and RAM memory is big. This construction is supported by associating a concept with a set of possible **adjectives**, which has a precise meaning defined in a specific ontology. Adjectives are not grouped to form a scale (as above), because adjectives can be translated to a set of statements in terms of attributes, as described in the example. The statements translated from adjectives are, in turn, situated in an ordered domain according to an attribute type, or in a scale.

Finally, types are generic representations for classes of particular **instances**. Instances of primitive types are **literals**, which are specific values assigned for **slots**, which in turn hold a value associated with an attribute. This value is one of the possible values of the primitive type associated with the attribute. For instance, if the attribute is associated with an enumeration, the literal assigned for the slot must be an enumeration value. The different literals are shown on the right hand side of Figure 3.2. **Concept instances**, on the other hand, are a composition of these slots, which are place holders for values (instances) assigned for each of the attributes that are associated with the concept of the instance.

3.2 Propositional Formulae

In different parts of preference specifications, users use propositional-formula-like constructions, for example, to express conditions (*if the laptop is a Mac*) or to restrict possible attribute values (*screen size between 14" and 15"*). So, we have developed a representation for propositional logic formulae, which are used for different purposes in our model, such as building constraints and conditions.

We captured these constructions in the model shown in Figure 3.3, which allows modelling formulae with three logic operators **not** (`NotFormula`), **and** (`AndFormula`), and **or** (`OrFormula`). We define a generic entity, namely `AtomicFormula`, to represent **atomic formulae**, and in particular we define three kinds of them, detailed below.

– **Attribute value specification**, which indicates a restriction over a domain of an attribute. For instance, "*Screen.size = 15 inches*". As an attribute is always part of a concept, we represent an attribute by <*concept*>.<*name*>, where *concept* is the concept that the attribute is part of, and *name* is its name. We represent concept names initiated with uppercase, and attribute names initiated with lowercase.

  An attribute is associated with a concept, and this concept can be the type of different attributes, which are part of different concepts. Consequently, preferences for an attribute value (of a concept) can be different, when this concept is related to different concepts, i.e. different contexts, e.g. preferences for colour depend on which concept colour is associated to: *I prefer white*

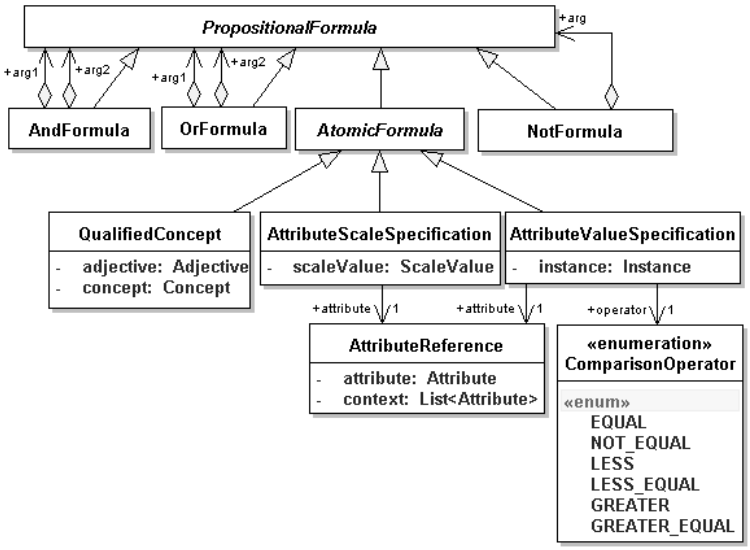


Figure 3.3: Propositional formula model.

houses and *I prefer red cars*. Therefore, this type of formula is represented as shown below:

<context><attribute><comparison operator><instance>.

An example is:

context: Trip.returnFlight, Flight.company
attribute: Company.group
comparison operator: = (equal)
instance: OneWorld

In this example, the context is composed of two attributes for defining the context of the attribute *Company.group*. Note that the type of a previous attribute in the context list must match the concept to which the following attribute is part of (the type of *returnFlight* is *Flight*). The same occurs with the last attribute of the context and the attribute. In Figure 3.3, context and attribute together are represented as an attribute reference.

– **Attribute scale specification**, which is represented as

<context><attribute><scale value>.

It also establishes a restriction over values of a particular attribute, but instead of specifying specific values, a scale value associated with the attribute is specified. Example:

context: Laptop.ramMemory
attribute: RAMMemory.size
scale value: Big

– **Qualified concept**, which qualifies a concept with an adjective, e.g. "*fast laptop*." In theory, the notion of context could also be adopted to associate the concept being qualified with a particular context, but this was not observed in our study of how humans express preferences, so we did not include it in our metamodel.

3.3 Preference Metamodel

In this section, we describe the preference metamodel, divided into multiple parts. We first present, in Section 3.3.1, an overview of all preference types and priorities of our metamodel, and how we model conditions and contexts associated with those preferences. Next, we describe preference constructions to express goals and constraints (Section 3.3.2). Constraints, in turn, are used to construct more sophisticated preferences, namely preference statements, which are presented in Section 3.3.3. Then, we detail how to express preferences over preferences, i.e. priorities (Section 3.3.4). Finally, Section 3.3.5 shows a set of examples of the different kinds of preferences presented.

3.3.1 Overview

Our metamodel is composed of different kinds of preferences and how to express priority among them. All of these are shown Figure 3.4, which overviews our metamodel, showing its main entities and how they are connected by generalisation or association. This figure also shows the two entities that are part of our preference metamodel, namely conditions and decision context, and indicate when preferences and priorities are applicable. These entities are represented in the same way, i.e. by a propositional logic formula, but they have two different meanings, as detailed next.

(i) **Condition.** Values assigned to certain attributes might impact preferences over other attributes, concepts, and so on. A condition specifies a set of values for attributes, to which a preference is subjected. Example: "*if Laptop.brand = Mac, <preference>*." Conditions can also be used for specifying priorities.

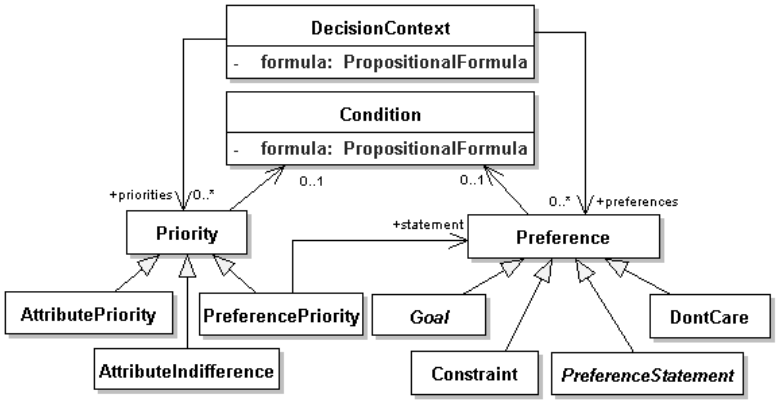
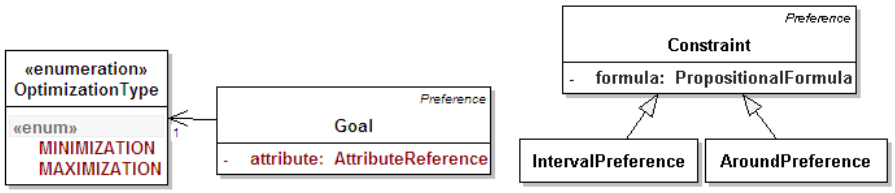


Figure 3.4: Overview of the preference metamodel.



3.5(a): Goal. 3.5(b): Constraint.

Figure 3.5: Goals and Constraints.

(ii) **Decision context.** Decision contexts model a state of world in which a particular set of preferences and priorities is relevant. Example: "*if the purpose of buying the laptop is for business work, <preference>*." Moreover, adjectives of the ontology as well as scales have a specific meaning for a particular decision context, i.e. "light laptop," for instance, can have different interpretations in personal or business contexts.

3.3.2 Simple Preferences

When users know the application area in which they are expressing preferences, they are aware of the possible values that concepts and attributes can have and they have preferences over these values, which impose restrictions on what these values should be. Moreover, users might not be interested in values or certain attributes. This kind of preferences are presented in this section, which we call **simple preferences**, as opposed to preferences presented in the next section, which involve expressing not only attributes and their values, but also a more sophisticated vocabulary, involving expressive speech acts and rates. Three kinds of preferences are introduced next: goals, constraints and don't care.

Goals (Figure 3.5(a)) represent an overall rule that indicates when an attribute value is preferred to another. Users state that they prefer to minimise or maximise (**optimisation type**) a particular attribute (**attribute goal**). An examples is: "*Minimise Laptop.price*."

A **constraint** (Figure 3.5(b)) represents a restriction over the values that can be selected for a particular attribute or set of attributes — it depends on which attributes are referred to in the propositional formula associated with the constraint. Users can specify any formula, but two predefined formulae are provided, as they are commonly expressed by users, as described next.

(i) **Interval preference**: it indicates that the value of an attribute should fall within a range of two provided values. Example: "*Laptop.hardDrive.size between 500GB and 750GB*." The propositional formula associated with this preference is an *and* formula composed of two attribute value specifications,

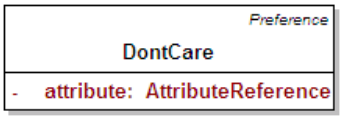


Figure 3.6: Don't care preference.

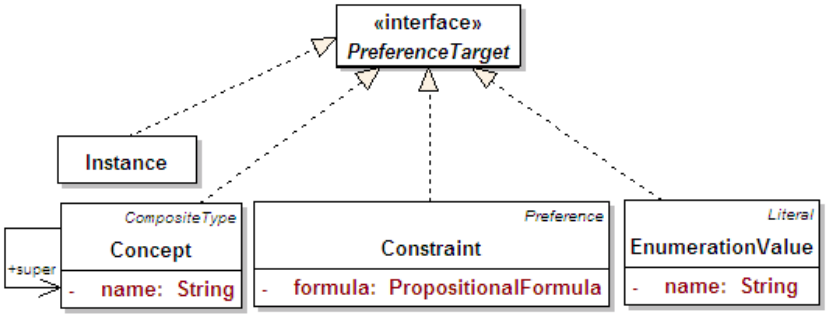


Figure 3.7: Preference targets.

one with the comparison operator *less than* (or *less than or equal to*) and the other with the comparison operator *greater than* (or *greater than or equal to*). Both attribute value specifications must refer to the same attribute.

(ii) **Around preference**: it indicates that an attribute should have a value that is close to a provided reference value, i.e. the closer the attribute value is to the reference value, the better. Example: "*Laptop.hardDrive.size around 500GB*." The propositional formula associated with this preference is an attribute value specification, whose comparison operator is *equal to*.

These two kinds of preferences do not state *how much* users prefer the restriction captured by this preference or make any kind of comparison — they are interpreted as "*I want to* ⟨*goal or constraint*⟩."

In addition, users might indicate that a certain attribute or value is totally irrelevant for them, by stating for instance "*I don't care about price.*" This means that the user does not impose any restriction to the values of this attribute, and is indifferent to all possible values that can be assigned to it. This kind of preference is referred to as **don't care**, and refers to an attribute as shown in Figure 3.6.

3.3.3 Preference Statements

Preference statements are sentences provided by users that state preferences over different entities of an application area, such as concepts and instances, and also the relevance of constraints. These entities and constraints are called **preference targets**, as they are targets of the provided preference statements. Figure 3.7 indicates the possible preference targets by showing which previously

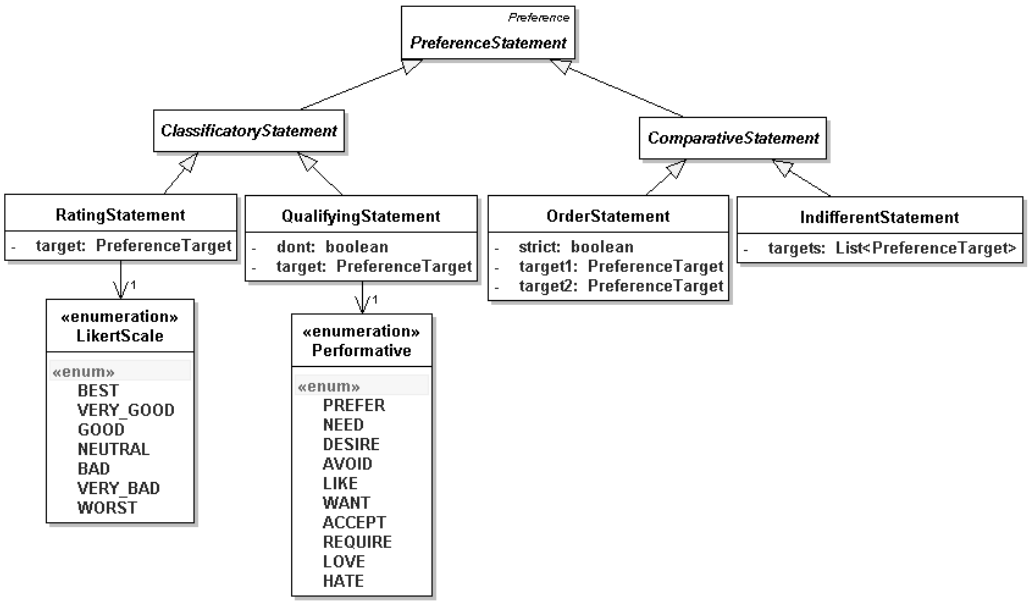


Figure 3.8: Preference statements model.

entities are a type of preference target. As preference statements are constructed by referring generically to targets, they can be any of the elements presented in Figure 3.7.

According to Hansson (Hansson 2001), preferences can be classified in two groups: (i) *monadic* preferences, which evaluate a single target, e.g. "*I like skiing*," and use terms such as "good," "very bad," and "worst;" and (ii) *dyadic* preferences, which indicate a relation between two targets, e.g. "*I prefer skying to surfing*," and use terms such as "better," "worse," and "equal in value to." These two main kinds of statements were indeed identified in our study, which are modelled as **classificatory statement** and **comparative statement**, respectively, as illustrated in Figure 3.8.

Our study showed that users not only classify preference targets as "good" and "bad" (**rating statement**) as suggested by Hansson (Hansson 2001), but also make extensive use of different expressive speech acts in their statement. This is captured by **qualifying statements**, which are speech acts (Searle 1969, Austin 1975) that classify targets with **expressive speech acts**, such as "*I need a light laptop*," "*I avoid Laptop.brand = Acer*." In addition, these expressive speech acts can also be used in the negative form, e.g. "*I don't need a light laptop*," represented by the attribute **don't** in qualifying statements. These different expressive speech acts are natural language expressions that indicate how hard (or soft) a preference is.

The other way of expressing preference statements is by stating an explicit comparison between two or more preference targets. The comparison can state that a target is preferred to another, strictly or not (**order statement**) or that a set of targets are equally preferred (**indifferent statement**).

3.3.4 Preference Priority

Users express different preferences about an application area, but these preferences can conflict with each other. In order to resolve these conflicts users indicate preferences over previously stated preferences, showing which are more relevant for them. This notion of preferences over preferences is represented as **priorities** in our metamodel.

The most common way that participants of our study indicated priority was numbering preference statements or restrictions, which is modelled with **preference priority** in our metamodel — see Figure 3.9. The other way was by stating relative importance over attributes, e.g. "*Laptop.quality is more important than Laptop.price*" (**attribute priority**) or "*Laptop.quality and Laptop.price are equally important to me*" (**attribute indifference**).

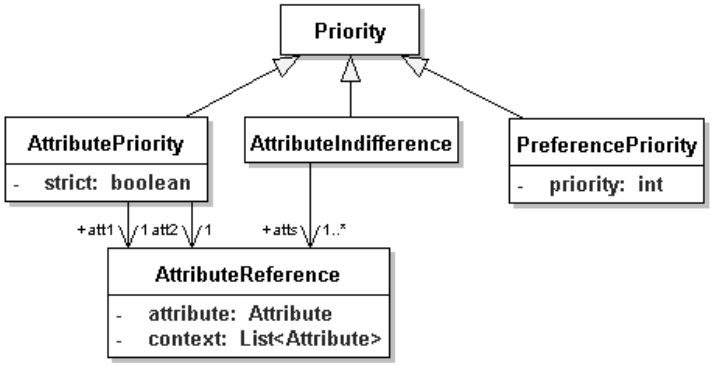


Figure 3.9: Preference priority model.

3.3.5 Interaction among Preferences and Targets

In previous sections, we presented different kinds of preferences: goals, constraints, don't care and four types of statements. Statements can refer to different entities of the ontology metamodel as well as to constraints, which are generically referred to as preference targets. In this section, we show how different statements interact with each target, by showing examples of their each possible combination. These examples are presented in Table 3.1(a). In addition, in order to make the remaining preferences and priorities clear for reader, we also provide examples of goals, don't care preferences, and the three types of priorities in Table 3.1(b). Therefore, Table 3.1 summarises most of the preference expressions that can be represented with our metamodel.

Table 3.1: Examples of preferences and priorities.

Type	Rating	Qualifying	Order	Indifferent
Ontology elements				
Concept	Laptop is good.	I need a laptop.	I prefer laptop to desktop.	I am indifferent to laptops and desktops.
Enumeration value	Color.pink is very good.	I accept Color.blue.	I prefer Color.blue to Color.green.	I am indifferent to Color.blue and Color.green.
Instance	HP is good.	I need a Dell.	I prefer HP to Dell.	I am indifferent to HP and Dell.
Constraints				
Constraint	Laptop.screen = small is very bad.	I like Laptop.screen = big.	I prefer Laptop.screen = big to Laptop.screen = small.	I am indifferent to Laptop.screen = big and Laptop.screen = small.
Interval	Laptop.screen.size between 14” and 15” is good.	I like Laptop.screen.size between 14” and 15”.	I prefer Laptop.screen.size between 10” and 13” to between 14” to 15”.	I am indifferent to Laptop.screen.size between 10” and 13”, and between 14” to 15”.
Around	Laptop.screen.size around 15” is good.	I like Laptop.screen.size around 15”.	I prefer Laptop.screen.size around 15” to around 10”.	I am indifferent to Laptop.screen.size around 15” and around 10”.

(a) Preference statements and targets interaction.

Goal	Minimise Laptop.weight.
Don’t Care	I don’t care about Laptop.bluetooth.
Priorities	
Attribute priority	Laptop.size is more important than Laptop.weight.
Attribute indifference	Laptop.size is as important as Laptop.weight.
Preference priority	1. Minimise price. 2. I need Screen.size ≥ 15”.

(b) Goals, don’t care preferences and priorities.

3.4 Final Remarks

In this chapter, we described a preference metamodel that contemplates patterns and expressions that are used by people and were identified in our study of how humans express preferences. The metamodel includes an ontology metamodel, whose instances describe an application area, such as laptops and their attributes. Propositional formulae are also represented, as many preferences refer to them to indicate preferred attribute values. Our metamodel consists of different types of preferences, such as goals and constraints, or more complex preferences that are associated with natural-language-like expressions, e.g. expressive speech acts. Many of these preferences are not part of existing preference metamodels, and this limitation of existing approaches is discussed in next chapter.

4
Related Work on Preference Representation

Given that we presented in the previous chapter a preference metamodel based on a study of how humans express preferences, we now introduce existing research work in the context of preferences that proposes representation models of preferences. These models may be part of approaches that include algorithms to reason about preferences, but our focus in this chapter is to describe how these approaches *represent* preferences, and to compare which kind of preferences can be *explicitly* expressed by the different proposed models, showing their limitations with respect to our metamodel.

Preference representation models are split into five groups. First, we present approaches that use constraints to represent preferences in Section 4.1. The second group consists of approaches that represent preferences using graphs, which are detailed in Section 4.2. Most of the approaches of these two groups are investigated within the artificial intelligence research area, but there are preference representation models proposed in the area of databases and semantic web, which are described in Sections 4.3 and 4.4, respectively. Finally, we present a recent approach that provides a non-parametric way of representing preferences in Section 4.5. We compare the presented research work in Section 4.6, and conclude in Section 4.7.

4.1
Constraint-based Approaches

4.1.1
Soft-constraints

Soft constraints model quantitative preferences by generalising the traditional formalism of hard constraints. Bistarelli et al. (Bistarelli et al. 1997) define a constraint solving framework where all such extensions, as well as classical Constraint Satisfaction Problems (CSPs), can be cast. The main idea is based on the observation that a semiring (i.e. a domain plus two operations satisfying certain properties) is all that is needed to describe many constraint satisfaction schemes. The domain of the semiring provides the levels of consistency (which can be interpreted, for example, as cost, degrees of preference, probabilities), and

the two operations define a way to combine constraints together. So, in a soft constraint, each assignment to the variables of a constraint is annotated with a level of its desirability, and the desirability of a complete assignment is computed by a combination operator applied to the local preference values.

Soft constraint-based approaches rely on Soft Constraint Satisfaction Problems (SCSPs), which are an extension of CSPs. The definition of a SCSP is based on the definitions of constraint system and constraint, and both involve the choice of a c-semiring — in this term, "c" stands for "constraint," meaning that this kind of semiring is a natural structure to be used when handling constraints. Basically, a SCSP defines the c-semiring being used, a set of attributes that describe an application area, and their associated domains. In addition, there is a set of constraints, each associated with a value that is interpreted according to the chosen c-semiring. Formally, a SCSP is defined as follows (Bistarelli et al. 1997).

1. **Constraint system**. A constraint system is a tuple $CS = \langle S, D, V \rangle$, where S is a c-semiring, D is a finite set, and V is an ordered set of attributes.

2. **Constraint**. Given a constraint system $CS = \langle S, D, V \rangle$, a constraint over CS is a pair $\langle def, con \rangle$, where

   - $con \subseteq V$, it is called the *type* of the constraint;
   - $def : D^k \rightarrow A$ (where k is the cardinality of con and A is the set of possible values representing the penalty, cost, preference, weight, etc.) is called the *value* of the constraint.

3. **Constraint problem**. Given a constraint system $CS = \langle S, D, V \rangle$, a constraint problem P over CS is a pair $P = \langle C, con \rangle$, where C is a set of constraints over CS and $con \subseteq V$. We also assume that $\langle def_1, con' \rangle \in C$ and $\langle def_2, con' \rangle \in C$ implies $def_1 = def_2$.

Bistarelli et al. use this framework to deal with *bipolar* preferences (Bistarelli et al. 2010), i.e. problems with both positive and negative preferences. They argue that soft-constraints *only can model negative preferences*, since in this framework preference combination returns lower preferences. So, they adopt the soft constraint formalism based on semirings to model negative preferences and also define a new algebraic structure to model positive preferences. Then, to model bipolar problems, these two structures are linked by a combination operator between positive and negative preferences to model preference compensation. Another extension to soft-constraints consists of *interval preferences* (Gelain et al. 2010), which instead of representing the *value* of the constraint as a specific number, it uses an interval, which may be easier to be specify.

4.1.2
Preference-based Problem Solving for Constraint Programming

An approach based on multi-objective optimisation was proposed by Junker (Junker 2008), considering a finite set of attributes X where each attribute $x \in X$ has a domain $D(x)$. The problem space of X is restricted by defining constraints on attributes in X. A constraint c has a scope $X_c \subseteq X$ and a "relation," which is expressed by a set R_c of assignments to the scope X_c.

Users must provide as input preferences on certain properties of the option. These criteria are mathematical functions from the problem space to an outcome domain. Formally, a criterion z with domain Ω is an expression $f(x_1, ..., x_n)$ where $x_1, ..., x_n$ are attributes from X and f is a function with signature $D(x_1) \times ... \times D(x_n) \rightarrow \Omega$. Function f can be formulated with the operators of the constraint language (e.g. sum, min, max, conditional expression) or by a table.

The user can compare the different outcomes in a domain Ω and formulate preferences between them. Preferences are modelled in form of a preorder $\succsim$ on Ω, which consists of a strict part $\succ$ and an indifference relation $\sim$. The user may also formulate *wishes* about the properties that an option should have. Such a wish is a soft constraint that should be satisfied if possible. A wish for constraint c can thus be modelled by a preference $\langle z_c, > \rangle$, which is abbreviated by $wish(c)$.

Users can inspect the conflicts between criteria and the way they have been resolved. If they are not satisfied with such a conflict resolution, they can change it by reordering the criteria. This importance is ordered in terms of a strict partial order $I \subseteq Z \times Z$ on the criteria set $Z := z_1, ..., z_n$.

4.2
Graphically-structured Approaches

4.2.1
CP-nets: Conditional Ceteris Paribus Preference Statements

Boutilier et al. (Boutilier et al. 2004) have proposed a graphical representation, namely CP-nets,[1] which can be used for specifying preference relations in a relatively compact and structured manner using conditional ceteris paribus (all else being equal) preference statements. The inference techniques for CP-nets focus on two questions: how to perform preferential comparison between outcomes, and how to find the optimal outcome given a partial assignment to the problem attributes.

In CP-nets, a set of variables $V = X_1, ..., X_n$ is assumed, over which the decision maker has preferences. Each variable X_i is associated with a domain

[1]This structures are conditional preference networks or CP-networks (CP-nets, for short).

$Dom(X_i) = x_1^i, ..., x_{n_i}^i$ of values it can take. An assignment x of values to a set $X \subseteq V$ of variables (also called an instantiation of X) is a function that maps each variable in X onto an element of its domain; if $X = V$, x is a complete assignment, otherwise x is called a partial assignment. The set of all assignments to $X \subseteq V$ is denoted by $Asst(X)$. Based on this assumption, two definitions are made.

- A set of variables X is preferentially independent of its complement $Y = V - X$, for all $x_1, x_2 \in Asst(X)$ and $y1, y2 \in Asst(Y)$, we have $x_1 y_1 \succcurlyeq x_2 y_1$ if and only if $x_1 y_2 \succcurlyeq x_2 y_2$.
- X is conditionally preferentially independent of Y given an assignment z to Z if and only if, for all $x_1, x_2 \in Asst(X)$ and $y_1, y_2 \in Asst(Y)$, we have $x_1 y_1 z \succcurlyeq x_2 y_1 z$ if and only if $x_1 y_2 z \succcurlyeq x_2 y_2 z$.

A CP-net over variables $V = X_1, ..., X_2$ is a directed graph G over $X_1, ..., X_2$ whose nodes are annotated with conditional preference tables $CPT(X_i)$ for each $X_i \in V$. Each conditional preference table $CPT(X_i)$ associates a total order $\succ_u^i$ with each instantiation u of X_i's parents $Pa(X_i) = U$.

In summary, the kinds of statements captured by CP-Nets are those that establish order among attribute values, conditioned to values set to other attributes (parent attributes).

4.2.2 TCP-nets: Modelling of Preference and Importance

Brafman et al. (Brafman et al. 2006) provide an extension of the CP-nets formalism in order to handle another class of qualitative statements — statements of **relative importance** of attributes. These statements have the form: "*It is more important to me that the value of X be better than that the value of Y be better*." A more refined notion of importance, which is also addressed, is that of **conditional relative importance**, having this form: "*A better assignment for X is more important than a better assignment for Y given that $Z = z_0$*."

The resulting extended formalism, TCP-nets (for tradeoffs-enhanced CP-nets), maintains the ideas of CP-nets, as it remains focused on using only simple and natural preference statements. In addition, it also uses the ceteris paribus semantics, and utilises a graphical representation of this information to reason about its consistency and to perform, possibly constrained, optimisation using it. The extra expressiveness it provides allows better modelling of trade-offs, with attribute importance relationships. When the TCP-net structure is "acyclic," the set of preference statements represented by the TCP-net is guaranteed to be consistent. TCP-nets are annotated graphs with three types of edges.

– A first type of (directed) edges, which comes from the original CP-nets model and captures direct preferential dependencies.
– A second (directed) edge type captures relative importance relations. The existence of such an edge from attribute X to attribute Y implies that X is more important than Y.
– A third (undirected) edge type captures conditional importance relations: such an edge between nodes X and Y exists if there exists a non-empty attribute subset $Z \subseteq V - \{X, Y\}$ for which $RI(X, Y \mid Z)$ (relative importance of X and Y conditioned on Z) holds.

In addition to the conditional preference table (CPT) of CP-Nets, in TCP-nets, each undirected edge is annotated with a conditional importance table (CIT). The CIT associated with such an edge (X, Y) describes the relative importance of X and Y given the value of the corresponding importance-conditioning variables Z.

4.3 Database Approaches

4.3.1 Scoring Function

Agrawal and Wimmers (Agrawal and Wimmers 2000) propose a framework for *expressing* and *combining* user preferences. In this framework, preferences for an entity are expressed by a numeric score between 0 and 1, vetoing it, or explicitly stating indifference (by default, indifference is assumed). The framework consists of three elements, as shown below.

– A set of **(base) types**, which typically include `ints`, `strings`, `floats`, `booleans`, etc.
– A data type called **score** that represents a user preference. Formally, this is $[0, 1] \cup \natural, \perp$. A score of 1 indicates the highest level of user preference, while a score of 0 indicates its lowest level. The "$\natural$" score, represents a veto, and the "$\perp$" score represents that no user preference has been indicated.
– A set of **record types**, which are pairs $name_1 : type_2, ..., name_n : type_n$ in which all n names (a name is simply a non-empty string) are different (although the types are allowed to be the same). In this case, $name_i$ is the name of a field in the record and $type_i$ is the type of that field. An option is a record, where each field takes on a value in the type of that field. In addition, a type is called wild if it contains "*," used to indicate a wild card that "matches" any value.

Based on these three elements, a preference function is defined as a function that maps options of a given record type to a score. Since sometimes a preference function is applied to an option with more fields than are present in the domain of the preference function, a projection operator is introduced to eliminate the extra fields.

4.3.2 Preference Formulae in Relational Queries

Chomicki (Chomicki 2003) proposes a framework for specifying preferences using logical formulae and their embedding into relational algebra. Preferences are defined using binary preference relations between tuples, which are specified using first-order formulae. The focus is mostly on *intrinsic* preference formulae, which can refer only to built-in predicates. Two infinite domains are considered in the approach: D (uninterpreted constants) and Q (rational numbers).

> Given a relation schema $R(A_1...A_k)$ such that U_i, $1 \leq i \leq k$, is the domain (either D or Q) of the attribute A_i, a relation $\succ$ is a preference relation over R if it is a subset of $(U_1 \times ... \times U_k) \times (U_1 \times ... \times U_k)$.

In addition, this preference relation has the following properties: irreflexivity, asymmetry, transitivity, negative transitivity and connectivity.

> A preference formula (pf) $C(t1, t2)$ is a first-order formula defining a preference relation $\succ$ in the standard sense, namely $t1 \succ_c t2$ iff $C(t1, t2)$. An intrinsic preference formula (ipf) is a preference formula that uses only built-in predicates.

Because two specific domains are considered, D and Q, there are two kinds of attributes, D-attributes and Q-attributes, and two kinds of atomic formulae: (i) equality constraints; and (ii) rational-order constraints. Without loss of generality, the author assumes that ipfs are in DNF (Disjunctive Normal Form) and quantifier-free. Moreover, atomic formulae are closed under negation (also satisfied by the above theories). *Indifferent* is also defined. Every preference relation $\succ_c$ generates an indifference relation $\sim_c$: two options $t1$ and $t2$ are indifferent ($t1 \sim_c t2$) if neither is preferred to the other one, that is, $t1 \succ_c t2$ and $t2 \succ_c t1$.

Finally, there is preference composition, which can be unidimensional or multidimensional. In unidimensional composition, a number of preference relations over a single database schema are composed, producing another preference relation over the same schema. In multidimensional composition, there is a number of preference relations defined over several database relation schemas, and a preference relation over the Cartesian product of those relations is

defined. Unidimensional composition can be: (i) *boolean composition*: union, intersection and difference of preference relations; (ii) *prioritised composition*: preference over preferences, represented by the $\triangleright$ operator; and (iii) *transitive closure*. Multidimensional composition, in turn, can be pareto composition and lexicographic composition.

4.3.3 Foundations of Preferences in Database Systems

A preference model tailored for database systems, proposed by Kießling (Kießling 2002), unifies and extends existing approaches for non-numerical and numerical ranking. He defined a declarative semantics of preference queries under the Best-Matches-Only (BMO) query model. His preference model includes a set of preference constructors, which allows the expression of different preferences, whose definition is given below.

Preference $P = (A, <P)$. Given a set A of attribute names, a preference P is a strict partial order $P = (A, <P)$, where $<P \subseteq dom(A) \times dom(A)$. $<P$ is irreflexive and transitive (which imply asymmetry). Important is this intended interpretation: "$x <P y$" is interpreted as "I like y better than x."

In order to build *base* preferences, there are two classes of constructors (detailed in Table 4.1): (i) *non-numerical base preferences*, which include positive and negative preferences; and (ii) *numerical base preferences*, which include preferences that specify preferred numerical values of an attribute using, for example, intervals. A preference term (i.e. a preference that is valid according to the provided constructors) can also be composed of other preferences. Given preference terms $P1$ and $P2$, P is a preference term if and only if P is one of the following.

1. Any **base** preference: $P := basepref_i$.
2. Any **subset** preference: $P := P1^{\subseteq}$
3. Any **dual** preference: $P := P1^{\partial}$
4. Any **complex** preference P gained by applying one of the following preference constructors:
   - Accumulating preference constructors:
     - Pareto accumulation: $P := P1 \otimes P2$
     - Prioritised accumulation: $P := P1 \,\&\, P2$
     - Numerical accumulation: $P := rank_F(P1, P2)$ (applied only to SCORE preferences)
   - Aggregating preference constructors:

Non-numerical base preferences	
POS(A, POS-set)	A desired value should be in a finite set of favourites $POS-set \subseteq dom(A)$. If this infeasible, any other value from $dom(A)$ is acceptable.
NEG(A, NEG-set)	A desired value should not be any from a finite set NEG-set of dislikes. If this is infeasible, any disliked value is acceptable.
POS/NEG(A, POS-set; NEG-set)	A desired value should be one from a finite set of favourites. Otherwise it should not be any from a finite set of disjoint dislikes. If this is not feasible either, any disliked value is acceptable.
POS/POS(A, POS1-set; POS2-set)	A desired value should be amongst a finite set POS1-set. Otherwise it should be from a disjoint finite set of alternatives POS2-set. If this is not feasible either, any other value is acceptable.
EXP(A, E-graph)	Let $E-graph = (val_1, val_2), ...$ represent a finite acyclic 'better-than' graph, V be the set of all val_i occurring in $E-graph$. A strict partial order $E = (V, <E)$ is induced as follows: (i) $(val_i, val_j) \in E-graph$ implies $val_i <E\ val_j$; and (ii) $val_i <E\ val_j \wedge val_j <E\ val_k$ imply $val_i <E\ val_k$. P is an EXPLICIT preference, if: $x <P\ y$ iff $x <E\ y \vee (x \notin range(<E) \wedge y \in range(<E))$.
Numerical base preferences	
AROUND(A, z)	The desired value should be z. If this is infeasible, values with shortest distance apart from z are acceptable.
BETWEEN(A, [low, up])	A desired value should be between the bounds of an interval. If this is infeasible, values with shortest distance apart from the interval boundaries will be acceptable.
LOWEST(A), HIGHEST(A)	A desired value should be as low (high) as possible.
SCORE(A, f)	Assume a scoring function $f : dom(A) \rightarrow \mathbb{R}$. Let $<$ be the familiar 'less-than' order on $\mathbb{R}$. P is called SCORE preference, if for $x, y \in dom(A)$: $x <P\ y$ iff $f(x) < f(y)$. No intuitive interpretation is given.

Table 4.1: Base Preference Constructors.

- Intersection aggregation: $P := P1 \blacklozenge P2$
- Disjoint union aggregation: $P := P1 + P2$
- Linear sum aggregation: $P := P1 \oplus P2$

4.3.4 Personalisation of Queries based on User Preferences

Koutrika and Ioannidis (Koutrika and Ioannidis 2006) have addressed query personalisation, by proposing (i) a model for representing and storing preferences in user profiles, (ii) a query personalisation framework that specifies which kind of personalised answer is generated given a query and a user profile; and (iii) query personalisation algorithms. Preferences may be expressed for values of attributes, and for relationships between entities, which indicate to what degree, if any, entities related depend on each other. The following preferences are part of the preference model.

(i) **Atomic Selection Preferences**. For any atomic selection condition q on attribute $R.A$ (of a relational table R_A, with D_A as its domain specific values), a user's preference for values satisfying (or not) q is expressed by the degree of interest in q, denoted by $doi(q)$, which is defined as follows:

$$doi(q) = \langle d_T(u), d_F(u) \rangle$$

where $\forall\ u \in D_A$, satisfying q, $d_T(u), d_F(u) \in [-1, 1]$ and $d_T(u) * d_F(u) \leq 0$. Based on this definition, three aspects of preferences can be modelled, as shown below.

(a) **Valence**. Preferences may be *positive* (expressing liking), *negative* (expressing dislike) or *indifferent* (expressing don't care). This is expressed by giving a positive or negative value to $d_T(u)$, $d_F(u)$.

(b) **Concern**. A user's concern is captured by the pair $\langle d_T(u), d_F(u) \rangle$, and $d_T(u)$ captures a user's concern for the *presence* of values u of $R.A$ (or any other path of the schema leading to $R.A$) that make q evaluate to true, while $d_F(u)$ captures a user's concern for the *absence* of the same values, i.e. for q evaluating to false.

(c) **Elasticity**. Preferences may be *exact* or *elastic* depending on whether the domain D_A is categorical or numeric. Constants are attributed to $d_T(u), d_F(u)$ to represent a constant preference for a value, and functions are adopted to represent preference that varies according to the value.

(ii) **Join Preferences.** Join preference indicates a preference for joining two entities, it expresses the dependence of the left part of the join on the right part (using database query vocabulary). A user preference for a join condition q is expressed by the degree of interest in q, $doi(q)$, defined as: $doi(q) = \langle d \rangle$, where $d \in [0, 1]$.

(iii) **Implicit Preferences.** User's preferences over the contents of a database can be expressed on top of a personalisation graph (Figure 4.1). This is a directed graph $G(V, E)$ and it is an extension of the database schema graph. Nodes in V are (a) *relation nodes*, one for each relation in the schema, (b) *attribute nodes*, one for each attribute of each relation in the schema, and (c) *value nodes*, one for each value that is of any interest to a particular user. Likewise, edges in E are (a) *selection edges*, from an attribute node to a value node; such an edge represents the potential selection condition connecting the attribute and the value, and (b) *join edges*, from an attribute node to another

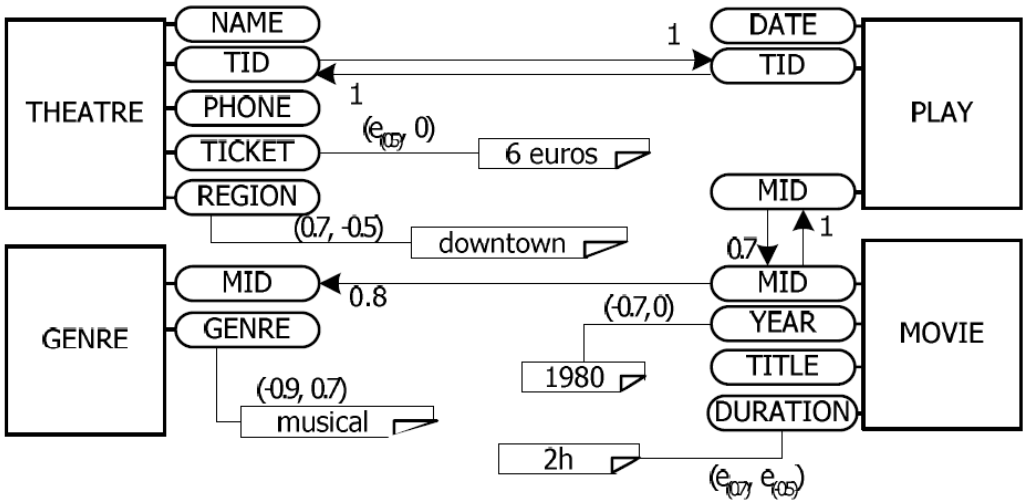


Figure 4.1: Personalisation Graph (Koutrika and Ioannidis 2006).

attribute node; such an edge represents the potential join condition between these attributes.

Implicit preferences are those derived from atomic preferences. An implicit join preference is mapped onto a path in the personalisation graph between two attribute nodes. An implicit selection preference is mapped to a path in the personalisation graph from an attribute node to a value node.

(iv) **Combination Preferences.** *Satisfaction* of an atomic or implicit selection preference $\langle q, d_T, d_F \rangle$ is equivalent to satisfaction of q if $d_T \geq 0$ or failure of q if $d_F \leq 0$. *Failure* of a preference is the exact opposite. Thus, the doi in the satisfaction of a preference is $d^+(u) = max(d_T(u), d_F(u))$. The degree of interest in the failure is $d^-(u) = min(d_T(u), d_F(u))$. The overall degree of interest in a combination of preferences is calculated using a *ranking function*. Three cases are distinguished: (a) all preferences are satisfied (positive combination), (b) none of the preferences is satisfied (negative combination), and (c) some preferences are satisfied and others not (mixed combination).

(v) **Preference Order.** The notion of degree of criticality is introduced for ordering preferences and selecting the top K of preferences. Intuitively, the most important or critical preference is that with the highest d^+, and the lowest d^-.

Even though there are five different kinds of preferences, only the first two (atomic selection and join preferences) are expressed by users; the remaining three are preferences derived from preferences represented as atomic selection and join preferences.

Preference	Definition
AroundPreference	It represents the preference for a value that is around a reference value.
IntervalPreference	It is similar to the AroundPreference, but it is associated with a range or interval of values.
ExtremalPreference	It represents the preference for maximising or minimising the value of a property.
NegativePreference	It indicates a value that is less preferred than others.
PositivePreference	It indicates a value that is more preferred than others.

Table 4.2: Types of AttributeValuePreference.

4.4 Semantic Web Approaches

4.4.1 OWLPref: a Declarative Preference Representation

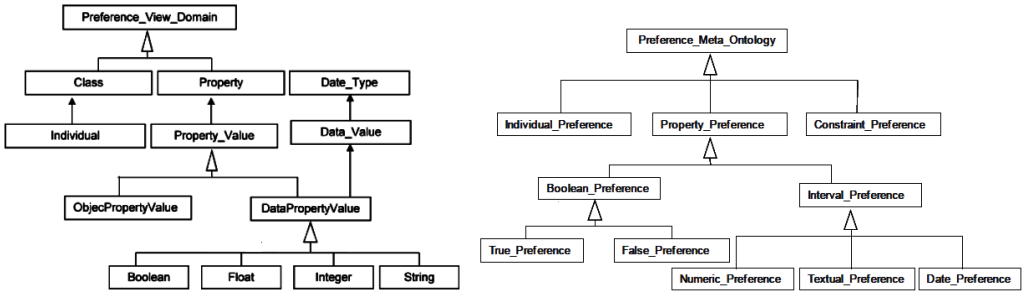
Ayres and Furtado (Ayres and Furtado 2007) proposed a declarative, domain-independent way of representing preferences in OWL, namely OWLPref. It is an ontology that defines different kinds of preferences, which are split into two groups: SimplePreference and CompositePreference. The latter makes compositions of the former.

There are three different types of simple preferences: (i) ClassPreference: it represents a preference than indicates that a certain class is preferred to another; (ii) AttributePreference: it represents a preference than indicates that a certain attribute is preferred to another; and (iii) AttributeValuePreference: it represents preferences for attribute values. The latter has five different subtypes, detailed in Table 4.2.

Composite preferences have three different types as well, and they indicate a relationship between two simple preferences. The first case, ParetoPreference, indicates when these two preferences are equally preferred. In order to indicate when two simple preferences are mutually exclusive, a DisjunctionPreference should be used. Last, an OrderPreference indicates which of two preferences is preferred to another.

Moreover, OWLPref allows the representation of ConditionalPreference. This concept models preferences that vary according to the context. It is composed of two properties: onCondition and hasPreference. The first allows representing the conditions to which the preference is applicable, while the second is a reference to the preference that is conditioned to the stated condition.

Finally, even if it is not explicitly defined, the authors mention that preferences can be organised into hierarchies, so that one can define priority among them.



4.2(a): PVD. 4.2(b): PMO.

Figure 4.2: Metamodels (Tapucu et al. 2008).

4.4.2 Metamodelling Approach to Preference Management

A metamodeling approach to preference management was presented by Tapucu et al. (Tapucu et al. 2008). They propose a preference metamodel, which consists of concepts and semantic relations to represent interests of users. Their motivation is that users may have the same type of preferences in different domains, and therefore metamodeling can be used to define similar preferences for interoperability in different domains. The metamodel structure consists of: (i) **FOAF ontology**, which has the personal data about the user; (ii) **Domain Ontology**, which defines the data about the domain knowledge; (iii) **Preference View Domain Ontology (PVD)** (Figure 4.2(a)), which decomposes the domain ontology and stores ontology resources in a hierarchy based on Ontology Definition Metamodel; and (iv) **Preference Meta Ontology (PMO)** (Figure 4.2(b)), which defines different preference types.

The PMO includes different types of preferences, which are listed below.

- **Boolean preferences** are modelled by means of OWL Boolean Data Type Property Construct. *Example*: Hotel Hilton Paris has sauna.
- **Constraint preferences** are modelled by means of the OWL Object Property construct. Constraint preferences show the related domain class. *Example*: Tennis Court has opening hours from 10:00 am to 22:00 pm.
- **Individual preferences** take value from domain class. *Example*: User prefers hotel Hilton Paris.
- **Interval preferences** have numerical (**Numeric preference**, e.g. User prefers hotel prices between 150-250 Euros), textual (**Textual preference**, e.g. User prefers the beer named "Aloha") and date (**Date preference**, e.g. User prefers to swim on the weekend) values either discrete or continuous.

4.4.3
Situated Preferences for Personalised Database Applications

Holland and Kießling (Holland and Kießling 2004) present a framework for modelling situations and situated preferences. They claim that, since preferences do not always hold in general, personalised applications have to consider the current situation of users. In this context, they propose a general metamodel for situations, which can be applied as foundation for situation models of applications. The metamodel consists of five entities, listed next.

- **Situation** is the most general entity type of situation models. It can contain any attributes describing the situational context of people, agents, applications, etc.
- **Timestamp** denotes the date and time of situations. Attributes can be SQL data types like date, time, time zone, etc.
- **Location** can describe the current position. Attributes are, for example, city, zip-code, or global positioning system (GPS) coordinates.
- **Influences** describe other aspects affecting a situation.
- **Personal Influences** denote human factors of a situation like physical state or current emotion.
- **Surrounding Influences** describe outer influences like weather condition or other people currently accompanying the user.

The metamodel is generic, so it defines only the main entities and their relationships. In addition, it focuses only on modelling situations and the approach is kept **independent from the underlying preference model**. Preferences can be associated with situations, and according these associations, they can be classified in three categories: (i) **Long-term preference**: preferences that hold generally; (ii) **Singular preference**: preferences that hold in exactly one situation; and (iii) **Non-singular preference**: preferences that hold in more than one situation.

4.5
Non-parametric Representation of User Preferences

Domshlak and Joachims (Domshlak and Joachims 2007) present an approach to ordinal utility revelation from a set of qualitative preference statements. According to the authors, their approach is able to handle heterogeneous preference statements both efficiently and effectively, consisting of a computationally tractable, non-parametric transformation into a space where ordinal utility functions decompose linearly.

The main type of preference statements addressed by the approach is **dyadic** statements (*I prefer A to B*). However, it has a particularity in comparison to existing approaches: statements refer to multiple features of alternatives seen in a unified non-parametric way, e.g. "*green sport car*," which is not defined by two parameters (green and sport). Additionally, the approach also covers **monadic** statements, by qualifying alternatives as good and bad. These qualifiers are defined in terms of the "better than" expressed in dyadic statements. Finally, statements as "**A is preferred to B more than C is preferred to D**" are also covered. These two types of preferences are interpreted based on dyadic statements. For monadic statements, one can establish what is indifferent using dyadic statements, and therefore good (bad) is more (less) than indifferent. And for the third type of statement, the difference between A and B must be greater than the difference between C and D. Statements are represented as a qualitative *preference expression*, as shown next.

$$S = \{s_1, ..., s_m\} = \{\langle \varphi_1 \oslash_1 \psi_1 \rangle, ..., \langle \varphi_m \oslash_m \psi_m \rangle\}$$

consisting of a set of preference statements $s_i = \varphi_i \oslash_i \psi_i$, where φ_i, ψ_i are logical formulae over X (set of attributes), $\oslash_i \in \{\succ, \succeq, \sim\}$, and $\succ, \succeq, \sim$ have the standard semantics of strong preference, weak preference, and preferential equivalence, respectively. The authors assume that attributes X are boolean (denoting $Dom(X_i) = \{x_i, \overline{x_i}\}$), and φ_i, ψ_i are propositional logic formulae.

4.6 Comparison of Preference Representation Models

Based on the introduced preference representation approaches, we summarise and compare them in Tables 4.3 and 4.4, showing which kind of preferences and targets each of the approaches addresses. It can be seen that most of the approaches address different kinds of preferences, but they are restricted in allowing stating these preferences only over attributes values. *Exceptions* are OWLPref (Ayres and Furtado 2007) and Tapucu et al.'s approach (Tapucu et al. 2008). In addition, even though Domshlak and Joachims's work (Domshlak and Joachims 2007) also refers to attribute values, it considers it in a *non-parametric way*. Three database approaches have some preferences in their preference model (Kießling 2002, Chomicki 2003, Koutrika and Ioannidis 2006) that are not shown in these tables, because these preference constructions do not correspond to how people express preferences, but to algebraic constructions necessary for the respective approaches.

It can be seen that the presented tables are very sparse, indicating that many existing preference models are very restricted. The approach that is able to represent

Table 4.3: Comparison of Preference Representation Approaches (1).

	Soft-Constraints	**(Junker 2008)**	**CP-nets**	**TCP-nets**	**Scoring function**	**(Chomicki 2003)**
Research Area	Constraint Programming	Constraint Programming	AI	AI	Databases	Databases
Preference Targets						
Concept						
Attribute						
Enumeration Value						
Instance						
Restrictive Preferences						
Optimisation (attribute)		Yes.				
Optimisation (value)						
Constraint / Attribute value	Yes.	Yes.	Yes.	Yes.	Based on record types.	Preference formula.
• Interval					BETWEEN	
• Around						
Preference Statements						
Rating		Wish for a constraint.			Score	
• Bipolar	Yes.					
Qualifying						
Order		Yes.	Yes.	Yes.		Yes.
Don't care					Wild type (*)	
Indifferent						Yes.
Relative Preference[a]						
Unknown					No preference ($\perp$).	
Preference Interaction						
Attribute priority				Yes.		
Attribute indifference						
Preference priority		Yes.			Veto ($\natural$).	Prioritised preference.
Preference indifference		Yes.				
Mutually Exclusive Preference						
Conditionality						
Condition			Yes.	Yes.		
Context						

[a] A is preferred to B more than C is preferred to D.

Table 4.4: Comparison of Preference Representation Approaches (2).

	(Kießling 2002)	**(Koutrika and Ioannidis 2006)**	**OWLPref**	**(Tapucu et al. 2008)**	**Situated Preferences**	**Non-parametric preferences**
Research Area	Databases	Databases	Semantic Web	Semantic Web	Databases	AI
Preference Targets						
Concept			ClassPreference	Class		
Attribute			AttributePreference	Property		
Enumeration Value						
Instance				Individual preferences		
Restrictive Preferences						
Optimisation (attribute)	LOWEST, HIGHEST		ExtremalPreference			
Optimisation (value)						
Constraint / Attribute value	Yes.	Atomic selection preferences.		Constraint preference		Yes.
• Interval			IntervalPreference	Interval preferences (numerical, textual, date)		
• Around	AROUND		AroundPreference			
Preference Statements						
Rating	EXP and SCORE	Atomic selection preferences and Join preferences.				Monadic statements
• Bipolar	POS, NEG, POS/NEG, POS/POS		PositivePreference and NegativePreference	Boolean preferences		
Qualifying						
Order			OrderPreference			Dyadic statements
Don't care						
Indifferent			ParetoPreference			Dyadic statements
Relative Preference[a]						Yes.
Unknown						
Preference Interaction						
Attribute priority						
Attribute indifference						
Preference priority	Prioritised preference and Numerical preference		OrderPreference			
Preference indifference	Pareto preference		ParetoPreference			
Mutually Exclusive Preference			DisjuntionPreference			
Conditionality						
Condition			ConditionalPreference	Yes.		
Context					Abstract metamodel for situations.	

[a] A is preferred to B more than C is preferred to D.

most of the preference types is OWLPref, but it still has limitations and allows only representing interval and around preferences, not having the flexibility to represent constraints. Considering preference types, note that there are two of them that are not represented in any of the approaches: (i) qualifying preferences; (ii) attribute indifference. Qualifying preferences, with the use of expressive speech acts, were widely used in our study of how humans express preferences, and this is an important limitation of existing approaches.

The comparison tables present three preference types that are not part of our proposed metamodel: (i) bipolar; (ii) relative preferences; and (iii) unknown. The first is a categorisation of constraints, but users, when expressing preferences, use expressive speech acts or rates to make this distinction. Moreover, these expressive speech acts and rates indicate how positive or negative preferences are, thus being more expressive than bipolar preferences. The second type of preferences not explicitly represented by our metamodel is a limitation in comparison with existing approaches, but they were not observed in our study. This type of preference indicates a form of resolving trade-offs between options, and our study indicates that people tend to not state this kind of preference (as none of the participants provided it), unless they are specifically asked to do so. Finally, unknown preferences are equivalent to the absence of preferences, i.e. if no preference is given with respect to a target, nothing can be inferred.

4.7 Final Remarks

In this chapter, we reviewed existing preference representation models, discussing the types of preferences they can explicitly represent, and the targets of those preferences. We showed that most of these models are very limited, and even in those that allow the representation of many preference types, there are important preferences that cannot be explicitly represented, such as qualifying preferences. Although there are few preference types that are part of existing models and cannot be represented in our metamodel, they either are not typically adopted by people or are expressed in a different way, according to our study of how humans express preferences. As our goal is to provide means for expressing preferences in a way close to natural language, our metamodel does not consider these preference types.

Representing preferences adequately is an important issue in the context of preference research, as it allows capturing and storing user preferences, and indicates the kinds of preferences that should be taken into account in preference handling approaches. It is extremely important to provide means for reasoning about those preferences, to support user decision making or even automate this process, and this is the issue that will be addressed in the next part of this thesis.

Part II

Preference Reasoning

In the previous part, we presented a metamodel that allows representing preferences provided by users. The vocabulary that this metamodel provides was built based on patterns and expressions extracted from a study of how humans express preferences, so it models high-level preferences adopted by people. While proposing this preference metamodel, we were not concerned with the feasibility or existence of a way of reasoning about the set of preferences that can be modelled with our metamodel. We have focused on developing a metamodel that shows which kind of information should be ideally used as input for algorithms to reason about preferences, in scenarios where users state their preferences, i.e. preferences are not captured implicitly. So, in this part, starting from our preference metamodel, we tackle the problem of reasoning about preferences in order to choose one option of a set of available.

We begin with a literature review of work in the context of reasoning about preferences, presented in Chapter 5. As taking into account individual preferences is relevant for different purposes, such as content personalisation, decision support systems and automation of user tasks, this problem was investigated in different research areas within computer science, such as artificial intelligence and databases. To capture similarities and differences between these different works, we present them in the form of a systematic review, thus enabling their comparison. We identify issues left unaddressed.

Next, we propose in Chapter 6 a technique for reasoning about preferences and making decisions, addressing these identified issues. Preferences that our technique receives as input are expressed in a language that is built based on a restricted version of our preference metamodel. In addition, this technique uses principles of human decision making, in order to deal with trade-off situations. We also compare our technique with existing work and evaluate it with information collected in our study of how humans express preferences.

5
A Systematic Review of Reasoning about Preferences

Preferences are involved with many problems of computer science. People who have complex decisions to be made can be assisted with decision support systems, and such decisions must be made based on preferences of a particular individual. Information provided in the web is nowadays massive, and even with search tools it may be hard to find the information that is needed in a short time as it is common to obtain lots of websites as results of a search. These different scenarios lead to the emergence of research on preference in many research areas, such as artificial intelligence (AI) and databases. As a consequence, there is a large number of approaches proposed. In addition, new approaches are typically compared to existing ones in the same research area; however, there is limited investigation of the relationship of approaches across different areas.

Work on preferences investigates different problems, often associated with one of these topics: preference elicitation (how to obtain preferences from users), preference representation (how to model preferences), and preference reasoning (how to use preferences for making decisions, ordering search results, and other purposes). In this chapter, we present a systematic review[1] of approaches that propose mechanisms to *reason* about preferences, which include algorithms and algebras. The goal of this comparison is to obtain a clear overview of the approaches and find out how existing approaches differ with respect to preference reasoning. As this is a topic investigated in many research areas, our aim is to provide a comprehensive review of these approaches to identify the kinds of issues they are addressing, which structures they use as input, how they work and their limitations. We begin by describing the evaluation framework adopted to analyse each investigated approach in Section 5.1. Approaches are then presented from Section 5.3 to Section 5.7, organised in the following way.

We first describe a background on Multi-Attribute Utility Theory (Section 5.2), which is one of the oldest approaches to help decision makers to make choices, and inspired much of the existing work on reasoning about preferences in different research areas. It established definitions that were adopted in most of the

[1]We use the term *systematic* because this literature review was performed in a systematic way. Nevertheless, the review is not a *systematic review* in the sense of social sciences (Petticrew and Roberts 2006).

presented approaches. The first group of approaches we present consists of those based on *utility functions* (a function that represents preference quantitatively), which propose specific models to represent preferences and forms to derive utility functions from these models, which are supposed to be more natural for users (Section 5.3). Section 5.4 describes approaches that extend Constraint Satisfaction Problems (CSPs) to incorporate *soft constraints*, i.e. constraints that can remain unsatisfied. Utility functions and CSPs are two classical approaches for dealing with preferences and making decisions, and approaches detailed in Section 5.5 take another direction: they propose new *graphical structures* to represent and reason about preferences. Databases are another research area that has been investigating preferences, and works in this area are presented in Section 5.6. These works propose *extensions of query languages* to incorporate preferences and algorithms to provide query results taking the specified preferences into account. Finally, more recently, researchers from argumentation in AI investigated the explicit use of preferences in *argumentation frameworks* to make decisions, which is discussed in Section 5.7. After presenting all these approaches, we further discuss them and show how they are related in Section 5.8, followed by Section 5.9, which concludes this chapter.

5.1 Review Method

Our comparison of approaches to reasoning about preferences follows an evaluation framework composed of seven different criteria, which are described below. Criterion that is not applicable for an approach is omitted in its corresponding section.

Goals. Many of the approaches have the goal of answering a user question that is related to preferences and decision making (see next criterion), so they propose preference representation models and/or algorithms to answer those questions. Other approaches start from an existing (lower-level) preference representation model, which has associated algorithms, and propose means for transforming high-level preference models into low-level ones. Therefore, in these criterion we classify approaches according to the following goals: (i) proposition of *preference representation models*; (ii) proposition of *algorithms*, *algebras* or *semantics* to answer preference-related questions or interpret preference constructions; and (iii) proposition of X *to* Y *transformation*, i.e. transform preferences represented in a model X to preferences represented in a model Y.

Questions. *Which are the preference-related questions addressed by the proposed approach?* As stated before, many of the approaches aim to answer preference-related questions. So, in cases that the approach has this goal, we show which are the questions addressed by each approach. The types of questions are detailed below, from which the first three have been defined by Boutilier et al. (Boutilier et al. 2004).

- *Outcome optimisation* — which is the optimal outcome according to a given preference structure?
- *Dominance queries* — is an outcome o preferred to another outcome o'? If so, it is said that o *dominates* o'.
- *Ordering queries* — is an outcome o' not preferred to another outcome o? If so, it is said that o is *consistently orderable* over o' with respect to the given preference structure.
- *Non-dominated outcomes* — which are the non-dominated outcomes according to a given preference structure?

Input. *What is the input required by the approach?* If the approach relies on a particular structure, specific for the approach, we provide details. Some of these structures were already introduced in Chapter 4, such as CP-nets.

Interpretation. *Which is the interpretation adopted for preference statements?* Most of the approaches use preference statements (or a particular representation of it) as input, but they can be interpreted in different ways: (i) *Ceteris paribus* (Hansson 1996) — meaning "all else being equal", i.e. if one states *"I prefer value x to value x' with regard to attribute X,"* it means that this preference can be considered only when the values of all other attributes are equal; (ii) *Not ceteris paribus* — meaning, when a preference is provided, it is applied to any context. For example, if one states that "lowest" price is a preference for cars, any cheaper car is preferred to any more expensive car.

How it works. An explanation of how each approach works is provided but, as our goal is not to give an extensive description of the involved algorithms and all their steps, we give only a broad understanding of them.

Complexity. *Which is the complexity of the proposed algorithm?* Some approaches do not provide complexity analysis, so this aspect is omitted in their sections.

Limitations. *Which are the limitations of the proposed approach?* Presented approaches may have limitations in different directions, such as having algorithms that takes exponential time to be executed or preferences expressed in a way that are hard to elicit. In Chapter 4, which focused on preference representation, we detailed the kinds of preferences addressed by most of the approaches described here and their particular representation models. Therefore, the lack of expressivity of the approaches will not be reported in this chapter, as it was discussed previously.

5.2 Background on Multi-Attribute Utility Theory

Multi-Attribute Utility Theory (MAUT) is a set of methods designed to handle decision problems involving multiple objectives and trade-offs. According to Dyer (Dyer 2005), MAUT has become synonymous in the view of many scholars with the theory proposed by Keeney and Raiffa (Keeney and Raiffa 1976), which emphasised the use of multi-attribute preference models based on the theories of von Neumann and Morgenstern (von Neumann and Morgenstern 1944), who presented a set of axioms about preferences and utilities such that any decision maker satisfying these axioms has a utility function.

MAUT is concerned with the valuation of the *outcomes* of an *option* of a decision maker. A problem is described in terms of *attributes* $X = \{x_1, ..., x_n\}$, and each of them can have a value assigned according to their respective domains $D_1, ..., D_n$, which establish the range of possible values of an attribute. The set of *all possible outcomes* is the cartesian product of attribute domains $D_1 \times ... \times D_n$. *Feasible outcomes* are a subset of all possible outcomes, and are those that consist of valid combinations of attribute values. A (cardinal) utility function is a function that maps outcomes to a real value that represents the preference for each outcome. The higher is the value, the more preferred the outcome is. In Table 5.1, we give the definition of the three main terms (option, outcome and attribute) adopted in MAUT and decision making, which were already introduced above, together with alternative terminology, as different approaches may use different terms. In some cases, options lead to only one possible outcome, and in such cases the term option and outcome are used as synonyms, for example choosing a product A to be bought (option) leads to the single consequence of buying this product (outcome), therefore A can be used to refer to both the option and the outcome.

Keeney and Raiffa (Keeney and Raiffa 1976) discuss decisions that can be made under *certainty* or *uncertainty*. The first refers to situations in which the outcome, represented by multiple attributes, of an option is known. In these scenarios, the main issue is to resolve trade-offs among preferences, which typically conflict because in general they cannot be maximised at the same time. For instance,

Term	Definition	Alternative Terminology
Options	Elements of the same conceptual class, which is the subject of the decision making process	Actions, Alternatives, Record, Tuples
Outcomes	Results obtained by choosing an option	Payoffs, Consequences
Attribute	Characteristics used to describe options	Variables, Features

Table 5.1: Main terms adopted in decision making.

two preferences might be maximise quality and minimise costs, but lower costs are normally achieved by compromising quality. In the second, the cases in which uncertainty is present, or in other words *risky* situations, there is a probability associated with the possible outcomes of an option. In their work, Keeney and Raiffa consider that this probability is given.

A preference representation function under certainty is referred to as a *value function* v. v associates a real number with each point x in the outcome space, representing the preference structure of the decision maker, provided that

$$\forall\, x', x'' \in D\ x' \sim x'' \Leftrightarrow v(x') = v(x'') \text{ and } x' \succ x'' \Leftrightarrow v(x') > v(x'')$$

$\sim$ and $\succ$ represent *indifference* and *preference*, respectively. Based on v, the goal is to find an outcome $x \in D$ to maximise $v(x)$.

When the decision making is based on assumption of the existence of value functions, two concepts (*preference independence* and *mutual preference independence*) play an important role, as they are related to properties of the value function, which can facilitate its elicitation process. The definition of these two concepts are given below.

Preference Independence. An attribute set Y, where $Y \subset X$, is *preferentially independent* of its complement $\overline{Y}$ if the preference order of two outcomes involving different values assigned to Y does not depend on the fixed values assigned to attributes in $\overline{Y}$. For example, if the laptop colour is preferentially independent of the remaining laptop attributes, and one states that prefers *silver* to *black*, by fixing values of the remaining attributes, any silver laptop is preferred to any black laptop.

Mutual Preference Independence. The attributes $x_1, ..., x_n$ are mutually preferentially independent if every subset Y of X is preferentially independent of its complementary set.

These definitions are used to identify a very important property of value functions: if the mutual preference independence is held, then it can be proved

that the form of the value function is *additive*, and this form of representation is one of the most common approaches for evaluating multiattribute alternatives. In this representation, attribute values are considered independent, and therefore the value of an outcome can be calculated by adding values of individual attributes. This approach is ideal because it is *compact*, as a specific value does not need to be specified for each possible outcome, whose number is exponential on the size of attribute domains. In this thesis, we adopt the term compact as an adjective to approaches that with little information (preference statements, utility values) one can derive a preference order between two outcomes.

When uncertainty is considered, the preference representation function is referred to as *utility function*, which takes into account the probability of outcomes. We now introduce definitions related to decisions under this kind of scenario — many of these definitions come from the formalisation of *expected utility theory* by von Neumann and Morgenstern (von Neumann and Morgenstern 1944).

- A *lottery* is any option with an uncertain outcome. *Examples*: investment, roulette, football game.
- A *probability* of an outcome (of a lottery) is the likelihood that this outcome occurs. *Example*: the probability often is estimated by the historical frequency of the outcome.
- The *probability distribution* of the lottery depicts all possible outcomes in the lottery and their associated probabilities. The probability of any particular outcome is between 0 and 1 and the sum of the probabilities of all possible outcomes is equal to 1.

The definitions of (mutual) preference independence are then extended by considering *uncertainty*: an attribute x_i is said to be **utility independent** of its complementary attributes if preferences over *lotteries* with different values of x_i do not depend on the fixed values of the remaining attributes. Attributes $x_1, x_2, ..., x_n$ are **mutually utility independent** if all proper subsets of these attributes are utility independent of their complementary subsets.

We discuss next goals and limitations of MAUT, which can be related to many of the approaches that will be presented, which rely on utility or value functions.

Goals. A preference representation model in the form of utility functions. The approach presented in MAUT does not focus directly on presenting algorithms to answer questions related to preference reasoning, because once utility functions are correctly defined for a decision maker, typical preference tasks and associated questions, such as comparing and ranking outcomes, can be easily performed and

answered, as it requires only a numerical comparison. The challenge is to reveal the utility function of decision makers.

Limitations. The main issue related to MAUT and other utility function-based approaches is the elicitation process. The process of obtaining the information required to generate a good utility function requires considerable effort on the part of the user. Users may be able to express preferences in other kinds of preference statements, but revealing the utility function underlying such preferences is a challenging task. Moreover, existing techniques for revealing utility functions typically rely on presenting users with several pairs of options and requesting them to compare these pairs, which is a time-consuming task that users may not be willing to perform, unless the consequences of making a wrong choice are very important for the decision maker.

5.3 Utility Function-based Approaches

In this section, we present approaches that use utility functions[2] to reason about preferences and, as discussed above, the main challenge is to identify which is the utility function of the decision maker. The first presented approach proposes a graphical structure to represent utility functions and algorithms, and the two remaining approaches propose methods to transform qualitative preference statements into utility functions. As our focus is not the elicitation process, we do not describe approaches that propose revealing utility functions by means of user interaction.

5.3.1 CUI networks

CUI networks are claimed by Engel and Wellman (Engel and Wellman 2008) as a compact graphical representation of utility functions over multiple attributes. These networks model multiattribute utility functions using the concept of conditional utility independence — CUI stands for (Conditional) Utility Independence — which requires a (cardinal) preference order over a subset of the attributes to be independent of another subset of attributes.

Goals. A preference representation model (CUI networks) and algorithms.

Questions. Outcome optimisation.

[2]Although these approaches claim to adopt utility functions, they rely on *value* functions according to the definition of Keeney and Raiffa (Keeney and Raiffa 1976).

Input. The required input for answering outcome optimisation queries based on CUI networks is a set of CUI conditions. These conditions indicate when a set of attributes Y is utility independent of a set of attributes X, meaning that the utility of an assignment for Y does not depend on X. In addition, the utility of the assignment for Y may depend on values assigned to a set of attributes Z (conditionality). Considering these sets of attributes, the utility function can be decomposed into

$$U(X, Y, Z) = f(X, Z) + g(X, Z)U(X', Y, Z), g(\cdot) > 0$$

where X, Y, Z are sets of attributes, X' is an assignment for X. The function sums the utility of attributes in X (conditioned to Z), with the utility of attributes in Y (conditioned to Z) — considering a weight $g(\cdot) > 0$ of the utility of Y. In addition, since $U(X', Y, Z)$ is a function only of Y and Z (its value does not change for different assignments X'), it can also be written as $U(Y, Z)$.

The set σ of CUI conditions on the attribute set $S = \{x_1, ..., x_n\}$, such that for each $x \in S$, σ contains a condition of the form $CUI(S\backslash(x \cup P(x)), x \mid P(x))$. The CUI condition indicates a set of attributes $P(x)$, which separates the rest of the attributes from x. If we apply this condition to x_1, for example, we have $U(S) = f_1(x_1, P(x_1)) + g_1(x_1, P(x_1))\, U_{x_1^0}(S\backslash\{x_1\})$.

Interpretation. Not ceteris paribus.

How it works. The proposed approach consists of initially transforming a given set of CUI conditions into a graphical model. For accomplishing this, a procedure is described by Engel and Wellman (not detailed here), which builds a graph whose nodes are attributes based on this given set of conditions and an order on the set S, namely the CUI network. The goal of providing a graphical form for CUI conditions is that, according to the authors, CUI networks provide a potentially compact representation of the multi-attribute utility function, via functional decomposition to lower-dimensional functions that depend on a node and its parents.

Based on CUI networks, two optimisation algorithms for discrete domains were developed. This first is applied only to CUI trees, which are a CUI network in which no node has more than one child (upside down version of a standard directed tree). The main idea of the algorithm is to select an optimal value function (OVF) for a node based on parents and child. However, as the algorithm runs from roots to leaf, the OVF of the child is calculated after visiting the parent, so when the value for the child is set, the parent values may be corrected, and this is propagated to the roots of the tree. The second algorithm applies to general directed acyclic graphs (DAGs). In the tree case, correcting the value of the child of a node x is

sufficient in order to separate x from the rest of the graph, excluding ancestors. Each value of the child is considered at a time, so it also determines the values for all the ancestors. In a general DAG it is no longer sufficient for the OVF to depend on the children, because they do not provide sufficient information to determine the values of the ancestors of x. So this notion is generalised to be the scope of x ($Sc(x)$), which is a set of nodes on which the OVF of x must depend, in order for an iterative computation of the OVF to be sound. With this generalisation, the DAG algorithm is similar to the tree algorithm. Further details can be seen elsewhere (Engel and Wellman 2008).

Complexity. For CUI networks structured as a tree, in case the numeric data at the nodes is available, factoring in the time it takes to recover the utility value for each outcome (which is $O(n)$), the algorithm runs in time $O(n^2 max_i \mid D(x_i) \mid^2)$, where n is the number of attributes and $D(x_i)$ is the domain of attribute x_i. For CUI networks structured as a DAG, the performance of the optimisation algorithm is exponential in the size of the largest scope $Sc(x)$ (plus one).

Limitations. This approach is restricted to dealing only with conditional utility independent functions, which, on the one hand, are a weaker independence condition in comparison to additive independence and, on the other hand, are still a limitation, because it assumes a particular kind of preference independence. In addition, as other quantitative approaches, it requires capturing numeric utility values from users, which is not a trivial task.

5.3.2 Utility functions for Ceteris Paribus Preferences

McGeachie and Doyle (McGeachie and Doyle 2008) present a set of methods for translating preference information from a qualitative representation to a quantitative representation. They consider ceteris paribus preferences, represented with a propositional language augmented with an ordering relation used to express preferences over propositional combinations of a set of elementary attributes. This qualitative representation is converted to an ordinal utility function in order to be used in reasoning processes.

Goals. A transformation from *qualitative statements* to *utility functions*.

Input. The input required by the method is preference statements interpreted under the ceteris paribus semantics and represented in a formal logic. A restricted logical language $\mathcal{L}$ is adopted, using only two logical operators: $\neg$ (*negation*) and $\wedge$

(*conjunction*) to construct finite sentences over a set of atoms, each corresponding to an attribute from a space of binary attributes describing possible worlds. A complete consistent set of literals is a *model*.

A preference order is a complete preorder (reflexive and transitive relation) $\succsim$ over the set of all models of $\mathcal{L}$. When, given two models m and m', $m \succsim m'$, it is said that m is weakly preferred to m'. If $m \succsim m'$ and $m' \not\succsim m$, it is said that m is strictly preferred to m', written $m \succ m'$. If $m \succsim m'$ and $m' \succsim m$, then it is said m is indifferent to m', written $m \sim m'$.

Interpretation. Ceteris paribus.

How it works. The general idea of the approach is to generate an ordinal UF from a set C of ceteris paribus preferences over attributes F. An initial step is performed, which translates the provided statements represented in a logical representation of ceteris paribus preferences to an attribute-vector representation. The latter is then used to produce the UF in the following way. The structure of preference statements in the attribute-vector representation is used to infer additive utility independence among attributes. Next, subutility functions are defined for each utility independent set of attributes. This is done based on the representation of preorders consistent with the preferences by building a graph over assignments to the attributes. Finally, to assign relative weights of satisfaction of different attributes, a linear programming problem is solved. In the end, a utility function that can be used to evaluate the utility of different assignments to values of F is built.

Complexity. This work has two different complexity questions, which are associated with: (i) time and space required to construct the utility function u; (ii) the time and space required to evaluate $u(m)$ on a particular model m. Both of them take exponential time, in worst cases. Constructing u involves solving a satisfiability problem, where time required depends on the number of rules involved in conflicts on each utility independent set. On the other hand, computing $u(m)$ involves computing $u_i(m)$, where u_i is a minimising utility function, for all the utility independent attribute sets. Each of the subutility functions can be exponential in the size of the utility independent set.

Limitations. The main limitations of the approach presented by McGeachie and Doyle (McGeachie and Doyle 2008) are that it is able to deal only with boolean attributes, and the intractability of the solution.

5.3.3 Learning Utility Functions with SVM

Domshlak and Joachims (Domshlak and Joachims 2007) proposed an approach that takes a new direction for reasoning about preferences. First, it adopts a new form of representing preferences: instead of seeing an option as values (parameters) specified for a set of attributes, it linearises the possible combinations of attribute values and each of these combinations is seen in a *unified non-parametric way*, i.e. as a unique specification. For instance, "red big suitcase" is not an option (suitcase) parameterised with values assigned to colour and size, but as a single option. Second, machine learning techniques are adopted to generate a utility function to model user preferences.

Goals. A transformation from *qualitative statements* to *utility functions*.

Questions. Experiments used utility functions constructed with the approach to generate an *ordering* and rank of k best options for users.

Input. The presented approach receives as input qualitative preference statements, which can be of three different types: (i) **dyadic** statements; (ii) **monadic** statements; and (iii) **A is preferred to B more than C is preferred to D**. These are represented with logic-based qualitative *preference expressions*, as detailed in Section 4.5.

Interpretation. Not ceteris paribus. Each parameter u_i of the utility function can be seen as representing the *marginal utility* of the interaction between the attributes associated with u_i when these take specific values. This means that each individual statement does not state a particular interpretation, such as preference independence, but gives a contribution for (in)dependency among attributes.

How it works. The two main aspects from this work are the proposed underlying preference representation, i.e. how the the qualitative preference statements are represented, and how the utility function is generated.

The situation explored by Domshlak and Joachims is when the system cannot assume a significant independence structure on outcome attributes. So, the basic idea is that if no useful preferential independence information in the original representation space is provided, a different space is adopted in which no such independence information is required. This new space is the following: assuming that the attributes X are all binary-valued, there is a map from the options χ into a new, higher dimensional space $\mathcal{F}$ using a certain mapping $\Phi = \chi \mapsto \mathcal{F} = \mathbb{R}^{4n}$.

The mapping Φ is not arbitrary, and it establishes a connection between the dimensions χ and $\mathcal{F}$, in which each element of $\mathcal{F}$ is a combination of values of attributes. This mapping is used for representing preference statements. Each element of $\mathcal{F}$ is associated with weights, and these must be in accordance with provided statements. Statements are used as constraints over weights together with an objective function in an optimisation problem. For calculating weights, techniques frequently used in machine learning in the context of Support Vector Machiness (SVMs) (Vapnik 1998) are adopted.

Complexity. Solving the optimisation problem of this approach poses several complexity issues. First, though this constraint system is linear, it is linear in the exponential space $\mathbb{R}^{4n}$. Second, the very description size of the optimisation problem, and, in fact, of each individual constraint of it, can be exponential in n. Nevertheless, Domshlak and Joachims show that these complexity issues can be overcome by using some duality techniques from optimisation theory and Reproducing Kernel Hilbert Spaces (RKHS).

Limitations. This work is highly associated with machine learning. It interprets each pairwise comparison as likes and dislikes of options with similar values for attributes and builds a model with values that generalises these comparisons, which can also be a comparison with the whole set of options, when monadic statements are provided. Therefore, there is a statistical generalisation, and for achieving good results with this approach a large amount of statements may be needed. This is reflected in the future work reported by the authors, whose goal is to specify the number of preference statements that a user has to provide for inferring a utility function that is effective.

5.4 Constraint Programming

Constraint Satisfaction Problems (CSPs) are mathematical problems defined as a set of objects (options) whose state must satisfy a number of constraints or limitations. Constraints can be seen as preferences that should ideally be satisfied, but when no solution is found for an over constrained problem, these (soft) constraints can be relaxed. Typically, each constraint is associated with a penalty for their not satisfaction (or a degree of preference for their satisfaction), and there is an objective of minimising penalty (or maximising preference satisfaction). In this section, we present approaches that use CSPs for dealing with preferences.

5.4.1 Semiring-based Constraint Satisfaction

Different approaches were proposed to deal with soft constraints, i.e. when constraints are used to formalise desired (preferred) properties rather than requirements that cannot be violated. Examples of these approaches are *fuzzy* and *weighted* constraints, which associate a value with constraints indicating the *preference* for a constraint satisfaction and the *cost* of not satisfying a constraint, respectively. Bistarelli et al. (Bistarelli et al. 1997) defined a constraint solving framework where all such extensions, as well as classical CSPs, can be cast. The main idea is based on the observation that a *semiring* (i.e. a domain plus two operations satisfying certain properties) is all that is needed to describe many constraint satisfaction schemes.

Goals. A preference representation model in the form of an abstract CSP.

Questions. The main question addressed by soft constraints is to find a solution for over constrained problems, so we can say that the typical question answered is *outcome optimisation*, in which the optimal outcome is that satisfying the most important constraints. In this approach, each constraint of a CSP is associated with a preference value (or a penalty) and the goal is to find a solution that maximises the overall preference value (or minimises the penalty of not satisfied constraints).

Input. The required input for soft constraint-based approaches is a Soft Constraint Satisfaction Problem (SCSP), which was introduced in Section 4.1.1, which consists of a constraint system and a set of constraints, which are associated with a *value* representing the penalty for unsatisfied constraints or the preference for their satisfaction.

Interpretation. Not ceteris paribus.

How it works. Semiring-based constraints rely on a simple algebraic structure, called a c-semiring since it is very similar to a semiring, to formalize the notion of satisfaction degrees, or preference levels. Recall that, in this term, “c” stands for “constraint,” meaning that this kind of semiring is a natural structure to be used when handling constraints.

The structure is specified by a set E of satisfaction degrees, where two binary operators are defined: $\times_s$ specifies how to *combine* preferences, while $+_s$ is used to induce a *partial ordering* on E. Additional axioms, including the usual semiring

Table 5.2: Different specific frameworks modelled as c-semirings (Meseguer et al. 2006).

Semiring	**E**	$\times_s$	$+_s$	$\succcurlyeq_s$	0	1
Classical	t, f	$\wedge$	$\vee$	$t \succcurlyeq_s f$	f	t
Fuzzy	[0, 1]	min	max	$\geq$	0	1
k-weighted	$0, ..., k$	$+^k$	min	$\leq$	k	0
Probabilistic	[0, 1]	xy	max	$\geq$	1	0
Valued	E	$\otimes$	min_v	$\preccurlyeq_v$	$\top$	$\bot$

axioms, are added to precisely capture the notion of satisfaction degrees in soft constraints.

A c-semiring is a 5-tuple $< E, +_s, \times_s, 0, 1 >$ such that:

- E is a set, $0 \in E$, $1 \in E$;
- $+_s$ is an operator closed in E, associative, commutative and idempotent for which 0 is a neutral element and 1 an annihilator;
- $\times_s$ is an operator closed in E, associative and commutative for which 0 is an annihilator and 1 a neutral element; and
- $\times_s$ distributes over $+_s$.

Compared to a classical semiring structure, the additional properties required by a c-semiring are the idempotency of $+_s$ (to capture a lattice ordering) and the existence of a minimum and a maximum element (to capture hard constraints).

In the SCSP framework, the values specified for the tuples of each constraint are used to compute corresponding values for the tuples of values of the attributes in con (set part of the constraint problem), according to the semiring operations: the multiplicative operation is used to combine the values of the tuples of each constraint to get the value of a tuple for all the attributes, and the additive operation is used to obtain the value of the tuples of the attributes in the type of the problem. More precisely, this is the definition of the operations of *combination* and *projection* over constraints.

The c-semiring is a generic framework for representing soft constraint problem solvers, and by capturing their commonalities in a generic framework one can design generic algorithms and properties instead of several apparently unrelated, but actually similar properties, theorems and algorithms. Different specific frameworks were presented as soft constraint networks (Meseguer et al. 2006, Bistarelli et al. 1997), which are summarised in Table 5.2.

Complexity. Since semiring-based constraints properly generalise classical constraints, this task is NP-hard. As in the classical case, perhaps the most direct

way to solve a soft constraint network is searching in its state space, exploring the set of all possible assignments. Since an optimal solution is an assignment that minimises the violation degree (or equivalently, maximises the satisfaction degree), solving optimally a soft constraint network is an optimisation problem, thus harder than solving classical constraint networks. Different techniques were proposed to optimise optimal solutions, such as defining lower and upper bounds of branch and bound algorithms.

Limitations. As other quantitative approaches, soft constraint-based approaches have the limitation of extracting from users numerical values for preferences, what is not intuitive for them. In addition, solving SCSPs is a NP-hard task. Finally, these approaches do not deal with multiple objectives, the only objective is to maximise the constraints that are satisfied (considering their weight), but additional objectives such as minimise price cannot be represented. Only intervals for acceptable prices can be provided as constraints.

Extensions

There are extensions of the c-semiring framework, whose goal is to represent other kinds of preferences. Next, we briefly describe two of these extensions.

Representing Bipolar Preferences. Bistarelli et al. (Bistarelli et al. 2010) state that preferences on a set of possible choices are often expressed in two forms: negative and positive statements. The former restricts the set of acceptable options, indicating what users do not want, and the latter indicates options that users prefer with respect to other acceptable options. These two kind of preferences are referred to as *bipolar* preferences. The approach described for bipolar preferences proposes a tool to represent these two types of preferences in a single framework and provides algorithms that, given as input a problem with these preferences, return its best solutions.

Representing Intervals. Gelain et al. (Gelain et al. 2010) argue that it is difficult to specify precise values associated with constraints to represent preferences, and it is more reasonable to consider intervals instead of specific values. The authors then define several definitions for optimal solutions in such problems, providing algorithms to find optimal solutions and also to test whether a solution is optimal.

5.4.2 Preference-based Problem Solving for Constraint Programming

Junker (Junker 2008) points out that traditional optimisation approaches compile preferences into a single utility function and use it as the optimisation objective when solving the problem, but they do not explain why the resulting solution satisfies the original preferences, and do not indicate the trade-offs made during problem solving. The author then argues that the whole problem solving process becomes more transparent and controllable by the user if it is based on the original preferences.

He tackled the problem of multi-objective optimisation by decomposing it into alternative sequences of single-criterion optimisation problems, which can be solved by standard optimisers. The chosen sequence gives information that explains the optimality of the solution. Based on the explanation, the user can either accept the solution or modify the preferences. The problem solver then modifies the solution correspondingly. Preferences thus allow the user to interact with the problem solver and to control its behaviour.

Goals. A preference representation model (introduced in Section 4.1.2), and an algorithm based on constraint problem solvers to find optimal outcomes.

Questions. Outcome optimisation. In addition, based on an optimal solution, users may ask two questions.

1. Why can't the criteria z have a value better than ω^* (the optimal solution)?
2. Why hasn't the value ω been chosen for the criteria z?

Input. A set of preferences are modelled in the form of a preorder $\succsim$ on criteria and their respective domains, consisting of a strict part $\succ$ and an indifference relation $\sim$ (see Section 4.1.2 for details), and a set of available options, which are described in terms of values for attributes that have specific domains, and constraints over possible values.

Interpretation. Not ceteris paribus.

How it works. As opposed to combinatorial problems with preferences, which are classically solved by compiling all preferences into a single utility function and by determining an option satisfying the constraints that has maximal or nearly maximal utility, Junker uses individual preferences with constraint solvers, in order

to allow the optimiser to give an explanation of optimality in terms of the original preferences.

First, an atomic optimisation step is performed, which finds options satisfying the constraints C that assign a $\succ$-maximal value to a single criterion z. This is possible if the user provides a total order over the domain of z, but if it is not the case, all possible total orders are generated based on the provided partial order, and the constraint solver runs for each possible total order. Maximal options in this case are the union of maximal options of each of them.

In order to address multiple criteria optimisation, importance order is introduced on the preferences and to decide trade-offs in favour of the more important preferences. Lexicographic optimisation is then used to define an ordering on the option space based on this importance principle, which leads to finding a *lexicographically optimal option*. As the importance among the criteria is unknown, permutations of all possible orders are done. If users are unsatisfied with the results, they can establish the importance among the criteria, which is a strict partial order $I \subseteq Z \times Z$. Thus, only permutations that respect this strict partial order will be considered.

Finally, pareto-optimality is adopted to capture all the possible trade-offs, because using a lexicographic approach gets a best value for a more important criterion z_1, and a less important criterion z_2 is completely penalised. In this case, pareto-optimal options are found, which are those that pareto-dominates other options, i.e. optional options are better in at least one criterion, and equally good in the other ones. As there is no direct way to transform a pareto-optimization problem into a solved form of the optimisation problem even if it is based on totally ordered preferences, the author introduces the notion of wishes, which establish penalisation limits for criteria.

Complexity. CSPs are known to be NP-hard, but this solution is based on the use of standard optimisers, such as constraint-based branch-and-bound, which improve performance.

Limitations. There are some steps in the work described by Junker that are not completely detailed, such as the mapping of preferences $\langle z, > \rangle$ to a utility function u. In addition, one of the most challenging problems in preference reasoning is dealing with trade-off. When a solution is optimal for all criteria, it is trivially optimal. However, requiring users to establish penalisation limits for resolving trade-offs has two main drawbacks: (i) for domains that users are not very familiar with, they might have a vague idea of which is a good limit to establish the trade-off; and (ii) this solution may lead to a situation in which the more important criterion is

maximised to a value that sets the maximum penalty to the less important criterion; however users may prefer something in between.

5.5 Graphically-structured Approaches

As seen in previous sections, the use of utility functions and CSPs has drawbacks associated with preference representation as they require not trivial preference elicitation processes. In this section, we present approaches that aim at addressing them by proposing new forms of representing preferences and relationships among attributes. These representations, which rely on graph structures, have the goal of capturing preferences in a intuitive way and also to be used in algorithms to reason about preferences.

5.5.1 CP-nets

CP-nets represent preferences in a compact graphical form and, according to Boutilier et al. (Boutilier et al. 2004), represent preference statements in a natural way. These statements are conditional qualitative preference statements interpreted under the ceteris paribus semantics.

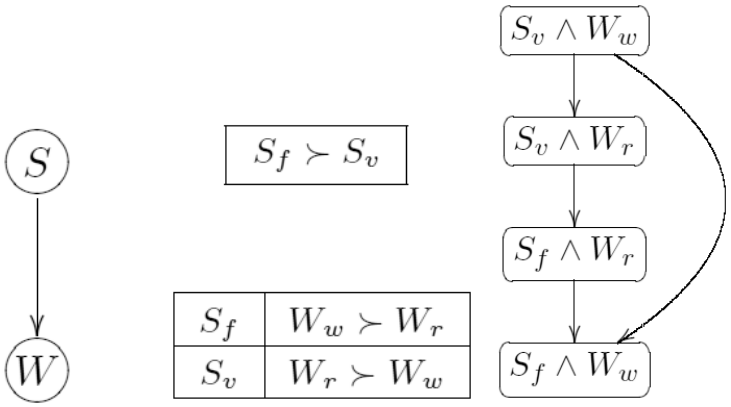
Goals. A preference representation model (CP-nets) and algorithms.

Questions. Outcome optimisation, dominance queries and ordering queries.

Input. A CP-net (introduced in Section 4.2.1) over attributes $V = X_1, ..., X_2$ is a directed graph G over $X_1, ..., X_2$ whose nodes are annotated with conditional preference tables $CPT(X_i)$ for each $X_i \in V$. While the required input for outcome optimisation requires only a CP-net as input, and optionally a partial assignment for attributes, the input for both dominance and ordering queries is a CP-net N and two outcomes o and o'.

Interpretation. Ceteris paribus.

How it works. In order to provide a better understanding of CP-nets, we first introduce a simple example (Boutilier et al. 2004) that expresses preferences over dinner configurations. This network, depicted in Figure 5.1(a), consists of two attributes S and W, standing for soup and wine, respectively. Fish soup (S_f) is strictly preferred to vegetable soup (S_v), while preference between red (W_r) and



5.1(a): Preferences for soup and wine. 5.1(b): Induced graph.

Figure 5.1: Example of a CP-net (Boutilier et al. 2004).

white (W_w) wine is conditioned on the soup to be served: red wine is preferred if served with a vegetable soup, and white wine if served with a fish soup.

The semantics of a CP-net is defined in terms of the set of preference rankings that are consistent with the set of preference constraints imposed by its conditional preference tables (CPTs). Figure 5.1(b) shows the preference graph over outcomes induced by the previously described CP-net. An arc in this graph directed from outcome o_i to o_j indicates that a preference for o_j over o_i can be determined directly from one of the CPTs in the CP-net.

Next we briefly describe how each of the preference-related questions are answered using CP-nets.

Outcome Optimisation. In order to generate the optional outcome of a CP-net, the network is sweep from top to bottom. The *forward sweep procedure* (Boutilier et al. 2004) is provided, which constructs the most preferred outcome. It considers, in order to be more general, a given evidence constraining possible outcomes in the form of an instantiation z of some subset $Z \subseteq V$ of the network attributes. The algorithm sets $Z = z$, and instantiate each $X_i \notin Z$ in turn to its maximal value given the instantiation of its parents.

Ordering queries. For acyclic CP-nets, Boutilier et al. define a corollary that provides an algorithm for answering ordering queries. The corollary states that given an acyclic CP-net N and a pair of outcomes o and o', if there exists an attribute X in N, such that: (i) o and o' assign the same values to all ancestors of X in N; and (ii) given the assignment provided by o (and o') to the parents of X, o assigns a more preferred value to X than that assigned by o', then $N \nvDash o' \succ o$.

Dominance queries. Boutilier et al. demonstrate the equivalence of answering dominance queries with the task of determining the existence of an improving (or worsening) sequence of attribute value flips with respect to the given CP-net. A sequence of improving flips from one outcome to another provides a proof that one outcome is preferred, or dispreferred, to another in all rankings satisfying the network. The authors also showed that this task can be reduced to a special subclass of classical planning problems, and presented several techniques that can be used in a generic search procedure for an improving flipping sequence.

Complexity. CP-nets were investigated to be used with different algorithms related to reasoning about preferences, and these had their complexity analysed with different graph topologies. Outcome optimisation queries can be answered using the forward sweep procedure, taking time linear in the number of attributes. The complexity of two of the comparison queries (dominance and ordering) has been analysed for different graph topologies of CP-nets, which can be seen in Table 5.3. These results are associated with boolean attributes, but ordering results also hold for multi-valued attributes.

Table 5.3: Complexity Analysis of CP-nets (boolean attributes).

Graph topology	Dominance	Ordering
Directed Tree	$O(n^2)$	$O(n)$
Polytree ($indegree \leq k$)	$O(2^{2k}n^{2k+3})$	$O(n)$
Polytree	NP-Complete	$O(n)$
Singly Connected ($indegree \leq k$)	NP-Complete	$O(n)$
DAG	NP-Complete	$O(n)$
General Case	PSPACE-Complete	NP-Hard

Given a CP-net N over n attributes and a set of complete assignments $o_1, ..., o_m$, ordering these assignments consistently with N can be done using ordering queries only, in time $O(nm_2)$.

Limitations. This work leaves different open theoretical questions, mainly related to the complexity analysis of other scenarios in which CP-nets are used to reason about preferences. In addition, CP-nets address representing preferences over discrete domains, and common preferences, such as price minimisation, cannot be directly represented. Moreover, CP-nets, when identifying optimal outcomes, first optimise parent attribute values for later assigning values for their children. This leads to a lexicographic importance among attributes, which may capture trade-off situations in a not appropriate way. This issue is investigated in TCP-nets, presented in Section 5.5.4, which add importance relations and conditional relative importance statements to the the conditional ceteris paribus statements supported by CP-nets.

5.5.2 Combining CP-nets and Soft Constraints

Domshlak et al. (Domshlak et al. 2003) establish a connection between the CP-nets and soft constraints machinery. As dominance queries are hard to be answered using a CP-net structure (Section 5.5.1), the authors propose constructing a semiring structure with a given acyclic CP-net, in order to be able to answer this kind of queries.

Goals. A transformation from *CP-nets* to *SCSPs*.

Questions. Domshlak et al. focus on answering dominance queries *in polynomial time*. This is achieved by using a semiring structure, so the main question is how to transform a CP-net into this structure, so that dominance queries can be answered in polynomial time.

Input. An acyclic CP-net.

Interpretation. Ceteris paribus.

How it works. This work consists of approximating CP-nets via soft constraints, i.e. defining a SCSP based on a CP-net. This provides a uniform framework to combine user preferences with both hard and soft constraints. So, given an acyclic CP-net, a corresponding SCSP is constructed in two steps. First, a constraint graph, named SC-net, is built. Second, preferences and weights for the constraints in the SC-net are computed, and this computation depends on the actual semiring framework being used. Basically, the SC-net constructed consists of one node for each node of the CP-net plus new nodes for nodes from the CP-net that have more than one parent. Arcs correspond to hard and soft constraints, and the latter is associated with weights and penalties.

For calculating these values, two alternative semiring frameworks are presented, based on (i) min+ and (ii) Soft-constraint Lexicographic Ordering (SLO) semirings. In both cases, the computation of preferences and weights ensures information preserving and satisfies the cp-condition, i.e. approximations preserve the ceteris paribus property.

Complexity. Given an acyclic CP-net N with the node in-degree bounded by a constant, the construction of the corresponding SC-net based on weighted SCSP

N_c is polynomial in size of N. No complexity analysis was presented for SLO soft constraints.

Limitations. This approach aims at solving an issue of CP-nets — tractability of dominance testing — by approximating a CP-net ordering via soft constraints. Even though this goal is achieved, precision is compromised to some degree. Different approximations (such as min+ and SLO) can be characterised by how much of the original ordering they preserve, the time complexity of generating the approximation, and the time complexity of comparing outcomes in the approximation. Incomparable outcomes with min+ are considered equal or ordered in either ways, and with SLO, an strict order will always be defined.

5.5.3 UCP-networks

A directed network representation for utility functions that combines certain aspects of quantitative and qualitative approaches for preferences was proposed by Boutilier et al. (Boutilier et al. 2001). The UCP-network formalism is an extension of the CP-network model (Boutilier et al. 2004) that allows one to represent quantitative utility information rather than only preference orderings.

Goals. A preference representation model (UCP-networks) and algorithms.

Questions. Outcome optimisation and dominance queries.

Input. The required input of this approach is a network representation, namely UCP-networks (or UCP-nets), which combine aspects of CP-nets and generalised additive independence (GAI) models. Sets of attributes X_i are said to *generalised additive independent* if the expected value of the utility function u is not affected by correlations between the X_i, but it depends only on the marginal distributions over each X_i.

UCP-nets are similar to CP-nets in that they are also a graph whose nodes are attributes, and an arc from one node to another indicates that preferences over an attribute are conditioned to values of a parent attribute. However, the graph structure is restricted to a DAG and instead of annotating nodes with CPTs, a table associates values of an attribute and its parents with a utility value. In other words, a UCP-net extends a CP-net with conditional utility information. For purposes of elicitation and computation, it is often convenient to normalise utilities over the range [0, 1].

Every UCP-net specifies a GAI decomposition of its underlying utility function, i.e. the utility of an outcome is calculated by summing the different

factors $f_i(X_i, U_i)$, where X_i is each individual attribute, and U_i is the parents of X_i. In addition, the GAI decomposition must respect the conditional preference independence imposed by the network.

Interpretation. Ceteris paribus.

How it works. For dominance queries, the utilities of two outcomes are compared, and they are obtained by extracting and summing the values of each factor in the network. On the other hand, for outcome optimisation queries, the CP-net structure is exploited and the forward sweep procedure (Boutilier et al. 2004), introduced in Section 5.5.1, is used.

One of the main reasons to move from qualitative to quantitative preference models is to support decisions under (quantified) uncertainty about outcomes of options. So, when the distributions induced by options can be structured in a Bayes net (Jensen 2001), UCP-nets can be used to help structure the decision problem. The approach presented (Boutilier et al. 2001) for this purpose uses GAI factorisation of utilities afforded by the UCP-net.

Complexity. Both outcome optimisation and dominance queries can be done in time linear in the size of the network, due to the use of CP-nets with utility functions.

Limitations. An initial problem of this approach is that determining if a quantified network is in fact a UCP-net requires a case-by-case analysis for each "extended family" in the network involving a number of tests exponential in the size of the extended families (each consisting of an attribute, its parents, its children, and its children's parents). However, it is not so problematic, given that families are not expected to have a large size. The major issue is related to the adoption of a quantitative approach, which brings problems to the elicitation process. The authors assume that both structure and local value functions are something that users will often be able to provide without too much difficulty.

5.5.4 TCP-nets

CP-nets represent the class of conditional qualitative preference statements. Brafman et al. (Brafman et al. 2006) address a limitation of this approach by extending CP-nets to capture another class of preference statements: those with conditional relative importance. These statements have the following form: "it is more important to me that the value of X be high than that the value of Y be high."

CP-nets were extended with new types of arcs, and questions to be answered based on this new graphical representation of preferences were investigated.

Goals. A preference representation model (TCP-nets) and algorithms.

Questions. Outcome optimisation, dominance and ordering queries. However, little attention has been given to ordering queries, as they are weaker than dominance queries.

Input. A TCP-net (which is a CP-net that also allows to represent attribute importance, and was presented in Section 4.2.2) is the only required input for outcome optimisation, and it can also be provided with an optional partial assignment for attributes. The required input for both dominance and ordering queries is a TCP-net N and two outcomes o and o'.

Interpretation. Ceteris paribus.

How it works. Informally, the semantics of TCP-nets is as follows. A strict partial order $\succ$ satisfies:

- *the conditional preferences for attribute* X, if any two complete assignments that differ only on the value of X are ordered by $\succ$ consistently with the ordering on X values in the CPT of X. This ordering can depend on the parent of X in the graph;
- *the assertion that* X *is more important than* Y, if given any two complete assignments that differ on the value of X and Y only, $\succ$ prefers that assignment which provides X with a better value; and
- *the assertion that* X *is more important than* Y *given some assignment* z *to attribute set* Z, if given any two complete assignments that differ on the value X and Y only, and in (both of) which Z is assigned z, $\succ$ prefers that assignment which provides X with a better value.

A particular class of TCP-nets is investigated by Brafman et al., because it is a class of networks that are proven satisfiable. This class of TCP-nets consists of conditionally acyclic TCP-nets, whose key property is that they induce an "ordering" over the nodes of the network. A TCP-net is conditionally acyclic if its induced dependency graph is acyclic and for every assignment w to the union of all selector sets (i.e. sets of the attributes associated with conditional importance arcs) of the network, the induced w-directed graphs are acyclic. Therefore, verifying if

a TCP-net is conditionally acyclic becomes a relevant issue. This is performed in polynomial time in certain situations, which are identified and detailed elsewhere (Brafman et al. 2006); however, this verification is generally hard.

Outcome Optimisation. For computing the most preferred outcome of an acyclic TCP-net and a (possible empty) partial assignment x on its attributes, the forward sweep procedure introduced in Section 5.5.1 can be used. Nevertheless, it is not possible if a set of hard constraints is provided. This happens because the key difference between processing an acyclic CP-net and a conditionally acyclic TCP-net is that, while the former induces a single partial order of importance over attributes, the latter induces a hierarchically structured set of such partial orders. So, a branch-and-bound algorithm is presented for computing the optimal outcome in this scenario, which consists of two parts: *Search-TCP* and *Reduce*. The *Search-TCP* algorithm is guided by the underlying TCP-net N. It proceeds by assigning values to the attributes in a top-down manner, assuring that outcomes are generated according to the preferential ordering induced by N. The *Reduce* algorithm, in turn, refines a TCP-net N with respect to a partial assignment K': it reduces both the CPTs and the CITs involving this attribute, and removes this attribute from the network.

Dominance queries. Much like in CP-nets, a dominance query $\langle N, o, o' \rangle$ with respect to a TCP-net can be treated as a search for an improving flipping sequence from the (purported) less preferred outcome o' to the (purported) more preferred outcome o through a sequence of successively more preferred outcomes, such that each flip in this sequence is directly sanctioned by the given TCP-net.

Complexity. Brafman et al. present a theorem that states that every conditionally acyclic TCP-net is satisfiable. Based on this theorem outcome optimisation and dominance queries in the context of this class of TCP-nets are investigated, with the following complexity analyses provided.

- Determining that a TCP-net is conditionally acyclic is co-NP-hard.
- TCP-nets have the optimal outcomes generated using the forward sweep procedure, and therefore the complexity for performing this task also applies for TCP-nets.
  - If a set of hard constraints is provided, determining the set of preferentially non-dominated outcomes is not trivial, and a

branch-and-bound algorithm is used, whose worst case leads to exponential time complexities.

– Dominance testing with respect to CP-nets, and thus TCP-nets, is NP-hard.

Limitations. As TCP-nets are an extension of CP-nets, they inherit some of the limitations of CP-nets, which are related to the complexity of algorithms to reason about preferences and the types of attributes addressed. In addition, even though TCP-nets capture importance relations between attributes, the way that these relations are interpreted lead to the maximisation of the most important attribute and a high penalisation for the least important ones.

5.5.5 Graphically Structured Value Function Compilation

Brafman and Domshlak (Brafman and Domshlak 2008) provide an approach for compiling the information captured by CP-nets and TCP-nets to value functions. As reasoning about preferences represented by value functions is easy as they establish an order among outcomes by ordering computed values, the challenge here is to convert these graphical structures into them. The authors assume that preference statements were already elicited or informed by users, and represented as a TCP-net — or a CP-net, in case there is no (conditional) relative importance between pairs of attributes. The graphical structure plays an important role in analysing and compiling these statements.

Goals. A transformation from *TCP-nets (and CP-nets)* to *value functions*.

Input. The input required is qualitative preferences represented in either a CP-net or a TCP-net, therefore the approach is able to deal with the kinds of statements represented by these graphical structures.

Interpretation. Ceteris paribus.

How it works. This work proposes to use a TCP-net to initially organise qualitative preference statements obtained from the user. Then, this information is compiled to a value function that maintains the qualitative structure and independence assumptions implicit in this TCP-net. This obtained value function is used as the model of user's ordinal preference, and as new information comes from the user, this value function is refined, while still maintaining independence assumptions implied by the original TCP-net, if possible.

A TCP-net can be represented as a value function, as it defines a (partial) order among outcomes. So, the main issue is to investigate whether there exists a generalised additive (GA) value function defined over small factors "implied" by the structure of the network, i.e. find a structured value function that, in some sense, is as *compact* as the original TCP-net.

For CP-nets, the concept of a CP-family is defined, which is the set of an attribute and its parents. That is, the goal is to define a GA value function whose factors are the families of the CP-net. It is shown that every acyclic CP-net is GA-decomposable over its CP-families, by constructing a system of linear inequalities (CP-conditions) and finding a solution for it. A similar approach is presented for TCP-nets — every acyclic TCP-net is GA-decomposable over its TCP-families — a TCP-family is the set of an attribute, its parents and the attributes that impose conditionality to the target attribute.

The approach is also extended to incorporate new item-level rankings provided by users. It is shown that this value function compilation is not only efficient and sound, but also complete, that is, the ability to generate the value function is guaranteed.

Complexity. A system of linear inequalities L, either for CP-nets or TCP-nets, is locally exponential. However, if the maximum cardinality of all extended CP-families (an attribute, its parents, its children, and its children's parents) of a CP-net N is a constant k, a value function consistent with N can be constructed in time polynomial in the size of N. It also holds for TCP-nets.

Limitations. Regarding preferences representation and interpretation, as this work relies on TCP-nets, limitations described for them (and CP-nets), are also applied. Moreover, the authors themselves point out that their work raises numerous open theoretical questions, including (Brafman and Domshlak 2008): (i) when (if at all) GA-decomposition is complete for cyclic TCP-nets, or even just cyclic CP-nets? (ii) what is the most compact form of GA-decomposition that is complete for all consistent TCP-nets?

5.6 Query-based Approaches

Most of the presented approaches were investigated in the context of AI, but more recently preferences have also been taken into account in research related to databases. This was mainly motivated by a scenario that emerged from the web, as users search online store databases and their specified preferences often result in a query over constrained, which has no result. In addition, web search engines provide

a huge amount of results for a set of keywords (a query), which can be reduced and ranked if preferences are taken into account. This section presents work that aims at integrating preferences into a query language.

5.6.1 Scoring Function

Agrawal and Wimmers (Agrawal and Wimmers 2000) proposed a framework for expressing and combining user preferences, based on the definition of an atomic scoring function, together with combination operations used to compute a score for each result tuple. The framework has two basic components: (i) a preference function that specifies user preferences; and (ii) a single meta-combining form *combine* that is based on value functions. Details of how the framework works are provided below.

Goals. A preference representation model and algorithms.

Questions. Ordering queries: how options are ranked according to preference functions provided by multiple users?

Input. Given a framework (detailed in Section 4.3.1) in which preferences for an entity in this framework are expressed by a numeric score between 0 and 1, vetoing it, or explicitly stating indifference, the required input to answer ordering queries is a database with available options and a set of preference functions, which are functions that map options to a score.

Interpretation. Not ceteris paribus. A preference function defines scores for specific options (or projections), and therefore a score is provided for a fully specified context, or, if the score is given for an option projection, it means that the score is also valid for any combination of values for the remaining option names (or attributes). In addition, when the wild type (*) is used for a name, the specified score will be considered for options with all possible values for that name too.

How it works. Preference functions consist of scores provided by (or elicited from) users for individual combinations of values of option names. Then, a preference function meta-combining form is defined, named *combine*, which takes a "*value function*" that states how to compute a new score based on the original scores and produces a preference function combining form, which takes a finite list of preference functions and produces the new preference function. *Combine* abstracts how different scores are combined: this function receives a function

as parameter, which determines how to combine scores. The example provided where the framework was proposed (Agrawal and Wimmers 2000) is a `FirstVeto` function, which establishes that scores of an specific individual are more important than of another.

Limitations. In terms of preference representation, this approach requires users to provide scores for each combination of values, if no elicitation method is used. In addition, no particular inference is performed to reason about preferences: scores order options in a particular order, and therefore ranking options is straightforward. Moreover, the provided preference function combining form is abstract, in the sense that resolving score combination is still part of specific instantiations of the framework, and the example provided (`FirstVeto`) is trivial.

5.6.2 Winnow

An extension to relational algebra was proposed by Chomicki (Chomicki 2003) as preference query formalisation, in which preferences are expressed as binary relations between tuples from the same database relation, which represent options to be selected. The approach defines a central algebraic operator (*winnow*), which selects the set of dominant options from a database according to provided preferences, and this operator has algebraic properties analysed, such as commutativity, commutation of selection or selection, and distribution of winnow over union and different. Analysing such properties of winnow is important (Chomicki 2003) as they can be exploited for formulating efficient database query execution plans.

Goals. A preference representation model, an algebra and algorithms.

Questions. Non-dominated outcomes.

Input. As input, it is required a database with available options and a set of preference formulae, defined as introduced in Section 4.3.2.

Interpretation. Not ceteris paribus.

How it works. An algebraic operator called *winnow*, whose definition is presented below, picks from a given relation the set of the most preferred (dominant) options, according to a given preference formula. A preference query is a relational algebra query containing at least one occurrence of the winnow operator.

Winnow. *If R is a relation schema and C a preference formula defining a preference relation $\succ_C$ over R, then the winnow operator is written as $\omega_C(R)$, and for every instance r of R:*

$$\omega_C(R) = \{t \in r \mid \neg \exists\, t' \in r.t' \succ_C t\}$$

There are three possible algorithms (Chomicki 2003) to evaluate winnow. The first is a nested-loops algorithm, which compares every two options and discards dominated ones. This algorithm is correct for any preference relation $\succ$. The second, BNL, is an algorithm proposed by Borzsonyi et al. (Borzsonyi et al. 2001) in the context of a specific class of preference queries, namely skyline queries, but the algorithm is considerably more general than the previous one. And the third (Chomicki et al. 2005), SFS, is a variant of the second, in which a presorting step is used. The BNL and SFS algorithms require the preference relation to be a strict partial order.

Complexity. The evaluation of the winnow operator, i.e. identifying dominant options, is not part of the solution provided by Chomicki. However, the author provides a complexity analysis of properties (irreflexivity, asymmetry, transitivity, negative transitivity and connectivity) of preference formulae, which can be seen elsewhere (Chomicki 2003). The best case complexity of the simple nested-loops algorithm presented is of the order of $O(n)$; n being the number of options in the input. In the worst case, the complexity is of the order of $O(n^2)$.

Limitations. The main issue investigated by this approach is algebraic properties of an operator added to relational algebra; however, the problem of identifying the most preferred options is abstracted. Moreover, the goal is to identify non-dominanted options, and there is no way to tell which is the optimal option from this subset of options. Finally, even though any preference can be expressed with a preference formula, including establishing a full total order among options, it is not compact, which is relevant when preferences are provided by users. The preference formula below, which expresses "*I prefer white wine with fish, and red wine with meat*," illustrates this issue.

$$\begin{aligned}(d, dt, w, wt) \succ_C (d0, dt0, w0, wt0) \equiv (d = d' \wedge dt = \text{`}fish' \wedge wt = \text{`}white' \\ \wedge dt' = \text{`}fish' \wedge wt' = \text{`}red') \\ \vee (d = d' \wedge dt = \text{`}meat' \wedge wt = \text{`}red' \\ \wedge dt' = \text{`}meat' \wedge wt' = \text{`}white')\end{aligned}$$

5.6.3 Best-Matches-Only Query Model

Kießling (Kießling 2002) defines preferences as strict partial orders and proposes several preference constructors in order to support the accumulation of single preferences into more complex ones. The author also provides a collection of algebraic laws to manipulate such preference constructions, as well as defines how to evaluate preference queries under the Best-Matches-Only (BMO) query model and decomposition algorithms for complex preference queries.

Goals. A preference representation model, and an algebra.

Questions. Non-dominated outcomes.

Input. The input needed for querying non-dominated outcomes is a database with available options and a set of preferences, built with preference constructors, presented in Section 4.3.3. In summary, there are two classes of constructors: (i) non-numerical base preferences, which include positive and negative preferences; and (ii) numerical base preferences, which include preferences that specify preferred numerical values of an attribute using, for example, intervals. A preference term (i.e. a preference that is valid according to the provided constructors) can also be composed of other preferences.

Interpretation. Not ceteris paribus.

How it works. The BMO result set contains only the best matches with respect to the strict partial order of a preference P. It is a selection of unordered result of options. All options t, $t' \in BMO$ have equal or incomparable values regarding the preference P.

A preference query is defined by $\sigma[P](R)$ declaratively as follows: $\sigma[P](R) = \{t \in R \mid t[A] \in max(P^R)\}$. $\sigma[P](R)$ evaluates P against a database set R by retrieving all maximal values from P^R. Preference queries behave non-monotonically, in that they are sensitive (holistic) to the quality of a collection of values. In addition, the following manipulations are provided in order to evaluate preference queries:(i) preference hierarchies; (ii) decomposition of '+' and and '♦'-queries; (iii) decomposition of '&'-queries; and (iv) decomposition of '⊕'-queries. These are accumulating and aggregating preference constructors introduced in Section 4.3.3.

Limitations. This approach is similar to the winnow operator (Section 5.6.2), and the same limitations reported for it are applicable to the BMO query model.

5.6.4 Query Personalisation based on Preferences

The preference model proposed by Koutrika and Ioannidis (Koutrika and Ioannidis 2006) is associated with preference selection algorithms, generation of personalised answers and raking functions. These parts work together in order to provide personalised results for a database search query, according to the preferences of a specific user. Users have several preferences, and as only some of them are relevant for a specific query, algorithms are proposed for selecting preferences related to a query according to various criteria. Based on a set of selected preferences and a query to be executed, the approach generates personalised query and results, which are ranked based on the estimated user interest.

Goals. A preference representation model, and algorithms.

Questions. Ordering queries, with a focus on the context of databases: how query results are selected and ranked according to user preferences?

Input. The personalisation process has two main steps, and the output of the first (*preference selection*) is used as input of the second (*generation of personalised results*). Preference selection receives as input a set of user preferences, a query to be executed, and a parameter K, which is a criterion for choice. The second step requires the query to be executed, the set of preferences selected in the previous step of query personalisation, and an explicit or implicit specification for L, which indicates the number of preferences that should at least be satisfied. Preferences, as detailed in Section 4.3.4, may be expressed for values of attributes, and for relationships between entities, which indicate to what degree, if any, entities related depend on each other.

Interpretation. Not ceteris paribus. The degree of interest given for each part of attribute-value is independent of other attributes.

How it works. The first step of the query personalisation process deals with the extraction of the top (most critical) K preferences related to a query. The selection is based on a syntactic level, which means that a preference is related to a query, if it maps to a path attached to the query graph. K is a criterion based on the degree

of criticality of preferences, which may specify for instance the top X preferences, or preferences with a degree of criticality above a threshold c_0.

With the execution of this first step, the top K preferences are integrated into the user query and a personalised answer is generated. This answer should be: (a) *interesting* to the user, in that it should satisfy (at least) L from the top K preferences; (b) *ranked* based on the degree of interest; and (c) *self-explanatory*: for each option returned, the preferences satisfied or not should be provided in order to explain its selection and ranking.

Two approaches are described for the generation of personalised answers: *Simply Personalized Answers (SPA)*; and *Progressive Personalized Answers (PPA)*. The first integrates the top K preferences into the initial query and builds a new one, which is executed. A personalised query is formulated as the union of a set of sub-queries, each one mapping to one or more of the K preferences selected. The last integrates preferences into sub-queries as in the SPA methodology. 1-to-many absence preferences are integrated as if they were presence ones. Hence, two sets of sub-queries are formed: a set of subqueries involving presence and 1-to-1 absence preferences, Q_s, and a set of subqueries involving 1-to-many absence preferences, Q_a. After the execution of each sub-query, a degree of interest d_t is calculated for each result using any *ranking function* for positive, negative or mixed combinations of preferences. A class of ranking functions is provided, and we refer to reader to the provided reference (Koutrika and Ioannidis 2006) for details. The list of options returned, R, is ordered in decreasing degree of interest — those that do not achieve a minimum degree of interest are discarded, and the remaining ones are given as the result of the query.

Limitations. Besides the problem of relying on a quantitative preference model, which are hard to elicit, the approach lets the *ranking function* as user-defined function, which is one of the main issues of the decision making process. Finding a numeric value that represents user preference for an option by combining individual preferences is not trivial, and this is left as a variable point of the approach.

5.6.5 OWLPref

The declarative, domain-independent way of representing preferences in OWL, namely OWLPref (Ayres and Furtado 2007), described in Section 4.4.1, has an associated API, which maps concepts from the OWLPref to queries of the SPARQL Preference (Siberski et al. 2006) query language. The authors support this choice by arguing that SPARQL is currently the most important and used language for querying semantic data, and also because they focused on the

ontology modelling, and not the inference engine for query execution, which is provided by SPARQL. The idea is that each preference defined in OWLPref has a defined mapping to a SPARQL preference statement, nevertheless this mapping is not detailed (Ayres and Furtado 2007), and it is not straightforward — there are kinds of preferences in OWLPref, e.g. `AroundPreference`, that do not have a corresponding constructor in SPARQL Preference.

As the concrete preference reasoning is performed by SPARQL Preference and the mapping process from OWLPref to it is not provided, we dedicate the remainder of this section to detail this SPARQL extension that deals with preferences. SPARQL Preference is extensively based on the winnow operator (Chomicki 2003), which was described in Section 5.6.2.

Goals. A preference representation model (extension of a query language) and formal definitions for each construction (semantics).

Questions. Non-dominated outcomes. More precisely:

1. Which are the solutions non-dominated by any other solutions? (*Skyline queries*)
2. Which are the solutions that satisfy all the (hard and soft) constraints? If none, by relaxing some or all soft constraints, which are best answers? (*Soft constraints*)

Input. A database with available options and a SPARQL Preference query, which is expressed in SPARQL with the `PreferringClause` extension, as shown below.

```
SolutionModifier ::= PreferringClause? OrderClause? LimitClause?
                     OffsetClause?
PreferringClause ::= 'PREFERRING' MultidimensionalPreference
MultidimensionalPreference ::= CascadedPreference
                               ('AND' CascadedPreference)*
CascadedPreference ::= AtomicPreference
                       ('CASCADE' AtomicPreference)*
AtomicPreference ::= BooleanPreference | HighestPreference |
                     LowestPreference
BooleanPreference::= Expression
HighestPreference::= 'HIGHEST' Expression
LowestPreference ::= 'LOWEST' Expression
```

Interpretation. Not ceteris paribus.

How it works. The SPARQL Preference syntax allows modelling two kinds of preferences and two kinds of preference interaction, whose semantics is described informally as follows. For a formal definition, we refer the reader to elsewhere (Siberski et al. 2006).

- **Boolean preferences**. Boolean preferences are specified by a boolean expression over solutions of an ontology defined in OWL DL (Patel-Schneider et al. 2004). An option S_i dominates an option S_j, if the boolean expression is evaluated to true for S_i, and false for S_j.
- **Scoring preferences**. This kind of preference is specified by an expression, which evaluates to a number or a value in other SPARQL domains that have total ordering. They express the preference for maximising or minimising a certain value. Therefore, an option S_i dominates S_j, if S_i has a higher (lower) value than S_j, when a `HighestPreference` (`LowestPreference`) is defined.
- **Multidimensional Preferences**. This kind of preference interaction indicates that two preferences must be addressed in a combined form. For any solutions S_i and S_j, the domination relation to combine independent preferences $\succ_{|C1\ AND\ C2|}$ establishes that S_i is dominated by S_j in neither $C1$ nor $C2$, and that S_i dominates S_j in either $C1$ or $C2$.
- **Cascaded Preference**. A cascaded preference indicates that a preference $C1$ has a higher priority than $C2$. So, for any options S_i and S_j, the domination relation to combine prioritised preferences $\succ_{|C1\ CASCADE\ C2|}$ states that either S_i dominates S_j according to $C1$, or they are incomparable according to $C1$, and S_i dominates S_j according to $C2$.

Based on the definition of dominance among options, a SPARQL Preference query returns as result all non-dominated solutions (*skyline queries*). If a solution S_i dominates S_j according to a preference $C1$, but S_j dominates S_i according to a preference $C2$, both options are presented as the result of the query.

Limitations. The major problem of this approach occurs when a best option is not found, according to stated preferences, which is a scenario that typically occurs, because trade-off among conflicting preferences is very common. For instance, if a user wants to buy a laptop and she states that she prefers to minimise price and maximise performance. The typical case is that price increases as the performance is better and, therefore, according to these preferences no laptop dominates another

(there is none, or only a few laptops, that are more expensive and have worse performance), and consequently the approach cannot select among these laptops.

5.7 Preferences in Argumentation Frameworks

Based on the different presented works, it can be seen that reasoning about preferences has been target of research in different areas of computer science, including AI, databases, and semantic web. Recently, this research topic has been investigated in the context of argumentation (Rahwan and Simari 2009), which can be abstractly defined as the interaction of different arguments for and against some conclusion. Amgoud et al. (Amgoud et al. 2008) discuss how to make decisions through preference-based argumentation, but they basically illustrate an application of the use of argumentation frameworks for deciding between two options, and there is no explicit representation of preferences. As a consequence, this work is not detailed in this section. Modgil (Modgil 2009) extended the classical Dung's argumentation theory (Dung 1995) in order to incorporate arguments that claim preferences between other arguments, thus incorporating metalevel-argumentation-based reasoning about preferences in the object level. This proposed extension is thus detailed next.

Goals. A preference representation model — Extended Argumentation Framework (EAF) — and semantics for identifying which are the sets of justified arguments, i.e. the possible extensions of a given EAF.

Input. The required input of this approach is an Extended Argumentation Framework (EAF), which is an extension of the argumentation framework proposed by Dung (Dung 1995). An argumentation framework models arguments, abstracting from the underlying logic that represents them, as a directed graph in which arguments are represented as nodes and a defeat relation is represented as arrows. This structure is then used for determining which arguments are the justified. EAFs accommodate arguments that claim preferences between other arguments, i.e. they explicit capture the notion of preferences in argumentation frameworks. Formally, an EAF is defined as follows.

An Extended Argumentation Framework (EAF) is a tuple $(Args, \mathcal{R}, \mathcal{D})$ such that $Args$ is a set of arguments, and:

- $\mathcal{R} \subseteq Args \times Args$,
- $\mathcal{D} \subseteq Args \times \mathcal{R}$,
- If $(X, (Y, Z)), (X', (Z, Y)) \in \mathcal{D}$ then $(X, X'), (X', X) \in \mathcal{R}$.

Interpretation. The preference relation proposed by Modgil (Modgil 2009) addresses preferences over attacks, and not over (a pair) of outcomes, as it is the case of the other works presented in this chapter. Therefore, the interpretation of preference statements does not apply for this work.

How it works. The main extension of EAFs is the inclusion of a second attack relation $\mathcal{D}$ that ranges from arguments X to attacks $(Y, Z) \in \mathcal{R}$, where $\mathcal{R}$ is the standard binary attack relation in a Dung framework. If $(X, Y), (Y, X) \in \mathcal{R}$, i.e. arguments X and Y attack each other, and there is a $(Z, (Y, X)) \in \mathcal{D}$, we can say that the attack (X, Y) is preferred to (Y, X). Additionally, preference arguments expressing contradictory preferences attack each other, where these attacks can then themselves be attacked by preference arguments. Preferences are not defined by some externally given preference ordering, but are themselves claimed by arguments.

As Dung's argumentation framework was extended, it is essential to redefine some concepts, e.g. when a set is conflict free and the acceptability of arguments (we refer the reader to Dung's work (Dung 1995) for these definitions), i.e. the semantics underlying Dung's framework should also be extended. Modgil starts by defining when an argument A defeats another argument B, which are in $S \subseteq Args$. It occurs when there is an $(A, B) \in \mathcal{R}$ and there is no argument $C \in S$, such that $(C, (A, B)) \in \mathcal{D}$. It is said that $A\ defeat_S\ B$. Conflict free set is intuitively defined as follows. If $A, B \in S$ ($S \subseteq Args$) and A attacks B, then S is *conflict free* only if B does not attack A and there is a C that defence attacks the attack from A to B. Finally, an argument $A \in Args$ is *acceptable* with respect to $S \subseteq Args$, if and only if for all B such that $B\ defeat_S\ A$, there is a $C \in S$ such that $C\ defeat_S\ B$ and there is a reinstatement set for $C\ defeat_S\ B$ (reinstatement set of an argument X is a set of arguments that defeats any argument that attacks X). With these definitions, the admissible, preferred, complete and stable extensions of an EAFs are defined in the same way as for Dung's framework.

Limitations. This work (Modgil 2009) has a different purpose in comparison of other works presented in this section, in that it proposes a general framework for supporting decision making, which is not restricted to choosing a preferred outcome from a set or a pair of outcomes. Therefore, it is not clear how the typical preference statements (discussed in other sections) can be represented and, with them, answering the typical questions related to preference reasoning. In addition, complexity is not a topic discussed, because, with the definition when an argument defeats another, any algorithm to identify extensions of argumentation frameworks can be used. However, only certain kinds of extensions

of argumentation frameworks with particular graph structures can be identified in polynomial time.

5.8 Discussion

In this section, we present a comparison of the presented approaches and discuss relevant points related to them. As stated in the introduction, our goal is not to rate these approaches to choose the best one, but to better understand them and their relationship. We start by summarising the key characteristics of the approaches for reasoning about preferences, which are depicted in Table 5.4. For each approach, we show six different aspects, which are presented in respective columns.

- **Area**: indicates to which research area the approach is mainly related.
- **Qualitative/Quantitative Representation and Reasoning**: classifies the representation and the reasoning method adopted by the approach as quantitative or qualitative.[3] Quantitative approaches represent preferences as a number (or a function that produces a number), such as ratings that capture the value of an option. Qualitative approaches, on the other hand, allow comparing outcomes without specifically stating how much an outcome or attribute value is preferred to another ("*I prefer X to Y*") or qualitatively evaluate them ("*I like X very much*").[4] *Representation* is related to how preferences captured from users are represented. This representation can be directly manipulated or can be transformed into another representation to be used by the algorithm of the approach (*reasoning*). For instance, the approach "Graphically Structured Value Function Compilation" adopts a qualitative representation (CP-net or TCP-net) but converts it to a quantitative representation (value function) for reasoning about the provided preferences.
- **Goals**: the goals shown in this table follow the classification introduced in Section 5.1. The acronyms stand for: *PRM* — Preference Representation Model; *ALGO* — Algorithms; *ALGE* — Algebra; *SEM* — Semantics; *TRANS* — Transformation.
- **Questions**: the questions shown in this table follow the classification introduced in Section 5.1. The acronyms stand for: *OO* — Outcome Optimisation; *DQ* — Dominance Query; *OQ* — Ordering Query; *NDO* — Non-dominated outcomes.

[3]Table 5.4 shows this term in italics in order to make a better visual distinction between the terms qualitative and quantitative.

[4]This distinction between quantitative and qualitative preferences is widely adopted by researchers on preference reasoning, even though an ordinal scale is still quantitative, but imprecise.

– **Input**: provides a general classification for the required input of the approach. It can be utility (or value) functions, a particular structure (e.g. CP-nets), qualitative preference statements (we do not specify here of which kinds), or quantitative preference statements (we do not specify here the domain used to rate outcomes or attribute values).
– **Interpretation**: indicates how preference statements are interpreted by the approach. There are three possible options: (i) *CP* (Ceteris Paribus) — when the approach uses the "all else being equal" interpretation for statements; (ii) *NCP* (Not Ceteris Paribus) — any other possible interpretation for provided statements, e.g. in the approach "Learning Utility Functions with SVM" statements have a marginal contribution for the utility function; and (iii) *N/A* (Not Applicable) — this is only the case of "Preferences in Argumentation Frameworks." As in this approach arguments are abstract and are not necessarily preference statements, it is not possible to state a specific interpretation adopted by the approach.

By analysing these approaches, we can observe that they address three main problems: (i) given a representation of qualitative statements, how can the typical questions (outcome optimisation and dominance queries) related to reasoning about preferences be answered? (ii) given a representation of qualitative statements, how can it be transformed into a utility function? and (iii) how can we represent compact utility functions? In problem (i), preferences are considered easier to elicit than in quantitative approaches, as qualitative statements are considered closer to the user vocabulary, but are difficult to be reasoned about. Problem (iii) is related to approaches that rely on utility functions to reason about preferences, and with them one can easily compare outcomes. However, eliciting utility functions is not a trivial task, and for particular contexts, combinations of attribute values have utility values that cannot be represented in a compact form, such as assuming utility independence among attributes. Therefore, problem (ii) aims at obtaining the main benefits of (i), i.e. easy representation, and (iii), i.e. easy reasoning, but the challenge now is the translation of qualitative to quantitative statements. In addition, considering all possible combinations of attribute values is a combinatorial problem, therefore it is very important to take into account the complexity of reasoning about preferences. We do not discuss complexity in this section, as the worst case takes exponential time to be computed for almost all approaches, and some of them do not provide complexity analysis.

All the approaches aim at helping users to make decisions of choosing from among alternatives, and we can classify this aim into two categories: (a) helping users to make decisions in situations that they are confused and it is hard for them to compare outcomes and identify the best one; and (b) helping users to make

Table 5.4: Comparison among Approaches to Reason about Preferences.

Approach	**Area**	**Quantitative/Qualitative**		**Goals**	**Questions**	**Input**	**Inter-pretation**
		Representation	**Reasoning**				
CUI Networks (Engel and Wellman 2008)	AI	Quantitative	Quantitative	PRM, ALGO	OO	Utility Functions	NCP
Utility Functions for Ceteris Paribus Preferences (McGeachie and Doyle 2008)	AI	*Qualitative*	Quantitative	TRANS		Quantitative Preferences	CP
Learning Utility Functions with SVM (Domshlak and Joachims 2007)	AI	*Qualitative*	Quantitative	TRANS		Quantitative Preferences	NCP
Semiring-based Constraint Satisfaction (Bistarelli et al. 1997)	Constraint Programming	Quantitative	Quantitative	PRM	OO	Quantitative Preferences	NCP
Preference-based Problem Solving for Constraint Satisfaction (Junker 2008)	Constraint Programming	*Qualitative*	Quantitative	PRM, ALGO	OO	Soft Constraint Satisfaction Problem	NCP
CP-nets (Boutilier et al. 2004)	AI	*Qualitative*	*Qualitative*	PRM, ALGO	OO, DQ, OQ	CP-net	CP
Combining CP-nets and Soft Constraints (Domshlak et al. 2003)	AI	*Qualitative*	Quantitative	TRANS	DQ	Acyclic CP-net	CP
UCP-networks (Boutilier et al. 2001)	AI	*Qualitative*	Quantitative	PRM, ALGO	OO, DQ	UCP-net	CP
TCP-nets (Brafman et al. 2006)	AI	*Qualitative*	*Qualitative*	PRM, ALGO	OO, DQ	TCP-net	CP
Graphically Structured Value Function Compilation (Brafman and Domshlak 2008)	AI	*Qualitative*	Quantitative	TRANS		CP-net or TCP-net	CP
Scoring Function (Agrawal and Wimmers 2000)	Databases	Quantitative	Quantitative	PRM, ALGO	OQ	Quantitative Preferences	NCP
Winnow (Chomicki 2003)	Databases	*Qualitative*	*Qualitative*	PRM, ALGO, ALGE	NDO	Qualitative Preferences	NCP
BMO Query Model (Kießling 2002)	Databases	*Qualitative*	*Qualitative*	PRM, ALGE	NDO	Qualitative Preferences	NCP
Query Personalisation based on Preferences (Koutrika and Ioannidis 2006)	Databases	Quantitative	Quantitative	PRM, ALGO	OQ	Quantitative Preferences	NCP
OWLPref (Ayres and Furtado 2007) SPARQL Preference (Siberski et al. 2006)	Semantic Web Databases	*Qualitative*	*Qualitative*	PRM, SEM	NDO	Qualitative Preferences	NCP
Preferences in Argumentation Frameworks (Modgil 2009)	Argumentation (AI)	*Qualitative*	*Qualitative*	PRM, SEM		Extended Argumentation Framework	N/A

decisions in situations that they know (or have an idea of) what is best, but the set of possible outcomes is so large that this task becomes time-consuming and requires too much effort of the user. Even though, in the end, the problem to be solved is the same in these two categories, they differ in a very relevant aspect: engagement of users in providing information about their preferences. A typical scenario for (a) is a company manager that has to decide an action to be taken, and this action has a crucial impact in the profit of the company. In addition, many attributes with uncertain values are associated with this action. Therefore, this manager is willing to spend a significant amount of time to precisely specify preferences among options and related attributes. And a typical scenario for (b) is web users that make searches with keywords and receive in return a large amount of results, which they have to filter and rank.

MAUT has emerged with the goal described in (a), i.e. helping a confused decision-maker to evaluate complex alternatives when their outcomes are uncertain and have several relevant attributes. Therefore, adopting utility functions to represent preference is reasonable (but still difficult), as users are willing to spend time in eliciting them, i.e. when the consequences of making a wrong decision compensates the time and effort spent in eliciting (and building new) preferences. Even though there is work (McGeachie and Doyle 2008, Domshlak and Joachims 2007) that obtains utility functions from qualitative statements, it still has limitations reported in the approaches presented in this chapter.

Query-based approaches address the scenario described in (b), and their aim is not to totally automate the decision making process, but to provide a reduced amount of results to users, possibly ranked. Considering preferences in queries allows users to be more restrictive when providing constraints for queries and still obtaining results that contain useful information — if the query is over constrained it is likely that to return no result. Therefore, this family of approaches aim at identifying non-dominated outcomes, and those dominated are discarded as they have no advantage with respect to the non-dominated outcomes. Choosing among non-dominated outcomes is a task that is still left for the user, but some heuristics, such as considering lexicographic importance among attributes, can be used to rank them, and give guidance to the users. There are two approaches (Agrawal and Wimmers 2000, Koutrika and Ioannidis 2006) that are quantitative, but for defining a total order of outcomes requires making assumptions, such as independence among attributes. Moreover, expecting quantitative values from users may require sophisticated elicitation processes.

In order to deal with over constrained problems, CSPs have been extended to incorporate constraints that can remain unsatisfied, with a no-satisfaction penalty

associated with them. These approaches have the objective to minimise the penalty of unsatisfied constraints (or maximising preferences), and other objectives, e.g. minimise price, cannot be specified. In addition, trade-off situations among conflicting objectives cannot be modelled either. SCSPs can help in solving a significant class of problems, e.g. meeting scheduling problems, but there are other kind of problems that cannot be addressed, such as those that include multi-objectives and trade-off.

Finally, there are the graphically-structured approaches, which structure qualitative preference statements in a graph and adopt the ceteris paribus interpretation. For graphs with certain properties, these approaches can be efficiently used to define optimal outcomes and, with the proposed techniques (Domshlak et al. 2003), can also answer dominance queries efficiently. However, two outcomes can be compared only when "all else is equal," and in practice it is often not the case. When other attribute values differ, outcomes are considered incomparable and no decision can be made regarding dominance. Moreover, these approaches can deal only with discrete attribute domains, and this is also a very restrictive assumption. Furthermore, trade-off is only captured in TCP-nets, but a lexicographic approach is adopted, unless a fully specified CIT is provided.

5.9 Final Considerations

In this chapter, we presented a systematic review of research work that aims at helping and automating decision makers to make choices taking into account their preferences. We were not limited to a specific research area, and included works in the context of decision theory, artificial intelligence, constraint programming, databases and semantic web. Each of these works was presented following an evaluation framework, which facilitates their comparison. Our review allows understanding not only each individual work but also, with our discussion, how they are related, which kind of issues they typically address and their limitations.

This study shows that a main issue related to reasoning about preferences is dealing with trade-off situations. Both utility functions and (conditional) qualitative statements can capture them, but it is difficult to express this kind of preferences in a compact form. For instance, if additive utility independence is not identified among attributes, each possible outcome will have a particular utility that cannot be derived from individual attribute values. And, in the case of qualitative statements, each full assignment for attributes must be compared. Moreover, users typically do not provide these kind of preferences, because they are constructed just when users face a concrete decision making situation (Lichtenstein and Slovic 2006).

Moreover, many of the presented approaches are able to reason only about a

restricted set of preferences, thus constraining users while expressing preferences. Dealing with heterogeneous types of preferences, as those of our preference metamodel, which include natural-language-like expressions, is not a trivial task. In next chapter, we present a decision making technique to address these issues of existing work.

6
An Automated Decision Maker with User-centric Principles

In order to tackle the problems of the existing approaches to reasoning about preferences discussed in the introduction and in the previous chapter, we present in this chapter a novel technique for automated decision making based on preferences and available options. This technique is able to handle qualitative preferences expressed in a high-level language and incorporates psychological processes to simulate human decision to resolve trade-offs. Our technique thus chooses one option from a finite set available, based on user preferences that have natural-language-like expressions, such as expressive speech acts (e.g. *like*, *accept* or *need*).

Section 6.1 describes the scope of decisions we are addressing and the assumptions of this work. Section 6.2 presents the high-level preference language that our technique is able to process, which is based on our preference metamodel. We also introduce a running example that exemplifies the use of this language and that will be used throughout the chapter. Next, Section 6.3 describes our technique, whose steps are detailed from Section 6.4 to 6.7. We compare the technique with existing work and present its evaluation in Section 6.8, concluding in Section 6.9.

6.1
Scope and Assumptions

As introduced earlier, our goal is to provide users with an automated decision making technique, which is able to make decisions on their behalf so as to automate their tasks. As decisions may be characterised in many different ways, we define in this section the *scope* of the decision problems we are dealing with. A decision problem consists of choosing one option o based on preferences from a finite set of available options, Opt, where all $o' \in Opt$ are of the same concept, for example a set of *laptops* or a set of *hotels*. Each concept is associated with a finite set of attributes, Att, and each $a_i \in Att$ is associated with a domain D_i, which establishes the values allowed for that attribute. Each domain D_i: (i) consists of a set of values x_{ij}; (ii) can be *discrete* or *continuous*; and (iii) can be *ordered* or *unordered*. For example, real numbers are an ordered continuous domain, integers are an ordered discrete domain, and colours are a unordered discrete domain. We refer to domains

composed of numbers as *numeric*. This way of describing options is a *restricted view* of our ontology model presented in Chapter 3. We left out of the scope of our technique the ability of handling: (i) *adjectives*; (ii) *scales*; and (iii) *proxy attributes*.

Our focus is on decisions that users are able to make themselves, but as these decisions may involve a large number of options, are tedious, and possibly repetitive for users, they prefer to delegate them instead of demanding time and effort in their execution — and they want the system to choose the option they would choose themselves. It is important to highlight that this is different from the goal of MAUT (Keeney and Raiffa 1976), which is a classical approach that was proposed to help people to make *critical* decisions with conflicting preferences, and requires user interaction to identify a function that represent preferences quantitatively.

Given the scope we are addressing, we now detail our *assumptions*. First, users have a set of preferences over the problem that the decision has to be made, i.e. their known preferences. Second, we consider a *unitary decision maker*, that is, provided preferences are given by a single user, and the goal is to increase the satisfaction of this user with the choice. Third, we assume a *consistent* set of preferences. User may provide conflicting preferences, e.g. "I prefer higher quality and lower price," but they are not inconsistent, such as "I prefer A to B, and I prefer B to A." Finally, available options, attributes and their domains are *given* and are, therefore, inputs of our technique, together with user preferences.

6.2 Preference Language and Running Example

As discussed in our study of how humans express preferences (Chapter 2), there are different forms in which they do so, and our goal is to provide them with a language in which they can inform their preferences in a way as close as possible to natural language. Based on our preference metamodel (Chapter 3), we derived a high-level preference language, whose EBNF is presented in Table 6.1, which includes seven types of preferences and means of specifying priorities among attributes and preferences. While constraints, qualifying and rating are monadic preferences, goals, orders and indifferences are dyadic.

The different types of preferences in our language have the same meaning of the corresponding types in our preference metamodel. However, there are some restrictions to the metamodel, which are constructions left unaddressed by our technique. These restrictions are associated with preference targets that cannot be used. We next present them, and also discuss how these limitations could be addressed. We highlight that most of these restrictions, and also those related to preference targets mentioned above, can be held at a higher level of abstraction: features in a user interface can allow users to express preferences that are translated

$$
\begin{aligned}
\mathit{preference} &::= [\mathit{condition}]\,(\mathit{constraint} \mid \mathit{qualifying} \mid \mathit{rating} \mid \mathit{goal} \\
&\quad \mid \mathit{order} \mid \mathit{indifference} \mid \mathit{dontCare}) \\
\mathit{condition} &::= \textbf{if}\ \mathit{formula}\ \textbf{than} \\
\mathit{formula} &:: \mathit{expression} \mid \mathit{formula}\ \textbf{and}\ \mathit{formula} \\
&\quad \mid \mathit{formula}\ \textbf{or}\ \mathit{formula} \mid \textbf{not}\ \mathit{formula} \\
\mathit{expression} &::= \mathit{attribute}\ (= \mid \neq \mid > \mid \geq \mid < \mid \leq)\ \mathit{value} \\
\mathit{constraint} &::= \mathit{formula} \\
\mathit{qualifying} &::= \mathit{expressive\_speech}_a\ \mathit{ct}\ \mathit{formula} \\
\mathit{rating} &::= \mathit{formula}\ \mathit{rate} \\
\mathit{goal} &::= (\textbf{minimise} \mid \textbf{maximise})\ \mathit{attribute} \\
\mathit{order} &::= \mathit{attribute} = \mathit{value} \succ \mathit{attribute} = \mathit{value} \\
\mathit{indifference} &::= \textbf{indifferent}\ \mathit{formula}\ \{\mathit{formula}\} \\
\mathit{dontCare} &::= \textbf{dont\_care}\ \mathit{attribute} \\
\mathit{expressive}_s\mathit{peech}_a\,\mathit{ct} &::= [\textbf{don't}]\,(\textbf{prefer} \mid \textbf{need} \mid \textbf{desire} \mid \textbf{avoid} \mid \textbf{like} \mid \textbf{want} \\
&\quad \mid \textbf{accept} \mid \textbf{require} \mid \textbf{love} \mid \textbf{hate}) \\
\mathit{rate} &::= \textbf{best} \mid \textbf{very\_good} \mid \textbf{good} \mid \textbf{neutral} \mid \textbf{bad} \\
&\quad \mid \textbf{very\_bad} \mid \textbf{worst} \\
\mathit{priority} &::= [\mathit{condition}]\,(\mathit{attribute\_priority} \\
&\quad \mid \mathit{attribute\_indifference} \mid \mathit{preference\_priority}) \\
\mathit{attribute\_priority} &::= \mathit{attribute} \rhd \mathit{attribute} \\
\mathit{attribute\_indifference} &::= \mathit{attribute} \sim \mathit{attribute} \\
\mathit{preference\_priority} &::= \mathbb{Z}.\ \mathit{preference}
\end{aligned}
$$

Table 6.1: Preference language.

to a set of preferences in our language.

- *Preferences over enumeration values.* Referring to an enumeration value, such as *I prefer red*, is to establish preferences over values of an enumeration in a generic way (e.g. the *colour* preference indicates that I prefer *red cars*, *red t-shirts* or *red coats*), and this is not allowed. However, our restricted language still allows users to express preferences over enumeration values, but preferences should be expressed in the context of each specific attribute, as they cannot be expressed generically.
- *Preferences over concepts*. Preferences over concepts, such as "*I prefer LCD to CRT monitors,*" are also not allowed. Preferences over concepts indicate a *is-a* relationship ("*LCD is-a monitor*"). Thus, in order to support concepts, the language would have to be extended with a new comparison operator (*is-a*) to be used in expressions.
- *Preferences over instances.* An instance represents a particular entity, which can be defined by assigning a value to each attribute of the entity class. Therefore, an instance can be represented in our restricted language with a

constraint preference, which makes a value assignment for an attribute that is used as an identifier.

– *Each preference is associated with a single attribute.* We restrict preferences to refer to only one attribute (this restriction is not extended to conditions). As a consequence, propositional formulae of constraints cannot refer to different attributes, e.g. "*I prefer a laptop with a 15" screen and integrated camera.*"

– *Order statements refer only to equality expressions.* Even though order preferences can refer only to equality expressions, some of the forbidden targets can be expressed in a different allowed manner, e.g. "*I prefer a laptop with a 14" or 15" screen to one with a 17" screen*" can be expressed with two preferences: "*I prefer a laptop with a 14" screen to one with a 17" screen*" and "*I prefer a laptop with a 15" screen to one with a 17" screen*."

In order to illustrate our high-level preference language, we introduce an example in this section. Throughout this chapter, this example will also be used to illustrate different parts of our decision making technique. Suppose *Bob* is visiting a university, and needs to choose an apartment at which to stay. Each apartment is described in terms of *seven* attributes, described below, each associated with a domain.

1. *uni*: $\{x \mid x \in \mathbb{R}, 0 < x \leq 15\}$ — the distance, in kilometres, from the apartment to the university.
2. *station*: $\{x \mid x \in \mathbb{R}, 0 < x \leq 1.2\}$ — the distance, in kilometres, from the apartment to the closest underground station.
3. *market*: $\{x \mid x \in \mathbb{R}, 0 < x \leq 0.7\}$ — the distance, in kilometres, from the apartment to the closest supermarket.
4. *zone*: $\{x \mid x \in \mathbb{Z}, 1 \leq x \leq 6\}$ — the zone where the apartment is located. The underground coverage in the city in which the university is located is split into six zones. Zone 1 is the city centre, and the higher the zone number is, the farther it is from the centre.
5. *brand*: { A, B, C, D } — each apartment has a brand, associated with the company it belongs to.
6. *stars*: $\{x \mid x \in \mathbb{Z}, 1 \leq x \leq 5\}$ — a number that represents the apartment quality; the higher, the better.
7. *price*: $\{x \mid x \in \mathbb{R}, 95 \leq x \leq 125\}$ — the price of renting the apartment (per week).

Bob's preferences are shown as follows, using our preference language, numbered by their priority (1...15), and with the final line being an attribute priority.

Table 6.2: Available apartments.

Apartment	brand	market	price	stars	station	uni	zone
Ap_A	A	0.45 Km	£100	2	0.3 Km	5.0 Km	2
Ap_B	D	0.40 Km	£115	3	0.6 Km	2.2 Km	1
Ap_C	B	0.20 Km	£95	2	0.3 Km	10.0 Km	3
Ap_D	B	0.60 Km	£105	2	0.5 Km	6.0 Km	2
Ap_E	B	0.30 Km	£100	3	0.5 Km	3.5 Km	2
Ap_F	C	0.40 Km	£125	4	0.9 Km	2.0 Km	1

1. **don't accept** $zone > 2$
2. **prefer** $uni \leq 2.5Km$
3. **if** $uni \leq 2.5Km$ **then need** $station \leq 1Km$
4. **if** $uni \leq 2.5Km$ **then prefer** $station \leq 0.7Km$
5. **if** $uni \leq 2.5Km$ **then require** $price \leq$ £125
6. **if** $uni > 2.5Km$ **then need** $station \leq 0.7Km$
7. **if** $uni > 2.5Km$ **then require** $price \leq$ £105
8. **minimise** $station$
9. **minimise** $market$
10. **minimise** $price$
11. **prefer** $brand = A$ **or** $brand = B$ **or** $brand = C$
12. $brand = A \succ brand = B$
13. $brand = B \succ brand = C$
14. $stars = 2$ **good**
15. **maximise** $stars$

    – **if** $uni > 2.5Km$ **then** $station \rhd uni$

These preferences and priorities are used to make a choice on Bob's behalf. The decision problem is to choose one apartment from the available options, shown in Table 6.2.

6.3 Technique Overview

The goal of our decision maker is to simulate human reasoning in making decisions, allowing us to exploit natural user expressiveness of preferences (without the need for elicitation methods) and resolve trade-offs (that cannot be resolved with the provided preferences) in a way humans would do. As a result, we propose a decision making technique that is based on heuristics investigated in psychology that explain how people make decisions. More specifically, the overall process is inspired by *reason-based choice* (Shafir et al. 1998), which discusses that people make decisions by identifying reasons to accept and reject options, and

incorporates two principles (Simonson and Tversky 1992): *extremeness aversion* (avoiding extreme options, which are those that compromise too much an attribute because of another that provides a gain), and *trade-off contrast* (influencing the preference between two options with the cost-benefit relationship of all options).

We introduce our technique by showing the different steps, process and data that comprise it, presented in Figure 6.1. We make three main observations on this figure. First, it can be seen that monadic preferences, which indicate preferences with expressive speech acts or rates, are used in many processes of our technique, showing that the technique is *driven by these natural-language expressions*. Second, the technique has *variable parts*: as our technique use a particular interpretation of natural language expressions, this interpretation can differ in different applications. Moreover, as we discuss later, during the decision making process our technique calculates quantitative costs of options based on qualitative preferences, and different functions associated with this calculation can be adopted. Based on experimentation, we selected particular instances (adopted in this thesis) for these parts. Third, our technique is composed of *four main steps*, explained next.

Pre-processing. Our preference language allows the expression of heterogeneous types of preferences, and an integrated view that shows how they interact and how they evaluate individual option attributes is helpful to make a choice. The pre-processing step involves building computational models that compile information given by different preferences provided by users and represent options in a way that their positive and negative aspects are made explicit (according to those preferences). The first is the Preference Satisfaction Model (PSM), which combines the information given by monadic preferences, and the second, the Options-Attribute Preference Model (OAPM), indicates which attribute value is better considering two options.

Explication. Sometimes a preference provided by a user implies a further preference in addition to its literal meaning. For example, preference 14 may indicate that not only the preference is for 2-star apartments, but the closer the number of stars of the apartment is to 2 the better. So, in the explication step we consider *implicit* preferences that we can extract from the preferences explicitly given by users, and based on this information we update the previously produced OAPM.

Elimination. When people make a choice from a set, they first eliminate options that have no advantage when compared to another, i.e. a *dominated* option, or attributes with unacceptable values, i.e. non-compensatory attributes. In the elimination step, we discard these two kinds of options. While the OAPM

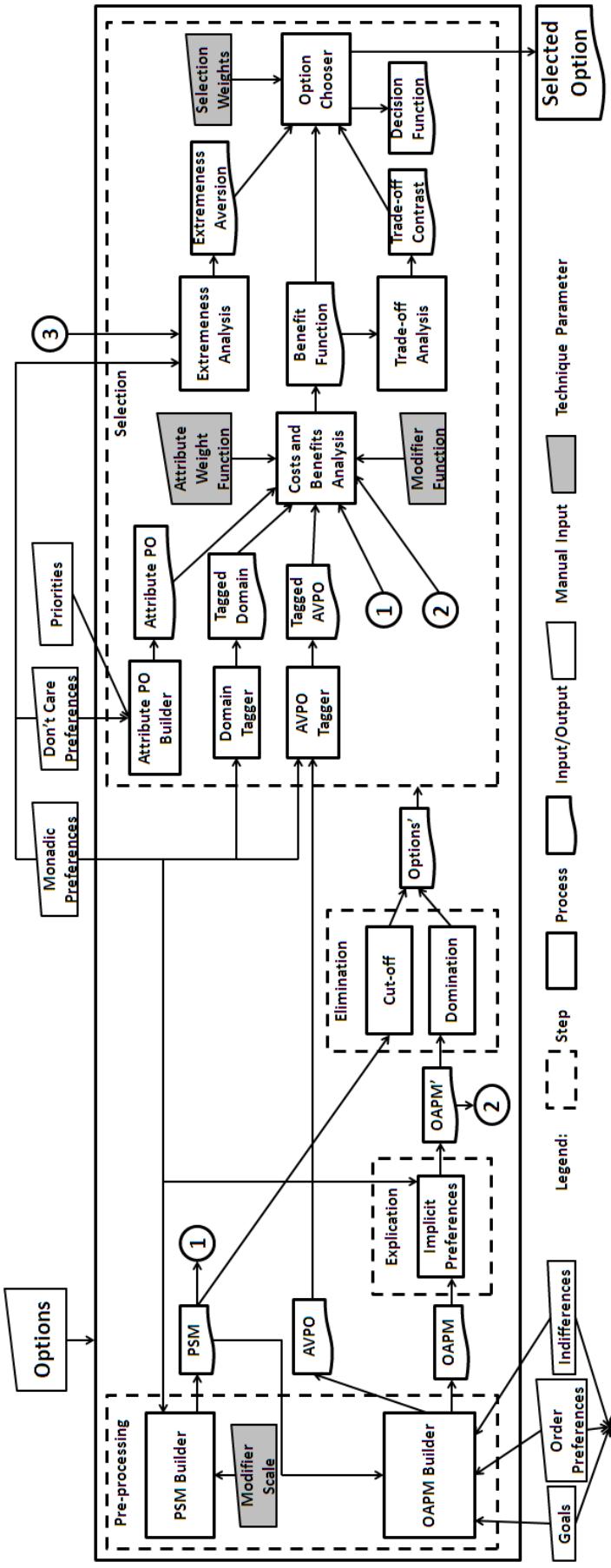


Figure 6.1: Technique Overview.

allows detecting dominated options in an easy way, as it shows the positive and negative aspects of each option compared to another, the PSM exposes options that do not satisfy hard constraints. Hard constraints in our technique are expressed using specific modifiers: *require*, *need*, *hate* and *don't accept* (this is subject to particular interpretations).

Selection. After eliminating the options above, we obtain a *consideration set*, which contains options that require trade-off resolution to make a choice. In order to make this selection, we first analyse option costs and benefits, by using the information compiled in our computational models to calculate an option costs with respect to another for each attribute. The overall costs of an option (w.r.t. another) is then a weighted sum of these individual attribute costs, by considering the provided priorities. Next, the trade-off between options and how these options compensate advantages with disadvantages are analysed (which are related to the trade-off contrast and extremeness aversion principles), and these factors are them combined with the previously calculated option costs.

It is important to highlight that, as opposed to many existing approaches, our technique does not focus on isolating two options and comparing them, as we use the whole set of options to evaluate preference between any two options. Our technique results in a partially ordered set, organised in four different levels, as described below.

1. **chosen option**, which is considered the optimal option;
2. **acceptable options**, which are in the consideration set, but were not chosen;
3. **eliminated options**, which were discarded because of non-compensatory attribute(s); and
4. **dominated options**.

We further make an observation on how we interpret provided user preferences. As it can be seen in our systematic review (Chapter 5), the most common interpretation of preferences adopted by approaches for reasoning about preferences is *ceteris paribus* (all other things being equal or held constant) (Hansson 1996). Under this interpretation, a preference that compares values of one or more attributes is taken into account only for comparing two options whose attribute values that are not targets of the preferences are equal. For example, the preference "*I prefer silver cars to white cars*" is considered only for comparing two cars that have the same brand, power, etc., i.e. everything else but colour has to be equal. This interpretation is very limited, because the common scenario is that options differ in more than one, if not several, attributes.

We do not adopt a ceteris paribus interpretation, but we are careful while using preferences provided by users. When users provide preferences, they consider attributes in isolation, and when they are used for comparing two options, the typical scenario is that preferences conflict with each other. In these cases, a trade-off must be resolved, which is typically done by users only while facing concrete decision making situation (Lichtenstein and Slovic 2006). And for resolving these trade-offs that emerge from conflicting preferences, our technique uses a psychology-inspired approach, as discussed above.

We describe next the steps of our technique. We first describe the computational models that are built in the pre-processing step of our technique (Section 6.4), which are later refined with implicit preferences (Section 6.5). These computational models are used to eliminate options (Section 6.6) and to choose an option (Section 6.7).

6.4 Pre-processing

In our approach, the first step to make a decision is to pre-process the available options and analyse them according to the preferences provided by users. As previously introduced, there are monadic and dyadic preferences, and as the former have only a single referent and the latter allow making a pairwise comparison, we process them separately, building two models based on them — the Preference Satisfaction Model (PSM) (Section 6.4.1) and the Options-Attribute Preference Model (OAPM) (Section 6.4.2). These computational models do not allow to conclude which available option is the "best" or decide which of two options is better, but they expose positive and negative aspects of available options, integrating the information provided by heterogeneous types of preferences.

6.4.1 Preference Satisfaction Model

The first model, named *Preference Satisfaction Model (PSM)*, consists of a table that captures how options satisfy preferences in terms of each attribute according to monadic preferences. This table associates option attributes with an expressive speech act or a rate (or their negation), meaning that the preference for an attribute value of a particular option is represented by a specific expressive speech acts or rate (e.g. the $price$ of option Ap_A satisfies ***require***). $e \in ExpressiveSpeechAct$ is an expressive speech act that comes from qualifying preferences, while $r \in Rates$ is a rate (e.g. *love* and *hate*) that comes from rating preferences. Expressive speech acts and rates are collectively referred to as ***modifiers*** ($M = ExpressiveSpeechActs \cup Rates$). Constraint preferences, which

are associated with no modifier, are considered to be associated with an implicit modifier, namely *want*. All these preferences are referred to as monadic preferences ($MP = Constraints \cup QualifyingPreferences \cup RatingPreferences$). The PSM is defined as follows.

Definition 6.1 *The **Preference Satisfaction Model (PSM)** is a partial map from a pair $\langle option, attribute \rangle$ to a modifier or its negation, indicating the most representative modifier that indicates preference for an attribute value of an option.*

$$PSM : Opt \times Att \mapsto \{\epsilon, \neg\} \times M$$

Before describing the PSM construction in detail, we introduce auxiliary functions. Each constraint, qualifying or rating preference p is characterised by (i) a modifier, $mod(p)$, e.g. ***need***; (ii) a formula, $form(p)$, e.g. $station \leq 1$; and (iii) optionally a condition, $cond(p)$, e.g. $uni \leq 2.5$ (examples are given considering preference 3). As we restrict preferences to refer to a single attribute, $att(p)$ is the attribute that is the referent of the preference, e.g. $station$. An option may or may not satisfy a constraint, and $sat(formula, o)$ replaces variables from the provided $formula$ with attribute values from option o and evaluates the formula for a boolean value, e.g. $sat(station \leq 1, Ap\_B) = true$. It is used to check either whether a preference is applicable to an option, i.e. the preference condition is satisfied by the option, or whether the option satisfies the preference itself. Given this notation, we define when a preference is applicable to an option.

Definition 6.2 *A preference p is applicable to an option o, $App(p, o)$, if and only if*

$$\nexists cond(p) \vee (\exists\, cond(p) \wedge sat(cond(p), o))$$

For example, preferences 3, 4 and 5 are applicable only to options Ap_B and Ap_F. Therefore, for each option, there is a subset of monadic preferences that is applicable to it, and each of them is related to a particular attribute. We can thus associate a set of modifiers (or their negation) with each option attribute — $AttMod(MP, o, a)$ — as shown below.

$$\begin{aligned} &AttMod(MP, o, a) ::= \\ &\quad \{\langle \epsilon, m \rangle \mid m = mod(p) \wedge p \in MP \wedge a = att(p) \wedge App(p, o) \wedge sat(form(p), o)\} \\ &\quad \cup \{\langle \neg, m \rangle \mid m = mod(p) \wedge p \in MP \wedge a = att(p) \wedge App(p, o) \wedge \neg sat(form(p), o)\} \end{aligned}$$

$AttMod(MP, o, a)$ may be empty. In the case of Ap_F and attribute $station$, for example, we thus have the following attribute modifiers.

$$AttMod(MP, Ap\_F, station) = \{\langle \epsilon, \textbf{need} \rangle, \langle \neg, \textbf{prefer} \rangle\}$$

Now that we have available all modifiers that indicate preference for a particular attribute of each of the options, we wish to select the most representative one. Expressive speech acts and rates, widely used by people, have an interpretation that is subjective and specific for each individual. Although they may have different meanings, such as expressing requirement or acceptance, all modifiers also express a degree of preference. Modifiers are categorised as *positive*, indicating a preference for an attribute value; *neutral*, indicating indifference and acceptance for an attribute value; and *negative*, indicating a preference against an attribute value. In addition, modifiers of each category can be stronger relative to each other. For example, consider the following two preferences: "*I need an apartment whose price is lower than £125*" and "*I prefer an apartment whose price is lower than £100.*" *Need* is stronger in the sense that it tells what *has* to be satisfied. However, when we have two options, the first satisfying only what is needed (e.g. an apartment that costs £110) and the second satisfying what it is needed and preferred (e.g. an apartment that costs £90), *the degree of preference* of the second is stronger than the degree of preference of the first.

We adopt a particular ranking that captures this notion to indicate the degree of preference of modifiers, namely *modifier scale*, presented in Figure 6.2, and each subset of modifiers that represents (according to this particular scale) the same degree of preference is associated with an index, which will be used later. The interpretation of modifiers is subjective, and therefore the modifier scale is one of the variation points of our technique, as indicated in Figure 6.1 — it can be instantiated in different ways for individual applications, or even customised to individual users. As discussed before, some of the modifiers indicate hard constraints, and those that are positive modifiers (*require* and *need*) are considered less strong than the other positive modifiers for the reasons above. Moreover, the modifiers that indicate hard constraints are in the same modifier scale than the others for modularity reasons: hard constraints are relevant only for eliminating options, so distinguishing modifiers that indicate them from the others is irrelevant in the other steps of the decision making process.

If more than one monadic preference applies to an option attribute, we use the adopted modifier scale to choose among them as the most representative. There are two possibilities. The first case is when there is at least one monadic preference whose formula is satisfied. According to the modifier scale, from the

Category	Modifier	Index
positive	want	9
	love, best	8
	desire, very_good	7
	prefer, like, good	6
	need	5
	require	4
neutral	accept, don't require, don't avoid	3
	neutral, don't love, don't hate	2
	don't need, don't desire	1
negative	don't prefer	-4
	avoid,	-5
	don't like, bad	-6
	don't want, very_bad	-7
	hate, worst	-8
	don't accept	-9

Figure 6.2: Modifier scale.

neutral modifiers (*don't need*, *don't desire*) to the most positive (*want*) there is a stronger preference for an attribute value; and from the neutral modifiers (*accept*, *don't require*, *don't avoid*) to the most negative (*don't accept*) there is a stronger preference against an attribute value. In case there are both positive and negative modifiers, there is inconsistency, which is not taken into account by our technique. Therefore, the strongest modifier from those satisfied (i.e. $\langle \epsilon, m \rangle$) is chosen.

The second case is when there is no monadic preference whose formula is satisfied. Again, there are two possibilities. If there is at least one (not satisfied) monadic preference whose modifier is positive, the weakest one is chosen. For example, if one option attribute is associated with $\langle \neg, \textbf{prefer} \rangle$ and $\langle \neg, \textbf{require} \rangle$, the second is selected, meaning that not even the weakest preference is satisfied. In case there is no monadic preference whose modifier is positive, then the weakest one is chosen (where the weakest is *accept*, *don't require* and *don't avoid*, and the strongest is *don't accept*). Given this informal description of how modifiers are selected, we present Algorithm 1, which describes how the PSM is built using the modifier indices. The PSM of the presented apartment decision problem built according to the proposed algorithm is shown in Table 6.3.

With this model one can see pros and cons against each available option. Even though one of the options might have only positive values, such as *like* and *require*, it does not mean it is the best option, because it may have only minimum acceptable values, and the trade-off with other options might indicate that another option is better. Moreover, other preferences, not processed yet, provide additional information, and this is what we will consider next.

Algorithm 1: PSM Builder

Input: MP: monadic preferences; Opt: options; Att: attributes; $scale$: modifier scale
Output: PSM: preference satisfaction model

foreach *Option* $o \in O$ **do**
  foreach *Attribute* $a \in A$ **do**
    $x \leftarrow null$;
    $m^* \leftarrow null$;
    foreach $\langle \epsilon, m \rangle \in AttMod(MP, o, a)$ **do**
      if $m^* == null \vee$ |IndexOf($m, scale$)|>|IndexOf($m^*, scale$)| **then**
        $x = \epsilon$;
        $m^* \leftarrow m$;
    if $m^* == null$ **then**
      foreach $\langle \neg, m \rangle \in AttMod(MP, o, a) \wedge Positive(m)$ **do**
        if $m^* == null \vee$ IndexOf($m, scale$) < IndexOf($m^*, scale$) **then**
          $x = \neg$;
          $m^* \leftarrow m$;
    if $m^* == null$ **then**
      foreach $\langle \neg, m \rangle \in AttMod(MP, o, a)$ **do**
        if $m^* == null \vee$ IndexOf($m, scale$) > IndexOf($m^*, scale$) **then**
          $x = \neg$;
          $m^* \leftarrow m$;
    $PSM[o, a] \leftarrow \langle x, m^* \rangle$;
return PSM;

Table 6.3: PSM of the Apartment Decision Problem.

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle \neg, prefer \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \neg, prefer \rangle$	$\langle \neg, prefer \rangle$	$\langle \neg, prefer \rangle$	$\langle \epsilon, prefer \rangle$
station	$\langle \epsilon, need \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \epsilon, need \rangle$	$\langle \epsilon, need \rangle$	$\langle \epsilon, need \rangle$	$\langle \epsilon, need \rangle$
market						
zone	$\langle \neg, don't\ accept \rangle$	$\langle \neg, don't\ accept \rangle$	$\langle \epsilon, don't\ accept \rangle$	$\langle \neg, don't\ accept \rangle$	$\langle \neg, don't\ accept \rangle$	$\langle \neg, don't\ accept \rangle$
brand	$\langle \epsilon, prefer \rangle$	$\langle \neg, prefer \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \epsilon, prefer \rangle$
stars	$\langle \epsilon, good \rangle$	$\langle \neg, good \rangle$	$\langle \epsilon, good \rangle$	$\langle \epsilon, good \rangle$	$\langle \neg, good \rangle$	$\langle \neg, good \rangle$
price	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$

6.4.2 Options-Attribute Preference Model

The second model that will aid us in the decision making process, namely Options-Attribute Preference Model (OAPM), is a table that captures comparison relationships between two options, from a perspective of individual attributes. This model shows for which attributes an option is better or similar to another — or no conclusion can be made with the provided preferences. For each OAPM value, which compares an option o to an option o' with respect to an attribute a, there are four possible preference values:

(i) $+$: the attribute value of o is better than o', i.e. $o \succ_a o'$;
(ii) $-$: the attribute value of o is worse than o', i.e. $o' \succ_a o$;
(iii) $\sim$: the attribute value of o is as preferred as o', i.e. $o \sim_a o'$;
(iv) ?: no conclusion can be reached.

The preference values that associate attribute values of two options are derived from provided user preferences. So, besides storing this information, the OAPM also keeps track of the reason for a preferred value. The OAPM is thus as follows.

Definition 6.3 *The **Options-Attribute Preference Model (OAPM)** is a total map from $\langle option_1, option_2, attribute\rangle$ to a preference value, indicating which attribute value of these options is the preferred one, and a reason that indicates the (explicit or implicit) preference that lead to this conclusion.*

$$OAPM : Opt \times Opt \times Att \mapsto \{+, -, \sim, ?\} \times Reason$$

The possible values of *Reason* can be a preference (e.g. maximisation goal), the PSM, or an implicit preference, specifically: *psm*, *max*, *min*, *avpo*, *indiff*, $\langle upper, p\rangle$, $\langle lower, p\rangle$, $\langle around, p\rangle$, and $\langle interval, p\rangle$ (they will be later described). For example, as preference 9 indicates that Ap_B is better than Ap_A with respect to the attribute $market$, the OAPM values are: $OAPM[Ap\_A, Ap\_B, market]$ = $\langle -, min\rangle$ and $OAPM[Ap\_B, Ap\_A, market]$ = $\langle +, min\rangle$ (the reason for the OAPM value is a minimisation goal). The initial OAPM state consists of all values set to ? and therefore, unless there are preferences that compare two attribute values, no conclusion is reached.

Note that the OAPM value $OAPM[o_1, o_2]$ is dual to $OAPM[o_2, o_1]$. The specification of our technique sets and uses both OAPM values to make it easier to understand, but its implementation can be optimised by representing just one of the values.

In the pre-processing step, OAPM values are set based on two kinds of information: (i) goal, order and indifference preferences and (ii) the previously built PSM. This information is processed separately in a specific order — PSM, goals, order preferences and indifference preferences — which makes the relationship between two attribute values established by a subsequently processed preference possibly override the current information present in an OAPM value. This is because a user may have general preferences for an attribute, but also have preferences for specific cases, such as stating that an attribute should be minimised and then providing order preferences for specific attribute values. In addition, preferences may refine other preferences, for instance, according to one preference a set of attribute values are considered equally preferred (e.g. preference 11), and specific preferences establish an order among the preferred values (e.g. preferences 12 and 13).

The next sections describe how the OAPM is constructed, in a declarative way. Presented formulae are representation of rules, which indicate the values to be set in

Table 6.4: Index used to compare PSM values.

PSM value	PSM Index
$\langle \epsilon, modifier \rangle$	$IndexOf(modifier, scale)$
$\langle \neg, negative\ modifier \rangle$	0
$\langle \neg, neutral\ modifier \rangle$	-1
$\langle \neg, positive\ modifier \rangle$	-2

the model. Rules are applied sequentially (following the order they are presented) for all pairs of options and attributes, and a subsequent rule may override the values set by a previous applied rule.

PSM

Monadic preferences in isolation do not allow us to compare attribute values, but, with the PSM, these preferences are situated in a context, and we can conclude that a value that is considered *best* is better than a value that is *good*, for instance. This idea is investigated by Hansson (Hansson 1990), who discusses the interpretation of "*good*" and "*bad*" in terms of "*better*." Our modifier scale, initially used to select most representative modifiers, is now used to compare modifiers associated with different options.

Note that the indices presented in Figure 6.2 have a gap between negative and neutral modifiers. Besides adjusting indices to give opposite index values to positive and negative modifiers, this gap is used to associate indices with unsatisfied modifiers, i.e. $\langle \neg, m \rangle$. When there is no satisfied modifier for an attribute value, we have three situations: (i) $\langle \neg, positive\ modifier \rangle$, there is one or more positive modifiers to evaluate that attribute value, but it does not satisfy them; (ii) $\langle \neg, neutral\ modifier \rangle$, there is no unsatisfied positive modifier, but there is one neutral that is unsatisfied; (iii) $\langle \neg, negative\ modifier \rangle$, there is no unsatisfied positive or neutral modifier, but the attribute value also does not satisfy a negative one. Situation (iii) is better than (ii), which is better than (i). When there is a satisfied modifier associated with each of two attribute values being compared — i.e. for both, the PSM value is $\langle \epsilon, modifier \rangle$ — the strongest modifier indicates the preferred value (or equally preferred if both options have the same degree of preference according to the scale), i.e. the one with the highest index. We thus use the indices of the modifier scale with additional ones (those shown in Table 6.4) to compare attributes values. Again, this raking is a variable point of our technique, and subject to different interpretations.

With these additional indices, we now choose the PSM value associated with the highest index. This way of stating which attribute value is better causes satisfied positive and neutral modifiers to be better than any other unsatisfied one, and satisfied negative modifiers worse than any other unsatisfied one. This

interpretation is adopted because users typically explicitly state what they want or do not want, being the absence of preference an indifference (weaker than the provided indifference), as people usually remember how the experiences felt when they were at their peak (best or worst) (Schwartz 2005).

Given this approach of establishing a preference relationship between attribute values based on (un)satisfied modifiers, i.e. PSM values, we show the rules used to set the OAPM values, which are applicable only to PSM values that are not null. $PSMIndex(PSM[o, a], scale)$ returns the index of the PSM value according to Table 6.4. Remember that the OAPM value is composed of a preference value (+, −, ∼, ?) *and* a reason.

$$PSMIndex(PSM[o_1, a], scale) = PSMIndex(PSM[o_2, a], scale) \rightarrow OAPM[o_1, o_2, a] = \langle \sim, psm \rangle \tag{6-1}$$

$$PSMIndex(PSM[o_1, a], scale) > PSMIndex(PSM[o_2, a], scale) \rightarrow OAPM[o_1, o_2, a] = \langle +, psm \rangle \wedge OAPM[o_2, o_1, a] = \langle -, psm \rangle \tag{6-2}$$

With respect to $station$, Ap_B is thus considered better than Ap_F, as the $PSMIndex$ associated with $\langle \epsilon, prefer \rangle$ (PSM value of $Ap\_B, station$) is 6, while the $PSMIndex$ of $\langle \epsilon, need \rangle$ is 5 (PSM value of $Ap\_F, station$).

Goals

The next set of preferences that is processed is goals, which is restricted to attributes whose domain is ordered, and indicates that an attribute value is considered better when its value is higher (maximisation goals) or lower (minimisation goals) than another. The rules used to set (or change) OAPM values, which are shown below, use two additional functions: (i) $type(goal)$, which is max when the goal is a maximisation, and min when it is a minimisation; and (ii) $val(o, a)$, which returns value of the attribute a of option o.

$$\exists\, g.(g \in Goal \wedge att(g) = a \wedge App(g, o_1) \wedge App(g, o_2) \wedge val(o_1, a) = val(o_2, a) \rightarrow OAPM[o_1, o_2, a] = \langle \sim, type(g) \rangle) \tag{6-3}$$

$$\begin{aligned} &\exists\, g.(g \in Goal \wedge att(g) = a \wedge App(g, o_1) \wedge App(g, o_2) \\ &\wedge ((type(g) = max \wedge val(o_1, a) > val(o_2, a)) \\ &\vee (type(g) = min \wedge val(o_1, a) < val(o_2, a))) \rightarrow \\ &OAPM[o_1, o_2, a] = \langle +, type(g) \rangle \wedge OAPM[o_2, o_1, a] = \langle -, type(g) \rangle) \end{aligned} \tag{6-4}$$

For example, because of preference 9, the OAPM values associated with

Ap_C and $market$ are set to $\langle +, min \rangle$ with respect to all other options.

Order Preferences

We now proceed to order preferences, which are those that establish an order among two attribute values by explicitly stating the preferred value. Order preferences are transitive, e.g. with preferences 12 and 13 we can derive that brand A is preferred to C. Therefore, from order preferences, we can derive a partial order of attribute values, namely *attribute value partial order (AVPO)*. However, as order preferences may be valid only according to a given condition, two different options may satisfy the conditions of different order preferences. Therefore an AVPO is constructed only with preferences applicable to an option and is specific to a pair of option and attribute. The order preferences of our example have no condition, thus the same partial order (AVPO) is built for all the options with respect to the brand attribute.

An AVPO is a forest (or possibly a tree) $\langle N, A \rangle$, where N is a set of nodes and A is a set of arrows that link nodes. Each node consists of expressions of order preferences, in the form of $attribute = value$ (e.g., $brand = A$), and an arrow from a node to another represents that the source node is preferred to the sink node. Algorithm 2 shows how an AVPO is constructed for a particular option and attribute. If the output is not a forest, there is a case of inconsistency, which is out of scope. In our example, the order preferences lead to the following AVPO for the $brand$ attribute for all options: $brand = A \rightarrow brand = B \rightarrow brand = C$.

Algorithm 2: AVPO Builder

Input: o: option; a: attribute; $Order$: order preferences
Output: $AVPO$: $\langle N, A \rangle$

1 Set⟨AVPONode⟩ $N \leftarrow \emptyset$;
2 Set⟨Arrow⟩ $A \leftarrow \emptyset$;
3 **foreach** *Order preference* $op \in Order$ **do**
4 **if** $\texttt{App}(op, o) \wedge \texttt{att}(op) = a$ **then**
5 $N \leftarrow N \cup \{ \texttt{LHS}(op), \texttt{RHS}(op) \}$;
6 $A \leftarrow A \cup \{\langle \texttt{LHS}(op), \texttt{RHS}(op) \rangle\}$;
7 **return** $\langle N, A \rangle$;

An attribute value of an option is preferred to another according to an AVPO if there is a path from the first to the second (typical tree algorithms are used (Cormen et al. 2001)), for which we use the notation $ExistsPath(AVPO[o, a], val_1, val_2)$, where val_1 matches a node whose expression is $attribute = val_1$. As different AVPOs may establish different preference relationships, we consider an attribute value of option o_1 better than that of option o_2 if there is a path in the AVPOs of both options, as shown in the next rule.

$$ExistsPath(AVPO[o_1, a], val(o_1, a), val(o_2, a)) \\ \wedge ExistsPath(AVPO[o_2, a], val(o_1, a), val(o_2, a)) \rightarrow \\ OAPM[o_1, o_2, a] = \langle +, avpo \rangle \wedge OAPM[o_2, o_1, a] = \langle -, avpo \rangle \quad (6\text{-}5)$$

All options are associated with the same AVPO with respect to $brand$ (presented above), as the same order preferences are applicable to them. According to this AVPO, Ap_A is better than Ap_C, Ap_D, Ap_E and Ap_F, with respect to this attribute.

Indifference Preferences

As opposed to order preferences, indifference is not transitive. A typical example illustrates the reason for this: a person is indifferent to two cups of tea with a difference of 0.1g of sugar on it. If transitivity is adopted, two cups of tea, one with no sugar and another with 1Kg of sugar, by transitivity, are considered equally preferred.

An indifference preference consists of a set of formulae, establishing indifference for two options' attribute values that satisfy formulae of the same indifference preference, but only if the condition (if any) of the preference is satisfied by both options, as detailed as follows.

$$\exists\, i.(i \in Indifference \wedge att(i) = a \wedge App(i, o_1) \wedge App(i, o_2) \\ \wedge \exists f, f'.(f \in form(i) \wedge sat(f, o_1) \wedge f' \in form(i) \wedge sat(f', o_2)) \rightarrow \\ OAPM[o_1, o_2, a] = \langle \sim, indiff \rangle) \quad (6\text{-}6)$$

By applying all the OAPM rules to our apartment decision problem, we produce as result the OAPM presented in Table 6.5.

6.5 Explication

Preferences provided by users always have a literal meaning. For example, the literal meaning of preference 2 of our running example is that apartments that are less than $2.5Km$ away from the university are preferred to those that are farther away than that. This sentence can also provide further information: if a maximum desired value is provided, and no minimum value, one can conclude that lower values are in general preferred to higher values. In addition, as this is a soft-constraint, i.e. it can remain unsatisfied if other attributes compensate this loss, the closer an option is to satisfying the preference, the better. In the case of preference 2, it means that between two apartments, both farther away than $2.5Km$ from the university, the preferred one is the closer. Preferences that can be derived from other explicit preferences are referred to as *implicit preferences*. We

Table 6.5: OAPM of the Apartment Decision Problem.

	Ap_B	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle -, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$
station	$\langle +, min\rangle$	$\langle \sim, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$
market	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle +, psm\rangle$	$\langle +, avpo\rangle$	$\langle +, avpo\rangle$	$\langle +, avpo\rangle$	$\langle +, avpo\rangle$
stars	$\langle -, max\rangle$	$\langle \sim, max\rangle$	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle -, max\rangle$
price	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle \sim, min\rangle$	$\langle +, min\rangle$

(a) Comparison with Ap_A

	Ap_A	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$
station	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$
market	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle \sim, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$
stars	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle \sim, max\rangle$	$\langle -, max\rangle$
price	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$

(b) Comparison with Ap_B

	Ap_A	**Ap_B**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$
station	$\langle \sim, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$
market	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$
zone	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$
brand	$\langle -, avpo\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, avpo\rangle$
stars	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle -, max\rangle$
price	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$

(c) Comparison with Ap_C

	Ap_A	**Ap_B**	**Ap_C**	**Ap_E**	**Ap_F**
uni	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$
station	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle \sim, min\rangle$	$\langle +, min\rangle$
market	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle -, avpo\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, avpo\rangle$
stars	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle -, max\rangle$
price	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$

(d) Comparison with Ap_D

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_F**
uni	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$
station	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle \sim, min\rangle$	$\langle +, min\rangle$
market	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle -, avpo\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, avpo\rangle$
stars	$\langle +, max\rangle$	$\langle \sim, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle -, max\rangle$
price	$\langle \sim, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$

(e) Comparison with Ap_E

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_E**
uni	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$
station	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$
market	$\langle +, min\rangle$	$\langle \sim, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle -, avpo\rangle$	$\langle +, psm\rangle$	$\langle -, avpo\rangle$	$\langle -, avpo\rangle$	$\langle -, avpo\rangle$
stars	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$
price	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$

(f) Comparison with Ap_F

are aware that there may be exceptions, and an implicit preference may be wrongly considered in certain cases — as it occurs with humans. This problem can be tackled with the addition of knowledge specific to an application area, e.g. by stating if ordered attributes should in general be maximised or minimised, or by learning what individual users usually mean by their provided preferences. Currently, we adopt a set of implicit preferences that in general derives correct preferences from explicit preferences.

Before introducing our set of implicit preferences, we will describe the context in which they will be considered. Implicit preferences do not override information obtained from explicitly provided preferences, that is, they change OAPM values only when two options are considered similar (w.r.t. an attribute) or no conclusion could be made. Therefore, the OAPM value for these two options must be either $\sim$ or ? and, if the value is $\sim$, it was not set due to an indifference preference, as we formally show in Equation 6-7. Moreover, our current implicit preferences are derived from monadic preferences, and as different monadic preferences may be used to derive different implicit preferences, they are derived when there is *only one monadic preference* that refers to an attribute and is applicable to a pair of options. Equation 6-8 is thus used to verify this restriction.

$$\begin{aligned} &Undecided(o_1, o_2, a) := \\ &\quad OAPM[o_1, o_2, a] = \langle ?, r\rangle \vee (OAPM[o_1, o_2, a] = \langle \sim, r\rangle \wedge r \neq indiff) \end{aligned} \tag{6-7}$$

$$
\begin{aligned}
&UniqueMonadic(p, o_1, o_2, a) := \\
&\quad p \in MP \wedge App(p, o_1, a) \wedge App(p, o_2, a) \wedge p(att) = a \\
&\quad \wedge \nexists p'.(p \neq p' \wedge p' \in MP \wedge App(p', o_1, a) \wedge App(p', o_2, a) \wedge p'(att) = a)
\end{aligned}
\tag{6-8}
$$

Our implicit preferences are also valid only for attributes whose domain is ordered. Finally, when keeping track why the OAPM is being updated, the reason is stored in the form of $\langle type, p \rangle$, where $type$ is the type of the applied implicit preference, and p is the preference that caused the update. This information is used in the selection step (Section 6.7). Now, we can proceed to our set of implicit preferences.

6.5.1 Upper bound

In the upper bound case, an upper limit is provided for an attribute (as in preference 2), which is given in a monadic preference whose formula is an instance of $attribute < value$ or $attribute \leq value$. A preference that satisfies this restriction returns true for $UpperBound(p)$. Due to this upper bound, we infer that there is a minimisation goal for this attribute. Note that monadic preferences may be associated with negative modifiers, and in the case the inference is the opposite: there a maximisation goal.

$$
\begin{aligned}
&Undecided(o_1, o_2, a) \wedge UniqueMonadic(p, o_1, o_2, a) \wedge UpperBound(p) \\
&\quad \wedge ((\neg Negative(mod(p)) \wedge val(o_1, a) < val(o_2, a)) \\
&\quad \vee (Negative(mod(p)) \wedge val(o_1, a) > val(o_2, a))) \rightarrow \\
&\qquad OAPM[o_1, o_2, a] = \langle +, \langle upper, p \rangle \rangle \wedge OAPM[o_2, o_1, a] = \langle -, \langle upper, p \rangle \rangle
\end{aligned}
\tag{6-9}
$$

6.5.2 Lower bound

As opposed to the previous case, a lower bound can be provided for an attribute, with monadic preferences whose formula are instances of $attribute > value$ or $attribute \geq value$, thus indicating that the goal is to maximise the value of this attribute (or to minimise it if the preference modifier is negative). Similar to above, $LowerBound(p)$ is true for preferences that satisfy the formula template. This is the case of preference 1, but as it is associated with a negative modifier, the inferred preference is a minimisation goal.

$$Undecided(o_1, o_2, a) \wedge UniqueMonadic(p, o_1, o_2, a) \wedge LowerBound(p)$$
$$\wedge ((\neg Negative(mod(p)) \wedge val(o_1, a) > val(o_2, a))$$
$$\vee (Negative(mod(p)) \wedge val(o_1, a) < val(o_2, a))) \rightarrow$$
$$OAPM[o_1, o_2, a] = \langle +, \langle lower, p \rangle\rangle \wedge OAPM[o_2, o_1, a] = \langle -, \langle lower, p \rangle\rangle \quad (6\text{-}10)$$

6.5.3 Around

If a desired value is given for an ordered attribute (a *reference value*), and this preference is not a hard constraint (i.e. it may be left unsatisfied), we infer that the closer the attribute value of an option is to the desired value, the better. Therefore, between two options, the preferred one is that whose value for this attribute has a shorter distance from the reference value. The rule that updates the OAPM according to the around implicit preference uses the following notations: (i) $Around(p)$, which is true when the monadic preference formula is an instance of $attribute = value$; (ii) $RefVal(p)$, which returns the reference value of a preference that satisfies $Around(p)$; and (iii) $AroundDist(o, a, p) :=\mid val(o, a) - RefVal(p) \mid$, which gives the distance between the attribute value of an option and the attribute reference value. Negative modifiers, as before, inverts the behaviour of the around preference.

$$Undecided(o_1, o_2, a) \wedge UniqueMonadic(p, o_1, o_2, a) \wedge Around(p)$$
$$\wedge ((\neg Negative(mod(p)) \wedge AroundDist(o_1, a, p) < AroundDist(o_2, a, p))$$
$$\vee (Negative(mod(p)) \wedge AroundDist(o_1, a, p) > AroundDist(o_2, a, p))) \rightarrow$$
$$OAPM[o_1, o_2, a] = \langle +, \langle around, p \rangle\rangle \wedge OAPM[o_2, o_1, a] = \langle -, \langle around, p \rangle\rangle \quad (6\text{-}11)$$

6.5.4 Interval

Instead of providing a single desired attribute value, users may provide an *interval*. In these cases, the provided monadic preference is associated with a formula that is an instance of $attribute > lowerBound$ **and** $attribute < upperBound$, or $\geq$ and $\leq$, instead of $>$ and $<$. We introduce: (i) $Interval(p)$, which is true when preference satisfies the formula template; (ii) $lb(p)$, which gives the interval lower bound; and (iii) $ub(p)$, which gives the interval upper bound.

Similarly to the around rule, the OAPM is updated based on the interval implicit preference using an auxiliary function to calculate the distance from

attribute values to the provided interval, which is depicted next. This function may be modified according to the provided interval, by changing $<$ to $\leq$. Note that the interval distance of attribute values that are in the interval is 0, and this makes the OAPM value between these two attribute values remain the same.

$$IntervalDist(o, a, p) =$$
$$\begin{cases} 0 & \text{if } lb(p) < val(o, a) < ub(p) \\ min(\mid val(o, a) - lb(p) \mid, \mid val(o, a) - ub(p) \mid) & \text{otherwise} \end{cases} \tag{6-12}$$

Given this function, we can now present the rule that updates the OAPM, which sets the attribute value of an option that has the lower distance from the interval than another as preferred; or the opposite, if the preference modifier is negative.

$$\begin{aligned} &Undecided(o_1, o_2, a) \wedge UniqueMonadic(p, o_1, o_2, a) \wedge Interval(p) \\ &\quad \wedge ((\neg Negative(mod(p)) \wedge IntervalDist(o_1, a, p) < IntervalDist(o_2, a, p)) \\ &\quad \vee (Negative(mod(p)) \wedge IntervalDist(o_1, a, p) > IntervalDist(o_2, a, p))) \rightarrow \\ &\qquad OAPM[o_1, o_2, a] = \langle +, \langle interval, p \rangle \rangle \wedge OAPM[o_2, o_1, a] = \langle -, \langle interval, p \rangle \rangle \end{aligned} \tag{6-13}$$

There are three cases to which implicit preferences are applicable in our running example, which are related to the preferences 1, 2 and 14. Now, the relationship between options with respect to the attribute $zone$ is established by the goal of minimising the value of this attribute (upper bound preference, with a negative modifier), and the attribute uni, which has also the goal of minimising the value of this attribute (upper bound preference, with a positive modifier). The unique modifier preference with respect to $stars$ indicates that the closer that the apartment stars are to 2, the better (around preference), but option attribute values are either already decided by the explicitly provided goal, or equal, thus the comparison remains $\sim$. Table 6.6 shows the updated OAPM, only with attributes that changed (for simplicity, we omit the preference that is part of the OAPM reason).

After executing the steps for building our two computational models that support the decision making process, and updating them by considering implicit preferences, we still have preferences provided by users that we have not taken into account, which are don't care preferences and priorities over preferences and over attributes. These will be used later, when resolving trade-off situations for choosing an option but, before, we use the constructed PSM and the OAPM to eliminate options, as presented next.

Table 6.6: Updated OAPM of the Apartment Decision Problem.

	Ap_B	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle -, psm \rangle$	$\langle +, upper \rangle$	$\langle +, upper \rangle$	$\langle -, upper \rangle$	$\langle -, psm \rangle$
zone	$\langle -, lower \rangle$	$\langle +, psm \rangle$	$\langle \sim, lower \rangle$	$\langle \sim, lower \rangle$	$\langle -, lower \rangle$
stars	$\langle -, max \rangle$	$\langle \sim, around \rangle$	$\langle \sim, around \rangle$	$\langle -, max \rangle$	$\langle -, max \rangle$

(a) Comparison with Ap_A

	Ap_A	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle +, psm \rangle$	$\langle +, psm \rangle$	$\langle +, psm \rangle$	$\langle +, psm \rangle$	$\langle -, upper \rangle$
zone	$\langle +, lower \rangle$	$\langle +, psm \rangle$	$\langle +, lower \rangle$	$\langle +, lower \rangle$	$\langle \sim, lower \rangle$
stars	$\langle +, max \rangle$	$\langle +, max \rangle$	$\langle +, max \rangle$	$\langle \sim, around \rangle$	$\langle -, max \rangle$

(b) Comparison with Ap_B

	Ap_A	**Ap_B**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle -, upper \rangle$	$\langle -, psm \rangle$	$\langle -, upper \rangle$	$\langle -, upper \rangle$	$\langle -, psm \rangle$
zone	$\langle -, psm \rangle$	$\langle -, psm \rangle$	$\langle -, psm \rangle$	$\langle -, psm \rangle$	$\langle -, psm \rangle$
stars	$\langle \sim, around \rangle$	$\langle -, max \rangle$	$\langle \sim, max \rangle$	$\langle -, max \rangle$	$\langle -, max \rangle$

(c) Comparison with Ap_C

	Ap_A	**Ap_B**	**Ap_C**	**Ap_E**	**Ap_F**
uni	$\langle -, upper \rangle$	$\langle -, psm \rangle$	$\langle +, upper \rangle$	$\langle -, upper \rangle$	$\langle -, psm \rangle$
zone	$\langle \sim, lower \rangle$	$\langle -, lower \rangle$	$\langle +, psm \rangle$	$\langle \sim, lower \rangle$	$\langle -, lower \rangle$
stars	$\langle \sim, around \rangle$	$\langle -, max \rangle$	$\langle \sim, around \rangle$	$\langle -, max \rangle$	$\langle -, max \rangle$

(d) Comparison with Ap_D

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_F**
uni	$\langle +, upper \rangle$	$\langle -, psm \rangle$	$\langle +, upper \rangle$	$\langle +, upper \rangle$	$\langle -, psm \rangle$
zone	$\langle \sim, lower \rangle$	$\langle -, lower \rangle$	$\langle +, psm \rangle$	$\langle \sim, lower \rangle$	$\langle -, lower \rangle$
stars	$\langle +, max \rangle$	$\langle \sim, around \rangle$	$\langle +, max \rangle$	$\langle +, max \rangle$	$\langle -, max \rangle$

(e) Comparison with Ap_E

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_E**
uni	$\langle +, psm \rangle$	$\langle +, upper \rangle$	$\langle +, psm \rangle$	$\langle +, psm \rangle$	$\langle +, psm \rangle$
zone	$\langle +, lower \rangle$	$\langle \sim, lower \rangle$	$\langle +, psm \rangle$	$\langle +, lower \rangle$	$\langle +, lower \rangle$
stars	$\langle +, max \rangle$	$\langle +, max \rangle$	$\langle +, max \rangle$	$\langle +, max \rangle$	$\langle +, max \rangle$

(f) Comparison with Ap_F

6.6 Elimination

One of the typical approaches adopted by users for making a choice is the elimination of options in a stepwise fashion until it remains a set of acceptable options, which ideally contains only one element. This is the main idea of the *Elimination by aspects* (Tversky 1972) and *Satisficing* (Simon 1955) approaches from psychology, and it often consists of an iterative approach of making decisions, which is not a characteristic of our technique. However, we can take an initial step in this direction, by eliminating options that have two properties: (i) dominated options (Section 6.6.1); and (ii) options that do not satisfy hard constraints (Section 6.6.2).

6.6.1 Eliminating Dominated Options

We begin by eliminating options that, for at least one attribute, are worse than another option, and that are not better than it for the remaining attributes. In this situation, we say that the options to be discarded are dominated by another.

Based on the information provided by the OAPM, which was constructed based on provided user preferences, we can define *domination*, which is a binary relation that indicates when an option dominates another, formally presented in Definition 6.4.

Definition 6.4 *Let* o_1 *and* o_2 *be two options in* Opt*, and* a *an attribute in* Att*.* $dominates(o_1, o_2)$ ***holds when:***

$$\exists\, a.(OAPM[o_1, o_2, a] = +) \wedge \forall\, a.(OAPM[o_1, o_2, a] \neq -)$$

For the domination relation to be true, o_1 must have an attribute value that is preferred to the attribute value of o_2. In addition, for all other attribute values, o_1 has to be at least as good as o_2. Therefore, we have a reason to reject o_2, and there is no other positive aspect of this option that can balance its negative aspects, w.r.t. o_1. Domination also holds in the absence of information about the attribute comparison of two options ($OAPM[o_1, o_2, a] =?$), if there is at least one attribute whose OAPM value is +.

Based on the definition of domination, we now define the set of dominated options, $Dominated$. As shown in Equation 6-14, all options dominated by at least one other option are in the $Dominated$ set and are discarded.

$$dominates(o_1, o_2) \rightarrow o_2 \in Dominated \tag{6-14}$$

In Tables 6.5(a) and 6.5(e), it can be seen that the OAPM has the value + or $\sim$, for all attributes of options Ap_A and Ap_E, when compared to Ap_D. Therefore, it can be said that $dominates(Ap\_A, Ap\_D)$ and $dominates(Ap\_E, Ap\_D)$, thus $Ap\_D \in Dominated$.

6.6.2 Applying Cut-off Values

The second set of eliminated options is composed of options that do not satisfy hard constraints of users. We consider hard constraints preferences that are either qualifying or rating statements with one of these four modifiers: (i) *don't accept*; (ii) *hate*; (iii) *require*; and (iv) *need*.

Other modifiers, such as *very good*, *want* and *very bad*, are also strong preferences from users, but they are not considered at this moment, because options have positive and negative aspects w.r.t. each other (otherwise it is a case of domination) and, even though an option has an attribute value that is *very bad*, it may be amortised by other positive aspects of this option. This argument is not valid for the four modifiers considered as hard constraints, because they are

interpreted as associated with *non-compensatory* attribute values, i.e. those that cannot be compensated with any benefit provided by other attributes.

In order to identify the options that are discarded due to cut-off values, we use the information captured by the PSM. First, we show how to select options associated with hard negative modifiers, and then those associated with unsatisfied hard positive modifiers. In the first case, the options selected to be part of the $CutOff$ set, i.e. the set of options discarded due to a cut-off value, are those that have at least one attribute associated with $\langle \epsilon, don't\ accept \rangle$ or $\langle \epsilon, hate \rangle$ in the PSM.

$$\begin{aligned} \exists\, a.(PSM[o, a] = \langle \epsilon, \text{“}don't\ accept\text{”} \rangle \\ \vee\ PSM[o, a] = \langle \epsilon, \text{“}hate\text{”} \rangle) \rightarrow o \in CutOff \end{aligned} \tag{6-15}$$

By construction, every option that satisfies a preference with either the “*don't accept*” or the “*hate*” modifier has the PSM value evaluated for these modifiers for the respective attribute. Therefore, these constraints are always respected.

Differently from the negative modifiers above, *require* and *need* are considered hard constraints unless another positive experience is provided. For example, assume these two preferences: “*I require an apartment at zone 1*”, and “*I accept one in zone 2*.” In this case, the second preference changes the first one to a soft preference, as it indicates an exception to the requirements. So, even though an apartment in zone 2 does not satisfy a requirement, it is not eliminated due to a cut-off. Therefore, *require* and *need* are usually hard constraints, but users may add acceptable exceptions, as in this example. So, we exclude options that satisfy neither a requirement or need, nor any other monadic preference whose modifier is neutral or positive, w.r.t. a particular attribute. However, if an option does not satisfy the requirement or need, but satisfies a monadic preference whose modifier is negative, then it is discarded when there is at least one option that satisfies the requirement or need. This interpretation is adopted because if users provide other preferences related to an attribute besides a requirement or need, it is an indication that they are a soft constraint, but still have a strong preference for their satisfaction. According to this informal explanation, we show below the rule used to select options to be rejected due to positive hard constraints.

$$\begin{aligned} \exists\, a.(PSM[o, a] = \langle \neg, require \rangle \vee PSM[o, a] = \langle \neg, need \rangle \\ \vee\ \exists\, o'.((PSM[o', a] = \langle \epsilon, require \rangle \vee PSM[o', a] = \langle \epsilon, need \rangle) \\ \wedge\ PSM(o, a) = \langle \epsilon, Negative(m) \rangle)) \rightarrow o \in CutOff \end{aligned} \tag{6-16}$$

By construction, if no monadic preference is satisfied by an option, and one of them is associated with a requirement (or need), the PSM value of this option will indicate an unsatisfied requirement (or need).

The PSM presented in Table 6.3 shows that $Ap\_C \in CutOff$, as $PSM[Ap\_C, zone] = \langle \epsilon, don't\ accept \rangle$. Note that an option eliminated because of domination may also be discarded due to a cut-off value, i.e. it is possible that $Dominated \cap CutOff \neq \emptyset$.

6.7 Selection

After performing the elimination step of our process, our set of available options is now reduced to a subset of options (which we refer to as $Acceptable$, but it is also referred to as *consideration set* in the literature), which requires us to resolve trade-offs to make a choice. It is important to highlight that, except for preference and attribute priorities (which will be taken into account in this step), we have already used the information provided by users to reject options, thus in order to make a decision we have to go beyond user preferences. This is the common situation that happens in the process of decision making, as users typically have preferences for individual attributes and resolve trade-offs in light of available options to establish a preference order among them. Our technique, inspired by human decision making, analyses cost and benefits of options and additional factors that humans typically adopt while making decisions.

Humans may also adopt other heuristics to make decisions (Payne et al. 1988), each requiring different amounts of cognitive effort. Heuristics are chosen based on the amount of effort required and the relevance of the decision to be made. Our approach does not aim to reproduce this particular behaviour, because it may be suboptimal when users are not willing to invest effort in the process. Our goal is to understand how people reason to resolve trade-offs, when demanding adequate time and effort to make a decision and this is the reasoning process we adopt to make automated choices. Avoiding extreme options (best for some attributes and worst for others) and analysing the trade-off contrast (influenced by other options), are the two main principles that humans adopt that our technique incorporates (Simonson and Tversky 1992).

The following sections describe the selection step of our technique in four parts. First, we describe how the benefits and costs of each option in the $Acceptable$ set, where $Acceptable = Options \backslash (Dominated \cup CutOff)$, are evaluated with respect to each individual attribute. Then, we show how we assess the overall benefits and costs of options with respect to each other based on the previous evaluation of each individual attribute. Later, we consider the two main principles that are typically adopted by people when making decisions. Finally, we show how all these evaluations are put together and used to make a final decision.

6.7.1 Cost-benefit Analysis

The first part of the process of selecting an option consists of evaluating each pair of options and assessing their relative benefits and costs, using the information provided by the OAPM. The costs of option o_1 w.r.t. o_2 are the benefits of option o_2 w.r.t. o_1, and vice-versa. We compute the cost of o_1 compared to o_2 w.r.t. an attribute a to a real value ranging from 0 to 1, captured by a function we build, represented as shown below.

$$AttCost : Opt \times Opt \times Att \rightarrow \{c \mid c \in \mathbb{R} \wedge 0 \leq c \leq 1\}$$

This value indicates *how much* one option is better than another, w.r.t. to each attribute, which will be used to evaluate the overall option costs. In essence, our cost function transforms qualitative information into quantitative values. One can argue that this quantitative values will comprise a utility function (Keeney and Raiffa 1976), as it is a weighted sum of values given for individual attributes, which represent how much an individual prefers each attribute value, but it is not, mainly because of two main differences. First, the cost function consists of *differences* between values as the cost is obtained by means of a *pairwise* comparison between options (as people usually do), and uses the *specific preferences* for each option pair, as there are many types of preferences applicable for pairs of options (note that using qualitative preferences to quantitatively evaluate preference for options is also a challenge). Second, the way we obtain these cost values is novel: we exploit natural-language-like expressions instead of submitting users to iterative processes, which demands high cognitive effort and time from users. Furthermore, this function is not decisive for making a choice: as introduced before, we will also add two factors that influences the preferences for options, which are associated with the two principles of human decision making. Therefore, option costs are not the unique factors that are needed for making the decision.

The attribute cost is 0, if the $OAPM[o_1, o_2, a] \neq \langle -, r \rangle$; otherwise, i.e. if $OAPM[o_1, o_2, a] = \langle -, r \rangle$, then we use the reason r to compute the $AttCost[o_1, o_2, a]$. The next sections describe this computation for the seven possible reasons r: PSM, order preferences, goal, lower bound, upper bound, around and interval. The remaining one, indifference, always causes OAPM values to be set to $\sim$, and therefore $AttCost$ is 0.

PSM

Our modifier scale (Figure 6.2) allows us to identify whether a modifier is stronger than another, and consequently the preference for a value compared to

another when different modifiers are used to qualify these values. However, as in this step we aim to asses *how much* one attribute value is preferred to another, we have to go beyond the order given by this scale. In order to make this assessment, we associate a numeric value with each index of PSM value, and therefore an option cost for the particular attribute with respect to another is calculated based on the difference between the values associated with the PSM values.

The association of a numeric value with each PSM value is given by a function that we refer to as f_m, which corresponds to a variation point of our technique. We have considered three different functions for generating a value for modifiers:

(i) **linear**: $f_{m_{li}}(index) = index$;

(ii) **quadratic**: $f_{m_{sq}}(index) = index^2$, if $index \geq 0$ and $f_{m_{sq}}(index) = -(index^2)$, if $index < 0$;

(iii) **log**: $f_{m_{lg}}(index) = \ln(\mid index \mid +1)$, if $index \geq 0$ and $f_{m_{lg}}(index) = -\ln(\mid index \mid +1)$, if $index < 0$.

This way of calculating the option cost for a particular attribute based on the PSM is shown in Equation 6-17. The value related to the modifiers is normalised to a value between 0 and 1, considering the possible modifiers (or their negation) that can be associated with any of the two options being compared. Therefore, for all monadic preferences that either $App(p, o_1)$ or $App(p, o_2)$, we select the $\langle \epsilon, m \rangle$ or $\langle \neg, m \rangle$ that has the maximum and minimum indices, i.e. we obtain which are the maximum and minimum values associated with a PSM value that these options can have, represented in Equation 6-17 as $max(maxIdx(o_1), maxIdx(o_2))$ and $min(minIdx(o_1), minIdx(o_2))$, respectively.

$$AttCost(o_1, o_2, a) = \frac{\mid f_m(PSMIndex([PSM[o_1, a], scale)) - f_m(PSMIndex([PSM[o_2, a], scale)) \mid}{f_m(max(maxIdx(o_1), maxIdx(o_2))) - f_m(min(minIdx(o_1), minIdx(o_2)))} \quad (6\text{-}17)$$

We have adopted the **log** function ($f_{m_{lg}}$) in our approach, chosen based on experimentation. This function makes the difference between strong modifiers, such as $want$ and $best$, smaller than the differences between modifiers in the middle of the scale, such as $neutral$ and $don't\ avoid$, and therefore the preference is stronger when we compare positive modifiers with negative modifiers. For example, the cost of values rated with $neutral$ compared to values qualified with $don't\ prefer$ is higher than the cost of values qualified with $want$ compared to values qualified with $desire$, even though for both the index differences are of two units.

When calculating the cost of the attribute uni with respect to options Ap_A and Ap_B, we obtain as the maximum and minimum possible values for

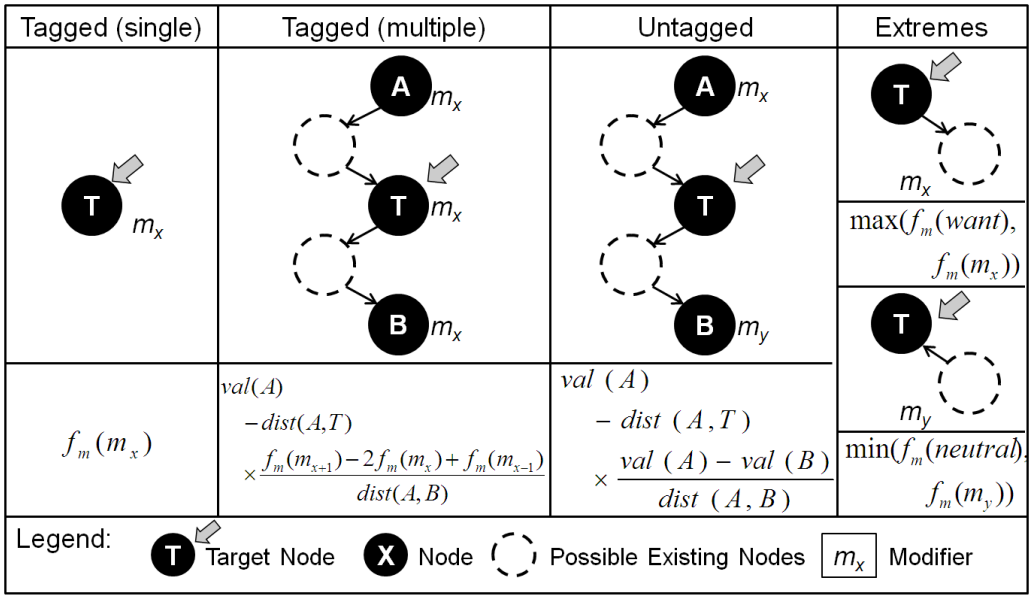


Figure 6.3: Calculating node values.

these options are those associated with $\langle \epsilon, prefer \rangle$ and $\langle \neg, prefer \rangle$, respectively, which are also the PSM values associated with Ap_B and Ap_A. Therefore, we have $\mid f_{m_{lg}}(6) - f_{m_{lg}}(-2) \mid / (f_{m_{lg}}(6) - f_{m_{lg}}(-2))$, which is 1.0 — the value of $AttCost(Ap\_A, Ap\_B, uni)$.

Order Preferences

AVPOs allow comparing attribute values and identifying the preferred one; however, as in our modifier scale, it is not possible to know how much one value is preferred to another. So, in order to obtain this information, we also associate numeric values with AVPO nodes, and for that we use information from the monadic preferences. Each AVPO is associated with an option and an attribute, and the monadic preferences considered are those that have their condition satisfied by that option and attribute. We initiate this process by tagging AVPO nodes with a modifier from the monadic preferences whose formula is satisfied by the domain value associated with the node: for selecting from among multiple modifiers, we follow the same rules used for building the PSM, but here we have only satisfied modifiers. Based on this tagging, the AVPO nodes are associated with a numerical value, as summarised in Figure 6.3 and explained in detail next.

Extreme Nodes For guaranteeing the existence of tagged values in the AVPO, we tag all most preferred values and all least preferred nodes (fourth column of Figure 6.3). The former is tagged with $want$ (in our current modifier scale, it is the strongest modifier) and the latter with $neutral$. If there are values in the partial order

tagged with a stronger modifier than $want$ or $neutral$, for instance $A \succ B \succ C$, and B is tagged with $avoid$, the extreme untagged nodes receive these tags for keeping consistency, i.e. A is tagged with $want$ (as the default) and C with $avoid$. We tag least preferred nodes with $neutral$ by default, because people typically provide an order for preferred or acceptable values, and do not mention not preferred ones — we confirmed this in our previous study (Chapter 2).

Tagged Nodes If the node is tagged with a modifier (first and second columns of Figure 6.3), it receives the value according to the values given for the modifier scale. However, there may be different values tagged with the same modifier and ordered in a sequential way, such as in the running example in which all nodes of the AVPO are tagged with the $prefer$ modifier.

In order to address this issue, Algorithms 3 and 4 are used to calculate the value of tagged AVPO nodes. The first searches for the most distant node that has the same modifier of the target modifier, either searching through the parents or children of nodes, according to a parameter up provided to the algorithm.

Algorithm 3: $LastEqual(tag, node, up, scale)$

Input: tag: modifier to be searched for;$node : AVPONode$; up : $boolean$ (flag to indicate if the search should be in the parents or the children of $node$; $scale$: modifier scale

Output: $\langle firstTagged, dist\rangle : \langle AVPONode, int\rangle$, first tagged node and the distance from it to $node$

```
1  List⟨ AVPONode ⟩ nodeList ← up ? Parents(node) : Children(node);
2  ⟨ AVPONode, int ⟩ lastEqual ← null;
3  double tagValue ← f_m(IndexOf(tag, scale));
4  foreach AVPONode next ∈ nodeList do
5      double nextTagValue ← null;
6      if Tag(next) ≠ null then
7          double nextTagValue ← f_m(IndexOf(Tag(next), scale));
8      if nextTagValue = null ∨ tagValue = nextTagValue then
9          ⟨ AVPONode, int ⟩ temp ← LastEqual(tag, next, up, scale);
10         if temp ≠ null then
11             temp_2 ← π_2(temp) + 1;
12         else if tagValue = nextTagValue then
13             temp ← ⟨next, 0⟩;
14         if lastEqual = null ∨ π_2(lastEqual) < π_2(temp) then
15             lastEqual ← temp;
16 return firstTagged;
```

Then, in Algorithm 4, two situations can happen. If the node tagged with a particular modifier is unique, its node value is given by the f_m function. In the second situation — more than one node is tagged with the same modifier — the most preferred value (brand A, in our running example) is associated with the numeric value related to the modifier, plus half of the difference between the modifier value and the modifiers whose index is increased in one unit, according the modifier scale

(in the example, *desire*). Similarly, the least preferred value (C) is associated with the value related to the modifier, minus half of the difference between the modifier value and the modifiers whose index is decreased in one unit (in the example, *need*). With these values, we divide the difference between the values associated with the most and least preferred values by the their distance (which is two, in the case of A and C) for obtaining the difference between any two values, which we refer to as $step$, and with it we are able to calculate the value of the remaining values. Therefore, the value associated with B is the value of A minus the distance between A and B times the $step$, as shown in lines 23 and 24 of Algorithm 4.

Algorithm 4: $TaggedNodeValue(node, scale)$

Input: $node$: $AVPONode$; $scale$: modifier scale
Output: $value$: $double$

```
1  int dist ← 0;
2  ⟨ AVPONode, int ⟩ above ← LastEqual(Tag(node), node, true, scale);
3  if above ≠ null then
4      dist ← π2(above);
5  ⟨ AVPONode, int ⟩ below ← LastEqual(Tag(node), node, false, scale);
6  if below ≠ null then
7      dist ← π2(below);
8  double tagValue ← fm(IndexOf(Tag(node),scale));
9  if dist = 0 then
10     return tagValue;
11 else
12     double max ← tagValue;
13     double temp ← fm(IndexOf(Tag(node), scale) +1);
14     if temp ≠ null then
15         max = max+ | temp − tagValue | /2;
16     double min ← tagValue;
17     double temp ← fm(IndexOf(Tag(node), scale) −1);
18     if temp ≠ null then
19         min = min− | temp − tagValue | /2;
20 if above = null then
21     return max;
22 else
23     double step ←| max − min | / dist;
24     return max − π2(above) × step;
```

Untagged Nodes If the node is not tagged with a modifier (third column of Figure 6.3), we first obtain the maximum and minimum surrounding node values of the target node, which is done through the execution of Algorithm 5. We find the closest tagged nodes that are preferred to the target node, and the closest tagged nodes less preferred than the target node. Then, we choose from these tagged nodes by selecting those that has the smaller difference from the target node: we consider their unsigned numerical value (as they are tagged, it is calculated in the way we explained above) and divide by the distance between them and the target value ($step$). Next, we calculate the numerical value for the node immediately above (or

below) the target node: we take the numerical value of the tagged node and subtract (or add) from it the step times the distance from the tagged node to the target node minus one, as we are calculating the value of the node immediately above or below. The lower (higher) numerical value calculated for this node is chosen, and therefore we guarantee that all more preferred nodes (than the target node) are associated with a higher numerical value and all less preferred nodes (than the target node) are associated with a lower numerical value.

Algorithm 5: $FirstTaggedNode(node, up, scale)$

Input: $node : AVPONode$; $up : boolean$ (flag to indicate if the search should be in the parents or the children of $node$; $scale$: modifier scale
Output: $\langle firstTagged, dist\rangle : \langle AVPONode, int\rangle$, first tagged node and the distance from it to $node$

```
1  List⟨ AVPONode ⟩ nodeList ← up ? Parents(node) : Children(node);
2  ⟨ AVPONode, int ⟩ firstTagged ← null;
3  double rate ← 0;
4  foreach AVPONode next ∈ nodeList do
5      ⟨ AVPONode, int ⟩ temp ← null;
6      if Tag(next) = null then
7          temp ← FirstTaggedNode(next, up, scale);
8      else
9          temp ← ⟨next, 0⟩;
10     temp2 ← π2(temp) + 1;
11     double tagValue ← fm(IndexOf(Tag(π1(temp)), scale));
12     double step ←| tagValue | /π2(temp);
13     if up then
14         double rateTemp ← tagValue − (π2(temp) − 1) × step;
15         if rate = null ∨ rateTemp < rate then
16             firstTagged ← temp;
17             rate ← rateTemp;
18     else
19         double rateTemp ← tagValue + (π2(temp) − 1) × step;
20         if rate = null ∨ rateTemp > rate then
21             firstTagged ← temp;
22             rate ← rateTemp;
23 return firstTagged;
```

After choosing the preferred and less preferred nodes, we calculate the difference between their numerical values and divide it by their distance ($step$), and then we calculate the numerical value related to the target node as above, as shown in Algorithm 6.

With this set of algorithms, the value of an AVPO node is given as shown below.

$$NodeVal(node) = \begin{cases} UntaggedNodeValue(node) & \text{if } Tag(node) = null \\ TaggedNodeValue(node) & \text{otherwise} \end{cases}$$

After associating tags and numerical values with the AVPO nodes, we can calculate the costs of an option with respect to an attribute using this information.

Algorithm 6: $UntaggedNodeValue(node, up, scale)$

Input: $node$: $AVPONode$; $scale$: modifier scale
Output: $value$: $double$

```
⟨ AVPONode, int ⟩ above ← FirstTaggedNode(node, true, scale);
double aboveValue ← TaggedNodeValue(π1(above), scale);
⟨ AVPONode, int ⟩ below ← FirstTaggedNode(node, false, scale);
double belowValue ← TaggedNodeValue(π1(below), scale);
double step ←| aboveValue − belowValue | /(π2(above) + π2(below));
return aboveValue − (π2(above) × step);
```

The cost associated with the attribute values of two options according to a given AVPO is shown in Equation 6-18, where $Node(AVPO[o, a], o_1)$ gives the node of the AVPO of option o and attribute a that is associated with the attribute value a of o_1. The cost is normalised in a similar manner as with the PSM. We use as maximum and minimum values those associated with indices of modifiers and their negation that tag any of the AVPO nodes. Finally, as there are two AVPOs, one associated with each option, the final cost is the average of their respective costs as shown in Equation 6-19.

$$AVPOAttCost(o, o_1, o_2, a) = \frac{| NodeVal(Node(AVPO[o, a], o_1)) - NodeVal(Node(AVPO[o, a], o_2)) |}{f_m(maxIdx(AVPO[o, a])) - f_m(minIdx(AVPO[o, a]))} \quad (6\text{-}18)$$

$$AttCost(o_1, o_2, a) = \frac{AVPOAttCost(o_1, o_1, o_2, a) + AVPOAttCost(o_2, o_1, o_2, a)}{2} \quad (6\text{-}19)$$

Goal, Lower Bound and Upper Bound

In the case that an option is considered worse than another (with respect to an attribute) due to a goal, upper or lower bound, we again exploit monadic preferences. Each attribute is associated with a domain, and we tag different domain values of attributes associated with goals (or lower and upper bound preferences) with values of modifiers of monadic preferences satisfied by the domain values. The tagging, according to the formula of monadic preferences, is as follows.

(i) $attribute = value$: the domain value $value$ is tagged with the value associated with the preference modifier.

(ii) $attribute > value_1$ **and** $attribute < value_2$: the domain values $value_1$ and $value_2$ are tagged with the preference modifier, plus and minus the difference from this modifier to the closest modifiers (as with AVPO nodes tagged with

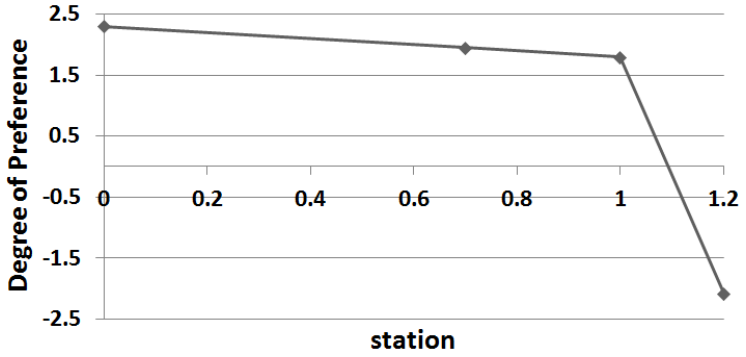


Figure 6.4: Tagging an attribute domain associated with a goal.

the same modifier). Note that it is possible to have ≥ and ≤ instead of > and <, respectively.

(iii) Domain boundaries, if not tagged, are tagged with the numeric values associated with *don't want* (minimum value), which is the minimum modifier that is not a hard constraint, and *want* (maximum value), in case of maximisation goal (or lower bound). If the goal is a minimisation (or upper bound), the boundaries tags are inverted. We do not use preferences with *don't accept* and *hate*, because these modifiers are hard constraints, but as in AVPOs, they are used to tag domain values only to keep consistency if there are monadic preferences that use them.

With this tagging, we keep the maximisation and minimisation goals but also associate a degree of preference (DoP) with specific domain values. For instance, based on preferences 3 and 4, we tag $station = 1.0$ with the value associated with *need*, and $station = 0.7$ with the value associated with *prefer*, obtaining the curve shown in Figure 6.4.

Now, we use these degrees of preference to measure attribute costs. As each domain value is now surrounded by two tagged values (or it is a tagged value), we are able to derive a degree of preference for all domain values, and the cost is the difference between them, normalised according to the maximum and minimum degrees of preference, which are given by the domain boundaries ($min(D_a)$ and $max(D_a)$).

Given an attribute value y_a, whose closest tagged attribute values are x_a, tagged with t_x, and z_a, tagged with t_z, we can calculate the parameters a and b of a linear function $DoP(x) = ax + b$, where $a = (t_x - t_z)/(x_a - z_a)$, and $b = t_x - ax_a$, and then calculate the degree of preference of y_a. If y_a is tagged, it already has an associate degree of preference.

$$AttCost(o_1, o_2, a) = \frac{\mid DoP(val(o_1, a)) - DoP(val(o_2, a)) \mid}{DoP(max(D_a)) - DoP(min(D_a))} \quad (6\text{-}20)$$

Even though goals can be either of maximisation or of minimisation, we use the same equation to calculate the attribute cost, as the difference is the same in both cases, and the cost is associated only with the option whose OAPM value is −.

Around

For assessing the cost using an around preference, we make a similar calculation as above, but many modifiers are not helpful in this case, as there is solely one monadic preference that is applicable to the options being compared — this is a requirement to apply the around implicit preference. We evaluate the attribute cost based on the difference of between attribute values and the reference value, which ranges from 0 (the attribute value is equal to the reference value, thus the distance from this value is 0) to the longest distance from the reference value, considering the attribute domain. As here we cannot use the difference between modifiers (as there is only one modifier, associated with the reference value), we use a function $f_d(dist)$ (currently, instantiated as a linear function), to evaluate cost in terms of the distance from the reference value, as shown in Equation 6-21, where p is the preference associated with the reason of the OAPM value $\langle around, p\rangle$.

$$AttCost(o_1, o_2, a) = \frac{f_d(\mid AroundDist(o_1, a, p) - AroundDist(o_2, a, p) \mid)}{f_d(max(\mid min(D_a) - RefVal(p) \mid, \mid max(D_a) - RefVal(p) \mid))} \tag{6-21}$$

Interval

Similarly to the around preference case, we assess the cost using an interval preference as a basis using the distance from a reference value, which is now an interval. The range of possible distances is from 0 (the attribute value is in the interval) to the longest distance from the interval extremes (considering the attribute domain), which are used by f_d to calculate cost. The difference between the attribute values from the compared options is given by $IntervalDist(o, a, p)$, introduced in Equation 6-12, where p is the preference associated with the reason of the OAPM value $\langle interval, p\rangle$.

$$AttCost(o_1, o_2, a) = \frac{f_d(\mid IntervalDist(o_1, a, p) - IntervalDist(o_2, a, p) \mid)}{f_d(max(\mid min(D_a) - lb(p) \mid, \mid max(D_a) - ub(p) \mid))} \tag{6-22}$$

Given these different ways of calculating the attribute cost $AttCost(o_1, o_2, a)$ based on the reasons for establishing a preference between attribute values of two options, we show the attribute costs for our running example in Table 6.7.

Overall Option Costs

Up to now, we have considered the costs of an option with respect to another by considering attributes in isolation, and now we look at the overall option costs

Table 6.7: Cost-benefit Analysis for the Apartment Decision Problem.

	w_i	**Ap_B**	**Ap_E**	**Ap_F**
uni	0.179	1.000	0.100	1.000
station	0.196			
market	0.132	0.071	0.214	0.071
zone	0.210	0.200		0.200
brand	0.094			
stars	0.030	0.027	0.027	0.054
price	0.158			
Cost		0.231	0.047	0.232

(a) $AttCost(Ap\_A, o', a)$

	w_i	**Ap_A**	**Ap_E**	**Ap_F**
uni	0.196			0.013
station	0.179	0.250	0.083	
market	0.132		0.143	
zone	0.210			
brand	0.094	1.000	1.000	1.000
stars	0.030			0.027
price	0.158	0.500	0.500	
Cost		0.218	0.207	0.098

(b) $AttCost(Ap\_B, o', a)$

	w_i	**Ap_A**	**Ap_B**	**Ap_F**
uni	0.179		1.000	1.000
station	0.196	0.166		
market	0.132			
zone	0.210		0.200	0.200
brand	0.094	0.018		
stars	0.030			0.027
price	0.158			
Cost		0.034	0.221	0.222

(c) $AttCost(Ap\_E, o', a)$

	w_i	**Ap_A**	**Ap_B**	**Ap_E**
uni	0.196			
station	0.179	0.500	0.250	0.333
market	0.132			0.143
zone	0.210			
brand	0.094	0.036		0.018
stars	0.030			
price	0.158	0.833	0.333	0.833
Cost		0.225	0.098	0.212

(d) $AttCost(Ap\_F, o', a)$

(also with respect to another option). This is performed by taking into account the priorities provided — which can be preference priority, attribute priority and attribute indifference — and building an attribute partial order ($attPO$) for each option, as different priorities can be applicable to different options.

As preferences, priorities pri also have a condition $cond(pri)$, and we present a similar applicability definition.

Definition 6.5 *A priority pri is applicable to an option o, $App(pri, o)$, if and only if*

$$\nexists cond(pri) \vee (\exists\, cond(pri) \wedge sat(cond(pri), o))$$

We initially take into consideration preference priorities (applicable to a particular option), which associates a number with preferences, meaning that the lower the number associated with the preference is, the more important it is. Each preference is related to a single attribute (according to our assumptions), and therefore the attribute order follows the order implied by the numbers associated with the preference, as presented in Algorithm 7 — as there may be many preferences associated with an attribute, we consider the lowest number.

Attribute priority and indifference modify the order given by preference priorities. First, if the attribute priority is inconsistent with the order of preference priorities, the attribute that was, before, considered less important becomes the attribute immediately more important than the other attribute referred in the given

Algorithm 7: $ProcessPreferencePriorities(priorities, allAtt)$

Input: $priorities$: preference priorities applicable to an option; $allAtt$: set of attributes
Output: $attPO$: attribute partial order

```
1  Set⟨Attribute⟩ N ← ∅;
2  Set⟨Arrow⟩ A ← ∅;
3  Set⟨Attribute⟩ parents ← ∅;
4  int i ← 1;
5  while priorities ≠ ∅ do
6      Set⟨Attribute⟩ currentAtt ← ∅;
7      while ∃ pri.(pri ∈ priorities∧ Priority(pri) = i) do
8          PreferencePriority pri ← Get(priorities, i);
9          Attribute a ← Attribute(pri);
10         if a ∉ N then
11             N ← N ∪ {a};
12             currentAtt ← currentAtt ∪ {a};
13         priorities = priorities\{pri};
14     if currentAtt ≠ ∅ then
15         foreach a ∈ currentAtt do
16             foreach p ∈ parents do
17                 A ← A ∪ {⟨p, a⟩};
18         parent ← currentAtt;
19     i ← i + 1;
20 foreach a ∈ (allAtt\N) do
21     N ← N ∪ {a};
22     foreach p ∈ parents do
23         A ← A ∪ {⟨p, a⟩};
24 return ⟨N, A⟩;
```

attribute priority. Algorithm 8 shows how this swapping process is performed. Second, if the attribute indifference is inconsistent with the order of preference priorities, the least important attribute becomes as important as the previously more important attribute. Algorithm 9 shows how this change is performed.

Finally, attributes associated with a *don't care* preference are excluded from attribute partial order. It is important to highlight that, as we assume consistency, priorities do not form a cycle.

In our running example, for option Ap_A , preference priorities result initially in the following order.

$$zone \succ \textbf{uni} \succ \textbf{station} \succ price \succ market \succ brand \succ stars$$

By considering the given attribute priority, $station$ and uni are swapped.

$$zone \succ \textbf{station} \succ \textbf{uni} \succ price \succ market \succ brand \succ stars$$

Given the attribute order, we consider the least important attributes as having the level 1 in the order, and the longest path in the order from the least important attributes to the most important ones is referred to as $length(attPO)$. Then, we use

Algorithm 8: $MoveAbove(att_1, att_2, A)$

Input: att_1, att_2: attributes to be swapped; A: attPO arrows

```
1  Set⟨Attribute⟩ oldParents ← Parents(att1) ;
2  Set⟨Attribute⟩ oldChildren ← Children(att1) ;
3  foreach p ∈ oldParents do
4      foreach c ∈ oldChildren do
5          A ← A ∪ {⟨p, c⟩};
6      A ← A\{⟨p, att1⟩};
7  foreach c ∈ oldChildren do
8      A ← A\{⟨att1, c⟩};
9  foreach p ∈ Parents(att2) do
10     A ← A ∪ {⟨p, att1⟩};
11     A ← A\{⟨p, att2⟩};
12 A ← A ∪ {⟨att1, att2⟩};
13 foreach p ∈ oldParents do
14     if ¬ExistsPath(att1, p) ∧ ¬ExistsPath(p, att1) then
15         A ← A ∪ {⟨p, att1⟩};
16 foreach c ∈ oldChildren do
17     if ¬ExistsPath(att1, c) ∧ ¬ExistsPath(c, att1) then
18         A ← A ∪ {⟨att1, c⟩};
```

Algorithm 9: $MoveEqual(att_1, att_2, A)$

Input: att_1, att_2: attributes to be swapped; A: attPO arrows

```
1  Set⟨Attribute⟩ oldParents ← Parents(att1) ;
2  Set⟨Attribute⟩ oldChildren ← Children(att1) ;
3  foreach p ∈ oldParents do
4      foreach c ∈ oldChildren do
5          A ← A ∪ {⟨p, c⟩};
6      A ← A\{⟨p, att1⟩};
7  foreach c ∈ oldChildren do
8      A ← A\{⟨att1, c⟩};
9  foreach p ∈ Parents(att2) do
10     A ← A ∪ {⟨p, att1⟩};
11 foreach c ∈ Children(att2) do
12     A ← A ∪ {⟨att1, c⟩};
13 foreach p ∈ oldParents do
14     if ¬ExistsPath(att1, p) ∧ ¬ExistsPath(p, att1) then
15         A ← A ∪ {⟨p, att1⟩};
16         A ← A ∪ {⟨p, att2⟩};
17 foreach c ∈ oldChildren do
18     if ¬ExistsPath(att1, c) ∧ ¬ExistsPath(c, att1) then
19         A ← A ∪ {⟨att1, c⟩};
20         A ← A ∪ {⟨att2, c⟩};
```

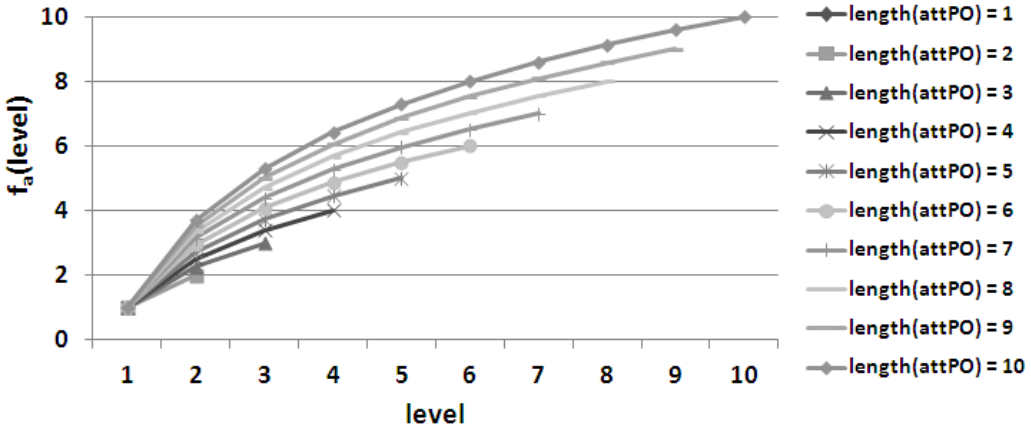


Figure 6.5: Attribute weights calculated with the logarithmic function.

a logarithmic function ($f_a(x) = \alpha \log x + \beta$) for calculating the attribute weights when considering the overall option benefits. We establish the following points for the function.

$$f_a(1) = 1 \tag{6-23}$$

$$f_a(length(attPO)) = length(attPO) \tag{6-24}$$

Point 6-23 indicates that attributes in the first level of the order have the minimum weight, which is 1, and point 6-24 shows that attributes in the last level have the maximum weight, which is $length(attPO)$. The logarithmic function, with the characteristics imposed by the points we established, gives a much higher priority to more important attributes, and these more important attributes have a smaller difference among them (in comparison with a linear function). This behaviour is shown in Figure 6.5, which shows the weight logarithmic function for one to ten levels of attributes. This is a default form we are adopting for calculating attribute weights, which was also selected based on experimentation, and it is a variation point. Next, we present how we calculate the parameters α and β of the logarithmic function for a particular level of attributes.

$$\alpha = \frac{length(attPO) - 1}{\log length(attPO)} \tag{6-25}$$

$$\beta = 1 \tag{6-26}$$

Based on the logarithmic function with the calculated parameters, the weight of each attribute $a_i \in Att$ is as shown below.

$$w_i = \frac{f_a(level(a_i))}{\sum_{a_j \in Att} f_a(level(a_j))}$$

Finally, now that we have the costs of an option o_1 with respect to o_2, for each individual attribute, and we also have the attributes weights, we calculate the overall benefits from o_1 with respect to o_2 using a weighted sum, as presented next. This function, which denotes the costs of all options w.r.t. each other option, calculated for our running example is shown in the last row of Table 6.7, which also details the attribute weights for each option.

$$Cost(o_1, o_2) = \sum_{a_i \in Att} w_i \times AttCost(o_1, o_2, a_i)$$

6.7.2 Trade-off Contrast

The result of not having dominated options in the set of acceptable options is that for any two options, one option is better for one or more attributes and the same applies to the other. As a consequence, a trade-off must be resolved for choosing one of the two options. According to Simonson and Tversky (Simonson and Tversky 1992), when people make choices they do not look only for the two options being compared, but analyse the cost-benefit relationship between two options compared with the cost-benefit relationship between all other options. This reasoning of comparing the trade-offs of the whole set of options is referred to as *trade-off contrast*, and is not in accordance with traditional decision making theory as it states that the preference between two options is independent of the other available options.

Therefore, we incorporate a new factor in the process of choosing an option, which is captured by a function that shows the trade-off between two options.

$$to : Opt \times Opt \rightarrow \mathbb{R}$$

We build the trade-off (to) as a partial function whose domain is every pair of options that satisfies $Cost(o_1, o_2) < Cost(o_2, o_1)$ and is associated with the options' cost-benefit relationship: $Cost(o_1, o_2)/Cost(o_2, o_1)$. Because $Cost(o_1, o_2) < Cost(o_2, o_1)$, to is always value in the interval [0, 1] and $Cost(o_2, o_1)$ cannot be 0. The average of all values of to is referred to as avg_{to}.

The trade-off between two options does not have a meaning in an isolated manner; when we have only two options, all we know is that one option has higher or lower cost than another. When there are other options, and the decision maker observes that the cost-benefit relationship is better for other options, this is seen as a negative aspect of the option and the benefits become smaller. That is, the option requires giving too much for receiving just a little in exchange.

Table 6.8: Trade-off Analysis for the Apartment Decision Problem.

	Ap_B	**Ap_E**	**Ap_F**
to			
ToContrast		0.191	

(a) Trade-off of Ap_A

	Ap_A	**Ap_E**	**Ap_F**
to	0.943	0.937	
ToContrast	0.017	0.017	

(b) Trade-off of Ap_B

	Ap_A	**Ap_B**	**Ap_F**
to	0.731		
ToContrast			

(c) Trade-off of Ap_E

	Ap_A	**Ap_B**	**Ap_E**
to	0.969	0.999	0.956
ToContrast	0.052	0.052	0.052

(d) Trade-off of Ap_F

Given the structure we built to store trade-offs, to, we now calculate the option costs with respect to trade-off, having as a basis the average of the trade-off between a particular option with the others — which is represented by the average of these trade-offs $avg_{to}(o)$ — and the trade-off among all options (which is represented by the average of all trade-offs avg_{to}). If $Cost(o_1, o_2) < Cost(o_2, o_1)$ and the trade-off relationship of o_1 is higher (i.e. worse) than avg_{to}, then we have one more cost of o_1 w.r.t. o_2. If the trade-off is lower (i.e. better) than the average, than it is counted as a benefit, and therefore as a cost for o_2. The function $ToContrast(o_1, o_2)$, which captures this notion of trade-off contrast, is shown below.

$$ToContrast(o_1, o_2) = \begin{cases} avg_{to}(o_1) - avg_{to} & \text{if } to(o_1, o_2) \text{ is defined} \\ & \text{and } avg_{to}(o_1) > avg_{to} \\ avg_{to} - avg_{to}(o_1) & \text{if } to(o_2, o_1) \text{ is defined} \\ & \text{and } avg_{to}(o_2) < avg_{to} \\ 0 & \text{otherwise} \end{cases} \quad (6\text{-}27)$$

The function $Cost$ calculated for our running example allows us to analyse the trade-off among options (to), which is shown in Table 6.8. The to averages are: avg_{to} = 0.923, $avg_{to}(Ap\_B)$ = 0.940, $avg_{to}(Ap\_E)$ = 0.731 and $avg_{to}(Ap\_F)$ = 0.975. Based on that we can calculate the trade-off contrast ($ToConstrast$), which is also presented in this table.

6.7.3 Extremeness Aversion

Another aspect that people take into consideration when making a decision is how extreme options are. Extreme options are those that have a large improvement for one attribute (or set of), e.g. quality, and a high penalisation for another attribute (or set of), e.g. price. In general, people avoid extreme options (Simonson and Tversky 1992), and this is referred to as *extremeness aversion*.

In order to evaluate how extreme options are, we compare option attribute values to the best possible values, measuring the distance between them. As best

values are subjective to each individual, we use the preferences applicable to each option to identify the best value for each attribute. As these values are better or equal to the particular option being analysed o, each attribute value of o can be associated with an attribute cost, which ranges from 0, i.e. the attribute value is equal to the best value, to 1, i.e. the attribute value is the worst possible value. Each of these costs is referred to as distance from best, or $bestDist(o, a)$.

The procedure is similar to making a cost-benefit analysis of the option being analysed with a hypothetic option whose attribute values are the best. Preferences to identify best values are processed in the inverse order of that used to build the OAPM, consequently we will keep the same precedence order, as earlier processed preferences may have their OAPM value overridden. Best values are identified in the following way, when attribute a of option o is being analysed. If there is a *don't care* preference associated with a (and is applicable to o), a is not taken into account.

(i) If $val(o, a)$ has an associated node in the $AVPO(o, a)$, the best value is that related to a source node of the AVPO. If more than one source node exists, we use the one with the highest associated numeric value.

(ii) If there is a goal associated with a (and the goal is applicable to o, we use the domain lower bound ($min(D_a)$) in case of a minimisation goal, and the domain upper bound ($max(D_a)$), otherwise.

(iii) If there are monadic preferences applicable to o, i.e. there is a modifier associated with its PSM value, we do not specify a particular best value but a PSM value that would be associated with the best value, which is $\langle \epsilon, m \rangle$ or $\langle \neg, m \rangle$, whose index is the highest and m is the modifier of a monadic preference applicable to o.

(iv) If none of the above can be applied, or monadic preferences are inconclusive to measure the distance between the attribute value and the best value, i.e. they are considered similar, we use implicit preferences. Then, the best value is: (a) $min(D_a)$, in case of an upper bound, (b) $max(D_a)$, in case of a lower bound; (c) $RefVal(p)$, in case of an around preference; and (d) either $lb(p)$ or $ub(p)$, which are an interval boundaries, in case of an interval preference.

Given this way of identifying best values, we calculate $bestDist(o, a)$ in the same way that attribute costs are calculated, but comparing options with best values instead of other options.

An extreme option has low costs for some attributes ($bestDist(o, a)$ close to 0) and high costs for others ($bestDist(o, a)$ close to 1), therefore we evaluate how extreme the option is by calculating the standard deviation of the function $bestDist$

for a particular option, for all attributes, which is a value between 0 and 1.

$$ext(o) = STDEV(\{bestDist(o, a_i) \mid i = 1...|Att|\})$$

The acceptable options of our apartment example are ordered according to their extremeness in the following way (from the least extreme to the most extreme): $ext(Ap\_B) = 0.344$, $ext(Ap\_E) = 0.346$, $ext(Ap\_A) = 0.361$, and $ext(Ap\_F) = 0.403$.

Finally, as the more extreme the option is, the more people avoid it, it is considered that the more extreme option, between two options, has a cost with respect to the other option. So, in order to capture this aspect, we define $ExtAversion : Options \times Options \rightarrow \mathbb{R}$, which represents the cost of the first option compared to the second, with respect to the extremeness aversion principle. This function, presented below, shows how the extremeness aversion is calculated: the more extreme option has a cost that is the difference between the extremeness values of the two options, and, as the less extreme option has no cost with respect to the other, the value is 0.

$$ExtAversion(o_1, o_2) = \begin{cases} ext(o_1) - ext(o_2) & \text{if } ext(o_1) > ext(o_2) \\ 0 & \text{otherwise} \end{cases} \quad (6\text{-}28)$$

6.7.4 The Decision Function: Comparing Relative Option Values

After executing the previous steps, we have analysed three aspects when comparing options: their costs, the trade-off relative to the set of available options, and how extreme they are. The last two aspects are also seen as costs (or benefits): if the trade-off is higher than the average, it is also considered as a cost, and a more extreme option has a cost when compared to a less extreme. So the final value of an option with respect to another combine these three aspects in a weighted sum of these costs, which can be seen in Equation 6-29, comprising our *decision function* — $d(o_1, o_2)$. We are now considering default weights (the last variation point of our technique), which are 0.25 for trade-off contrast and 0.15 for extremeness aversion. Based on the $d(o_1, o_2)$ function, we identify the chosen option, which is the option that has less or equal disadvantages ($d(o_1, o_2) \leq d(o_2, o_1)$) than every other option of the $Acceptable$ set, i.e. those that are better or equal to the other options. If different options have the same decision value with respect to another ($d(o_1, o_2) = d(o_2, o_1)$), and they are better than every other option, we randomly choose one of them.

$$\begin{aligned} d(o_1, o_2) = {} & (1 - w_{to} - w_{ea}) \times Cost(o_1, o_2) \\ & + w_{to} \times ToContrast(o_1, o_2) + w_{ea} \times ExtAversion(o_1, o_2) \end{aligned} \quad (6\text{-}29)$$

Table 6.9: Options for Illustrating the Impact of the User-centric Principles.

	A	**B**	**C**
uni	0.5 Km	1.75 Km	3.0 Km
price	£150	£100	£50

(a) Extremeness Aversion.

	A	**B**	**C**
uni	0.9 Km	1.8 Km	3.0 Km
price	£125	£95	£90

(b) Trade-off Contrast.

In order to demonstrate the effect of the trade-off contrast and extremeness aversion, we use an example that is smaller than our running example, but also involving a choice among apartments. The apartments are now described only in terms of the distance from the university (uni) and price ($price$), both real numbers, whose domains are [0.5, 3.0] and [£50, £150]. The preferences provided by the user are two goals: minimise the value of both attributes, and these attributes are equally important. For showing the impact of extremeness aversion, let A, B and C be three options, whose attribute values are detailed in Table 6.9(a).

By calculating option costs and benefits, we find out that the option costs are amortised for all the options, when they are compared in a pairwise fashion, i.e.

- $AttCost(A, B, uni)$ = 0.5 and $AttCost(B, A, price)$ = 0.5,
- $AttCost(B, C, uni)$ = 0.5 and $AttCost(C, B, price)$ = 0.5, and
- $AttCost(A, C, uni)$ = 1.0 and $AttCost(C, A, price)$ = 1.0.

Therefore, if we calculate the value of one option with respect to another without taking into account the extremeness aversion principle, we conclude that options are equally good, as $d(o_1, o_2) = d(o_2, o_1)$ for all the options. However, if we calculate the distance from the best attribute values for each available option, we can verify that $bestDist(A, uni) = 0$ and $bestDist(A, price) = 1$, and therefore $ext(A) = 0.5$. B, as it has the intermediate values, has its extremeness evaluated to 0 (the best distance for both attribute values is 0.5), and C has the opposite values of A, having its extremeness also evaluated to 0.5. As a consequence, now A and C have a cost with respect to B, which is chosen as the optimal option.

Given that we showed the impact of the extremeness aversion isolated from the trade-off contrast, we now introduce another set of options to explain the impact of the latter, described in Table 6.9(b), using the same preferences. First, we calculate the $Cost$ function in order to identify the costs of each option, which is composed of the weighted sum of the costs identified for each individual attribute. The first rows of Table 6.10 show these calculated values — weights used are 0.5 for both uni and $price$, as they are equally important. Considering solely these costs, A is the optimal option, as it has less costs than B and C, and B is better than C.

By comparing the trade-off, i.e. the ratio between benefits and cost, when the cost is smaller than the benefit, we can see that the cost paid to choose A

Table 6.10: Options for Illustrating the Impact of Trade-off Contrast.

	A	**B**	**B**	**C**	**A**	**C**
uni	0.0	0.36	0.0	0.48	0.0	0.84
$price$	0.3	0.0	0.05	0.0	0.35	0.0
$Cost(o_1, o_2)$	0.15	0.18	0.025	0.24	0.175	0.42
$to(o_1, o_2)$	0.833		0.104		0.416	
$ToContrast(o_1, o_2)$	0.174	0.0	0.000	0.347	0.174	0.0
$v(opt_1, opt_2)$	0.156	0.135	0.019	0.267	0.175	0.630
Balance	**0.021**			**0.248**		**0.455**

instead of B is very high to get the provided benefits (0.833), in comparison to the trade-off between B and C (0.104), and A and C (0.416). Therefore, as people tend to adopt the trade-off contrast principle (Simonson and Tversky 1992), choosing A seems to be not a "good deal," because the costs are too high to get A's benefits, in comparison with other options.

So, by considering the average (0.451) of the to function as a threshold for considering the trade-off relation as a benefit or cost, we now have the value of $ToContrast$ shown in its respective row, which is incorporated into the option costs using our default weight (0.25). Finally, the optimal and chosen option is now option B. The adoption of this principle is consistent with our previous study (Chapter 2) as many of the participants pointed out in their preferences that their choice is based on a "good cost-benefit relationship."

As it can be seen in our provided examples, both the extremeness aversion and trade-off contrast are incorporated as costs or benefits using weights, in order to evaluate the decision value of an option with respect to another. However, the individual costs associated with extremeness aversion and trade-off contrast that are added to this value may not be high enough to make the order established by the $Costs$ function change. Even together, the extremeness aversion and trade-off contrast may still not be high enough to make this change. Therefore, it is the interplay among the option costs, extremeness aversion and trade-off contrast that specify which option of every two options is preferred, and consequently make it possible to determine the chosen option.

Considering our apartment example, if the $Costs$ function were the only factor taken into account, the apartment Ap_F would have been chosen — note that $Cost(Ap\_B, Ap\_F)$ = 0.09768 and $Cost(Ap\_F, Ap\_B)$ = 0.09760. Nevertheless, by considering the other two principles we are adopting, we have a different result. The costs of Ap_B and Ap_F are almost equal, but Ap_B is the least extreme option, while Ap_F is the most extreme: Ap_F has the best values for some attributes (uni, $zone$ and $stars$), but a high penalisation for others ($station$ and $price$). Moreover, by analysing the cost-benefit relationship between options, we identify that the relationship between Ap_F and Ap_B, which is 0.999, is worse

Table 6.11: Decision Function of the Apartment Decision Problem.

	B	**E**	**F**
Cost	0.231	0.047	0.232
ToConstrast	0.000	0.191	0.000
ExtAversion	0.017	0.015	0.000
d	0.141	0.078	0.139

(a) Ap_A

	A	**E**	**F**
Cost	0.218	0.207	0.098
ToConstrast	0.017	0.017	0.000
ExtAversion	0.000	0.000	0.000
d	0.135	0.129	0.059

(b) Ap_B

	A	**B**	**F**
Cost	0.034	0.221	0.222
ToConstrast	0.000	0.000	0.000
ExtAversion	0.000	0.002	0.000
d	0.021	0.133	0.133

(c) Ap_E

	A	**B**	**E**
Cost	0.225	0.098	0.212
ToConstrast	0.052	0.052	0.052
ExtAversion	0.042	0.060	0.057
d	0.154	0.080	0.149

(d) Ap_F

than the average 0.922. Table 6.11 shows the values of all functions calculated for every pair of options of the $Acceptable$ set, and by considering the weighted sum of the costs, trade-off contrast and extremeness aversion, the **chosen option is** Ap_B.

Our criterion to choose the optimal option is the balance between costs and benefits of each option with respect to another. As this is calculated in a pairwise fashion, there may be situations in which there is a cycle, that is, $d(o_1, o_2) < d(o_2, o_1)$, $d(o_2, o_3) < d(o_3, o_2)$ and $d(o_3, o_1) < d(o_1, o_3)$. This situation arises because we use different criteria to compare the attribute values of each pair of options, for example, the price of o_1 and o_2 is compared based on a goal, and o_1 and o_3 based on a monadic preference. As user preferences are consistent, if we analyse only the price attribute, we will find no cycles; however, as the d function is calculated in a different manner according to different preferences, small differences in the scales may lead to a cycle when considering the overall option costs.

In order to choose one option when this situation occurs, we adopt the following strategy. We identify the set of options that are considered better than the highest number of options, and then, from these, we choose the one with the minimum-maximum balance for every option that is considered better than it. In other words, if option o can be chosen as the optimal option, and for every option o' that $d(o, o') > d(o', o)$, we calculate the maximum value for the difference between $d(o, o')$ and $d(o', o)$. And if this value is the lowest one when compared to the value of every other candidate option (options that are better than the same amount of options of o), than o is chosen.

In our experiment (see next section), which was performed with real user data, there was only 1 (of 113) occurrence of cycle. Even though this number is low, we use a workaround to solve this issue, and it is part of future work to completely eliminate the possibility of cycles.

Finally, we discuss the complexity of our technique. As it can be observed in

Table 6.12: Complexity Analysis of our Technique.

Pre-processing	
PSM	$O(\mid Opt \parallel P_e \mid)$
OAPM	$O(\mid Opt \mid^2 \mid Att \parallel P_e \mid)$
AVPO	$O(\mid Opt \mid^2 + \mid P_e \mid^2)$
Explication	$O(\mid Opt \mid^2 \mid Att \parallel P_e \mid)$
Elimination	$O(\mid Opt \mid^2 + \mid Opt \parallel Att \mid)$
Selection	
Cost	$O(\mid Opt \mid^2 \mid Att \mid + max(PP_i) \mid PP_i \mid + \mid P_i \parallel Att \mid)$
Extremeness Aversion	$O(\mid Opt \parallel Att \mid + \mid Opt \mid^2)$
Trade-off Contrast	$O(\mid Opt \mid^2)$
Decision Function	$O(\mid Opt \mid^2)$

the presented algorithms, our technique runs in polynomial time, and most of the algorithms require comparing each pair of options according to each attribute. We present the complexity of each part of our technique in Table 6.12, whose total is

$$O(\mid Opt \mid^2 \mid Att \parallel P_e \mid + \mid P_e \mid^2 + max(PP_i) \mid PP_i \mid + \mid P_i \parallel Att \mid) \quad (6\text{-}30)$$

where

- Opt is the set of available options,
- Att is the set of attributes,
- P_e is the set of preferences,
- PP_i is the set of preference priorities,
- $max(PP_i)$ is the maximum number associated with the preference priorities, and
- P_i is the set of attribute priorities and attribute indifferences.

6.8 Comparison with Related Work and Evaluation

One of the key advantages of our approach is the ability to handle different types of preferences. In this section, we thus compare our technique with existing approaches to reasoning about preferences in terms of the preference types they can handle. We also evaluate our approach empirically by comparing our choices with those of a human expert. As the input of our technique is high-level preferences, and existing approaches cannot handle all of them, our empirical evaluation does not make side-by-side comparison with existing work.

As discussed in previous chapter, many existing approaches are based on *utility functions (UFs)*. As with UFs it is possible to order available options, thus choosing among them, different approaches (McGeachie and Doyle 2008,

Table 6.13: Reasoning Approaches vs. Preferences.

Approach	Preference								Priority		
	Cd	Ct	G	O	Q	R	I	D	a	i	p
UF-based (McGeachie and Doyle 2008)				X							
SVM-based (Domshlak and Joachims 2007)				X		X	X				
SCSP (Bistarelli et al. 1997)		X									X
Bipolar preferences (Bistarelli et al. 2010)		X									X
Interval-valued SCSP (Gelain et al. 2010)		X									X
CP-Nets (Boutilier et al. 2004)	X			X							
TCP-Nets (Brafman et al. 2006)	X			X					X		
Scoring Function (Agrawal and Wimmers 2000)		X				X		X			X
Winnow (Chomicki 2003)		X		X			X				X
BMO (Kießling 2002)		X		X		X					X
Query Personalisation (Koutrika and Ioannidis 2006)						X					X
SPARQL (Siberski et al. 2006)		X	X								X

Legend — Cd: condition; Ct: constraint; G: goal; O: order; Q: qualifying; R: rating; I: indifference; D: don't care; a: attribute priority; i: attribute indifference; p: preference priority.

Domshlak and Joachims 2007) have been proposed to transform specific models that capture qualitative preferences (which are closer to how users express preferences) into UFs, i.e. quantitative preferences, which are consistent with the constraints established by the qualitative preferences. Some approaches (Bistarelli et al. 1997) extend Constraint Satisfaction Problems (CSPs) to incorporate *soft constraints* (that can remain unsatisfied), namely SCSP, associating a penalty (or preference) with each constraint, and creating an optimisation problem of minimising penalty (or maximising preference). In order to allow the representation of other kinds of preferences, extensions to traditional soft constraints approaches were proposed: (i) bipolar preferences (Bistarelli et al. 2010), which distinguish what users want (indicating preferred options), and what they do not want (restricting the set of acceptable options); and (ii) use of intervals (Gelain et al. 2010) to represent penalties (or preferences), as it may be difficult to specify precise values. A third group of approaches, mainly represented by CP-Nets (Boutilier et al. 2004) and TCP-Nets (Brafman et al. 2006), takes another direction, proposing new *graphical structures* to represent and reason about qualitative preferences. Finally, work in the area of databases proposes *extensions of query languages* (Agrawal and Wimmers 2000, Chomicki 2003, Kießling 2002, Koutrika and Ioannidis 2006, Siberski et al. 2006) to incorporate preferences and algorithms to provide query results according to specified preferences. Even though these approaches propose different solutions, they share the common goal of making a choice based on preferences. However, as shown in Table 6.13, they address limited kinds of preferences, restricting their natural expression by humans. Our technique is the only one that exploits natural language expressions — expressive speech acts — to make decisions on behalf of users.

Besides being able to deal with restricted kinds of preferences in comparison to our technique, these approaches only choose between two options when the preferences provided are sufficient to make the decision, i.e. if the decision involves

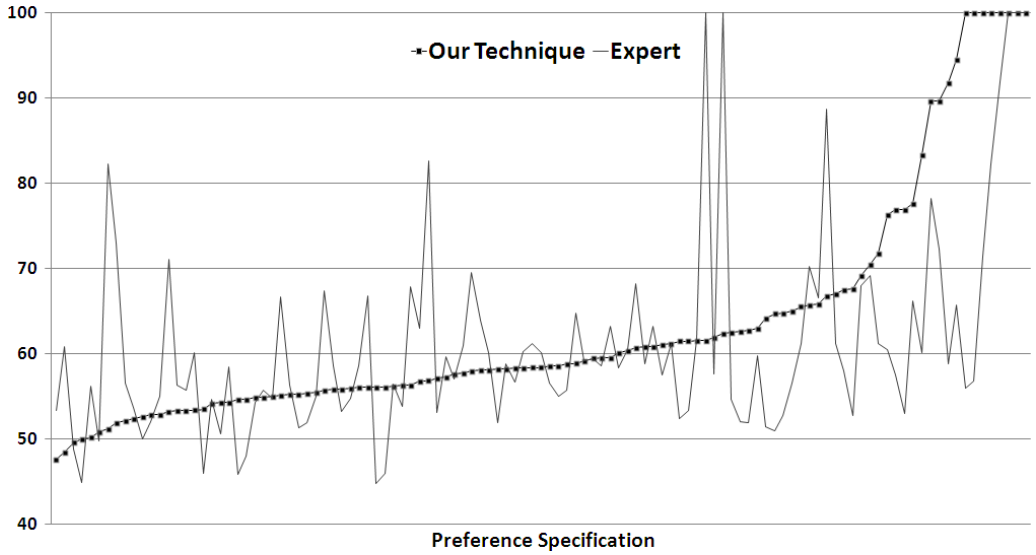


Figure 6.6: Expert vs. our technique: first choices.

trade-off, users must have previously resolved it and specified their preferences. However, as discussed by Tversky (Tversky 1996), people resolve trade-offs in light of available options, and do not provide such preferences. Our technique, on the other hand, resolve trade-offs using (i) preferences over individual attributes; (ii) priorities; and (iii) user-centric principles, with the aim of performing a similar reasoning that humans do.

Finally, note that this thesis is not concerned with preference elicitation methods, as our goal is to provide a preference language as close as possible to natural language, so that users can directly express their preferences, and based on such preferences make a choice. However, we do not exclude the possibility of combining our approach with elicitation methods, which can be simpler if the gap between provided user preferences and the preference model (or language) in which elicited preferences are captured is reduced, as our approach proposes.

The empirical evaluation of our technique is based on the study that also informed the preference language itself (Chapter 2). Participants provided preference specifications (in natural language) for use by an individual to buy a laptop on their behalf. Both these individuals, and domain expert, were given a laptop catalogue (with 144 laptops) from which to choose up to five options. The three relevant parts of the study used for our evaluation are the initial preference specification, the user choices and the domain expert recommendation. We use these to compare our decisions against those of the user and domain expert. Similarly to how the domain expert recommendation was assessed in the study, we calculate a similarity score SS, which compares the recommendation with the user choice, using Equation 2-1 — it takes into account the position of the up to five chosen laptops using a weighted average. SS ranges from 0 to 100, with 100 indicating a

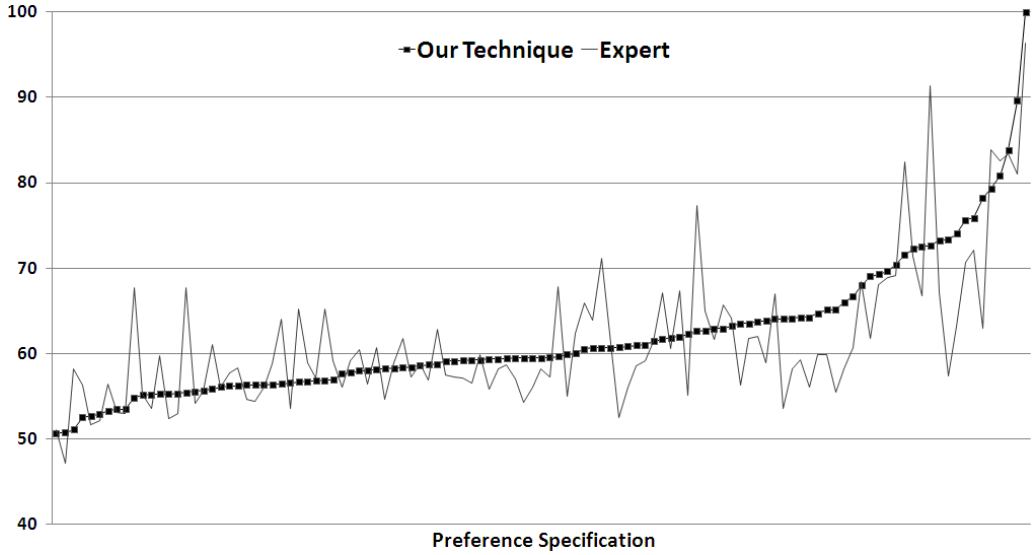


Figure 6.7: Expert vs. our technique: up to five choices.

match to user choice.

Using a graphical user interface developed to input preferences according to our preference language, we were able to store the preferences provided by 113 participants. Of the 192 user specifications, 79 (41%) use subjectivity or purpose, and therefore cannot be expressed in our language. For example, "*I'd like a laptop to carry on my backpack*." Moreover, of these 79, 9 have no expert recommendation, as they are too vague, such as "*I would never delegate this task [buy a laptop] to another person*." For the remaining specifications, we applied our technique (which takes an average of 3.6026 $seconds$ on an Intel Core i5 2.30GHz, 8GB of RAM, with standard deviation 1.4051, to be executed for each request, with 144 laptops, and 61 attributes), and obtained the similarity scores shown in Figures 6.6 and 6.7. The former shows the similarity score considering the first expert choice and the first choices of our technique compared to the first user choice, and the latter shows the comparison between the first up to five choices. If the domain expert recommended x laptops, we use the first x choices of our technique. Even though our technique does not rank acceptable options, as we calculate a numeric value to compare them, we use these numbers to obtain the first up to five choices. Both charts have their *x-axis* ordered according the SS calculated for our technique.

The results show that the values obtained for the (human) domain expert and our technique are not so different — considering only the first choices (labelled as *F*), our technique has M_{SS} = 63.19 and the domain expert has M_{SS} = 61.25; i.e. our technique has a better average SS than the expert. The same occurs for the up to five choices (labelled as *5*). These SS values, as well as standard deviation, minimum and maximum, are summarised in Table 6.14. The difference between the obtained similarity scores is *significant* when comparing only first choices, as determined by

Table 6.14: Analysis of Domain Expert and our Technique choices.

	Our Technique (F)	Expert (F)	Our Technique (5)	Expert (5)
Average	63.19	61.25	61.94	61.44
Standard Deviation	13.36	11.93	8.00	8.32
Minimum	47.68	44.80	50.72	47.12
Maximum	100.00	100.00	100.00	96.39

a Wilcoxon Signed Ranks test — $W(112) = 3711, p = 0.0497$ (F), and thus we reject the null hypothesis that domain expert and our technique choices are equal. However, this is not the case of the up to five choices: $W(112) = 3774, p = 0.1131$ (5). As a consequence, *we can conclude that our technique makes choices at least as good as those of the domain expert*.

6.9 Final Remarks

Previous studies (Lichtenstein and Slovic 2006) state that people have a set of preferences that they are aware of, which are used as a basis to make decisions and, when facing concrete decision making situations (Tversky 1996), they construct new preferences to resolve trade-offs that cannot be resolved with this set of known preferences. In this chapter, we presented an automated decision making technique that uses preferences expressed by users in a high-level language, which is close to how people express their known preferences in natural language. The technique resolves trade-offs based on priorities, which indicate attributes they consider more important, combined with user-centric principles, thus making decisions in a way similar to how humans do, with the aim of automating tasks on their behalf. Our technique makes a decision in a stepwise fashion: first, it evaluates preferences over individual attributes; second, it eliminates dominated options and those that do not satisfy cut-off values, obtaining a consideration set, which contains options that require trade-off resolution; third, it chooses one option from the consideration set evaluating option pros and cons, trade-off contrast and extremeness aversion, being the latter two the main principles from psychology we adopt. The empirical evaluation of our technique showed that it is able to make a choice on behalf of the users at least as good as that made by a human domain expert, considering our experiment.

Part III

User Explanation

As shown in the previous part, our goal is to provide a high-level way for users to specify their preferences so that an agent, provided with the ability to reason about these preferences and making decisions, can act on their behalf or support their decisions. However, users are willing to delegate their tasks to personal agents, only if: (i) they know exactly what the agent is going to do (Schiaffino and Amandi 2004); or (ii) users have built trust in the agent based on previous interactions with it. However, there is still lack of an in-depth investigation of the concrete system design features that could be developed to promote user trust (Chen and Pu 2010).

One of the most investigated ways of increasing user trust in their personal agents, which may be an expert or recommender system, is providing agents with the ability of giving explanation for users to justify their decisions. When an autonomous agent makes decisions based on user preferences, which may be complex, conflicting, and changing, it can be obscure and difficult for the user to understand and know why the agent has made certain decisions. Without understanding whether there is a rationale behind an action, the user is unlikely to trust the agent to act on her behalf, and consequently accept the system.

We begin this part by presenting a literature review (Chapter 7) of different approaches that aim at providing explanations for users about decision making and recommendation methods in order to increase user trust in decisions and recommendations. Then we describe a study (Chapter 8) performed to identify the kinds of arguments that people make to support a decision. This study allowed us to derive explanation templates and guidelines to be addressed by decision making methods and recommender systems. Finally, we present an approach (Chapter 9) for generating explanations, which follows the proposed templates and guidelines, and which is based on execution traces produced by our decision making technique described in the previous part.

7
Background on Explanation Approaches

The need for explaining why systems perform in a certain way or make particular decisions has emerged with the popularity of Expert Systems (ESs) in the 80's. More recently, explanations have been explored in the context of other areas that involve decision making, such as recommender systems. In this chapter we review different approaches proposed for justifying decisions made by computer systems by means of explanations. First, we overview how explanations were addressed in the context of ESs (Section 7.1). Next, we discuss explanation in recommender systems (Section 7.2), which are considered the successors of ESs (Tintarev and Masthoff 2007). Then, we describe approaches that focus on explaining relaxations performed in over-constrained problems (Section 7.3), followed by general approaches for explanations of decision making techniques based on multi-attribute preference models (Section 7.4). Finally, we conclude in Section 7.5.

7.1
Expert Systems: the Roots of Explanation

Expert Systems (ESs) are computer programs whose aim is to automate human decisions of a particular application area, and incorporate expert knowledge of this area. An ES consists of (a) a reasoning engine, (b) knowledge about the domain (which is typically captured in the form of rules), and (c) a knowledge base, which together are able to make decisions in a similar way that humans do, such as medical diagnostics. With the popularity of such systems, researchers and software developers observed that systems able to make decisions without giving an explanation of how an answer is obtained are very unlikely to be accepted, as users want to understand the rationale behind the choice to trust in it, and possibly to be able to justify the decision for other individuals. This intuition of the need for explanations is confirmed by the empirical study performed by Ye and Johnson (Ye and Johnson 1995), which concluded that explanation facilities indeed have a positive impact on the user acceptance of an ES's advice.

Different ways of providing explanations have been proposed, which were classified by Nakatsu (Nakatsu 2006) into three categories: (i) reasoning traces;

(ii) strategic knowledge; and (iii) deep justifications. The first (*traces*) is the most straightforward to be obtained. Most of the existing ESs are rule-based systems, and therefore the reasoning consists of applying rules according to a specific input in order to obtain a result. Each rule application can be seen as a trace of the reasoning process, which can be used to explain the decision made. These traces can be used to answer different types of questions made by user, such as "*why*" (why the system is asking a particular kind of information), "*how*" (how the conclusion was obtained), "*why not*" (why the system did not reach a particular conclusion). The first two types of questions are widely known, as they were addressed by one of the most famous ESs, namely MYCIN (Buchanan and Shortliffe 1984). More sophisticated forms of this approach consist of transforming computer-generated code to natural language, so that users can better understand displayed rules, and also omitting details according to the user knowledge.

The second type of explanation, *strategic* knowledge, consists of making the problem-solving strategies explicit to the end user. Therefore, the explanation is less detailed than with the use of reasoning traces, and solely the overall line-of-reasoning is made visible to the end user. What is presented for the users is a strategic knowledge structured for the end user, which can be a tree of goals, or a tree of topics. The third explanation type, deep *justifications* (also known as "*canned text*"), is an explicit description of the causal argument or rationale behind each inferential step taken by the ES. In this case, an explanation is tied to an underlying domain model that provides structural knowledge and taxonomic knowledge about the domain.

Besides showing the impact of explanations on the user acceptance of ESs, the study of Ye and Johnson (Ye and Johnson 1995) also investigated the impact of these three different types of explanation, and concluded that the most effective one is *justifications*. Nevertheless, it has the drawback of requiring developers to predict questions that users are going to make, and also codify answers, in order to produce human-like explanations.

Many ESs were implemented as a proof of concept of many of the ideas related to ES explanations, and they are implementations of these three kinds of explanations introduced above. For further information about implemented ESs that support explanation, we refer the reader to the review made by Lacave and Diez (Lacave and Diez 2004).

In addition, there are works that focus on other aspects of the incorporation of explanations into ESs, such as the kinds of arguments to be presented to users. Bohanec and Rajkovič (Bohanec and Rajkovič 1993) have proposed a different form of reasoning for ESs, which integrates ideas of multi-attribute decision making. They define the knowledge representation with a tree of criteria. Leafs

of the tree are referred to as *basic* criteria, and the remaining ones are called *aggregate* criteria. For example, as illustrated in the paper, an aggregate criterion, which is part of the criteria tree for the evaluation of microcomputer hardware, is "Basic Communication," which is composed of two basic criteria "Keyboard," and "Monitor." Each of these criteria is associated with an evaluation domain, such as $\{bad, acceptable, good, excellent\}$, and aggregate criteria are specified in terms of the values given for basic criteria, for instance, if a computer has a *good* monitor and an *unaccept* keyboard, the basic communication is specified as *unaccept*. With these "preferences" specified, one can make a decision to chose the "best" option, and give an explanation, based of the criteria that makes one option better than another. This approach has the limitation of not addressing how this information is obtained — as the authors state, they left the *knowledge acquisition* out of the scope. Nevertheless, their approach requires the specification of how each basic and aggregate criterion is evaluated, and this involves a combinatorial problem, thus requiring this information from users is infeasible.

On the other hand, some works have given attention to the architecture of ESs integrated with explanation ability. Shankar and Musen (Shankar and Musen 1999) have proposed a multi-agent framework, named WOZ, that provides explanations by using arguments. The framework proposes the use of a set of components, each having a particular responsibility in the explanation process, such as a user model that specifies the characteristics of the current user (e.g. the user's protocol expertise, domain expertise, familiarity with patient data, and visual preferences), and the explanation strategy, which contains the situation-action mappings. A "director" component is responsible for orchestrating all the other components. The framework proposes to make the explanation visualisation independent from its generation, but it is abstract in that *it is does not specify how the explanation is generated.*

The approach consists of having a base of *meta-arguments*, which conceptualise arguments for a *class of claims*. The elements of the meta-argument structure are stated with abstract descriptions. These meta-arguments are used to produce *concrete arguments* (instances of meta-arguments), which define arguments for a specific claim of a class of claims. Meta-arguments, and consequently arguments, follow a particular structure — the Toulmin's Argument Structure (Toulmin 2003) — whose components are detailed in Table 7.1.

El-Beltagy et al. (El-Beltagy et al. 1999) focused on an architecture based on agents, which separates different concerns related to explanation. The agents related to the explanation generation are: (i) terminology server, which provides terminology definitions and abstractions, as well as detailed term knowledge within a given predefined domain; (ii) "why" agent, which is responsible for justifying

Table 7.1: Toulmin's Argument Structure.

Component	Definition
Data	the particular facts about a situation on which a claim is made
Warrant	the knowledge that justifies a claim made using the data
Backing	the general body of information or experience that validate the warrant
Qualifier	the phrase that shows the confidence with which the claim is supported to be true
Rebuttal	the anomaly that shows the claim not to be true
Claim	the assertion or conclusion put forward for general acceptance

the question asking strategies using familiar domain-oriented knowledge; and (iii) domain-oriented justification agent, which generates a "plausible story" from the given inputs that justify the conclusion. The proposed approach does not detail how the responsibilities given for each agent are performed, and therefore the explanation generation itself is not described.

Said et al. (Said et al. 2009) argued that the form adopted to provide reasoning for ESs (or problem solving method) is tightly coupled to how explanations are generated, so they proposed a methodology for automatically generating explanations during and at the end of the reasoning process, considering different knowledge representation schemes, which are "generate and confirm hypotheses," and the "routine design generic task." They show, in an *abstract way*, which kind of information should be displayed as an explanation for each of these two approaches, for two different explanation primitives: (i) "why asking," in which users want to get an explanation for why the system is asking a certain question; and (ii) "how," so at the end of the consultation the user may want to get an explanation of how the system reached a certain decision or conclusion.

As we mentioned before, these works focused on general approaches for explanation (not on the implementation of particular systems) and architectures; however, they are abstract in a way that the explanation generation still remains as a challenge.

7.2 Explanation in Recommender Systems

Some researchers have referred to *recommender systems* as the successors of ESs (Tintarev and Masthoff 2007). Since their beginning, approaches for recommender systems have aimed at increasing the accuracy of such systems, in order to evaluate their quality. More recently, researchers have perceived the relevance of explanations, as it was the case with ESs. Tintarev and Masthoff (Tintarev and Masthoff 2007, Tintarev and Masthoff 2011) have performed a survey on explanations in recommender systems and have identified seven different aims related to approaches proposed for explanation, which are shown in Table 7.2.

Recommender system approaches are classified into three categories,

Table 7.2: Aims of Recommender Systems (Tintarev and Masthoff 2007).

Aim	**Definition**
Transparency	Explain how the system works
Scrutability	Allow users to tell the system it is wrong
Trust	Increase users' confidence in the system
Effectiveness	Help users make good decisions
Persuasiveness	Convince users to try or buy
Efficiency	Help users make decisions faster
Satisfaction	Increase the ease of usability or enjoyment

which are associated with the technique used to produce recommendations (Adomavicius and Tuzhilin 2005): (i) *content-based approaches*, which recommend to users products that are similar to the ones the user preferred in the past; (ii) *collaborative approaches*, which recommend items that people with similar tastes and preferences liked in the past; (iii) *hybrid approaches*, which combine collaborative and content-based methods. In addition, some approaches are said *preference-based*, as they aim at predicting an implicit ordering among products instead of predicting a rate that the user would give to the product.

Explanation approaches that are proposed for recommender systems address the different presented aims by providing explanations that show the rationale behind the adopted recommendation approach. For instance, if a collaborative approach is adopted, the user may receive as an explanation a histogram of ratings of the product given by similar users. In addition, simple clarifications, such as "*people who bought this product also bought X*," are considered effective.

There are other approaches that look for products similar to a specification (i.e. requirements or constraints) given by users. McSherry (McSherry 2005) focused on case-based reasoning approaches, in which products are seen as cases from which one should be selected when it is similar to the case (or specification) provided by the user. On each stage of his approach, McSherry asks the user a value for a new product attribute and presents the case that is more similar to the current specification. In addition, users can ask why a certain attribute value is asked, and the approach answers it based on candidate cases, indicating which cases would be selected or rejected based on the user answer. Moreover, in certain cases, the answer might lead to the termination of the dialog.

Another direction is explanation interfaces, which were investigated by Pu and Chen (Pu and Chen 2007). These interfaces organise recommended products in a way that make trade-off situations to be resolved explicit for users, thus facilitating the decision making process. For example, in systems that recommend digital cameras, a group of recommended products can be classified as "cheaper, but with lower resolution."

Recommender systems have been significantly improved by having

explanations incorporated into recommendations, but explanation approaches are attached to the approach used to recommend products and do not produce arguments that people adopt to choose or reject options. Stating that "someone like you chose this product" or "you like similar products" is not enough to justify a recommendation. The attachment between the recommendation approaches and explanation types, which typically have an one-to-one mapping, has been criticised by Papadimitriou et al. (Papadimitriou et al. 2011), and they proposed a new taxonomy for explanation types that is based on humans and (product) features. However, the new taxonomy classifies explanations in a different way, but does not solve the problem of producing explanations that only justify the recommendation and not giving the kinds of arguments that users are expecting.

Decoupling the recommendation approach from the explanation was investigated by Zanker and Ninaus (Zanker and Ninaus 2010). They argue that current recommender technology cannot be easily used for explaining why an item is proposed, because the semantics of the model gets lost when factorising and rotating matrices, which are the ideas behind recommender approaches. On the other hand, knowledge-based mechanisms that are associated with a transparent line of reasoning typically do not reach comparable accuracy levels as collaborative mechanisms do. So, the authors propose to decouple reasoning for recommendations from generating explanations. Their approach relies on a layered directed acyclic graph, whose certain paths from a start node to an end node constitute explanations. Even though the idea behind their approach (i.e. decoupling the recommendation from the explanation) is relevant, their briefly described proposal seems to be untractable as the graph is composed of all possible values of the product features (or attributes), and no further discussion is made in this direction.

Finally, content-based and collaborative-based approaches have explored application domains that involve products that people consume in a regular basis, such as movies, books, music, restaurants and news. However, they might be not suitable for the scenario we are exploring, in which people can express preferences for a decision that is not necessarily so frequently performed and therefore there might be not enough data to extract recommendations. Moreover, as Zanker and Ninaus discussed, the models relying recommender system approaches allow only explaining *how* a recommended items is chosen, but not *why*.

7.3 Explanations for Over-constrained Problems

A particular class of decision problems is Constraint Satisfaction Problems (CSPs), which involve imposing a set of hard constraints, i.e. constraints that must

be satisfied, and identifying a solution that satisfies all constraints. In situations that this is not possible, that is, the problem is over-constrained, one or more constraints that conflict with each other must be relaxed in order to find a solution for the problem. When a solution is automatically chosen for an over-constrained problem, this solution must be presented for users together with an explanation that justifies why a particular set of constraints was chosen to be relaxed. However, over-constrained problems can have an exponential number of conflicts, which explain the failure, and an exponential number of relaxations, which restore the consistency. So, in this context, there are two main issues: how to choose a relaxation, and how to perform this in an efficient way.

Junker (Junker 2004) has tackled this problem by defining preferred explanations and relaxations based on user preferences between constraints and computing them with a generic method, which not only works for CSPs, but also for any satisfiability problem such as propositional satisfiability or the satisfiability of concepts in description logic. The relaxation problem is defined as $\mathcal{P} := (\mathcal{B}, C)$, where $\mathcal{B}$ is the background containing the constraints that cannot be relaxed, and C is a set of constraints. A subset $\mathcal{R}$ of C is a relaxation of a problem $\mathcal{P}$ if and only if $\mathcal{B} \cup \mathcal{R}$ has a solution. In order to choose a relaxation, Junker assumes the existence of a strict partial order between the constraints of C, denoted by $\succ$, as a user typically prefers to keep the important constraints and to relax less important ones. As $\succ$ is an incomplete specification of ranking among constraints, three extensions are defined in order to produce a total order among constraints, and based on this total order, some conflicts are more relevant for the user than other conflicts. Because of that, Junker is able to define *preferred relaxations* and the analogous definition for *preferred conflicts*. An algorithm based on a divide-and-conquer approach, named *QuickXplain*, is proposed, which has a polynomial response time for polynomial CSPs. Other works have followed the ideas proposed by Junker, and proposed new approaches that aim at optimising the conflict detection in CSPs, such as the *FastXplain* (Schubert et al. 2010).

O'Sullivan et al. (O'Sullivan et al. 2007) argue that many other existing approaches to explanation generation in constraint-based settings are based on the notion of a (setwise) minimal set of unsatisfiable constraints, also known as a minimal conflict set of constraints. However, explaining a relaxation by stating that a conflict is minimal can be not intuitive for users, spurious or misleading (Friedrich 2004), and therefore users need more than one explanation in order to avoid drawing false conclusions. In order to produce better explanations, the notion of *representative set of explanations* is defined, which means that every constraint that can be satisfied is shown in a relaxation and every constraint that must be excluded is shown in an exclusion set. For finding these representative explanations,

the algorithm named *RepresentativeXplain* was proposed, which was evaluated in random and real world scenarios.

These works focus on a particular (but relevant) problem, but they are not sufficient for producing explanations for our problem, as it is not a CSP, and involves many other types of preferences besides constraints. In addition, choosing a minimal or less preferred set of constraints to relax can be the wrong choice in many cases, as users may prefer to relax many constraints and do not compromise too much few of them.

7.4 Explanation for Multi-attribute Preference Models

Some approaches have focused on explanations for Multi-Attribute Utility Theory (MAUT), considering utility functions for individual attributes and weights used to sum them, which represent the trade-off among these attributes. Klein and Shortliffe (Klein and Shortliffe 1994) have proposed the *Interpretive Value Analysis (IVA)*, which is a framework for explaining and refining multiattribute value functions automatically. The main idea of the explanation relies on two main concepts: (i) COMPELLINGNESS, which is the weighted difference between the values of a particular attribute (the approach actually considers objectives, which are represented in a tree, in which upper levels are higher-abstraction objectives) of two options and represents how strong an attribute is; and (ii) NOTABLY-COMPELLING, which evaluates if the COMPELLINGNESS is higher than k standard deviations above the average of the values of a particular attribute — "k" determines the degree to which the magnitude of the COMPELLINGNESS of a particular attribute must be an outlier to be considered NOTABLY-COMPELLING with respect to an option, and this parameter might be adjusted for particular users. Based on these two concepts, an explanation is built containing only attributes that are relevant, i.e. when they are "notably compelling." Moreover, Klein and Shortliffe also propose a way of translating all the numerical analysis to make a decision to natural language, so that a user that wants a detailed explanation for the decision can understand the reasoning process. We show below two explanations following this approach, which justify why an option is better than another. The first indicates that only the best of the two options has compelling attributes, while in the second both options have (but in the end SHELL.B is better than SHELL.C). Explanation variables are in *italics*.

- *Price* provides the most compelling reason for the choice.
- *Quality of documentation and quality of front end* are reasons to prefer *SHELL.B* over *SHELL.C*. *Reliability, interactive development facilities, and syntactic familiarity to data processing programmers* are reasons to prefer *SHELL.C* over *SHELL.B*.

A computational model, named Generator of Evaluative Arguments (GEA), was proposed by Carenini and Moore (Carenini and Moore 2006) in order to cover all aspects of generating evaluative arguments in a principled way, by effectively integrating general principles and techniques from argumentation theory and computational linguistics. The approach assumes a previously elicited additive multi-attribute value function (AMVF), which captures user preferences, and based on it, evaluative arguments are generated to support (or not) an option. The argument generation process is divided into two parts — the first consists of selecting arguments to be presented and the second transforms these abstract selected arguments into natural language. As particularities of text generation in natural language is not our concern, we will focus on the first part. Carenini and Moore give seven guidelines to select and produce arguments, such as how to distinguish supporting and opposing arguments based on AMVFs and when an argument is *compelling* (adopted from Klein and Shortliffe's work), and these guidelines are used to create an argumentation strategy, which generates the input of a text generation technique. The approach was evaluated empirically, by testing if tailoring an evaluative argument to the user preferences increases its effectiveness, and if differences in conciseness significantly influence argument effectiveness. While the second hypothesis was confirmed in the experiment, the first one was only marginally confirmed. Note that the most important part to generate explanations rely on Klein and Shortliffe's work.

An approach for selecting and generating arguments for the family of multi-attribute decision models parameterised by weights assigned to the criteria, such as expected utility model and the weighted majority model, was proposed by Labreuche (Labreuche 2011). His approach is based on the analysis of the values of the weights together with the relative scores of the options to be compared. For generating explanations, Labreuche proposes a set of *anchors*, which are a generic way of reasoning in the explanation and look for some changes in the weight vector v that yield an inversion of the prescription made by the decision model. The explanation focuses then on the criteria for which the weight vector has changed. Two strategies for the modification of the weights are considered: the replacement of v by some reference weights $w^{\mathcal{F}}$, and a permutation of the weights v among the criteria (which is associated with a branch-and-bound algorithm). A trivial anchor addresses the case of domination, and another last anchor covers the remaining cases. Examples of each anchor are presented in Table 7.3.

These approaches provide a substantial work on explanation generation for decision making models, with different types of arguments. Nevertheless, they still present limitations on generated explanations, as they do not cover many arguments that users adopt, such as the existence of cut-off values. In addition, there are

Table 7.3: Labreuche's Approach Examples.

Anchor "all"
y is preferred to *x* since *y* is better than *x* on ALL criteria.
Anchor "not on average"
Even though *x* is better than *y* on average, *y* is preferred to *x* since *y* is better than *x* on the criteria *selected pros* that are important whereas *y* is worse than *x* on the criteria *selected cons* that are not important.
Anchor "invert"
y is preferred to *x* since *y* is better than *x* on the criteria *selected pros* that are important and on the criteria *selected pros* that are relatively important, *x* is better than *y* on the criteria *selected cons* that are not important and on the criteria *selected cons* that are not really important, and [criterion *j* for which *y* is better than *x* is more important than criterion *i* for which *y* is worse than *x*] *for all* (i, j) *selected*.
Anchor "remaining case"
y is preferred to *x* since the intensity of preference *y* over *x* on *pros* is significantly larger than the intensity of preference of *x* over *y* on *cons*[, and all the criteria have more or less the same weights].

two main approaches for selecting arguments to be presented in the explanation (decisive criteria) — the use of a threshold (COMPELLINGNESS) and weight analysis — however, there is no consensus which of them (if any) selects the decisive criteria.

7.5 Final Remarks

In this chapter, we presented a review of works that propose approaches for generating explanations that justify decisions made by a software system. The need for explanations emerged from Expert Systems (ESs), which aim at supporting humans to make decisions and need explanations mainly for critical domains, such as medical diagnosis. Even though approaches for ESs advanced this research area, they rely on a huge set of rules that capture domain knowledge, and the bootle neck of this kind of system is still the knowledge engineering. Recommender systems, sometimes seen as the successors of ESs, are now incorporating explanations in their recommendations, but they do not provide explanations for and against recommended products, but only make the recommendation process more transparent for users. This kind of explanation increases the acceptability of recommender systems, but in many cases are not helpful for users to make a decision. In the same way, explanations are produced for justifying relaxations of over-constrained problems; however, this kind of explanation has some of the problems of recommender systems.

Promising, and more general, approaches have been proposed for explaining the result of multi-attribute preference models, using weights to specify trade-off among attributes. These approaches produced substantial results in the explanation generation. However, there are still many challenges to be overcome to produce explanations that support people to make decisions — such challenges include

a deep investigation of the kinds of arguments that are helpful, which will be investigated in the next chapter.

8
Guidelines and Patterns for Explanations

As presented in the previous chapter, there are many challenges that need to be overcome in order to produce effective explanations that support decision making performed by software systems. One of them is the identification of the kinds of explanation that users expect and need to understand the rationale behind decisions made in order to accept the them. In this chapter, we address this issue by describing an exploratory study performed to capture patterns that describe how people justify their choice among a set of available options. As we assume that the explanations provided by people are those that users expect to receive, we derive from our study results a set of guidelines and patterns, which serves as a basis for approaches whose aim is to explain to users why a particular option is chosen based on multi-attribute decision-making. We first describe this study in Section 8.1, then detail and discuss its results in Section 8.2 and our interpretation in Section 8.3. Finally, we present the guidelines and patterns derived from this study in Section 8.4, concluding in Section 8.5.

8.1
Study Description

The exploratory study presented in this chapter follows the same overall procedure adopted in our previous study, which was presented in Chapter 2. The framework that provided guidance for the study elaboration was proposed by Basili et al. (Basili et al. 1986), which includes the GQM template (Basili and Rombach 1988), used to define the goal of the study — and later to define research questions and select metrics for answering those questions. The goal of the present study, following the GQM template, is presented in Table 8.1. As highlighted previously, the work of Basili et al. focused on experimentation in the context of SE, but it is sufficiently generic to be applied to other areas, and the reason for using their guidelines and template is due to the experience of the researchers with SE studies.

In order to achieve this goal, the study we designed is based on a questionnaire made available online, in which participants (individuals that are part of the social network of the researchers) had to make a choice from a set of available options

Definition element	Our experiment goal
Motivation	To identify the kinds of explanations users expect to receive from software systems,
Purpose	characterise and evaluate
Object	explanations to justify a choice
Perspective	from a perspective of the researcher
Domain:people	as they are provided by people
Scope	of the context of the social network of the researcher.

Table 8.1: Goal Definition (GQM template).

and later justify their decision. The arguments given by participants were carefully analysed to understand their common characteristics and also the dependency between options and the arguments given. The options given in our study consist of hotels located in New York City, USA, and this decision was made due to three reasons.

(i) *New York is a widely known touristic city*, therefore participants are more likely to have a broad idea of close to where they would like to stay, prices they are accepting to pay, and so on.

(ii) *Researchers knowledge about the city*, thus we are able to select appropriate options for being part of the study.

(iii) *Massive amount of available hotels*, which is important as our study is based on real hotel data so participants take it more seriously.

The next sections provide further details about our study. We start by presenting the research questions of the study in Section 8.1.1, and then detail the study procedure in Section 8.1.2. We describe the participants of our study in Section 8.1.3, to later proceed to the analysis and interpretation of the results.

8.1.1 Research Questions

As shown in the previous section, our main objective while performing this study is to give guidance for explanation generation. It is accepted by the research community that explanations improve software systems that automate decision-making or provide recommendations by making its reasoning process or reasons for the choice more explicit, thus enhancing user acceptance and trust. Nevertheless, there is no consensus on what constitutes a *good* explanation, and what kind of information must be provided to users. Therefore, the present study aims to solve this issue in a user-centric way by identifying the kinds explanation

RQ1. Do users use a pattern to justify an option chosen from the set of those available?
RQ2. Is there a relationship between the type of explanation given to support the decision and the chosen option?
RQ3. Do users use a pattern to justify the rejected (not chosen) options?
RQ4. Is there a relationship between the type of explanation given to reject options and the rejected or chosen option?

(a) Research Questions.

EA1. Analysis of the arguments given to justify the chosen option and identification of commonalities among arguments given by different users.
EA2. Comparison among the arguments given to justify each different chosen option.
EA3. Analysis of the arguments given to reject options and identification of commonalities among arguments given by different users.
EA4. Comparison among the arguments given to reject options according to each different chosen and rejected option.

(b) Evaluation Approaches.

Table 8.2: Research questions and their evaluation approach.

that people give — and we assume that they are those that users expect to receive — to justify a decision, and then providing guidelines and patterns that allow defining "good" explanations from a user perspective. The aim of such explanations is to expose to users why a system chose a particular option, thus improving *effectiveness* and user *trust* in the decision. In our study we addressed four different research questions, presented in Table 8.2(a).

By answering these research questions, we are able to extract patterns for user explanations to be generated by decision-making systems (based on *RQ1* and *RQ3*), and also the context in which each pattern is adopted (based on *RQ2* and *RQ4*). These explanations are associated with both chosen and rejected options — the first two questions focus on patterns and their context for explaining the chosen option; and the last two address explaining why other options were rejected (or not chosen).

8.1.2 Procedure

In a nutshell, our study consists of collecting information provided by participants through a web-based questionnaire, and later analysing the collected data. Our aim was to obtain a high number of participants, and therefore anyone with Internet access could access the questionnaire (more details about the set of participants are given in next section). Our study involves decision-making and explanation about this process, and we chose hotels as the domain associated with the decision. The main reason for this design choice is that most of people are aware of the attributes that characterise hotels, and have preferences for individual attributes. Moreover, we chose to provide hotels in New York city for the reasons already presented. The applied questionnaire, which can be seen in Appendix C, consists of three parts, and each of which is explained next.

User Information Data. Our study does not assume that explanations depend on people characteristics, such as age or gender; however we collect some information about the participants to obtain demographic information (as we made the questionnaire available online, any individual can access it). The collected participant data is: (i) age; (ii) gender; (iii) location (city and country); and (iv) working/studying field.

Choosing Product. The study participant is then requested to imagine the scenario in which she is going to spend holidays in New York, and must choose a hotel for staying there from a set of options that we made available for her. As hotel rates for double rooms are very similar to those for single rooms, and people usually spend holidays with at least one friend, we include in this hypothetical scenario that the participant would travel with a friend and does not mind to share a bed with him or her. In order to make our scenario more realistic, we have selected real existing hotels to offer to participants. Hotels are described in terms of attributes associated with hotels and their rooms available at the `booking.com` website, presented in a table that allows a side-by-side comparison. We have selected *five* different hotels (*Hotel 91*, *Econo Lodge Times Square*, *The Hotel at Times Square*, *Comfort Inn Times Square*, *Renaissance New York Hotel 57*), viewing these options as forming three groups (not known to participants), as described below. Complete details about each hotel can be seen in Appendix C.

G-1 **Dominated option**. Although a dominated option (one that has no advantage and at least one disadvantage with respect to another) is generally not chosen, we add such an option (or at least something close to it) to capture arguments used to reject them. If we ignore small differences in room size, and discount parking price (which typically does not appear in catalogues of features), we can identify one hotel (Comfort Inn Times Square) dominated by another (The Hotel at Times Square) even though "Comfort Inn Times Square" actually has better parking price and a slightly better room size than "The Hotel at Times Square." The assumption (subsequently confirmed by our study) is that most participants focus on the main attributes and ignore small differences, so that "Comfort Inn Times Square" is dominated.

G-2 **Extreme options**. Extreme options compromise too much one attribute (e.g. quality) to improve another (e.g. price). People in general avoid such options, as stated by the extremeness aversion principle (Simonson and Tversky 1992), so we also selected extreme options to understand how participants explain their rejection or, if they choose

Subjective	Objective
• Justifications for acceptance • Justifications for rejection • Explanation types • Additional characteristics of justifications	• Chosen hotel • Chosen hotel vs. Explanation types for acceptance • Chosen hotel vs. Explanation types for rejection of other hotels • Rejected hotels vs. Explanation types for their rejection

Table 8.3: Data collected in our study.

them, why they do so. There are two extreme options: (i) much lower quality and much lower price (Hotel 91); and (ii) much higher quality and much higher price (Renaissance New York Hotel 57).

G-3 **Options that Require Trade-off Resolution**. Two options that have relative pros and cons require a trade-off to be made. As this may require a different form of explanation from either category above, we include options that clearly illustrate such a need for trade-off, "Econo Lodge Times Square" and "The Hotel at Times Square."

Reasons for Choice. The participant is asked to state why they choose a particular option, and why they reject the remaining options — we assume that if participants do not choose an option, they automatically reject it. In order to obtain useful responses, we highlight for the participant that *complete* answers should be provided and that arguments should be sufficiently strong to convince another person about the choice made.

In all this, the most important information collected is the provided justifications, expressed in natural language. The analysis of the study consists of carefully investigating these justifications to identify patterns and define explanation types from which, based on this initial analysis, we can extract quantitative data. Table 8.2(b) shows our approach to answering our research questions, which is mainly based on a classification of explanation types. In summary, the collected subjective and objective data are presented in Table 8.3.

8.1.3 Participants

The participants of our survey were selected using convenience sampling, which reached a total number of 100 participants. The sample was obtained based on the social network of the researchers involved in this study, by means of two forms of publishing the survey: (i) by e-mail, using the contact list of the researcher;

Gender	Male 58 (58%)		Female 42 (42%)	
Country	Brazil 78 (78%)	United Kingdom 8 (8%)	Canada 5 (5%)	Other 9 (9%)
Age	16-25 years 4 (4%)	26-35 years 61 (61%)	36-45 years 11 (11%)	>45 years 24 (24%)
Field of Work of Study	Informatics 54 (54%)	Education 11 (11%)	Management 7 (7%)	Other 28 (28%

Table 8.4: Demographic Characteristics of Participants.

and (ii) by Facebook,[1] which is a widely known social network. The distributed message consists of an invitation to participate of the survey and a request to forward the invitation for other people.

The survey was available for participation on October 12–24, 2011 and was initiated by 191 people, who answered at least one of the steps of the survey, from which 100 (52.36%) finished all the survey steps — the remaining surveys were discarded. The demographic characteristics of the participants that completed the survey are described in Table 8.4. Because we adopted the social network of this researcher to perform the study, most of the participants are aged between 26 and 35 years (61%) and are Brazilians (78%). Non-Brazilian participants are from 8 other countries: United Kingdom, Canada, Germany, United States of America, Switzerland, China, France and Netherlands, and the latter six were grouped into the "Other" category in Table 8.4, as only there are only a few participants from these countries.

8.2 Results and Analysis

Our collected data consists mainly of justifications expressed in natural language and, as these are qualitative data, we analysed them in a systematic way to extract quantitative information. In the section, we explain how we performed this analysis and show results obtained from our study, separating our findings according to the research questions we set out to answer. We focus on describing the obtained data and our qualitative analysis, and we leave for the next section further discussions and our interpretation. Note that, at various points, we label some findings with *"Evidence X,"* so that we can later refer to them to support our proposed guidelines.

Before proceeding to this detailing, we present the hotels chosen by our participants. This information is relevant for understanding the relationship between the chosen option and the corresponding justifications, as indicated by our research questions *RQ2* and *RQ4*. Figure 8.1 shows how many participants selected each

[1] `http://www.facebook.com`

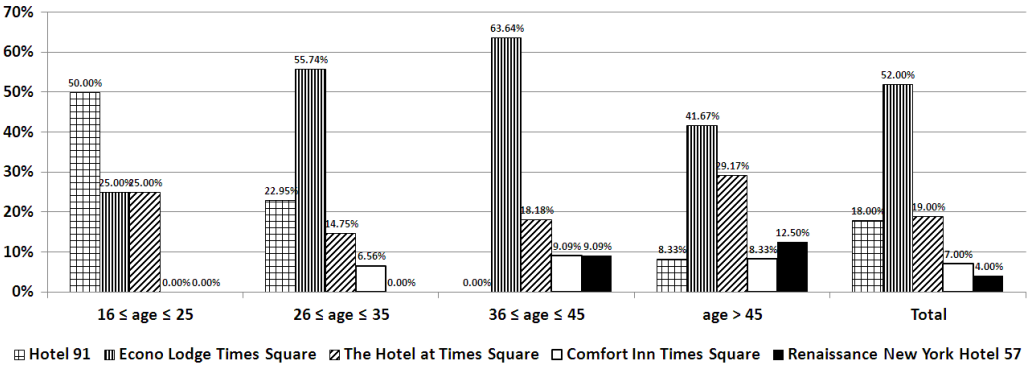


Figure 8.1: Hotel Choice.

hotel and, as expected, the majority of participants chose a hotel from group G-3. We also show the choice distribution according to age, and it can be observed that younger participants prefer cheaper options.

RQ1: Do users use a pattern to justify an option chosen from the set of those available? Each participant had to provide five justifications for their choice, being one of them a justification for why they chose a particular hotel. We analysed all provided justifications and derived from them a classification, which we refer to as *explanation types*, consisting of *six* different types that are described as follows. This classification emerged from the qualitative analysis of collected data, supported by the principles of content analysis from the social sciences (Braun and Clarke 2006). We exemplify each of these explanation types for the choice scenario in Table 8.5.

Critical attribute. For a group of participants, there is an attribute that plays a crucial role in the decision-making process, being in most of the cases the attribute *price*. In these situations, the justification focuses only on this crucial attribute, and the remaining ones are omitted. The same attribute is used to justify the chosen and all rejected options.

Dominance. The domination relationship can be used as an argument to justify a decision, but the acceptance of an option is justified using dominance only when it dominates all other options. This is an uncommon situation when choosing among products because, due to seller competition, there is typically a trade-off to be resolved, with options presenting both pros and cons. However, if domination *does* arise, the decision is extremely easy: one option may dominate another from a particular participant's perspective, as they might not care about a set of attributes, and the remaining ones create this ideal scenario to make the decision.

Explanation Type	Example of Justification for Acceptance
Critical attribute	*H_i is the cheapest option.*
Dominance	*H_i is better in all aspects.*
Main reason	*I chose H_i because it offers the benefit a_i.*
Minimum requirements	*From the hotels that satisfy my requirements, H_i is the cheapest.*
One-sided Reasons	*I chose H_i because it provides the benefits a_i and a_j.*
Pros and Cons	*Even though H_i is not the cheapest, it provides the benefits a_i and a_j.*

Table 8.5: Example of Justification for Acceptance.

Main reason. Some participants take into account many attributes to make a decision, but a particular option may be chosen (or rejected) when there is one attribute value that, together with its importance, is decisive for the choice. This most important attribute used in this kind of justification is specific to each option, differently from the critical attribute explanation type. We can observe that other attributes contribute for the decision by analysing the other justifications.

Minimum requirements. People usually have hard constraints, used to filter available options by discarding those that do not satisfy all of them — this can be seen as the establishment of cut-off values. If only one option satisfies all requirements, the decision becomes easy as the justification for option acceptance is that it satisfies all requirements. Furthermore, some participants provide a justification based on minimum requirements but, since more than one option satisfies these requirements, the participants also provide some criterion to distinguish between them, e.g. minimum price.

One-sided Reasons. Instead of only providing the main reason for acceptance, many participants focus on exposing only positive aspects (or negative, in case of rejection) of the option, even though the chosen option has disadvantages (or advantages) with respect to other options in relation to their preferences. This indicates the existence of a minimal set of attributes that caused the option to be chosen (or rejected).

Pros and Cons. The most complex type of explanation consists of making the option pros and cons explicit, and showing the reasoning process behind the choice. Based on an evaluation of these pros and cons, the participant states that the pros compensate for the cons (or do not, in case of rejection). In some cases, participants do not enumerate pros and cons, but only state "*this is (not) the best cost-benefit relationship*."

Figure 8.2: Explanation types used to justify each chosen hotel.

These explanation types indicate that justifications for choosing an option *do* follow patterns, and these can be used in systems for explanation generation. The right hand side of Figure 8.2 (which shows the explanation types used to justify each hotel) represents the total number of the different explanation types adopted by the participants, who mostly adopt *one-sided reasons* and *pros and cons* to explain their choices.

RQ2: Is there a relationship between the type of explanation given to support the decision and the chosen option? Given that we have identified patterns used to justify why a particular hotel is chosen, we now investigate if there is any relationship between the type of explanation given and the chosen option. Figure 8.2 shows how much each explanation type is adopted for each individual hotel.

The distribution of explanation types indicates three trends. First, it can be seen that most of the participants that chose "Hotel 91" (61.11%) justified their decision by giving information about a critical attribute. As price is an attribute extremely relevant for these participants, and what matters for them is basically that this hotel is the cheapest one. Some participants provided further positive information about the hotel (*one-sided reasons*, 22.22%), besides stating that it is the cheapest one — they provided other positive aspects that complement the fact that it is cheapest, i.e. they showed that even though the hotel is the cheapest, the quality that they require is not compromised.

The second observation is related to the hotels of G-3 group. As expected, the main adopted explanation types for choosing them are *one-sided reasons* and *pros and cons*, as it can be seen in Figure 8.2 and as it is shown in more detail in Table 8.6. The first explanation type is used to show that a whole set of hotel characteristics is responsible for the choice made. In general, participants that chose the "Econo Lodge Times Square" had excluded the cheapest hotel from the set of hotels being

Hotel	One-sided Reasons	Pros and Cons	Total
Econo Lodge Times Square	55.77%	25.00%	**80.77%**
The Hotel at Times Square	26.32%	63.16%	**89.47%**

Table 8.6: Main explanation types used for justifying hotels of the G-3 group.

considered in the decision, and they explained the benefits of this hotel to show that this hotel is suitable for them, i.e. there is no reason to pay more for another option if this hotel already provides what the participant wants. On the other hand, participants that chose "The Hotel at Times Square" made a detailed analysis of this hotel against the "Econo Lodge Times Square," i.e. they discussed their *pros and cons*, and showed that the higher price of the former justifies the benefits it provides, when compared against the latter. With respect to these two options, we point out one last comment: there are two participants (3.85%) that used dominance to justify why they chose "Econo Lodge Times Square." The participants ignored attributes that are not relevant for them, creating a scenario in which this hotel dominates all the others.

Finally, we discuss the results obtained for the dominated option and the most expensive option. It can be seen that there is no most adopted explanation type, and participants adopted different explanation types for justifying them. Only few participants chose these two options and, as it is not obvious why these options should be chosen, the participants gave their particular explanations to justify this decision. In the first case, "Comfort Inn Times Square," some participants were vague and said that they chose this hotel because it has the best cost-benefit relationship without giving further details. The remaining ones used as arguments the two attributes that this hotel is better than "The Hotel at Times Square," i.e. parking price and room size. The room size argument was also used with the expression of *intuition*: as the room is bigger, and the price is higher, the hotel "apparently" provides more comfort. For this same reason, some participants chose the 4-star "Renaissance New York Hotel 57," as comfort is the most important issue for them, and they are not concerned with price, and in their justification they explained this situation, i.e. for them the price of the hotel justifies the possible comfort it offers, and this is assumed because of the hotel stars. In one case, a participant said that she prefers the most expensive (critical attribute), as she wants to maximise comfort.

RQ3: Do users use a pattern to justify the rejected (not chosen) options?
Now, that we have already addressed the research questions related to choosing an option, we focus on the rejected options. By analysing justifications for rejecting options, we have observed the same explanation types used for justifying the chosen

Explanation Type	Example of Justification for Rejection
Critical attribute	*There are other options cheaper than H_i.*
Dominance	*There is no reason for choosing H_i, as it is worse in all aspects than H_j.*
Main reason	*I did not choose H_i because it does not offer the benefit a_i.*
Minimum requirements	*H_i is too expensive.*
One-sided Reasons	*I did not choose H_i because it has the disadvantages a_i and a_j.*
Pros and Cons	*Even though H_i provides the benefits a_i and a_j, its price does not compensate it.*

Table 8.7: Example of Justification for Rejection.

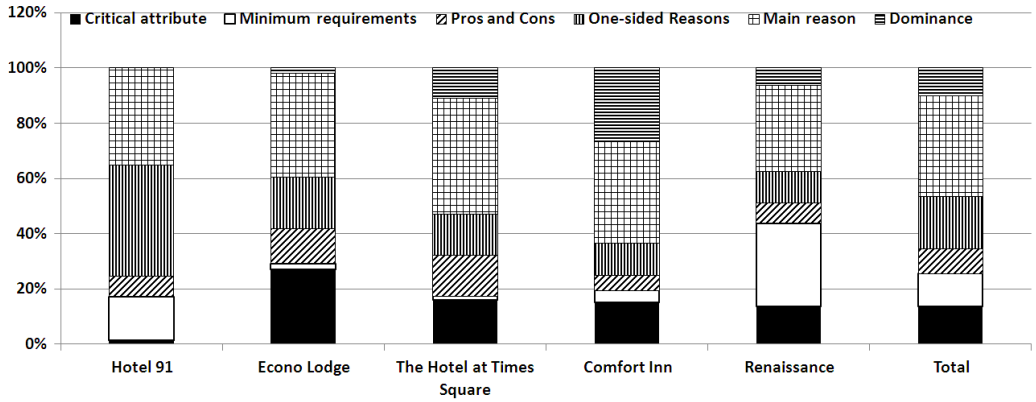
option. The description given for our set of explanation types show that they can also be applied for rejecting options, e.g. if an option does not satisfy the minimum requirements, than it is rejected due to this reason. In Table 8.7, we show examples of how each of these explanation types is used in the context of option rejection.

As it is the case with justifications for accepting an option, we also concluded that participants do use patterns for constructing arguments to reject options, and we next analyse the relationship between the adopted explanation types and the options involved in the decision-making process.

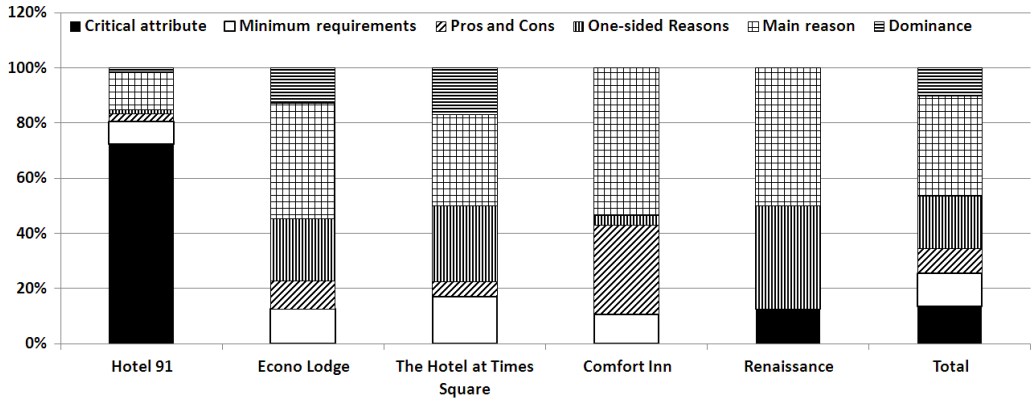
RQ4: Is there a relationship between the type of explanation given to reject options and the rejected or chosen option? In order to understand how participants choose a particular explanation type, we analyse the relationship between the types adopted to justify rejected options from two perspectives. The first consists of analysing justifications for rejection by relating them to the hotel that was rejected, i.e. we observe which explanation types were adopted to reject a particular hotel. The second perspective groups justifications according to the chosen hotel, i.e. we observe which explanation types were adopted to reject other options according to a particular chosen hotel. These two discussed views of justifications for rejection are presented in Figures 8.3(a) and 8.3(b).

There are many interesting aspects that can be observed in the collected data. *Critical attribute* is the type of explanation used when the decision is guided by it. For instance, if the participant wants to minimise price, the justification for the acceptance is that the chosen hotel is the cheapest, and the justification for the remaining rejected hotels is that they are more expensive (than the chosen hotel). Similarly, this situation happens with the more expensive hotel, in which the participant wanted to maximise the price (as a proxy to the comfort maximisation).

Dominance, on the other hand, is adopted when the chosen option dominates the rejected option, i.e. the comparison made in the explanation is always comparing



8.3(a): Explanation types used to justify the rejection of each hotel.



8.3(b): Explanation types used to justify the rejection of other hotels given a chosen hotel.

Figure 8.3: Rejection explanation types.

the chosen option with the others. In many situations, preferences (hidden in justifications) of participants, who chose "Econo Lodge Times Square," indicate that "The Hotel at Times Square" dominates "Comfort Inn Times Square;" however this is not given as an argument to discard the latter, but the participants seek for an explanation why "Econo Lodge Times Square" is better than "Comfort Inn Times Square" (*Evidence A*). Dominance was used as argument by participants that chose "Econo Lodge Times Square" when the set of attributes that matter for the participants indicated that this option dominates both "The Hotel at Times Square" and "Comfort Inn Times Square."

Some participants have hard constraints that they require to be satisfied by the chosen hotel, such as a maximum price that they are willing to pay, a maximum distance from the city centre or a minimum number of stars. In these situations, an option is rejected regardless the remaining options, and the justification given is that the option does not satisfy the participant *minimum requirements*.

Main reason and *one-sided reasons* indicate that there is an attribute (or a set thereof, in case of one-sided reasons) that is really important for the participant that, even though it is not part of a hard constraint, plays a *decisive role* in the

decision, i.e. because of this (these) attribute(s), the option is being rejected. This set of attributes is *kept as simple as possible* (*Evidence B*); e.g., some participants that chose "Econo Lodge Times Square" rejected "The Hotel at Times Square" and "Comfort Inn Times Square" because they do not have a refrigerator and are more expensive (than the chosen hotel). But, for justifying the "Renaissance New York Hotel 57" (which also does not have a refrigerator), they argued only that it is more expensive. It is important to note that the explanations given for *The Hotel at Times Square* and *Comfort Inn* are exactly the same, and there are many other cases in which the same explanation is given for different options rejected for the same reason (*Evidence C*). Finally, *pros and cons* are given as rejection arguments by participants when the decision between two (or three) options is difficult, so they expose these options' pros and cons to show that the chosen option has the best cost-benefit relationship. Thus, *pros and cons* are used only in the *absence of a decisive subset of attributes* (*Evidence D*).

In this way, the justification given for rejecting an option depends on both the chosen and rejected options, as the explanation given typically justifies why the rejected option is worse than the chosen one. Only in those cases in which the option is rejected due to a hard constraint (*minimum requirements*), the rejection explanation depends only on the option being rejected.

Further Observations. While analysing the collected data, we have also identified other relevant characteristics present in the provided justifications. We describe each of these characteristics below, and most of them can be used to suggest informal arguments to be used in systematic approaches for decision-making.

Explicit trade-off (TO). As already mentioned in the description of the *pros and cons* explanation type, some participants stated that the chosen hotel has the best cost-benefit relationship (or not the best, for rejecting a hotel), and sometimes just provided this argument without any details, e.g. "*For a trip like this, it seems the best cost-benefit among the 3-star hotels.*"

Preferences mentioned (PREF). Participants, when requested to justify their decision, provided arguments that are constructed based on their preferences (*Evidence E*); for example, a participant argue "*Absence of a fitness centre*" to justify a rejection, but this is due to the participant preference for a hotel *with* a fitness centre — and in some cases, participants made their preference explicit.

Insignificant difference (ID). The "Econo Lodge Times Square" has a US$5.00 difference from "The Hotel at Times Square." While some participants argue

that the benefits provided by the second does not compensate the price difference, others, who have chosen the second, stated that the price difference is insignificant, as it is very small, and both hotel prices can be considered the same. The same applies for room size or location, from the perspective of some participants.

Intuition (INT). One interesting characteristic of some provided justifications consists of inferring information of the hotel without any basis, i.e. some participants used their intuition to choose a hotel. For instance, one participant that chose "Econo Lodge Times Square" justified the rejection of "The Hotel at Times Square" by saying "*The name The Hotel **seems** to provide quality and, consequently, high price.*"

Price as a first class attribute (PRICE). The majority of participants (92%) mentioned the attribute "price" in their justifications, and evaluated options by comparing this attribute with all the other ones. This indicates that *cost* is not seen as any disadvantage that an option has when compared to another, but a fixed attribute that should be treated differently in the provided explanations (*Evidence F*).

Irrelevant attributes (IRR). When participants chose a hotel that does not offer as many benefits as the others, mainly when they chose the cheapest hotel, they used as an argument that those benefits are not important for them and, as they do not care about them, there is no reason for paying more for something that will not be used. Irrelevant attributes were mentioned in both acceptance and rejection justifications. For supporting a choice, participants state: "*Even though hotel H_i does not offer attribute a_i, this is not important to me*," and for rejecting an option, they say "*Even though hotel H_i offers attribute a_i, this is not useful to me*."

In Table 8.8, we show the percentage of participants whose justifications presented these identified characteristics. The table is split into each chosen hotel, and rows of each separate sub-table is related to the justification provided for each separate hotel. We highlight in gray the hotel that was chosen, therefore the row of a highlighted first cell is associated with justifications for acceptance.

As mentioned before, it can be seen that price should be treated as a first class attribute in explanations, as it is a crucial factor considered in the decision. In cases that a higher price is chosen, but this difference is very small, many participants acknowledge this fact. When the chosen option has a lower price, benefits provided by other options may be relevant to be mentioned, even though the decision maker does not care about it. In cases in which pros and cons of a set of options make the

Hotel 91						
Reason for	TO	PREF	ID	INT	PRICE	IRR
Hotel 91	0.00%	0.00%	0.00%	0.00%	100.00%	16.67%
Econo Lodge	5.56%	0.00%	0.00%	0.00%	77.78%	11.11%
The Hotel	0.00%	0.00%	0.00%	0.00%	83.33%	16.67%
Comfort Inn	0.00%	0.00%	0.00%	0.00%	83.33%	16.67%
Renaissance	0.00%	0.00%	0.00%	0.00%	94.44%	11.11%
Econo Lodge Times Square						
Reason for	TO	PREF	ID	INT	PRICE	IRR
Hotel 91	1.92%	3.85%	0.00%	3.85%	15.38%	0.00%
Econo Lodge	19.23%	1.92%	5.77%	1.92%	76.92%	1.92%
The Hotel	3.85%	3.85%	7.69%	1.92%	82.69%	13.46%
Comfort Inn	3.85%	1.92%	1.92%	0.00%	88.46%	7.69%
Renaissance	0.00%	0.00%	0.00%	0.00%	96.15%	13.46%
The Hotel at Times Square						
Reason for	TO	PREF	ID	INT	PRICE	IRR
Hotel 91	5.26%	0.00%	0.00%	0.00%	5.26%	0.00%
Econo Lodge	0.00%	5.26%	47.37%	5.26%	42.11%	0.00%
The Hotel	36.84%	10.53%	5.26%	5.26%	63.16%	0.00%
Comfort Inn	0.00%	5.26%	5.26%	0.00%	68.42%	5.26%
Renaissance	5.26%	0.00%	0.00%	0.00%	73.68%	0.00%
Comfort Inn Times Square						
Reason for	TO	PREF	ID	INT	PRICE	IRR
Hotel 91	28.57%	14.29%	0.00%	14.29%	28.57%	0.00%
Econo Lodge	28.57%	14.29%	0.00%	0.00%	14.29%	0.00%
The Hotel	28.57%	0.00%	0.00%	0.00%	28.57%	0.00%
Comfort Inn	28.57%	0.00%	0.00%	0.00%	42.86%	14.29%
Renaissance	14.29%	0.00%	0.00%	0.00%	85.71%	0.00%
Renaissance New York Hotel 57						
Reason for	TO	PREF	ID	INT	PRICE	IRR
Hotel 91	0.00%	25.00%	0.00%	25.00%	25.00%	0.00%
Econo Lodge	0.00%	25.00%	0.00%	0.00%	25.00%	0.00%
The Hotel	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Comfort Inn	0.00%	0.00%	0.00%	0.00%	25.00%	0.00%
Renaissance	0.00%	0.00%	0.00%	25.00%	0.00%	0.00%

Table 8.8: Results for additional characteristics observed in justifications.

decision hard, an explicit statement that a particular option has the best cost-benefit relationship might be helpful. Finally, participants typically to not support their arguments with their preferences.

8.3 Interpretation

Our study investigates explanations given by people to justify their choices, providing reasons to accept or reject options. In this section, we provide an interpretation for our analysis, which explains how participants, and more generally people, choose a particular type of explanation for their decisions.

8.3.1 Explanation for Choice

We provided five different options for participants, chosen in order to characterise options with certain particularities. Options that have quality lower than most of the other options available, but also lower price, are justified by the "*critical attribute*," which in this case is price. Therefore, other option details are not relevant, and do not need to be part of the explanations. In some cases, as the cheapest option being offered in our study already provides some comfort (as the hotel has 2 stars, shower, etc.), some participants mentioned that they chose the cheapest hotel, as it satisfies all their minimum requirements. Moreover, irrelevant attributes can be mentioned as part of explanations for this kind of option, in order to make the argumentation stronger. Explanations for the most expensive hotel (and with highest quality), although also characterised as an extreme option, does not follow this same reasoning. This kind of option is justified with all positive aspects it provides, or a main one — that is generally only offered by expensive options. In this scenario, price is typically not mentioned, as it is not a concern.

The majority of the participants have reduced their choice to options that require trade-off, by first discarding some options due to a minimum requirement, such as maximum price. There are mainly two kinds of justifications given for supporting a choice from this set of options. When the chosen option is the cheapest in this set, people use only the main reason or the benefits provided by the chosen option. On the other hand, when the option is not the cheapest, a more detailed explanation is needed, therefore the pros and cons related to the chosen option should be exposed, with the aim of showing that pros justify cons. For making this argument stronger, explicitly mentioning that the option provides the best cost-benefit relationship is helpful.

The last kind of option, namely the dominated option, is never chosen. In our study, the option that represents this group of options (G-1) has few advantages

with respect to the others, which are the room size and parking price. As these attributes are extremely relevant for some of the participants, they supported their choice evaluating pros and cons or by stating these essential aspects, in summary, this option was considered as an option of the G-3 group by those participants.

8.3.2 Explanation for Rejection

Having described how explanations are constructed to justify a chosen option, we now discuss how explanations for rejecting an option are built. A very important aspect of the explanation for rejecting an option, is that the *explanation is driven by the chosen option* in many cases.

As in explanations for acceptance, when an extreme option is chosen, explanations for rejecting options have a different behaviour for the two different extreme directions. When the chosen option is the cheapest and with lowest quality, the explanation given for all the remaining options are the same, which says that the other options are more expensive than the chosen one. So the typical explanations for this scenario are "*because of the price*" or "*because this option is more expensive [than the chosen option]*." When the chosen option is the most expensive option and with the highest quality, the arguments used to reject the remaining options are the most decisive aspects that are not offered by these options. For example, if one individual considers "fitness centre" and "bar" as important, but the former is more important than the latter, this individual would justify the rejection of "Econo Lodge Times Square" by stating that it does not have "fitness centre," even though it also does not have a "bar." Both aspects should be mentioned in the explanation if they, individually, would not change the choice made. For this kind of extreme option, dominance is never used because, as the chosen option is the most expensive, it does not dominate any other option.

Next, we discuss explanation for rejection when the chosen option is part of the G-3 group, which has a different explanation for each kind of rejected option. When there is a dominated option, this is the argument to be given, but *only if the chosen option dominates this option*. In addition, people that choose an option from the G-3 group, typically discard some options due to a cut-off value, i.e. options that do not satisfy minimum requirements, which are often part of the G-2 group. In this situation, the reason for cutting this option off should be given as the explanation, such as "*too expensive*" or "*too far away*." Finally, to reject other acceptable options, the reasoning is similar to that described for the most expensive option, i.e. the option decisive aspects must be exposed. If pros and cons of the rejected option have a similar balance to the chosen option, then this should be discussed in detail in the explanation, in order to show that pros do not compensate cons.

8.4 Guidelines and Patterns

This study provides us with a means of understanding how users construct arguments to justify a choice, by explaining why an option is chosen and why the remaining ones are rejected. Moreover, based on the results from this study, we are able to contribute to our ultimate goal of providing guidance that serves as a basis for the development of explanation approaches. To this end, we introduce guidelines and patterns derived from our study in this section. For each guideline, we indicate the evidence that supports it.

8.4.1 Guidelines

1. Provide chosen-option-oriented explanations. *(Evidence A)* The explanation generation process must be guided by a previously chosen option. The goal of the explanation is not to expose all the reasoning process used to make the decision, but to provide the main arguments that justify a chosen option and reject the remaining ones. After the choice is made, the explanations given should answer two main questions: (i) what makes the chosen option better than the others; and (ii) what makes the other options worse than the chosen option.

An example of the application of this guideline is the case of domination, which is used as a justification only if the chosen option dominates the rejected option.

2. Keep it simple. *(Evidence B)* The explanation given to a user should be as simple as possible, even justifying the decision with a single sentence; e.g. *A is the cheapest option*. Therefore, the less complex the explanation, the better. The next three guidelines are associated with this, and provide concrete ways of keeping the explanation simple.

3. Focus on the most relevant criteria. *(Evidence D)* In the given explanation, only the *decisive criteria* should be mentioned; i.e. the minimum set of attributes that causes an option to be selected or rejected. These decisive criteria should be derived from the comparison of the chosen option against the others.

For example, the case described above in which the rejection of the "Econo Lodge Times Square" is justified only by the fact that it does not have a fitness centre, although it also does not have a bar.

4. Group similar options. *(Evidence C)* An explanation to reject an option can also be given to reject other options. So, rejected options should be grouped when they are rejected for the same reason, and presented as a group and not individually.

5. Back up explanations with user preferences, but provide them only if asked. *(Evidence E)* Characteristics mentioned in explanations are relevant, because of the preferences being considered in the decision-making process; e.g., "*I chose this option as it is the cheapest*" (explanation), and "*I want to minimise costs*" (preference). People usually do not explicitly state their preferences to justify their decisions but, if a decision is made on someone's behalf, it is fundamental to back up an explanation with their preferences. As this information is not always needed, and as simpler explanations are better, preferences must be provided as part of explanations only upon request.

6. Use *cost* as a first class attribute. *(Evidence F)* An option is chosen by an individual when they believe that the cost being paid for that option compensates for the benefits it provides. Benefits is a subset of all possible positive characteristics that an option can have, for example, a hotel that provides breakfast, "big" room, "good" location, etc.; nevertheless benefits always come with a cost, which in the hotel case is its price, but in other scenarios it can be time or effort. The trade-off between benefits and costs is the key issue in the process of decision-making, so the option attributes that define the option costs should be made explicit and used as first class attributes in the explanation provided to justify a decision.

8.4.2 Patterns

Based on our study, we derived patterns of explanations, which can be used for supporting a decision made by a software system. Moreover, we identified the components these patterns must have, which comprise a template for an explanation pattern catalog. These components are: (i) a *classification*; (ii) a *context* in which the pattern should be applied; (iii) a *template* for the explanation; (iv) the pattern *description*; (v) an *example*; (vi) *preferences* that back up the explanation; and (vii) optionally, *extensions* to the pattern. Patterns are classified (item (i)) according to three attributes, explained below.

- *Explanation goal: accept/reject/both*. An explanation can justify a chosen option (*accept*), a rejected or not chosen option (*reject*), or both (*both*).
- *Target: decision/option*. An explanation pattern can provide guidance to generate an argument that justifies the decision as a whole (*decision*), or the

generation of an argument that supports the acceptance or rejection of a single option (*option*).

– *Position: absolute/relative*. When a pattern target is *option*, the explanation given can be based solely on the target option (*absolute*), or make a statement that explicitly compares the option to another (*relative*).

Next we present each of our proposed patterns. Patterns are presented ordered according to their complexity, i.e. the simpler the explanation associated with a pattern is, the earlier it is presented. According to our second guideline, the explanation should be as simple as possible so, if two patterns can be used in a particular situation, the simplest must be applied.

Pattern 1: Critical Attribute

Classification:

– *Explanation goal*: both
– *Target*: decision

Context: this pattern is applied in two situations: (i) there is an attribute that is extremely important for the user and this is the only one to be taken into consideration; and (ii) all available options satisfy all constraints and there is one criterion to choose the best.

Template:

(Option) chosen option was chosen because it has the best value for critical attribute .

Description: some users have a single criterion to choose an option, and may additionally have a set of constraints that is satisfied by all options. In these situations, the decision becomes trivial, as well as its associated explanation, which consists of stating that the chosen option was selected according to this single criterion.

Example: the user wants to buy the cheapest flight from Rio de Janeiro to London, regardless number of stops, flying time, airline company and so on.

Back up preference: preference that establishes criterion used to make the choice, and possibly other preferences satisfied by all options.

Extensions: rejected options might have characteristics not present in the chosen option, which are typically considered as benefits; however, for the decision maker, these characteristics are irrelevant. The explanation in this case can be extended by stating that the chosen option does not have such characteristics, but these are not relevant given the provided preferences.

Pattern 2: Cut-off

Classification:

- *Explanation goal*: reject
- *Target*: option
- *Position*: absolute

Context: an option does not satisfy a user requirement (hard constraint), or does not satisfy a constraint that is very relevant for the user, but there are other options that satisfy it.
Template:

(Option) rejected option was rejected because it does not satisfy constraints associated with attribute.

Description: in many situations, users have a set of requirements that **must** be satisfied (or hard constraints), and therefore an option that does not satisfy at least one of these requirements cannot be chosen. As not satisfying at least one of these constraints is enough for rejecting an option, only the constraint associated with the most important attribute for the user is part of the explanation. In addition, there are cases in which some requirements are not hard constraints, because the user might accept options that do not satisfy it when there is no other available options. When there are options that satisfy these "almost-hard" constraints, options that do not satisfy it can be rejected with an explanation of a not satisfied hard constraint.
Example: the maximum amount of money that a user will spend in a hotel room is US$300.00 for two nights. A hotel that costs US$350.00 for two nights is rejected because it is too expensive, i.e. due to a cut-off value.
Back up preference: not satisfied hard-constraints (or "almost-hard").

Pattern 3: Domination

Classification:

- *Explanation goal*: reject
- *Target*: option
- *Position*: relative

Context: the chosen option dominates a rejected option.
Template:

There is no reason to choose (option) rejected option, as (option) chosen option is better than it in all aspects, including cost.

Description: when an option dominates another, there is no argument that justifies considering the latter better than the former. Therefore, exposing this fact is enough to explain the rejection of the dominated option. However, the domination argument must be used only if the option that dominates the rejected option is the chosen one.
Example: there are two hotel rooms available for the user: standard queen room and superior queen room. The difference between them is that the superior queen room is bigger, it has a sitting area, it has a bath besides the shower, and, because of these extra features, it is more expensive. All user constraints are satisfied by both options, and she does not care about these three extra features, but she cares about the price. Therefore, according to the preferences of this user, the superior queen room is dominated by the standard queen room.
Back up preference: preferences that establish that individual attributes of the chosen option are considered better than the rejected one.
Extensions: in this pattern, attributes that users do not care about might be mentioned to support the domination relationship. See more details in the extensions of Pattern 1: Critical Attribute.

Pattern 4: Minimum Requirements^{-}

Classification:

- *Explanation goal*: reject
- *Target*: option
- *Position*: relative

Context: user established a set of minimum requirements for options, and a way of choosing from those that satisfy it. According to the requirements, some options were discarded. Other option attributes might have been used for making the decision, but only one of them makes the difference.
Template:

Even though (option) _rejected option_ *satisfies all your requirements, it has a worse value for* _attribute_ *than (option)* _chosen option_ .

Description: this pattern addresses justifying rejected options that differ only by a single attribute (that matters for the user) from the chosen option, and these rejected options, and also the chosen one, satisfy a set of user requirements. This scenario might happen when provided preferences consist only of these minimum requirements plus a preference to choose among them or when, even though when the decision involves a careful evaluation of pros and cons of each individual option, after choosing a particular option, one or more options are distinguished from it by a single attribute, relevant for the decision. Options that are in this category

can be rejected using a simple explanation — rather than a more complex one, possibly used in the decision-making process — consisting only of the requirements satisfaction and a single attribute that makes the difference.
Example: a user wants to stay in a 2-star hotel, whose price is up to US$150.00 per night, and is within the city centre, with breakfast. Given these requirements, the cheapest one. The chosen hotel costs US$120.00 per night, there is one that costs US$130.00; and all the remaining attributes have the same values for both options. The rejection of the second hotel is explained using this pattern, as there are other hotels that do not satisfy the price constraint.
Back up preference: user requirements and preference used to choose from the options.

Pattern 5: Minimum Requirements^{+}

Classification:

- *Explanation goal*: accept
- *Target*: option
- *Position*: absolute

Context: user established a set of minimum requirements for options, and a way of choosing from the ones that satisfy it. According to those requirements, some options were discarded. Other option attributes might have been used for making the decision, but only one of them made the difference.
Template:

Besides satisfying all your requirements, (option) chosen option has the best value for attribute .

Description: users, in certain decisions, establish a set of minimum requirements that reduces the set of available options to a subset, in which options differ only by a single criterion from the chosen option. This is a situation in which the decision becomes easy, and also the explanation, which consists of acknowledging users that, from those options that satisfy their requirements, the chosen option is the preferred one according to a particular criterion.
Example: a user wants to stay in a 2-star hotel, whose price is up to US$150.00 per night, and is within the city centre, with breakfast. Given these requirements, the cheapest one. The chosen hotel costs US$100.00 per night, and other available hotels provide the same features and are more expensive. Therefore, the explanation for the chosen hotel is that it is the cheapest from those that satisfy requirements.
Back up preference: user requirements and preference used to choose from the hotels.

Extensions: in this pattern, attributes that users do not care about might be mentioned to support the decision. See more details in the extensions of Pattern 1: Critical Attribute.

Pattern 6: Decisive Criteria

Classification:

- *Explanation goal*: both
- *Target*: option
- *Position*: absolute

Context: even though there are other attributes that contribute to the option acceptance (or rejection), there is a subset of them that would confirm this decision regardless the values of the other attributes.

Template:

(Option) option was [chosen | rejected] because of its set of decisive attributes .

Description: options, when compared, might have different pros and cons. However, there are attributes that are the most important ones in the decision, and other attributes — which can make a difference in particular cases — do not impact on the decision between two options. Therefore, the only attributes that must be part of the explanation are those that impact on the decision, leaving the remaining attributes out of it.

Example: three hotel options were given to a user. One is a 3-star hotel, cheaper than the other two options and has a refrigerator in the room. The second is also a 3-star hotel, more expensive than the former, with a better location. The last is a luxury 4-star hotel, much more expensive than the others and, as the second option, also does not have a refrigerator in the room. While the rejection of the second is justified by the absence of the refrigerator *and* its price; the rejection of the third is justified only because of its price, because this was the decisive criteria for not choosing it.

Back up preference: preferences over the set of decisive attributes.

Pattern 7: Trade-off Resolution

Classification:

- *Explanation goal*: both
- *Target*: option
- *Position*: absolute

Context: there is no set of attributes that are decisive.

Template:

Template for rejected options:

Even though (option) _rejected option_ *provides better* _pros_ *than (option)* _chosen option_ *, it has worse* _cons_ *.*

Template for the chosen option:

Even though (option) _chosen option_ *does not have the best value for* _cons_ *, its values for* _pros_ *compensate its cons.*

Description: a set of decisive attributes does not exist in all situations. Options might provide different pros and cons in a way that all attributes are important for making the decision, therefore, all option attributes that differ for these options have to be evaluated, and their evaluation has to be informed to the user.

Example: a user is provided with two hotel options. Both of them are 3-star hotels, the first is cheaper (US$115.00 per night) and its room has a refrigerator, and the second is more expensive (US$130.00 per night), better located (two blocks closer to the city centre), and has breakfast included. As, according to the user preference in our example, location has a higher priority than price, and their difference is strong enough when considering the included breakfast and the refrigerator, the second hotel is chosen. The explanation thus states that even though the second hotel has a lower price and a refrigerator, it has a worse location and does not include breakfast.

Back up preference: all user preferences used to evaluate pros and cons of options that require trade-off analysis.

Extensions: in situations in which pros and cons of the chosen option create a balance that is very similar to the one of another option, it might be not obvious for the user why pros compensate cons. So, additionally, it can be explicitly told to the user, i.e. inform the user that the chosen option provides the best cost-benefit relationship.

8.5 Final Considerations

In this chapter, we presented a study performed to understand how people justify their decisions, by giving explanations why they chose a particular option from the set of those available, and why remaining options are rejected. The study consisted of providing participants (a hundred people) with a set of carefully chosen hotel options, and requesting them to give reasons for the choice. Based on collected data, we identified explanation types that are patterns of justifications given by people, and how they are selected to be given as explanation — for both chosen and rejected options. Assuming that explanations given by people are the explanations that users expect to receive as reasons for a choice, our study allowed us to propose a set of guidelines and patterns for the development of explanation approaches. We now will show how we produce explanations for choices made by our decision-making technique, which take into consideration this guidance that we derived from our study.

9
Generating Explanations to Justify Choice

In Chapter 6, we proposed a decision making technique that incorporates principles adopted by humans to make decisions, in order to facilitate extracting a rationale behind decisions and produce explanations for choices made. We have also identified in our study of how explanations can justify choices the kinds of explanation users expect to receive in order to understand and accept a decision made by a software system on their behalf. We now propose in this chapter an explanation technique that bridges the gap between these two approaches — we generate explanations that meet identified requirements based on the reasoning process of our decision making technique.

Before detailing our explanation generation approach, we introduce in Section 9.1 the notation adopted in this chapter, and remind the reader of structures of our reasoning technique, used in our explanation approach. Then, we show in Section 9.2 how we select attributes to be part of explanations, which are parameters of the explanation templates presented in previous chapter. As different explanations can be given to justify a choice, we describe in Section 9.3 how to select the type of explanation to be given and also how to put the selected parameters together with the explanations, showing how to generate explanations. Section 9.4 shows an example of generated explanations for our apartment example. Finally, we compare our approach with existing work and present a performance evaluation in Section 9.5.

9.1
Notation

Our explanation generation approach follows the same notations adopted in Chapter 6, and is based on many of the structures produced in our decision making technique. o_c is an option chosen from the set of those available, Opt, and the remaining ones, i.e. $Opt_r = Opt - \{o_c\}$, are rejected options o_r. Options are characterised by a set of attributes, Att. Variables o and a, possibly with an index, represent an option and an attribute, respectively. We also have a set of modifiers M, which consists of expressive speech acts (e.g. *like*, *love*, *hate*, *don't accept*) and rates (e.g. *good* and *bad*), and these are used to express monadic preferences. Some of these modifiers indicate hard-constraints to be considered in the decision making

process. Finally, we recall structures built during the execution of our decision making technique, which are used to generate explanations.

- $PSM[o, a]$: Preference Satisfaction Model (PSM) is a partial mapping from an option and an attribute to a modifier (or its negation). It represents how the value of an option attribute is evaluated in terms of a modifier.
- $OAPM[o_i, o_j, a]$: Options-Attribute Preference Model (OAPM) is a mapping from a pair of options and an attribute to {+, −, ∼, ?}. It shows if the attribute value of an option o_i is better (+), worse (−), as good as (∼) the value of the same attribute of an option o_j. If there is no information available, the OAPM value is "?".
- $d(o_i, o_j) = (1 - w_{to} - w_{ea}) \times Cost(o_i, o_j) + w_{to} \times ToContrast(o_i, o_j) + w_{ea} \times ExtAversion(o_i, oj_2)$: the decision function represents how much o_i is negatively evaluated with respect to o_j. It can be seen as the cons of o_i (w.r.t. o_j), as opposed to $d(o_j, o_i)$, which are its pros. $d(o_i, o_j)$ is the weighted sum of three factors: the costs provided by attribute values ($Cost(o_i, o_j)$); and the trade-off contrast ($ToContrast(o_i, o_j)$) and extremeness aversion ($ExtAversion(o_i, oj_2)$), which are two principles that humans adopt when making decisions (Simonson and Tversky 1992).
- $Cost(o_i, o_j) = \sum_{a \in Att} w_a(o_i) \times AttCost(o_i, o_j, a)$: the costs of o_i with respect to o_j are captured by a real value [0, 1], which is calculated as the weighted sum of the cost provided by each individual attribute. Attribute weights may vary for each option, as there may be priorities expressing conditional importance of attributes.

9.2 Explanation Parameters: Selecting Relevant Attributes

Explanation patterns presented in Chapter 8 give templates for explanations, which are parameterised by a single attribute, or sets of attributes. In this section, we show how these attributes (or this attribute) are selected to be part of the explanations. We will follow the same order adopted in the previous chapter to present patterns.

9.2.1 Single-attribute Selection

The first explanation pattern consists of identifying an attribute that is critical for making the decision. An attribute is critical when it is the reason for choosing a particular option, and the values of other attributes are not relevant. In addition, as described in the Critical Attribute pattern, there may be constraints — used

to express extreme cases that users do not accept — but they are satisfied by all options. A critical attribute is identified based on the OAPM, as its values associated with the chosen option are +. For the remaining attributes, there are two possibilities for the OAPM value: (i) ∼, which indicates that constraints were given, but the other options also satisfy them; and (ii) ?, meaning that no preference was given with respect to a particular attribute. The definition of *critical attribute* is presented below, which indicates that if there is an attribute that is critical, it is unique.

Definition 9.1 *Let o_c be the option chosen from the set Opt by a decision making technique. An attribute $a_{crit} \in Att$ is said the* ***critical attribute*** *of the decision, or $CriticalAttribute(o_c)$, if for all options $o_r \in Opt_r$, we have $OAPM[o_c, o_r, a_{crit}] = +$, and for all the other attributes $a \in Att$ and $a_{crit} \neq a$, o_c is considered as good as o_r ($OAPM[o_c, o_r, a] \neq \sim$) or there is no preference given over this attribute, i.e. $OAPM[o_c, o_r, a] \neq$ "?".*

In many situations, people have hard-constraints, which eliminate options, whose at least one attribute value is non-compensatory, that is, it is not possible to compensate this value, regardless the values of other attributes. Our decision making technique considers four modifiers as hard-constraints, which are *require*, *need*, *hate* and *don't accept*. Therefore, one might expect that options that do not satisfy preferences associated with *require* or *need*, or satisfy preferences associated with *hate* or *don't accept* have their rejection explained due to a cut-off value. Indeed, this intuition is the case, but we include other attributes in the cut-off group.

When users provide monadic preferences associated with a negative modifier, they indicate which values are not desired for certain attributes. Therefore, a rejected option has a cut-off attribute when its value satisfies a preference associated with a negative modifier, even if it is not considered a hard-constraint, that is, an undesirable value of an attribute option can be used as a reason to reject the option even though this preference is not a hard-constraint. Nevertheless, in some situations, options provide combinations of attribute values that make users, or an automated technique, to choose a certain option even though one or more of its attributes have an undesired value. Therefore, an option cannot have its rejection justified by a cut-off attribute if the chosen option also has the same undesired attribute value, or even worse. So, we first analyse the chosen option to verify the strongest negative modifier associated with its values, and then we explain rejected options by cut-off attributes when they either do not satisfy a hard-constraint or are associated with a negative modifier that is stronger than the modifiers associated with the chosen option.

To formally define this idea, we first define, in Equation 9-1, $min_{mod}(M_S)$, where $M_S \subseteq M$. This function returns the minimum modifier — i.e. the strongest negative, of a set of modifiers — and is based on the function f_m, which associates

modifiers with a real value, respecting the order established by the modifier scale (and in our decision technique we are currently using a logarithmic function).

$$min_{mod}(M_S) := m \mid m \in M_S \wedge \forall\, m'.(m' \in M_S \wedge m \neq m' \wedge f_m(m) \leq f_m(m')) \quad (9\text{-}1)$$

$min_{mod}(M_S)$ is used to capture the most negative modifier associated with an option, with a given lower bound (*don't need*), that is, if no negative modifier is associated with this option, the selected modifier is *accept*. Equation 9-2 defines the function $mostNegative(o)$, which makes this modifier selection.

$$\begin{aligned} mostNegative(o) := min_{mod}(\{\text{“}don't need''\} \cup \{mod \mid mod \in M \\ \wedge\, IsNegative(mod) \wedge \exists\, a.(PSM[o, a] = \langle empty, mod\rangle)\}) \end{aligned} \quad (9\text{-}2)$$

Finally, we define below when an attribute is considered a cut-off, following the informal description discussed previously.

Definition 9.2 *Let mod be a modifier from the set M, and $o_c \in Opt$ and $o_r \in Opt_r$. An attribute $a_{cutOff} \in Att$ said a **cut-off**, or $a_{cutOff} = CutOff(o_r, o_c)$, if we have:*

$$\begin{aligned} PSM[o_r, a_{cutOff}] = \langle \neg, \text{“}require''\rangle \vee PSM[o_r, a_{cutOff}] = \langle \neg, \text{“}need''\rangle \\ \vee\, (PSM[o_r, a_{cutOff}] = \langle empty, mod\rangle \wedge f_m(mod) < f_m(mostNegative(o_c))) \end{aligned}$$

If more than one attribute satisfy this property, we select most important one, i.e. if $w_{a_i}(o_r) < w_{a_j}(o_r)$, then the selected attribute is a_i.

The third pattern, namely Dominated Option, relies on the notion of domination, which is characterised by an option (dominant) that is better than another (dominated) with respect to one attribute, and at least as good with respect to the others. As this explanation does not involve any parameters, we do not discuss it in this section. So, next we address two patterns together: Minimum Requirements$^-$ and Minimum Requirements$^+$. In the scenario of these patterns, users have provided a set of constraints that lead to the elimination of options due to cut-off attributes, allowing the identification of a *consideration set*. In addition, the chosen option has no reason to be rejected, i.e. it satisfies all positive constraints and do not satisfy the negative ones, that is, for all attributes att we have $PSM[o_c, a] \neq \langle empty, IsNegative(modifier)\rangle$ and $PSM[o_c, a] \neq \langle \neg, IsPositive(modifier)\rangle$. If we have this scenario in the decision making process, and also there is one attribute that is decisive to choose one option from consideration set, which we refer to as *tie-breaker attribute*, we adopt these patterns to explain chosen and rejected options — excluding those rejected due to domination or cut-off attributes. The definition of the tie-breaker attribute is as follows.

Definition 9.3 *Let $a_{tieBreaker}$ and a be attributes from Att, and $o_c \in Opt$. $a_{tieBreaker}$ is said a **tie-breaker attribute**, or $TieBreaker(o_c)$, if there exists an option $o'_r \in Opt_r$*

rejected due to a cut-off value, i.e. $\exists\, a.(CutOff(o'_r, o_c) = a)$*, and for all the remaining rejected options* $o_r \in Opt_r$ *that* $\nexists a.(CutOff(o_r, o_c) = a)$*, we have* $OAPM[o_c, o_r, a_{tieBreaker}] = +$*. In addition, there is no* a' *that* $a' \neq a_{tieBreaker}$ *and* $OAPM[o_c, o_r, a_{tieBreaker}] = +$*, i.e.* $a_{tieBreaker}$ *is unique.*

We now proceed to the last two patterns, involving multiple attributes.

9.2.2 Multi-attribute Selection

In order to identify the *decisive criteria* to justify a decision made, required by the Decisive Criteria pattern, we first make definitions of concepts needed for identifying them. When two options are compared in our decision making technique, we identify pros and cons of these options with respect to each other. These are captured by the sets $Att^+(o_i, o_j)$ and $Att^-(o_i, o_j)$, which are sets of attributes that are pros and cons of o_i, respectively, and are defined as follows.

Definition 9.4 *Let* $o_i \in Opt$ *and* $o_j \in Opt$*. Then we define.*

$$Att^+(o_i, o_j) = \{a \mid a \in Att \wedge w_a(o_j) \times AttCost(o_j, o_i, a) > 0\}$$
$$Att^-(o_i, o_j) = \{a \mid a \in Att \wedge w_a(o_i) \times AttCost(o_i, o_j, a) > 0\}$$

As our technique identifies *how much* an option is preferred to another with respect to each attribute, we calculate the total pros and total cons of an option as shown below.

Definition 9.5 *Let* $o_i \in Opt$ *and* $o_j \in Opt$*. Then we define*

$$Pros(o_i, o_j) = \sum_{a^+ \in Att^+(o_i, o_j)} w_{a^+}(o_j) \times AttCost(o_j, o_i, a^+)$$
$$Cons(o_i, o_j) = \sum_{a^- \in Att^-(o_i, o_j)} w_{a^-}(o_i) \times AttCost(o_i, o_j, a^-)$$

The definition of decisive criteria is different for rejected and chosen options. The decisive criteria for rejecting an option consist of the subset of attributes whose values are enough for rejecting this option. For example, assume than an option X is chosen, and it has better values for the attributes a and b than an option Y has. In addition, $Cons(Y, X) > Pros(Y, X)$ — as we take into account trade-off contrast and extremeness aversion for making a choice, this may not hold. If we do not consider the benefit of X (cost of Y) with respect to b, and cons are still higher than pros, we can say that what matters is only the value of a of this option for

making the decision. This intuition is formalised below and, as more than one set of attributes may have this described characteristic, we also define a precedence for choosing one of these sets — we choose the simplest sets (in terms of the number of attributes), and among these, the strongest one (in terms of total pros).

Definition 9.6 *Let o_c be the option chosen from the set Opt by a decision making technique, and o_r a rejected option. The* ***decisive criteria*** *$D \subset Att^{-}(o_r, o_c)$ is the set of attributes such that $\sum_{a \in D} w_a(o_r) \times AttCost(o_r, o_c, a) < Pros(o_r, o_c)$. If there is an $S \subset Att^{-}(o_r, o_c)$ that also satisfies this property, and $S \neq D$, D is the decisive criteria if and only if*

$$(|D| < |S|) \vee (|D| = |S| \wedge \sum_{a \in D} w_a(o_r) \times AttCost(o_r, o_c, a) > \sum_{a \in S} w_a(o_r) \times AttCost(o_r, o_c, a))$$

Besides defining the decisive criteria for rejecting an option, it is essential to provide efficient means of identifying it, and this can be done by the execution of Algorithm 10. This algorithm includes attributes to the decisive criteria in a stepwise fashion, always including the attribute a_i whose value $w_{a_i}(o_r) \times AttCost(o_r, o_c, a_i)$ is the highest. At the moment that cons of the included attributes are higher than the pros, the algorithm stops, and returns the decisive criteria. If all attributes should be considered to make cons higher than pros, or if cons lower than pros (i.e. the rejection of o_r depends on the user-centric principles), there is no decisive criteria. In order to show that this proposed algorithm satisfies the conditions of Definition 9.6, we present the theorem below.

Theorem 9.7 *$DecisiveCriteria^{-}(o_r, o_c)$ returns the decisive criteria for rejecting o_r, or an empty set if this minimal set does not exist.*

Proof. We prove this theorem by contradiction. Assume that $D = DecisiveCriteria^{-}(o_r, o_c)$ and there is a subset $Out \subseteq D$, which is not part of the decisive criteria, and a subset $In \subseteq Att^{-}(o_r, o_c) \backslash D$, which is part of the decisive criteria. If $|In| > |Out|$, then $|In \cup (D \backslash Out)| > |D|$, and according to the Definition 9.6, $In \cup (D \backslash Out)$ is not the decisive criteria. If $|In| = |Out|$, as for all $attI \in In$ and $attO \in Out$, $w_{attI_i}(o_r) \times AttCost(o_r, o_c, attI_i) \leq w_{attO_i}(o_r) \times AttCost(o_r, o_c, attO_i)$ — as we sort $Att^{-}(o_r, o_c)$ — $\sum_{a \in In \cup (D \backslash Out)} w_a(o_r) \times AttCost(o_r, o_c, a) < \sum_{a \in D} w_a(o_c) \times AttCost(o_r, o_c, a)$, therefore contradicting the definition of decisive criteria. Finally, we analyse the case when $|In| < |Out|$. As the while loop ends at the first time that accumulated cons becomes higher than $Pros(o_r, o_c)$, and the last attribute added to D has the lower value for $w_a(o_r) \times AttCost(o_r, o_c, a)$, if we

remove any of the attributes of D, accumulated cons will become lower or equal to $Pros(o_r, o_c)$. So, $\sum_{a \in In \cup (D \setminus Out)} w_a(o_r) \times AttCost(o_r, o_c, a)$ has to be higher than $\sum_{a \in D \setminus \{x\}} w_a(o_r) \times AttCost(o_r, o_c, a)$, for all $x \in Out$; however, this is not possible, because we sort $Att^{-}(o_r, o_c)$, as shown in the previous case. And this completes the proof. ■

Algorithm 10: $DecisiveCriteria^{-}(o_r, o_c)$

Input: o_r: a rejected option; o_c: chosen option
Output: D: subset of Att containing the decisive criteria

```
1  SortedAtt⁻ ← Sort(Att⁻(o_r, o_c),
   a_i ≻ a_j ↔ w_{a_i}(o_r) × AttCost(o_r, o_c, a_i) > w_{a_j}(o_r) × AttCost(o_r, o_c, a_j));
2  AccumulatedCons ← 0;
3  D ← ∅;
4  while AccumulatedCons ≤ Pros(o_r, o_c) ∧ SortedAtt⁻ ≠ ∅ do
5      a ← Last(SortedAtt⁻);
6      SortedAtt⁻ ← SortedAtt⁻ − {a};
7      AccumulatedCons = AccumulatedCons + w_a(o_r) × AttCost(o_r, o_c, a);
8      D ← D ∪ {a};
9  if |D| < |Att⁻(o_r, o_c)| then
10     return D;
11 else
12     return ∅;
```

After showing how to identify the decisive criteria of rejected options, we analyse the case of the chosen option. The decisive criteria for justifying a chosen option can be either the set of attributes that the chosen option has the values better than at least half of the other options have, and no worse for the others; or (if this set does not exist), the decisive criteria for rejecting the option that has the lower pros and cons balance, when compared to the chosen option, which is seen as the "second best option." In both cases, we do not consider options rejected due to domination ($Expl(o, o_c) = \Psi_{dom}$) or cut-off attributes ($Expl(o, o_c) = \Psi_{cutOff}$). In order to identify the set of attributes of the first case, we define the concept of *best attributes* below.

Definition 9.8 *Let o_c be the option chosen from the set Opt by a decision making technique. The best attributes $B \subset Att$ is the set of attributes such that for all $a \in B$ and for all rejected options $o_r \in Opt_r*$ and $Opt_r* = Opt_r - \{o \mid Expl(o, o_c) = \Psi_{cutOff} \vee Expl(o, o_c) = \Psi_{dom}\}$, we have $OAPM[o_c, o_r, a] = +$, for at least $\frac{|Opt_r*|}{2}$ options, and $OAPM[o_c, o_r, a] = \sim$ for the remaining ones. Moreover, B is maximal in the sense of $\subset$.*

And now, we define the decisive criteria for the chosen option, which describes the two cases introduced above. Note that the decisive criteria for rejecting the option that has the lower pros and cons balance may not exist. This happens

because the lower pros and cons balance can be lower than 0, as we take into account trade-off contrast and extremeness aversion for making a choice.

Definition 9.9 *Let o_c be the option chosen from the set Opt by a decision making technique. The decisive criteria $D \subset Att$ is the best attributes B of o_c. If $B = \emptyset$, then D is the decisive criteria of an o_r, i.e. $DecisiveCriteria^{-}(o_r, o_c)$, such that $Pros(o_c, o_r) - Cons(o_c, o_r)$ is minimal, for all $o_r \in Opt_r$. Moreover, D exists if and only if $|D| \neq \emptyset$.*

The decisive criteria for a chosen option can be obtained by running Algorithm 11. The first part of the algorithm (lines 3–12) tries to identify the best attributes, and if they do not exist, then the second part (lines 14–15) tries to find the decisive criteria compared to the second best option.

Algorithm 11: $DecisiveCriteria^{+}(o_c)$

Input: o_c: chosen option
Output: D: subset of Att containing the decisive criteria

```
1  Opt_r* ← Opt − {o | o = o_c ∨ Expl(o, o_c) = Ψ_cutOff ∨ Expl(o, o_c) = Ψ_dom};
2  D ← ∅;
3  foreach a ∈ Att do
4      in ← true;
5      counter ← 0;
6      foreach o_r ∈ Opt_r* do
7          if OAPM[o_c, o_r, a] =~ then
8              counter ← counter + 1;
9          else if OAPM[o_c, o_r, a] ≠ + then
10             in ← false;
11     if in ∧ counter < |Opt_r*|/2 then
12         D ← D ∪ {a};
13 if D = ∅ then
14     o_r ← o | o ∈ Opt ∧ min(Pros(o_c, o) − Cons(o_c, o));
15     D ← DecisiveCriteria^−(o_r, o_c);
16 return D;
```

A chosen or rejected option may not be associated with a set of attributes, which are the decisive criteria for making the decision in different cases, and therefore the **Decisive Criteria** pattern cannot be applied, so the last explanation pattern — **Trade-off Resolution**— has to be adopted to justify the choice for the user. We next describe these cases for the chosen option, and then later rejected options.

For explaining a chosen option, which does not have a set of attributes that are the decisive criteria of the decision, we have three cases to analyse, which consist of the reasons why there is no decisive criteria. First, a chosen option o_c may not have one or more attributes that are better than the attributes of all other options, and also the pros and cons balance of second best option may

be negative, that is, $Pros(o_c, o_r) < Cons(o_c, o_r)$ — meaning that the trade-off contrast and/or extremeness aversion are responsible for choosing o_c instead of o_r. For explaining this scenario, we have two alternatives, which depend on the existence of a set $D \subset Att$, which $D = DecisiveCriteria^{-}(o_c, o_r)$. When D exists, the provided explanation highlights that o_r has D pros (i.e. "*even though* o_r *is better considering* a_x, a_y, *etc.*"), and states that o_c has a better cost-benefit relationship, as $ToContrast(o_r, o_c) > 0 \vee ExtAversion(o_r, o_c) > 0$. When these decisive criteria do not exist, we have a procedure to select both decisive pros and decisive cons, shown in Algorithm 12, which identifies the maximal set of pros that should be considered for enabling the existence of a decisive criteria for rejecting o_c. Therefore, $DecisiveProsCons(o_c, o_r)$, for an o_r whose pros are higher than cons when compared to the chosen option, identifies the cons that should be shown in the "*even though*" part of the explanation, and also the pros that should be mentioned, which compensate cons. Moreover, the cost-benefit relationship is also highlighted as the trade-off contrast and extremeness aversion play an important role in the decision.

Algorithm 12: $DecisiveProsCons(o_i, o_j)$

Input: $o_i, o_j \in Opt$

Output: $\langle P, C \rangle$: subsets of Att, which represents pros and cons of o_i

1 $SortedAtt^{+} \leftarrow$ `Sort(`$Att^{+}(o_i, o_j)$, $a_i \succ a_j \leftrightarrow w_{a_i}(o_j) \times AttCost(o_j, o_i, a_i) > w_{a_j}(o_j) \times AttCost(o_j, o_i, a_j)$`)`;

2 $RemainingPros \leftarrow Pros(o_i, o_j)$;

3 $P \leftarrow \emptyset$;

4 $C \leftarrow \emptyset$;

5 **while** $C = \emptyset \wedge SortedAtt^{+} \neq \emptyset$ **do**

6 $a \leftarrow$ `Last(`$SortedAtt^{+}$`)`;

7 $SortedAtt^{+} \leftarrow SortedAtt^{+} - \{a\}$;

8 $RemainingPros = RemainingPros - w_a(o_j) \times AttCost(o_j, o_i, a)$;

9 $P \leftarrow P \cup \{a\}$;

10 $C \leftarrow DecisiveCriteria^{-}(o_i, o_j, RemainingPros)$;

// $DecisiveCriteria^{-}(o_i, o_j, RemainingPros)$ `above is` $DecisiveCriteria^{-}(o_i, o_j)$ `but considering only the remaining pros`

11 **if** $C = \emptyset$ **then**

12 $C \leftarrow Att^{-}(o_i, o_j)$;

13 **return** $\langle P, C \rangle$;

In case o_c has the best pros and cons balance, but none of the attributes have the best values in comparison with other acceptable options (i.e. the ones not excluded due to a cut-off value or domination), we use the second best option — the option o_r that has the minimum pros and cons balance ($Pros(o_c, o_r) - Cons(o_r, o_c)$) — to explain the decision. This scenario is explained by finding the decisive criteria for rejecting the second best option, but this case was already covered in previous section. Therefore, there is only one case left, that is, o_c has the best pros and cons

balance, but there is no decisive criteria to choose it over the second best option. The explanation given in this case is based on the same algorithm adopted before, but used in the opposite direction — $DecisiveProsCons(o_r, o_c)$ — we identify key attributes of the second best option, which are removed so that we can identify decisive criteria, and the explanation states that even though o_r (the second best option) has better values associated with the key attributes (o_c's disadvantages), the values of the attributes that are the decisive criteria compensate these disadvantages.

Rejected options may also not have associated decisive criteria, since all attributes that characterise the cons of a rejected option may play a role in the decision between this option and the chosen option, or the trade-off contrast and extremeness aversion may have played a crucial role in the decision. So, to justify rejected options in these situations, the reasoning to build an explanation is similar to that made for explaining the chosen option. First, we analyse if the rejected option o_r has a better pros and cons balance than the chosen option $Pros(o_r, o_c) > Cons(o_r, o_c)$. If so, we adopt the same approach used previously. (a) If there is a set of attributes that characterises the decisive criteria for choosing o_r instead of o_c, i.e. $DecisiveCriteria^{-}(o_c, o_r)$, we highlight these positive aspects of o_r, and state that, nevertheless, o_r has a worse cost-benefit relationship when compared to o_c. (b) If there is no decisive criteria, we select the decisive pros and cons $\langle P, C\rangle = DecisiveProsCons(o_c, o_r)$, and besides mentioning only the cost-benefit relationship of o_c, we also highlight its decisive pros. This procedure is also the one applied when $Pros(o_r, o_c) \leq Cons(o_c, o_r)$, but there is no decisive criteria to justify the decision.

We have shown different ways of explaining the trade-off resolution in order to justify a chosen or a rejected option. In Table 9.1 we summarise these different scenarios and, as explanations that follow the Trade-off Resolution Pattern receive as parameters pros and cons to be made explicit, we show how they are obtained. In some scenarios, a constant argument — better or worse cost-benefit relationship — may be part of the explanation.

9.3 Choosing and Generating an Explanation

After showing how parameters are selected to be part of explanations, we now present how we choose an explanation to be given. First, we introduce the representation of each explanation type in Table 9.2. This representation is the information that we need for generating an explanation according to the templates proposed in our explanation patterns. We extend these patterns by including Domination as an explanation of a chosen option, which is applied when the chosen option dominates all the other ones. This is not reported as a pattern, as this situation

Chosen Option			
$\exists\, o_r.(Pros(o_c, o_r) < Cons(o_c, o_r))$?	**Decisive Criteria**	**Pros**	**Cons**
Yes	$DecisiveCriteria^{-}(o_c, o_r) \neq \emptyset$	Cost-benefit relationship	$DecisiveCriteria^{-}(o_c, o_r)$
Yes	$DecisiveCriteria^{-}(o_c, o_r) = \emptyset$	P returned by $DecisiveProsCons(o_c, o_r)$ Cost-benefit relationship	C returned by $DecisiveProsCons(o_c, o_r)$
No	$DecisiveCriteria^{-}(o_{2^{nd}Best}, o_c) \neq \emptyset$	Decisive Criteria pattern	
No	$DecisiveCriteria^{-}(o_{2^{nd}Best}, o_c) = \emptyset$	C returned by $DecisiveProsCons(o_{2^{nd}Best}, o_c)$	P returned by $DecisiveProsCons(o_{2^{nd}Best}, o_c)$
Rejected Options			
$Pros(o_r, o_c) > Cons(o_r, o_c)$?	**Decisive Criteria**	**Pros**	**Cons**
Yes	$DecisiveCriteria^{-}(o_c, o_r) \neq \emptyset$	$DecisiveCriteria^{-}(o_c, o_r)$	Cost-benefit relationship
Yes	$DecisiveCriteria^{-}(o_c, o_r) = \emptyset$	C returned by $DecisiveProsCons(o_c, o_r)$	P returned by $DecisiveProsCons(o_c, o_r)$ Cost-benefit relationship
No	$DecisiveCriteria^{-}(o_r, o_c) \neq \emptyset$	Decisive Criteria pattern	
No	$DecisiveCriteria^{-}(o_r, o_c) = \emptyset$	P returned by $DecisiveProsCons(o_r, o_c)$	C returned by $DecisiveProsCons(o_r, o_c)$

Table 9.1: Trade-off explanations: selection of pros and cons to be shown.

Explanation Type	Representation	Parameters
Critical Attribute	$\Psi_{crit}(o_c, a)$	$o_c \in Opt$ $a \in Att \wedge att = CriticalAttribute(o_c)$
Domination	$\Psi_{dom^+}(o_c)$ or $\Psi_{dom^-}(o_r, o_c)$	$o_c, o_r \in Opt$
Cut-off	$\Psi_{cutOff}(o_r, a)$	$o_r \in Opt$ $a \in Att \wedge a = CutOff(o_r, o_c)$
Minimum Requirements$^+$	$\Psi_{minReq^+}(o_c, a)$	$o_c \in Opt$ $a \in Att \wedge a = TieBreaker(o_c)$
Minimum Requirements$^-$	$\Psi_{minReq^-}(o_r, o_c, a)$	$o_c, o_r \in Opt$ $a \in Att \wedge a = TieBreaker(o_c)$
Decisive Criteria	$\Psi_{decisive}(o, target, atts)$	$o \in Opt$ $target \in \{chosen, rejected\}$ $atts \subset Att$
Trade-off Resolution	$\Psi_{tradeOff}(o, target, atts_P, atts_C, cb)$	$o \in Opt$ $target \in \{chosen, rejected\}$ $atts_P, atts_C \subset Att$ Pros and Cons $cb \in \{true, false\}$ Cost-benefit relationship is an argument?

Table 9.2: Explanation Types.

is very unlikely to occur but, as it is possible, we take it into consideration.

Explanations presented in Table 9.2 are all possible explanations that can be given either to justify choosing an option or rejecting an option. In some situations, more than one explanation can be given for justifying a decision, but we choose one of them based on a precedence order, which is shown below for the chosen option.

$$\Psi_{crit} \triangleright \Psi_{dom} \triangleright \Psi_{minReq^+} \triangleright \Psi_{decisive} \triangleright \Psi_{tradeOff} \tag{9-3}$$

In order to select an explanation according to this order, we propose Algorithm 13 that makes this selection. In this algorithm, we use $dominates(o_i, o_j)$ presented in Definition 6.6.1 (Chapter 6). The remaining procedures or functions were introduced in this chapter.

Similarly, for rejected options, we also establish a precedence order for the possible explanation types, as presented below. Algorithm 14 describes how an explanation is selected for a particular rejected option. As in the previous algorithm, the main idea is to verify if the conditions for using a pattern are satisfied, following the precedence order.

$$\Psi_{crit} \triangleright \Psi_{cutOff} \triangleright \Psi_{dom} \triangleright \Psi_{minReq^-} \triangleright \Psi_{decisive} \triangleright \Psi_{tradeOff} \tag{9-4}$$

9.4 The Apartment Example: Illustrating our Approach

Now, we come back to our running example introduced in Chapter 6 to show how explanations to justify the choice for the apartment Ap_B are generated. As there is neither an attribute that is a critical attribute for the decision, nor one that is a tie-breaker, the Critical Attribute and the Minimum Requirements patterns cannot be applied. Next, we discuss each option individually and provide for them an explanation for their rejection, or choice, in case of Ap_B.

Algorithm 13: $Explanation(o_c)$

Input: $o_c \in Opt$: chosen option
Output: Ψ: explanation to justify the choice

if $\exists\, a.(CriticalAttribute(o_c) = a)$ **then**
  return $\Psi_{crit}(o_c, CriticalAttribute(o_c))$;
if $\forall\, o_r.(dominates(o_c, o_r))$ **then**
  return $\Psi_{dom}(o_c)$);
$ok \leftarrow true$;
foreach $a \in Att$ **do**
  $\langle x, mod \rangle \leftarrow PSM[o_c, a]$;
  if $(x = empty \wedge IsNegative(mod)) \vee (x = \neg \wedge IsPositive(mod))$ **then**
    $ok \leftarrow false$;
if $ok \wedge \exists\, a.(TieBreaker(o_c) = a)$ **then**
  return $\Psi_{minReq^+}(o_c, TieBreaker(o_c)))$;
$D \leftarrow DecisiveCriteria^+(o_c)$;
if $D \neq \emptyset$ **then**
  return $\Psi_{decisive}(o_c, accept, D)$;
else
  if $\exists\, o_r.(Pros(o_c, o_r) < Cons(o_c, o_r))$ **then**
    $D \leftarrow DecisiveCriteria^-(o_c, o_r)$;
    if $D \neq \emptyset$ **then**
      return $\Psi_{tradeOff}(o_c, accept, \emptyset, D, true)$;
    else
      $\langle P, C \rangle \leftarrow DecisiveProsCons(o_c, o_r)$;
      return $\Psi_{tradeOff}(o_c, accept, P, C, true)$;
  else
    $o_{2^{nd}Best} \leftarrow o \mid o \in Opt \wedge min(Pros(o_c, o) - Cons(o_c, o))$;
    $\langle P, C \rangle \leftarrow DecisiveProsCons(o_{2^{nd}Best}, o_c)$;
    return $\Psi_{tradeOff}(o_c, accept, C, P, false)$;

Algorithm 14: $Explanation(o_r, o_c)$

Input: $o_r \in Opt$: a rejected option; $o_c \in Opt$: chosen option
Output: Ψ: explanation to justify the choice

if $\exists\, a.(CriticalAttribute(o_c) = a)$ **then**
  return $\Psi_{crit}(o_c, CriticalAttribute(o_c))$;
if $dominates(o_c, o_r))$ **then**
  return $\Psi_{dom}(o_r, o_c))$;
if $\exists\, a.(CutOff(o_r, o_c) = a)$ **then**
  return $\Psi_{cutOff}(o_r, CutOff(o_r, o_c))$;
if $Explanation(o_c)$ *is-a* Ψ_{minReq^+} **then**
  return $\Psi_{minReq^-}(o_r, TieBreaker(o_c)))$;
$D \leftarrow DecisiveCriteria^-(o_r, o_c)$;
if $D \neq \emptyset$ **then**
  return $\Psi_{decisive}(o_r, reject, D)$;
else
  if $Pros(o_r, o_c) > Cons(o_r, o_c)$ **then**
    $D \leftarrow DecisiveCriteria^-(o_c, o_r)$;
    if $D \neq \emptyset$ **then**
      return $\Psi_{tradeOff}(o_r, reject, D, \emptyset, true)$;
    else
      $\langle P, C \rangle \leftarrow DecisiveProsCons(o_c, o_r)$;
      return $\Psi_{tradeOff}(o_r, reject, C, P, true)$;
  else
    $\langle P, C \rangle \leftarrow DecisiveProsCons(o_r, o_c)$;
    return $\Psi_{tradeOff}(o_r, reject, P, C, false)$;

Ap_A**.** Option Ap_A is neither dominated by Ap_B nor has a value that makes it be cut-off. As the $DecisiveCriteria^{-}(Ap\_A, Ap\_B) \neq \emptyset$, rejecting this option is explained by stating the criteria that were decisive for its rejection. The explanation is $\Psi_{decisive}(Ap\_A, reject, \{zone, uni\})$, written as follows — we use the full attribute names, i.e. instead of "*uni*," we use "*distance from the university*."

> *Option* ***Ap_A*** *was rejected because of its* ***zone*** *and* ***distance from the university****.*

Ap_B**.** Option Ap_B is the chosen option, and, as mentioned before, there is no critical or tie-breaker attribute to justify the decision, and Ap_B also does not dominate all the remaining options. By executing Algorithm 13 the explanation type returned is decisive criteria, as $DecisiveCriteria^{+}(Ap\_B) = \{zone\}$, and therefore the explanation is $\Psi_{decisive}(Ap\_B, accept, \{zone\})$. Based on this returned explanation, we are able to generate the following statement, according to the pattern template.

> *Option* ***Ap_B*** *was chosen because of its* ***zone****.*

Ap_C**.** Option Ap_C has many attributes that are better than other options have, but its attribute $zone$ has a value that is not acceptable for the user. So, $CutOff(Ap\_C, Ap\_B) = zone$, and the given explanation is $\Psi_{cutOff}(Ap\_C, zone)$, informally written as shown below.

> *Option* ***Ap_C*** *was rejected because it does not satisfy constraints associated with* ***zone****.*

Ap_D**.** Even though option Ap_D is dominated by Ap_A, domination is not used as an explanation because Ap_A is not the chosen option. As $DecisiveCriteria^{-}(Ap\_D, Ap\_B) = \{uni\}$, the decisive criteria pattern is then used.

> *Option* ***Ap_D*** *was rejected because of its* ***distance from the university****.*

Ap_E**.** As Ap_E is not dominated by Ap_B, has acceptable attributes values and has no decisive criteria to justify its rejection, we have to identify the pros and cons that support the choice for Ap_B. $Pros(Ap\_E, Ap\_B) < Cons(Ap\_E, Ap\_B)$, but $DecisiveCriteria^{-}(Ap\_E, Ap\_B) = \emptyset$, so we have to execute $DecisiveProsCons(Ap\_E, Ap\_B)$. As a result, we have $\langle\{brand\}, \{uni\}\rangle$, and thus the explanation is $\Psi_{tradeOff}(Ap\_E, reject, \{brand\}, \{uni\}, false)$.

> *Even though option **Ap_E** provides better **brand** than the chosen option, it has worse **distance from the university**.*

Ap_F**.** The explanation for Ap_F is similar to that for Ap_E, but in this case $Pros(Ap\_F, Ap\_B) > Cons(Ap\_F, Ap\_B)$, indicating that the user-centric principles played an essential role in the decision. So executing $DecisiveProsCons(Ap\_B, Ap\_F)$, we have as result $\langle\{price\}, \{brand\}\rangle$, and thus the explanation is $\Psi_{tradeOff}(Ap\_F, reject, \{brand\}, \{price\}, true)$.

> *Even though option **Ap_F** provides better **brand** than the chosen option, it has worse **price** and **cost-benefit relationship**.*

9.5 Comparison with Related Work and Performance Evaluation

We now will discuss a comparison of our approach with two existing approaches (Klein and Shortliffe 1994, Labreuche 2011), which also address the selection of attributes to be part of the explanations. These approaches focus on explaining why one particular option is better than another, and not the whole explanation — they mainly focus on the attribute selection process. As many explanations are similar for the different options, we show the explanation generated for options Ap_E and Ap_F.

Both approaches assume that the decision is made based on MAUT, and the value of an option (how much an option is preferred) is a real number between 0 and 1, calculated by the weighted sum of values (how much an attribute is preferred) of attributes. We use our $Cost(o_i, o_j)$ function to select attributes using these approaches. Klein and Shortliffe's approach (Klein and Shortliffe 1994) relies on the concept of compellingness, which is used to select attributes whose values are above a threshold (calculated based on the average and standard deviation of values associated with option attributes). Labreuche's approach (Labreuche 2011), on the other hand, has different strategies for selecting attributes. Table 9.3 shows which attributes are selected for each of these approaches.

It can be seen that these two approaches and ours differ on the selected decisive criteria. Klein and Shortliffe's approach selects attributes based on a threshold, which in some cases does no capture all attributes needed to support the decision or selects too many. Therefore, as Labreuche argued, there is no formal reason why an attribute should be selected. Labreuche addressed this limitation using another approach — he analyses attributes weights (comparing them with average weights or switching them). As a consequence, in some scenarios, attributes that are not important (by having very low weight) and are associated with small

Approach	Ap_E	Ap_F
Klein and Shortliffe's approach	The attribute **distance from the university** provides the most compelling reason to prefer **Ap_B** over **Ap_E**.	**Ap_B** is preferred to **Ap_E** since criterion **zone** for which **Ap_B** is better than **Ap_E** is more important than criterion **brand** for which **Ap_B** is worse than **Ap_E**.
Labreuche's approach	The attributes **distance from the university**, **brand**, and **stars** are reasons to prefer **Ap_F** over **Ap_B**. The attributes **distance from the station**, and **price** are reasons to prefer **Ap_B** over **Ap_F**.	**Ap_B** is preferred to **Ap_F** since the intensity of preference **Ap_B** over **Ap_F** on **distance from the station**, and **price** is significantly larger than the intensity of preference of **Ap_F** over **Ap_B** on **distance from the university**, **brand**, and **stars**.

Table 9.3: Comparison of selected decisive criteria.

values are selected as part of explanations, and they may be irrelevant for the decision. Our approach follows Klein and Shortliffe's idea that the combination of attribute weights and values (in our case costs) are both important for selecting the decisive criteria, but we give a formal reason why an attribute should be selected as part of the explanation. It can be seen that both approaches selected all pros and cons to explain option Ap_F, which occurred because there is no compelling attribute according to the first approach, and only the remaining case anchor of the second approach applies to this option. This situation is more likely to happen when the user-centric principles have an important role in the decision.

After making this comparison with related work, we present a performance evaluation of our approach. We have implemented the proposed algorithms, and used this implementation to evaluate our approach and also the distribution of explanation types in a real scenario. This evaluation is based on the study of how humans express preferences in which we have preferences written in natural language by people, which were expressed with our preference language. These preference specifications (113 in total) were used to evaluate our decision making technique. We now use them to produce explanations based on the reasoning traces generated during the execution of the decision making technique.

On average, our technique takes $2.12ms$ (on a Intel Core 2 Quad $2.66GHz$, $4GB$ of RAM) to generate explanations, standard deviation 1.85, minimum $0.0ms$ (actually, $< 0.0ms$), and maximum $17ms$. This indicates that our explanation technique performs well in a real scenario: 144 options, 61 attributes, and realistic sets of preferences provided by people. All algorithms are executed at maximum in $6ms$, and sometimes less than $1ms$, as reported in Table 9.4, which also shows the percentage of each generated explanation types (distribution column). The *Explanation Generation Time* column reports the times to execute the algorithms

Explanation Type	Distribution	Explanation Generation Time (ms)				Pattern time (ms)		
	%	Min	Max	Avg	StDev	Pos	Neg	All
Chosen Option								
Critical Attribute	0.00%						0.03448	0.03448
Domination	1.72%	0.00	0.00	0.00000	0.00000			
Minimum Requirements	5.17%	0.00	1.00	0.83333	0.37268			
Decisive Criteria	58.62%	0.00	6.00	0.75000	0.81123	0.05882	0.30000	0.14815
Trade-off Resolution	21.55%	0.00	3.00	1.08000	0.56000			
Random	12.93%	1.00	2.00	1.06667	0.24944			
Total	100.00%	0.00	6.00	0.85345	0.71020			
Rejected Options								
Critical Attribute	0.00%							
Cut-off	13.03%	0.00	1.00	0.00117	0.03420	0.00000	0.00000	0.00000
Domination	42.78%	0.00	1.00	0.00383	0.06178	0.00014	0.00176	0.00097
Minimum Requirements	0.44%	0.00	1.00	0.01389	0.11703			
Decisive Criteria	31.64%	0.00	1.00	0.02188	0.14628	0.01409	0.00942	0.01285
Trade-off Resolution	11.06%	0.00	3.00	0.05656	0.23803	0.03065	0.00524	0.02803
Random	1.07%	0.00	1.00	0.01705	0.12944			
Total	100.00%	0.00	3.00	0.01420	0.11975			

Table 9.4: Explanation Evaluation.

$Explanation(o_c)$ and $Explanation(o_r, o_c)$, which generate explanations for the chosen and rejected options, respectively, while the *Pattern time* column shows the times to execute the algorithm used to generate an explanation of a particular pattern, such as the execution of the algorithm $DecisiveCriteria^{+}(o_c)$ to generate the explanation for the chosen option according to the Decisive Criteria pattern.

Besides the explanation types presented in this chapter, there is an additional one: *random*. This explanation type was added to address a scenario not discussed before, which consists of having options with the exact same values for attributes. Therefore, in this case we select one of them randomly. Note that the random choice is made between two or three options with the exact same values, and not among the whole set of available options.

The difference in pattern execution times is mainly because the first explanation generated is for the chosen option, but to verify the applicability of the Domination and Minimum Requirements^{+} patterns, we must analyse dominated options and cut-off attributes. As this information is stored when first evaluated, it does not need to be evaluated again to obtain explanations for rejected options.

9.6 Final Considerations

In this chapter, we presented a technique to generate explanations for users to justify choices made by our decision making technique. The technique is based on proposed guidelines and patterns, and provides a means of identifying parameters of explanation templates, which are part of the patterns. Our technique not only identifies these parameters, but also provides an algorithm to choose which explanation should be used in different cases. We illustrated the explanation generation with an apartment example, showing how we justify a chosen option,

and the rejection of the remaining ones. Moreover, we presented a performance evaluation of our approach, showing its efficiency, and the distribution of explanations produced in a real scenario. As this evaluation does not covers user acceptance of our explanation generation technique, we present in next chapter a user study performed to evaluate not only this technique, but also our preference metamodel and decision technique.

Part IV

Final

10
Evaluating our Approach with a User Study

We have presented an approach for automated decision making, which involves contributions in three different directions: (i) representation of high-level qualitative preferences; (ii) preference-based decision making with user-centric principles; and (iii) explanations to justify choices. Our preference metamodel and language (Chapter 3) are justified by our study of how humans express preferences (Chapter 2), the decision making technique (Chapter 6) that is able to process such language was evaluated by a comparison with a human domain expert, and our explanation generation technique (Chapter 9) is justified by our investigation of how people explain their choices (Chapter 8). In this chapter, we evaluate through a user study these three parts of our work, focusing on the explanation and its impact on the trust and confidence of users on choices made by the decision making technique, and the comparison with existing explanation approaches. The description of our empirical evaluation is presented in Section 10.1, and its results are detailed in Section 10.2. Finally, we discuss the threats to the validity of our study in Section 10.3.

10.1
Study Description

This last study that we conducted in the context of this thesis consists of an experimental evaluation to which we adopted the same framework (Basili et al. 1986) used in our previous studies, detailed in Chapters 2 and 8. Our study goal, according the GQM template (Basili and Rombach 1988) is shown in Table 10.1.

The study we designed to achieve this goal consists of within-subjects user study, in which we use a developed application that allows participants to (i) express their preferences in a high-level language; (ii) receive a choice made by our decision making technique; and (iii) receive different explanations that justify this choice. With this application, we ask for participants' feedback with respect to these three dimensions and use this information to evaluate our approach. Moreover, the domain chosen for this study is *choosing a mobile phone to buy*, as mobile phones can be described in terms of attributes of different types, in general people have a

Definition element	Our experiment goal
Motivation	To assess the impact of different explanations on automated decision making,
Purpose	evaluate
Object	the user understanding, trust and confidence on choices made
Perspective	from a perspective of the researcher
Domain:people	as users receive explanations to justify those choices
Domain:system	from a decision making system
Scope	in a within-subjects study.

Table 10.1: Goal Definition (GQM template).

set of preferences they are aware of to make this choice (known preferences), and there are plenty of mobile phones available.

In next sections, we describe the research questions and hypotheses (Section 10.1.1), the procedure (Section 10.1.2) and participants (Section 10.1.3) of our study.

10.1.1 Research Questions and Hypotheses

Even though the main goal of this study is to investigate the impact of explanations on the user understanding, trust and confidence on choices made, and compare existing explanation approaches with respect to these dimensions, we also evaluate other aspects that our approach includes: (i) the language expressiveness; (ii) the decision making technique's choices; and (iii) the impact of explanations. Therefore, there are different research questions related to this study, presented below.

RQ1. Is our high-level preference language expressive enough for users to provide their preferences about a domain?

RQ2. Does our decision making technique make choices on user's behalf that they consider good?

RQ3. Do explanations increase the user understanding, trust and confidence on why a particular choice is made?

RQ4. Do different kinds of explanations (generated using using our approach (Chapter 9), Klein and Shortliffe's (Klein and Shortliffe 1994) and Labreuche's (Labreuche 2011)) have a different impact on the user understanding, trust and confidence on why a particular choice is made?

By answering these research questions, we are able to identify issues of our existing approach with respect to missing available preference types for users to

express themselves, and the quality of the decisions made by our technique, thus indicating points that should be addressed as future work. Moreover, we are able to conclude how much an explanation changes the evaluation of a choice by a user, and identify the best explanation generated by different approaches available.

But, as we mentioned before, our main focus is on the impact of explanations on automated decision making systems and the comparison between the different explanation approaches, and we list below our null hypotheses related to them.

$H1_0$: *The **choice quality** perceived by users on the choice made does not change after presenting them an explanation that justifies it.*

$H2_0$: *The **trust** of users in the choice made does not change after presenting them an explanation that justifies it.*

$H3_0$: *The user **decision confidence** on the choice made does not change after presenting them an explanation that justifies it.*

$H4_0$: *The three investigated kinds of explanations have the same impact on the **understanding of why (transparency)** why choices were made.*

$H5_0$: *The three investigated kinds of explanations have the same impact on the **choice quality** perceived by users on the choice made.*

$H6_0$: *The three investigated kinds of explanations have the same impact on the **trust** in choices made.*

$H7_0$: *The three investigated kinds of explanations have the same impact on the user **decision confidence** on choices made.*

10.1.2 Procedure

In order to answer our research questions and test our hypotheses, we have designed a user study in which participants interact with a developed software system, which implements the three components of our approach, namely the preference language, the decision technique and the explanation generation technique, and an interface to collect and display data. Moreover, it is also able to generate explanations with the approaches proposed by Klein and Shortliffe (Klein and Shortliffe 1994), and Labreuche (Labreuche 2011). The study consists of within-subjects comparing the impact of these three explanation approaches, besides analysing other aspects related to our decision making approach. Furthermore, our study involves making a decision about mobile phones, creating a hypothetical scenario in which participants are going to buy a mobile phone and need to choose one model from a set of available. As already explained, we chose mobile phones as the domain of the decision to be made as it fits the requirements of using our approach — most of people are aware of the attributes

that characterise mobile phones, and have preferences over individual attributes. Furthermore, we can retrieve a mobile phone database in a relatively easy way.[1] In addition, this domain is different from the domain of our previous studies — choosing a laptop (Chapter 2) and a hotel (Chapter 8) — from which we derived our preference language and explanation patterns, respectively.

Each participant has to go through seven steps while taking part of the study, each of which is described next. Screenshots of the application developed to be used in the study can be seen in Appendix D. The application has two available languages: English and Portuguese.

1. **Participant Data.** In order to collect demographic information of the participants, they are required to provide the following data: (i) age; (ii) gender; (iii) location (city and country); and (iv) working/studying field.

2. **Preferences.** The study participants are requested to imagine a situation in which they are going to buy a new mobile phone. In addition, in this scenario, they are provided with an intelligent system that will make a choice on their behalf and asks them to specify their preferences and restrictions over the mobile phone they want. Participants are able to specify their preferences using our language (presented in Chapter 6) through an interface that has many features, such as choosing explanation types with radio buttons, then selecting preferences parameters with combo boxes, setting preference formulae in a similar way to specifying rules in e-mail clients, and so on. Before providing their preferences, the participants receive a brief tutorial on how to interact with the interface and explanations about the language constructions. The application also is able to list the mobile phone attributes, and descriptions of each preference and priority type. For recording purposes, we store how long participants take to specify their preferences.

3. **Preference Language Evaluation.** After specifying their preferences, we request the participants to evaluate the interface and language they used in terms of two aspects: *perceived ease of use* and *perceived effort*, whose associated questions that are asked to participants are shown in the "Preference Language Evaluation" part of Table 10.2, and the possible answers for such questions are according to a 7-point Likert scale. These measured variables as well as others that are adopted in next steps of the study are part of a user evaluation framework of recommender systems (Chen and Pu 2010), whose questions were adapted to our study. Moreover,

[1]Imported from the Best Buy store (`http://www.bestbuy.com`), available through a REST API located at `https://bbyopen.com/developer`.

participants are requested to describe any preferences that they could not express in our language.

4. **Choice Analysis and Evaluation.** Based on the provided participants' preferences, we choose an option using our decision making technique, and present to the participant: (i) the chosen mobile phone; (ii) the next four mobile phones of the acceptable set ranked according to the decision function of our technique (so as to form five chosen options, which was deemed an adequate number in our previous study); and (iii) the remaining mobile phones initially hidden, but the participants can see them upon request to analyse all the 191 available mobile phones. Now, with this presented choice, participants are asked to evaluate it by answering the questions in the "Choice Evaluation" part of Table 10.2, which are related to variables: *choice quality*, *decision confidence* and *trust in choice*. They also have to specify which mobile phone they would choose, if they had to make the choice themselves.

5. **Explanation Impact.** After evaluating the chosen mobile phone, participants are provided with explanations to justify the choice made generated using our technique. The participants are then requested to answer questions in the "Explanation Impact and Explanation Comparison" part of Table 10.2, which are the same presented in the previous step together with a question related to the *transparency* variable. The goal of asking the same questions again is to evaluate if the choice quality perceived by users, and their trust and decision confidence change after receiving explanations. The questions related with these measures are initialised with the answers previously provided by the participants.

6. **Explanation Comparison.** The participants now receive the three possible explanations — generated by (i) our approach; (ii) Klein and Shortliffe's approach; (iii) Labreuche's approach — in a side-by-side comparison, and have to answer the same questions of the previous step. As our approach was already presented in the previous step, its answers are already initialised. Participants are requested to compare the three given explanations and evaluate them, and they are notified that they can review their opinion about the previously present explanation. In order not to create a pre-defined explanation order, we change the order of the explanations for different participants.

7. **Approach Evaluation.** Finally, participants have to answer final questions that evaluate the approach as a whole, which are shown in the "Approach Evaluation" part of Table 10.2.

Measured Variable	Question responded on a 7-point Likert scale from "strongly disagree" to "strongly agree"
Preference Language Evaluation	
Perceived ease of use	I find this interface to provide my preferences easy to use.
Perceived effort	Providing my preferences in this language required too much effort (*reverse scale*).
Choice Evaluation	
Choice quality	This application made really good choices.
Trust in choice	I feel that this application is trustworthy.
Decision confidence	I am confident that the choice made is really the best choice for me.
Explanation Impact and Comparison	
Transparency	I understand why the products were returned through the explanations in the application.
Choice quality	This application made really good choices.
Trust in choice	I feel that this application is trustworthy.
Decision confidence	I am confident that the choice made is really the best choice for me.
Approach Evaluation	
Perceived usefulness	This application is competent to help me effectively make choices I really like.
Intention to purchase	I would accept this choice if given the opportunity.
Intention to return	If I had to search for a product online in the future and an application like this was available, I would be very likely to use it.
Intention to save effort in next visit	If I had a chance to use this application again, I would likely make my choice more quickly.
Enjoyment	I found my visit to this application enjoyable.
Satisfaction	My overall satisfaction with the application is high.

Table 10.2: Measured Variables — adapted from (Chen and Pu 2010).

In the last four steps of the study, participants are also able to provide further comments. With this collected information we are able to extract issues of our current approach, evaluate our language, decision making technique, and existing explanation approaches (including ours). In next section, we present the participants involved in our study.

10.1.3 Participants

As in our previous studies, we selected participants using convenience sampling, by making invitations for volunteers via email to the social network of the researchers involved in this study. However, as participants were observed while taking part of the study, only participants in the same locations (in two

Gender		Male 21 (60%)		Female 14 (40%)	
City		Porto Alegre 19 (54.29%)	Rio de Janeiro 14 (40%)	Other 2 (5.71%)	
Age	<16 years 1 (2.86%)	16-25 years 9 (25.71%)	26-35 years 17 (48.57%)	36-45 years 2 (5.71%)	>45 years 6 (17.14%)
Field of Work of Study		Informatics 16 (45.71%)	Engineer 5 (14.29%)	Law 5 (14.29%)	Other 9 (25.71%)

Table 10.3: Demographic Characteristics of Participants.

different Brazilian cities) of the researchers could be selected — two participants were visiting from other locations. The demographic characteristics of the study participants, which are 35 in total, are described in Table 10.3.

10.2 Results and Analysis

Now, we will present the data collected in the study we performed, and discuss its results, which are split into four parts: (i) the analysis of provided preferences and the evaluation of the preferences language (Section 10.2.1); (ii) the evaluation of the choice made by our decision making technique and the impact of the given explanation (Section 10.2.2); (iii) the comparison between the different provided explanations (Section 10.2.3); and (iv) the evaluation of the overall approach (Section 10.2.4).

10.2.1 Preferences and Language Evaluation

After providing their personal data, participants had to specify their preference about mobile phones. As explained before, we briefly introduced to the participants the interface for providing preferences. The types of preferences (qualifying, rating, goal, etc.) were presented in a radio button list, from left to right (see Figure D.2(a) in Appendix D), and in order to avoid the bias of users always starting by the same first preference type (qualifying), we introduced the preferences in the opposite direction, starting from the *don't care*. On average participants took $15 min$ to specify their preferences, including the time to give the brief tutorial.

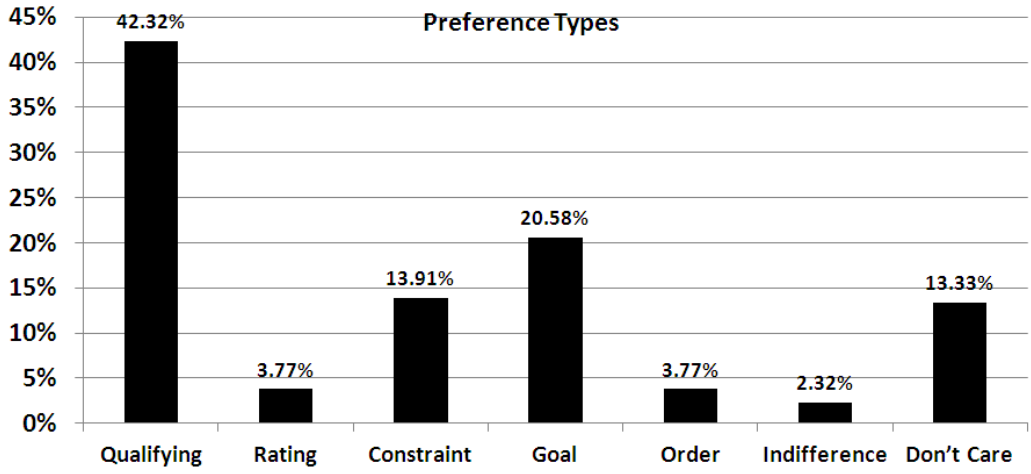
By observing the interaction of participants with the interface, we noticed they first took a few moments to get familiar with it, and then explored the available preference types, priorities and attributes. Many participants began providing their preferences by specifying the attributes they do not care about, for later concentrating on the characteristics they desire. As participants had the list of attributes available to them, they often looked at this list to remember their preferences — after providing a set of preferences, some checked the list again,

and said "*let me check if I forgot something*." People that have no or little interest in mobile phones seemed to not know what to specify, mainly because the attributes provided were not at a high level, such as "easy to use." Therefore, in order to explicitly provide preferences in this way, it is important for users to have at least some knowledge about the domain.

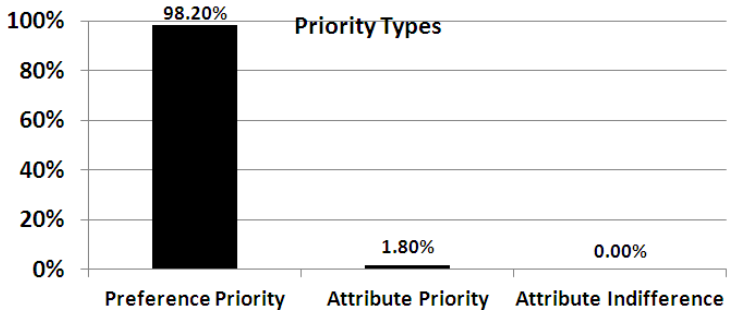
A set of histograms of the provided preferences is shown in Figure 10.1, where we present the distribution of the different preference types used, the different priority types used, and the expressive speech acts and rates used in qualifying and rating preferences. It can be seen in Figure 10.1(a) that *qualifying* is the most used preference type (42.32%), which indicate that people tend to use expressive-speech-act-based statements, which is one of the main advantages of our language with respect to existing preference languages. The most frequent set of preferences provided by participants is a combination of qualifying preferences with goals. The majority of participants — 26 (74.29%) — adopted preference priorities among preferences (i.e. numbered preferences), instead of priorities among attributes, and some split preferences into groups, that is, a set of preferences of priority 1, priority 2, and so on. Finally, the most common expressive speech act used was *want* (57%), but a wide variety of other expressive speech acts and rates were used.

Participants were then requested to evaluate their experience in providing preferences through the given interface and language, and the obtained results are shown in Figure 10.2. We observed during the execution of our study that it was not straightforward for older people (age > 45 years) to use the interface (including those who work with informatics), and also for them to provide their preferences as they are not familiar with the domain. Therefore, they are responsible for the worst scores with respect to the *perceived ease of use* (M = 5.63 and SD = 1.19) and the *perceived effort* (M = 3.06 and SD = 1.59) measurements. Even though the interface was considered ease to use for most of the participants, they reported that it is not intuitive, and without the given explanation it would not be easy to interact with it.

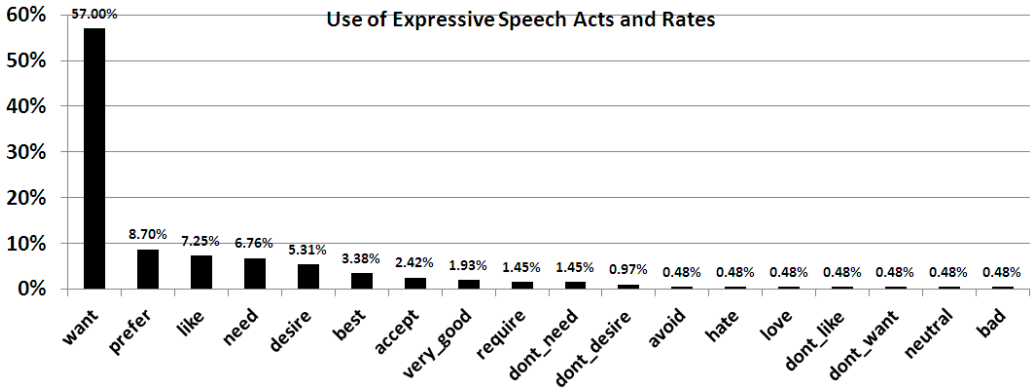
With regard to the effort spent in providing preferences, 45.72% participants (strongly) disagreed that providing preferences in our language requires too much effort. This issue can be divided into two parts: effort required to provide preferences and effort to express these preferences in our language. The additional question we asked of participants, which requests them to list any other preferences they wanted to express and they could not, helped us to distinguish what the participants meant. 25.71% of the participants pointed out limitations, but only 8.57% reported *language* limitations. Most of the limitations are related to missing attributes, such as alarm clock, operating system version and processor, or even subjective ones,



10.1(a): Preference Types.

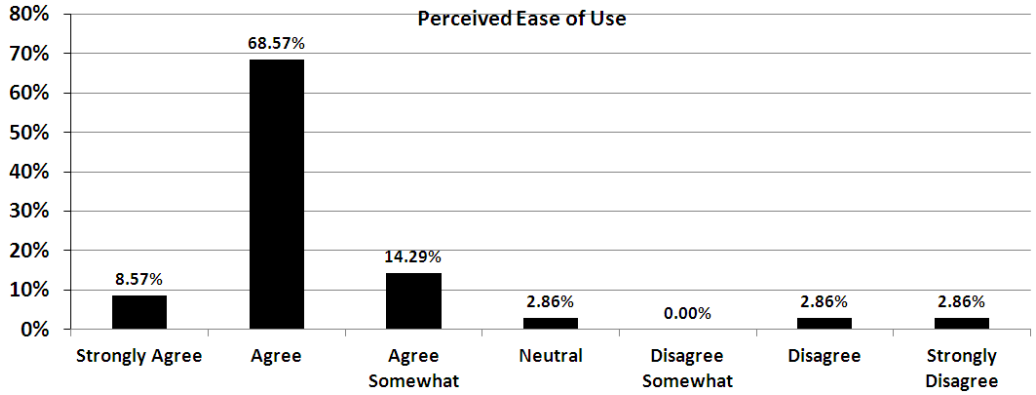


10.1(b): Priority Types.

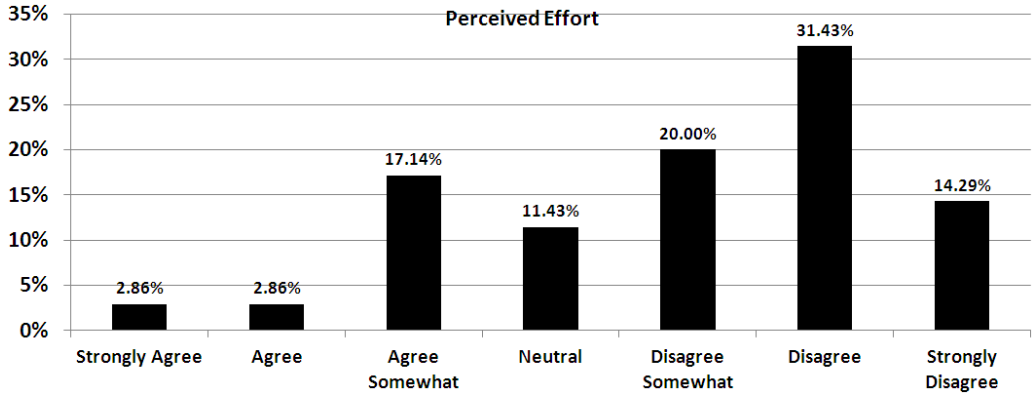


10.1(c): Use of Expressive Speech Acts and Rates.

Figure 10.1: Preference Analysis.



10.2(a): Perceived Ease of Use.



10.2(b): Perceived effort.

Figure 10.2: Language Evaluation.

e.g. usability, or ways of referring to attribute values, e.g. good or small, which are not missing preference constructions, but restrictions of the domain. Some of these restrictions are associated with elements that are part of our preference metamodel (e.g. scales and adjectives) that our decision making technique is unable to handle, and others are related to the engineering of the domain. Based on the given participants' comments, we make three observations.

1. It is important to make it explicit to users how they can express hard constraints. Some participants asked how they could express a restriction that *must* be satisfied, but they were not informed during the study. From these, some intuitively used *require* or *need*, which indicate hard constraints in our technique, but others did not.

2. It is interesting to consider the possibility of extending the language or having a user interface that provides "shortcuts" for the expression of more complete specifications to allow users to express a subset of the most important attributes, which was a limitation pointed out by one participant.

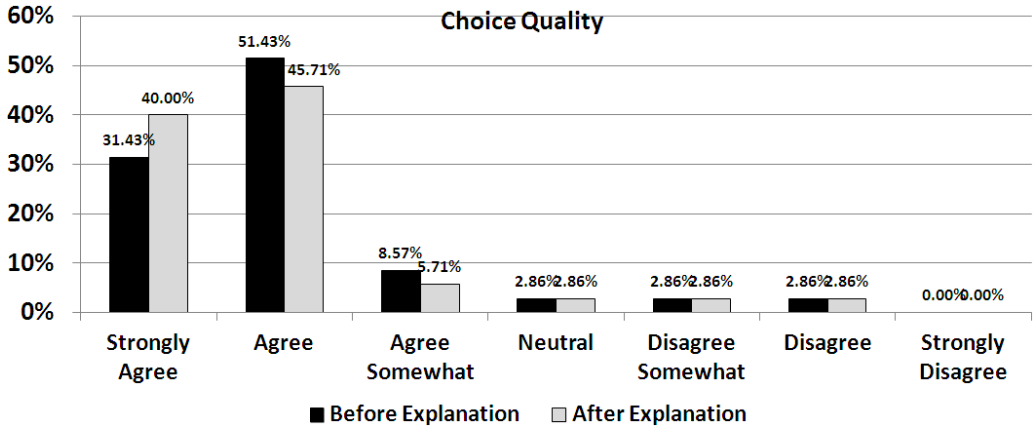
3. A single participant stated that the many different ways of expressing preference is confusing, while the others (97.14%) seemed to be comfortable with having many alternatives.

10.2.2 Choice Evaluation and Explanation Impact

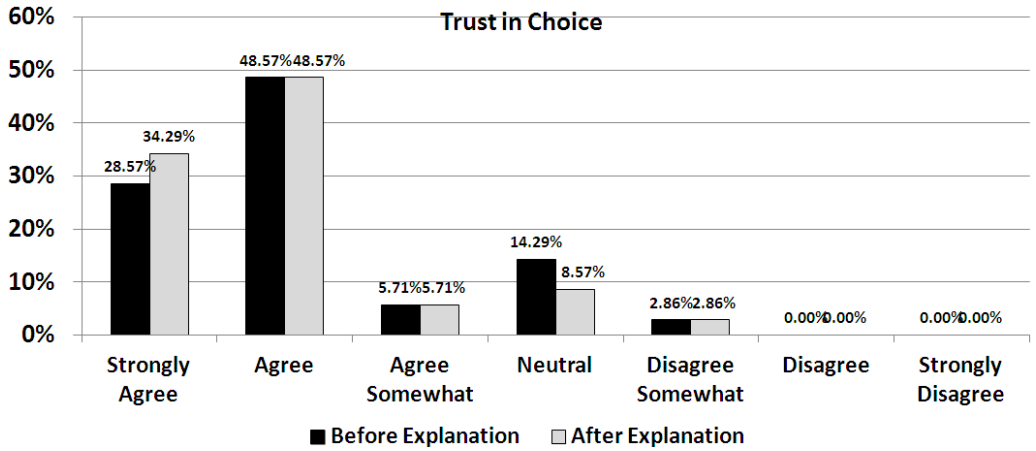
The next two steps of our experiment, namely choice evaluation and explanation impact, are discussed together. Based on the provided preferences, participants had a choice made on their behalf, and they had to evaluate this choice (and other selected options that are close to the chosen option) with respect to the *choice quality*, *trust in choice* and *decision confidence* — the obtained results are presented in Figure 10.3. It can be seen that our technique was evaluated with high levels of choice quality and trust, and also of decision confidence (but at a level lower than the other measurements) indicating that our technique is able to make adequate choices. We identified three situations in which the decision made was not good, detailed next.

- *Establishing a preference order without specifying preferred values.* If a preference like "I prefer X to Y" is specified, an order is established between X and Y, but not between them and the remaining values. However, participants that provided this kind of preference had in mind that they specified that X and Y are preferred to all the other values, and they could not see how this preference was taken into account in the presented choices.
- *Establishing a preference indifference without specifying preferred values.* Similarly to above, some participants stated that they are indifferent to X and Y, but did not specify that these are preferred values. Therefore, they also could not see how this preference was taken into account in the presented choices.
- *Providing a hard constraint that is not compatible with most of the models.* One participant provided a hard constraint that caused most of the models (which satisfy her other preferences) to be discarded, and received choices that did not satisfy her other preferences. As she did not pay attention to this constraint when analysing options, she did not agree with the choices made.

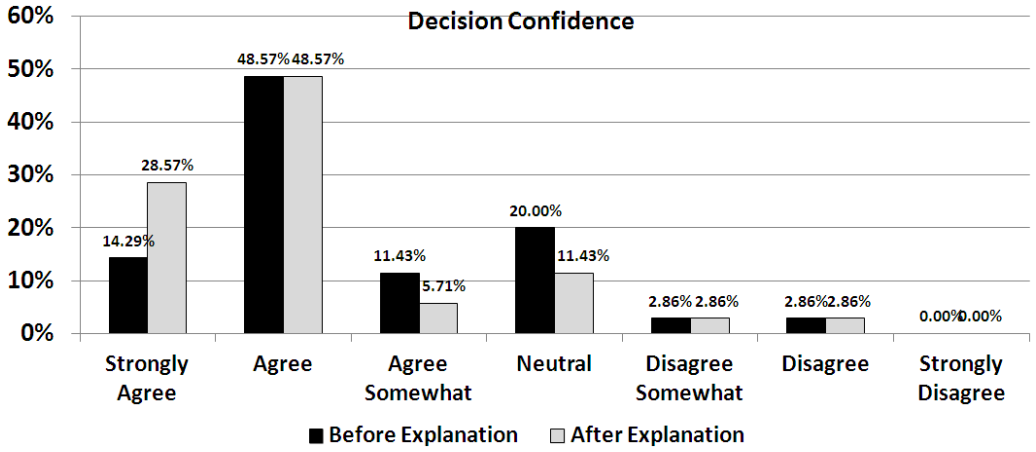
An interesting situation is that some participants did not (strongly) agree with the decision, but they said they understood it. These participants realised that they forgot to specify something in their preferences as the first choices had undesired characteristics, which they did not mention as a preference.



10.3(a): Choice Quality.



10.3(b): Trust in Choice.



10.3(c): Decision Confidence.

Figure 10.3: Choice Evaluation and Explanation Impact.

Measurement	Before Explanation		After Explanation		Wilcoxon Test	p-value
	M	SD	M	SD		
Choice Quality	5.97	1.12	6.09	1.15	$W(34) = 3.50$	0.102
Trust in Choice	5.86	1.09	6.03	1.01	$W(34) = 2.50$	0.084
Decision Confidence	5.43	1.22	5.80	1.23	$W(34) = 0.00$	0.006

Table 10.4: Choice evaluation and explanation impact measurements.

Even though our technique received high levels on the evaluation of choices made, only 26.47% of the participants would choose the option chosen by the technique, and 52.54% would choose one of the up to five selected options.

After receiving explanations generated with our approach, participants had to answer these same questions in order to evaluate the impact of the explanations. It can be seen some participants changed their evaluation from less positive to more positive rates, showing that the explanations tend to increase the choice quality perceived by users, and the trust and decision confidence on choices made. A Wilcoxon Signed Ranks test was conducted to compare these three measurements before and after explanations, and it showed that the explanations have a significant impact only on the decision confidence. Therefore, *we cannot reject hypotheses* $H1_0$ *and* $H2_0$*, but we reject hypothesis* $H3_0$. We summarise the results in Table 10.4.

Some participants provided only a few preferences over boolean attributes, which split the available mobiles phones into two groups: those that satisfy these preferences, and those that do not. In these cases, the explanations just expose this situation, stating that a random choice was made between options that satisfy all preferences, which explains the choice but it is not very helpful, thus most of the participants in this case did not change their evaluation rates. There was only one case in which a participant decreased her rates on these measurements, showing one shortcoming of our approach. This participant specified a preference for an attribute, and later she added a don't care preference related to this same attribute. Therefore, the decision support models (PSM and OAPM) took that preference into account, but the decision function ignored it because of the don't care preference. However, some of the explanations took into account the decision models and mentioned that attribute, and the participant complained that she stated she does not care about it.

Finally, participants had also to evaluate whether they could understand why the choice was made based on the provided explanations (*transparency*) and, as it can be seen in Figure 10.4, most of the participants agreed with that. We observed that participants expected that important attributes related to unsatisfied preferences should always be mentioned, which was not always the case. The average and standard deviation of the transparency measurement are $M = 6.20$ and $SD = 1.21$, respectively.

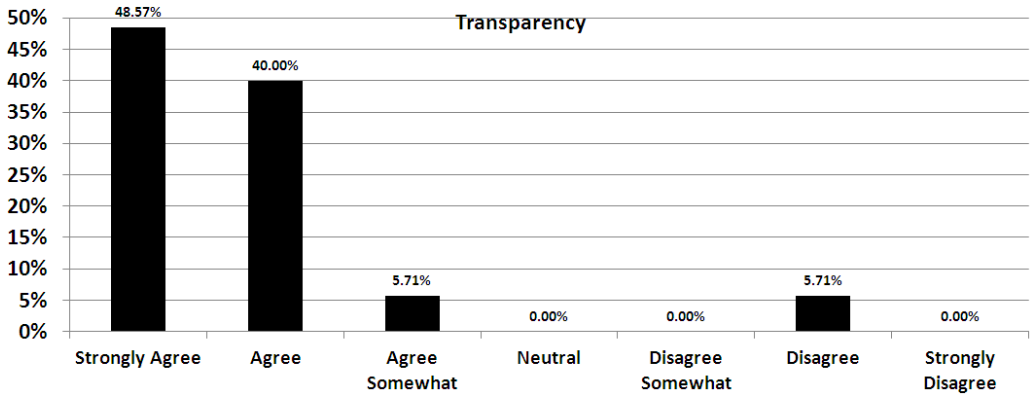


Figure 10.4: Explanation Impact — Transparency.

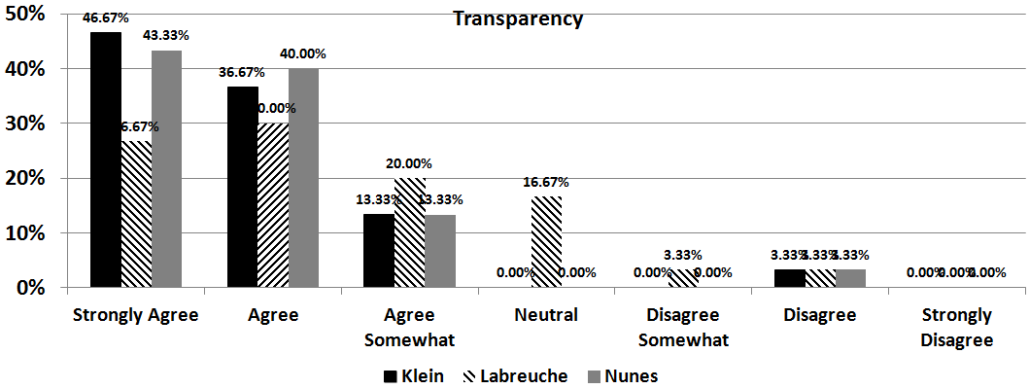
10.2.3 Explanation Comparison

After analysing the explanation impact, we compare explanations generated by different techniques. In order to do so, participants made a side-by-side comparison of three techniques, including ours, evaluating them with respect to the same criteria as above. Klein and Shortliffe's and Labreuche's approaches receive a utility function as input to generate an explanation that justifies why an option is better than another. As this comparison is made in a pairwise fashion, we can use the values of our cost function as input of these approaches.

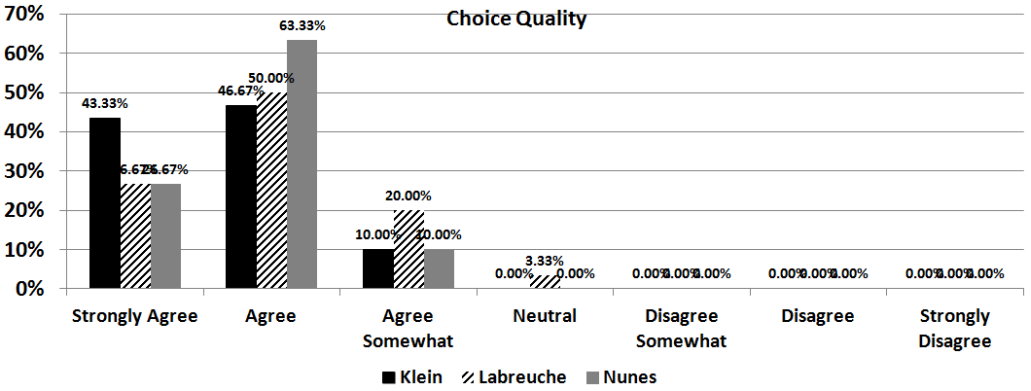
It is important to highlight that we established a timeout of $2min$ for each approach to generate explanations, as one of the possible explanations of Labreuche's approach is associated with a branch-and-bound algorithm, which can take a considerable amount of time to run (the algorithms of the other approaches run in polynomial time). This timeout was tested with a pilot study, and it showed that giving more time to the execution of the algorithm would make the participants lose engagement in the study. When this timeout is reached, all the explanations generated up to this moment are shown. While other approaches always executed in less than $2min$, there were five cases in which Labreuche's approach presented no results (which were discarded for comparing the different explanation techniques), and ten other cases that it presented six or less explanations. This already points out a limitation of Labreuche's approach.

We begin by presenting the collected data during this step, and we will later discuss relevant points we observed during the execution of this part of the study. The results of the comparison of transparency, choice quality, trust in choice and decision confidence among the three explanation approaches are shown in Figures 10.5 and 10.6.

First, we noticed that participants focused on the first selected options (this observation is also valid for the two steps above), and sometimes they did not even



10.5(a): Transparency.



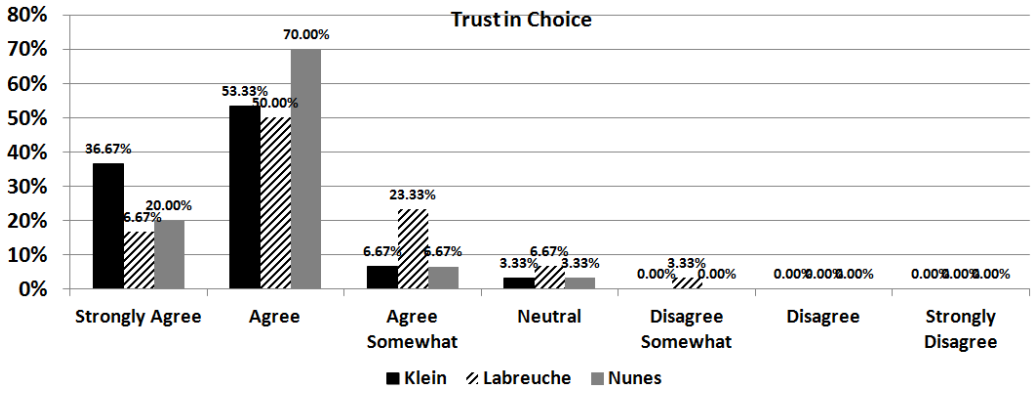
10.5(b): Choice Quality.

Figure 10.5: Explanation Comparison (I).

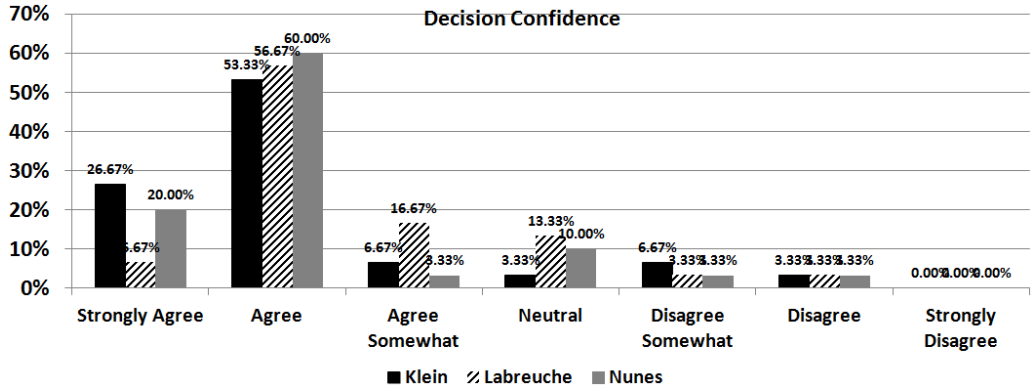
open the view in which all options with explanations are shown. In some cases, when participants had a specific preferred model, they searched for this model to verify why it was not chosen. Therefore, participants may have answered questions based on a subset of explanations.

The explanation approach that received the best scores on average was Klein and Shortliffe's. This approach provides only one type of explanation, which selects a subset of relevant attributes based on a threshold. As a consequence, their approach basically exposes the main positive (and possibly) negative aspects of options, helping the user to confirm that the choice made is right or wrong, but it does not actually explain the choice. In some cases, the explanation can be quite long, as the threshold may include many attributes.

The worst rates were given to Labreuche's approach. Based on our observations, this is mainly due to the complexity of their approach. In many cases the explanation given follows the *invert* pattern, which indicates the relationship of important pros and not important cons. Participants read these explanations more than once to understand its content, and there were reports that the explanation is complex and too long. Additionally, there were cases in which the domination



10.6(a): Trust in Choice.



10.6(b): Decision Confidence.

Figure 10.6: Explanation Comparison (II).

pattern was presented, and participants usually gave lower scores for explanations involving many situations of domination. What happened is that in these cases options were dominated because of the participants' preferences, and not actually because they were worse in all aspects, and this was not considered an appropriate explanation by the participants. For example, if a participant states that she wants a mobile phone with 32GB of internal memory, according to her preferences a mobile phone with 64GB is "worse" than one with 32GB, but in practice it is not.

This domination problem was also identified in our approach, which is one of the reasons why our approach received a few scores lower than Klein and Shortliffe's approach, and an example of this situation is shown in the *Domination* row of Table 10.5. Another problem we have identified is associated with the explanation for the chosen option, which highlights the key aspects that caused this option to be chosen. Many participants interpreted that the mentioned attributes in this explanation (mainly when there was only one attribute) were the only ones taken into account, such as the case shown in Table 10.5 (*Chosen Option Explanation* row). Because of these issues, which indicate possible improvements to be done, our approach was rated with lower scores in these two situations

Situation	Klein	Nunes
Domination	The attributes front-facing camera resolution (MP), and touch screen provide the most compelling reason to prefer mobile phone X over mobile phone Y.	There is no reason to choose option mobile phone Y, as option mobile phone X is better than it in all aspects.
Chosen Option Explanation		Option mobile phone X was chosen because of its brand.
More concise		Option mobile phone X was chosen because of its camera resolution (MP), and price.
	While height (cm) is a compelling reason to prefer mobile phone W over mobile phone X, price (US$) is a compelling reason for not choosing it.	Even though option mobile phone W provides better height (cm) than the chosen option, it has a worse price (US$).
	While height (cm) is a compelling reason to prefer mobile phone Y over mobile phone X, camera resolution (MP), touch screen, and price (US$) is a compelling reason for not choosing it.	Option mobile phone Y was rejected because of its touch screen and camera resolution (MP).
	While height (cm) is a compelling reason to prefer mobile phone Z over mobile phone X, downloadable games, camera resolution (MP), touch screen, and price (US$) is a compelling reason for not choosing it.	Option mobile phone Z was rejected because it does not satisfy constraints associated with downloadable games.
Different Text	The attribute QWERTY keyboard provides the most compelling reason to prefer mobile phone X over mobile phone Y.	Option mobile phone Y was rejected because of its QWERTY keyboard.

Table 10.5: Examples illustrating the main differences between Klein and Shortliffe's approach and ours.

(for 8 participants). However, *in general our approach received similar scores to their approach, and sometimes even higher* (for 8 participants as well), which are associated with situations in which Klein and Shortliffe's threshold is not adequate — as exemplified in the *More concise* of Table 10.5. We also noticed that the "*cost-benefit relationship*" causes a positive impact on the explanations, and when it was given to participants, they gave higher scores to our approach than to the others.

Finally, we make a comment on the text associated with the explanations — we implemented each approach following the text suggested by their respective authors. Explanation approaches focus on the algorithms that select parameters of templates to build explanation sentences. Nevertheless, in some cases Klein and Shortliffe's approach and ours selected the same attributes to present to participants, and their only difference was the text associated with it, as shown in row *Different Text* of Table 10.5, and the rates given to them were different. One participant also complained that Labreuche's approach mentions the mobile phone name too many times in the explanation. Issues related to the text of the sentences are easy to be

Measurement	Klein		Labreuche		Nunes		Friedman's Test	p-value
	M	SD	M	SD	M	SD		
Transparency	6.20	1.06	5.50	1.33	6.17	1.05	$\chi^2(2) = 2.7578$	0.0161
Choice Quality	6.33	0.66	6.00	0.79	6.17	0.59	$\chi^2(2) = 2.6958$	0.0193
Trust in Choice	6.23	0.73	5.70	0.95	6.07	0.64	$\chi^2(2) = 3.3428$	0.0024
Decision Confidence	5.80	1.27	5.40	1.13	5.73	1.20	$\chi^2(2) = 2.6817$	0.0201

Table 10.6: Explanation comparison measurements.

tackled, and must be taken into account as they are related to how users perceive the quality of the explanations.

In summary, Klein and Shortliffe's explanations are generally good, as showing option pros and cons is in general helpful, but there are cases in which a more specific explanation is better. This is what Labreuche's approach and ours aim to do, but they generate inadequate explanations in some cases. While Labreuche's approach turned explanations too complex for the participants, our approach managed to provide good explanations for them, with some exceptions, which were listed above.

In order to test whether the difference among the different explanation techniques is statistically significant, we used the Friedman's test. As it can be seen in Table 10.6, all the measurements differ significantly across the three explanation approaches. Therefore, *we can reject hypotheses* $H4_0$*,* $H5_0$*,* $H6_0$ *and* $H7_0$.

Given that we have a significant difference for the measurements, we further performed the post-hoc tests of Wilcoxon-Nemenyi-McDonald-Thompson, which shows us that the differences are due to the following.

- **Transparency**
  - Klein and Labreuche (p-value= 0.0161)
  - Nunes and Labreuche (p-value= 0.0255)
- **Choice Quality**
  - Klein and Labreuche (p-value= 0.0192)
- **Trust in Choice**
  - Klein and Labreuche (p-value= 0.0024)
- **Decision Confidence**
  - Klein and Labreuche (p-value= 0.0200)
  - Nunes and Labreuche (p-value= 0.0268)

Therefore, we conclude that (i) Klein and Shortliffe's approach provides a significant improvement for all measurements with respect to Labreuche's

approach; (ii) our approach provides a significant improvement for transparency and decision confidence with respect to Labreuche's approach; (iii) Klein and Shortliffe's approach and ours have no significant difference for all measurements.

10.2.4 Approach Evaluation

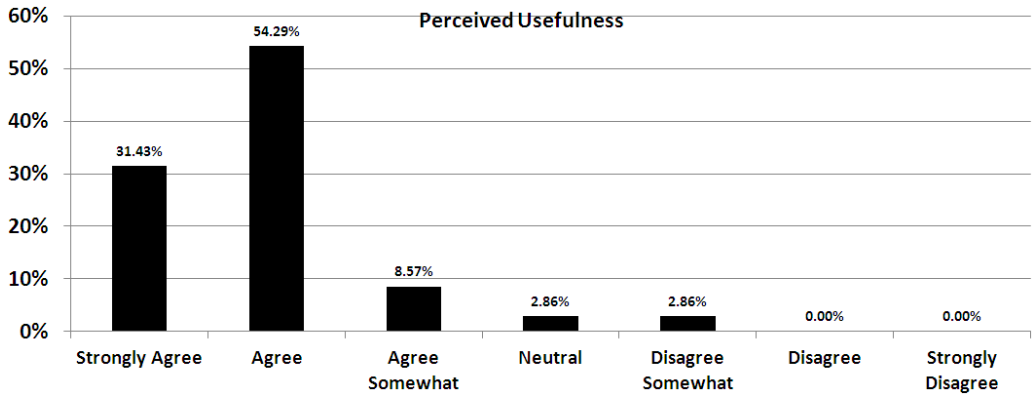
In the last step of the experiment, participants had to answer questions whose goal is to evaluate the whole approach: the experience while providing preferences, the choices made, and the explanations given. Figures 10.7 and 10.8 depict the results obtained, showing that a representative amount of participants answered (strongly or somewhat) agree for the *perceived usefulness* (94.29%), *intention to return* (97.14%), *enjoyment* (97.14%) and *satisfaction* (94.28%). There were participants that declared that they indeed needed a mobile phone and they were happy with the system and the recommended choices. Table 10.7 shows the average and standard deviation for all measurements.

Two of the measurements do not follow this case: *intention to purchase* and *intention to save effort*. Although many participants stated that the choice quality is good (first five options), they were not sure or disagreed that the chosen option is the best for them. Consequently, it may justify why 17.14% of the participants answered neutral or disagree with respect to the intention to purchase. In addition, this measurement depends greatly on purchase habits (e.g. impulsive vs. careful), being also a reason for the lower intention to purchase average. In fact, we included this measure in our study as it is part of Chen and Pu's framework (Chen and Pu 2010), but given that there are many variables that influence this measure (e.g. purchase habits), it may have a higher or lower score according to the personality of the participant and not her opinion with respect to the choice made.

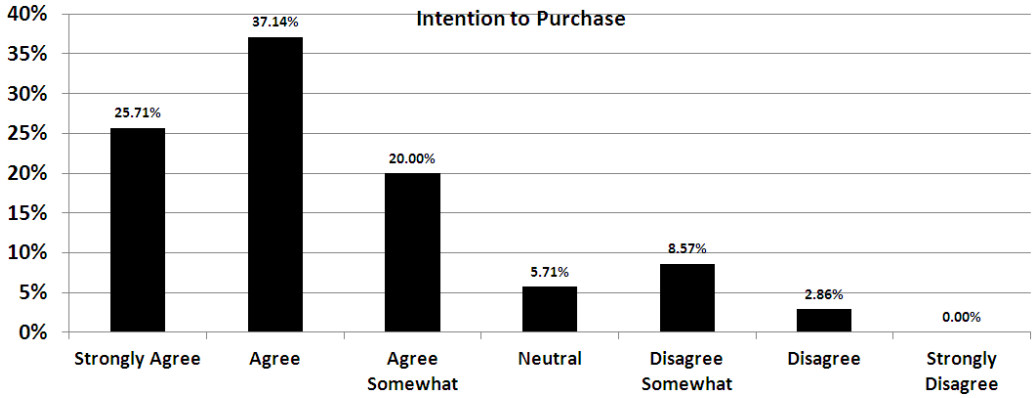
Regarding the intention to save effort, some participants claimed that even though using the system may help them with their choice, it would require more effort — but they are willing to use it anyway, because they believe it can help them to make a better choice than that they would do without support. One participant stated: "*if I wanted to buy an expensive, delicate, unique, etc. product, I would use the system because it is more sophisticated than I would choose myself.*"

Measurement	Average	Standard Deviation
Perceived usefulness	6.09	0.89
Intention to purchase	5.57	1.33
Intention to return	6.23	1.06
Intention to save effort	5.94	1.19
Enjoyment	5.83	0.92
Satisfaction	6.06	0.84

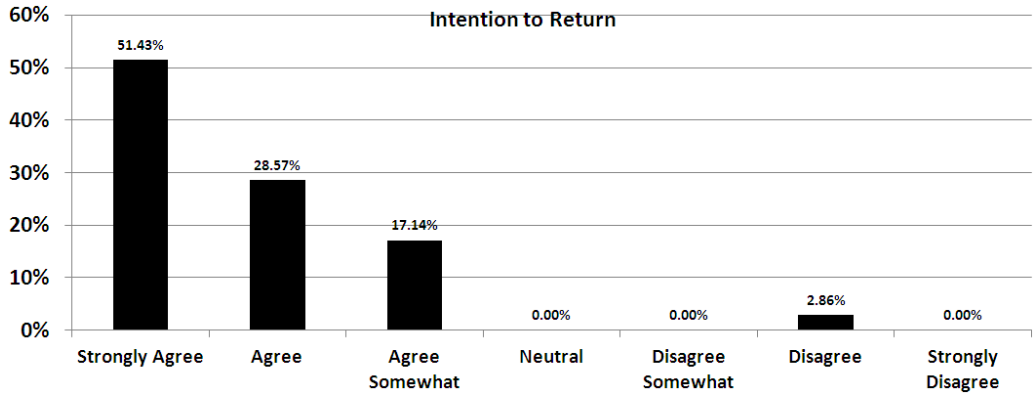
Table 10.7: Approach evaluation measurements.



10.7(a): Perceived usefulness.

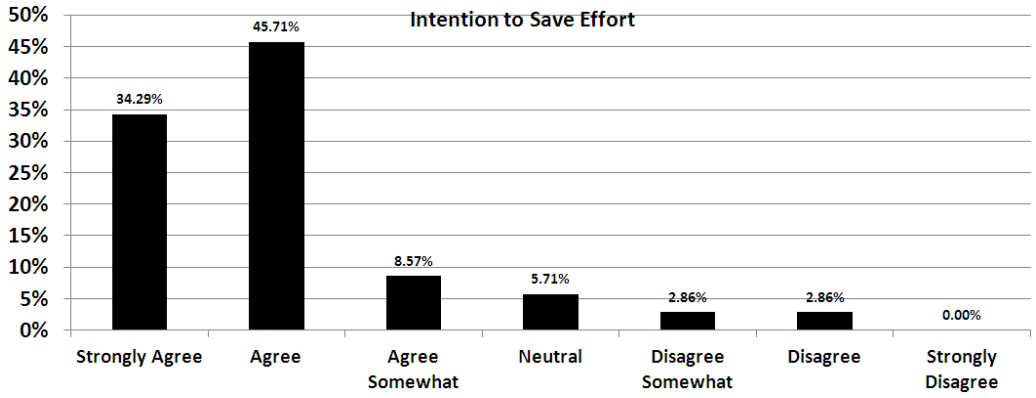


10.7(b): Intention to Purchase.

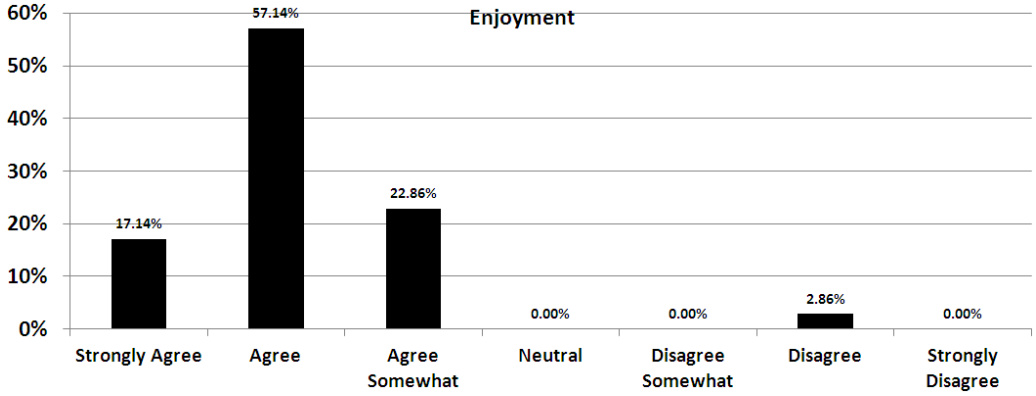


10.7(c): Intention to Return.

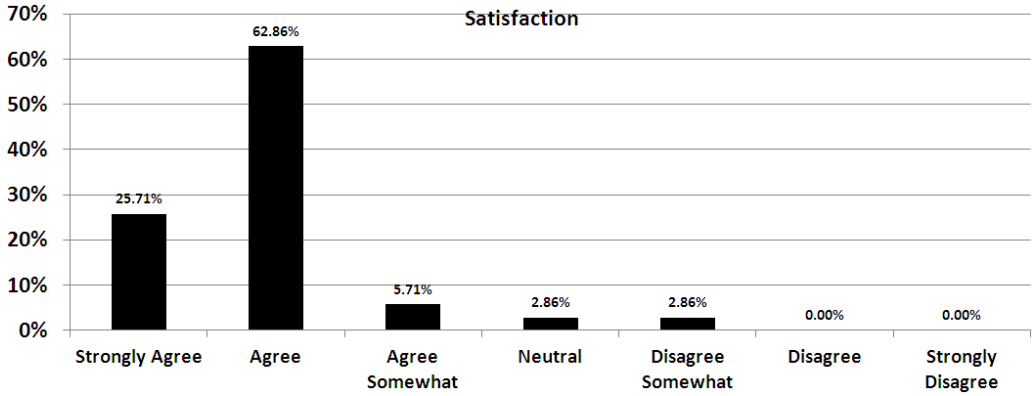
Figure 10.7: Approach Evaluation (I).



10.8(a): Intention to save effort in next visit.



10.8(b): Enjoyment.



10.8(c): Satisfaction.

Figure 10.8: Approach Evaluation (II).

10.3 Threats to Validity

In this section, we discuss threats to the validity of this study. We have identified three possible threats, which are listed below.

Construct Validity. In order to evaluate how different explanation approaches impact the trust in choice, participants had to answer a question related to this measure for each of the investigated approaches. However, we identified that participants may have answered the question related to trust from two perspectives: (i) trust in the decision maker (for choice evaluation and explanation impact); (ii) trust in the explanation given (for explanation comparison). Therefore, the trust in choice measure analysed for explanation comparison may be not related to the combination of the choice and explanation, as it was intended. Moreover, we observed that it was hard for the participants to isolate the three explanations to evaluate how confident they were in the decision when receiving each explanation. This problem can be tackled by performing another study using a between-subjects design, but it has the disadvantage that different participants may have different perceptions for the rates.

Internal Validity. Most of the participants had previous knowledge about the available mobile phones. As a consequence, when they answered questions to evaluate the choice quality, trust in choice and decision confidence, they may have used this knowledge. Therefore, the impact of the explanation can be lower than in the situation in which participants do not know options of the available set, such as when people search for hotels.

External Validity. The fact that our user study involved participants (i) of single geographic location (i.e. Brazil) and (ii) that are volunteers is a threat to the generalisation of the results of our study.

10.4 Final Remarks

In this chapter, we presented a user study, which was performed to evaluate different aspects of our approach: the preference language, the decision making technique and the explanation generation approach. The study consisted of allowing participants to interact with an application in which they had to (i) provide their preferences according to our language; (ii) analyse a choice made based on their preferences; (iii) analyse different explanations that justify this choice. Each of these steps also involves answering questions to evaluate our approach.

The results of our study show that our preference language is expressive enough for users to provide their preferences, but for some of them this process demands too much effort. Even though our approach does not include an interface for specifying preferences, participants considered the one used in the study easy to use. Our decision making technique was also positively evaluated, achieving high rates for choice quality, trust in choice and decision confidence. Moreover, explanations generated with our technique significantly increase the decision confidence. Our explanation technique was also compared with two main existing approaches, and results showed that our technique and also Klein and Shortliffe's approach are better than Labreuche's approach with respect to transparency and trust in choice. Although the difference between our explanation approach and that proposed by Klein and Shortliffe is not statistically significant — they are indeed considered equally good for many participants — the former outperforms the latter in some situations, at the cost of providing worse explanations in others.

Our study allowed us to identify shortcomings in the three aspects of our approach (language, decision making and explanation), which will help us to further improve our work. Finally, other studies can be conducted to refine the results of this study, tackling identified threats to its validity.

11
Conclusion

Making decisions over a possibly huge set of options is part of many of the tasks that humans face in their everyday lives. Such decisions are not only time-consuming, but also require cognitive effort demanded by humans, as choosing an option from an available set of options often requires resolution of trade-offs. Moreover, as nowadays the number of options available to users is massive, analysing all available options goes beyond their cognitive limitations, making them often unsatisfied with their choices.

Approaches to representing and reasoning about preferences, as well as explanation approaches to justify decisions made by computer systems, have been proposed in order to support and automate decision making, and this is the context of this thesis. We proposed a new preference metamodel, which allows the representation of high-level preferences. This metamodel was developed based on a study involving almost 200 participants, whose goal was to understand how humans express preferences and the expressions they use. With the aim of making decisions in a similar way to humans, and based on preferences they explicitly provide, we also presented a decision making technique, which receives as input preferences in a language that is based on our metamodel and chooses an option from a set available, taking into account user-centric principles.

In order for users to understand why an option was chosen and to trust the decision, we proposed an explanation generation technique that uses models built by our decision making technique to justify choices. Our explanation approach is based on guidelines and patterns that we derived from a study to justify choices, which involved 100 participants. Finally, we presented the results of a user study, performed to evaluate the different aspects of our approach. This evaluation shows that (i) our preference language is adequate for users to express their preferences; (ii) our decision making technique makes choices that users consider as having good quality; and (iii) the provided explanations allows users to understand why the choice was made and improves decision confidence. By making a side-by-side comparison of our explanation technique with the two similar existing approaches, we were able to determine that ours is significantly better than one of them with respect to transparency and trust in choice. Although our explanation technique

has no significant difference with respect to the second compared approach, the comparison showed the particular scenarios in which our explanation technique needs to be improved. Therefore, limitations of our approach were also identified with this user study, not only associated with explanations. For example, we identified cases where users provide dyadic preferences, but also indicate that values referred to by those preferences are preferred to all other possible values for that attribute, and that explanations must mention important attributes associated with unsatisfied preferences.

Next, we detail the contributions of this thesis and discuss future work.

11.1 Contributions

As the result of the work presented in this thesis, many contributions can be enumerated, which are detailed next. Some of them serve as a basis or to evaluate our three main contributions: the preference metamodel, the decision making technique, and the explanation technique.

Study of How Humans Express Preferences. Our first study, presented in Chapter 2, provided a deeper understanding of how humans express their preferences about a domain. We analysed and discussed different aspects of preferences specified by the study participants, such as how useful the provided preferences are for making a choice on their behalf, and how preferences change after the participants face a concrete decision making situation. We also derived from this study common expressions that humans use to state their preferences.

High-level Preference Metamodel. Considering the results of our study of how humans express preferences, we proposed a preference metamodel (Chapter 3) — which also includes an ontology metamodel to represent application areas and a metamodel to represent propositional formulae — that allows representation of different types of preferences, such as constraints, goals and qualifying statements, which use expressive speech acts to indicate preference. Initial versions of this metamodel were published elsewhere (Nunes et al. 2010a, Nunes et al. 2010b). The metamodel is represented in UML, but is also formally specified using the Z notation.

Systematic Review of Reasoning about Preferences. As many different areas of computer science investigate preferences, we provided a systematic review of reasoning about preference approaches in Chapter 5, including work in the context of decision theory, artificial intelligence, constraint programming, databases

and semantic web. Each piece of work was presented following an evaluation framework, which facilitates their comparison, and we further analysed them, discussing their positive and negative aspects.

User-centric Preference-based Decision Making Technique. We proposed an automated decision making technique (Nunes et al. 2012a, Nunes et al. 2012b), presented in Chapter 6, that uses preferences expressed by users in a high-level language, to resolve trade-offs based on priorities provided by users combined with user-centric principles. Our technique provides the novelty of exploiting different natural language expressions and user-centric principles in automated decision making. These two particularities of our technique consist of two ways of significantly improving research in this area: while expressive speech acts and other expressions give valuable information that can be used to generate low-level preference representations, such as utility functions; our (and possibly others) user-centric principles can be used to reduce the amount of preferences obtained from users, as they can predict how users would resolve trade-offs. Moreover, these principles of human decision making explain situations in which a decision made by a human is "irrational" according to classical decision theory, and by taking these principles into account, automated systems can make decisions that are more acceptable to users.

Explanation Guidelines and Patterns. We performed a study (Nunes et al. 2012c) that allowed us to investigate how humans justify their decisions (Chapter 8), by arguing why they choose a particular option from the set of those available, and why the remaining options are rejected. With the data collected from this study, we have derived *guidelines* and *patterns* of explanations to be given to users to justify decisions made by the system.

Explanation Generation Technique. We presented a means of generating explanations for users to justify choices made by our decision making technique (Nunes et al. 2013) (Chapter 9). This is based on proposed guidelines and patterns, and provides a means of identifying parameters of explanation templates, which are part of the patterns. The technique not only identifies these parameters, but also provides an algorithm to choose which explanation should be used in different cases.

Evaluation. In order to evaluate different aspects of our approach, we performed a *user study*, presented in Chapter 10, in which participants had to specify their preferences, receive a choice (or recommendation), receive an explanation for that choice, and finally receive alternative explanations for it. With this study,

we evaluated our preference language, our decision making technique and our explanation generation technique, compared to existing approaches. The results of this study showed that our approach performs well in these three different aspects, but also identified its limitations.

11.2 Future Work

The contributions of this thesis advance research work on preferences, with the proposal of a preference metamodel, a novel decision making technique and an explanation generation approach. However, our work has limitations, leading to ongoing and future work, some aspects of which are discussed as follows.

Replication Studies. Each of our studies was performed in the context of one domain only: laptops (study of how humans express preferences); hotels (study of explanations to justify choice); and mobile phones (user study performed to evaluate our approach). In order to confirm the results obtained from these studies, it is important to replicate them in other domains and with other subjects. Moreover, the recommendation of other domain specialists could be taken into account in our first study.

Preference Consistency. One of the assumptions of our decision making technique is that the provided preferences are *consistent*. This is unlikely to happen, as confirmed by our study of how humans express preferences in which none of the preferences provided was inconsistent, and there was only one case in which preferences were inconsistent in our user study. However, when inconsistency *does* arise, wrong decisions can be made, with possibly inadequate explanations, which compromise user acceptance. Therefore, it is important to elaborate an approach that is able to check whether a set of provided preferences is consistent.

Preference Elicitation. Although our preference metamodel allows the derivation of a language in which users can express preferences in a way close to natural language, the activity of providing preferences requires a significant effort, as our user study revealed. Therefore, it is important to consider an approach that is able to implicitly capture an initial set of preferences, so that users can refine this set later. The advantage of using a high-level language is that users are able to understand the elicited preferences, and possibly make changes. Furthermore, as identified in our study of how humans express preferences, other kinds of support could be provided, e.g. reminding users of (generally important) attributes that were not mentioned.

Preferences not Covered by the Decision Making Technique. The language in which preferences that are the input of our decision making technique are expressed corresponds to a restricted version of our preference metamodel. As our metamodel is associated with expressions and terms that humans use to state preferences, we still constrain users while providing their preferences. Therefore, the investigation of how these limitations can be addressed is important. While some of these expressions that were not addressed should be handled during the decision making process, others may be associated with the translation or interpretation of terms used as proxy for others. For example, if the user wants to *maximise mobility* (of a laptop), this can be translated into the minimisation of laptop dimensions and weight. Therefore, an user interface language can be used as an abstraction layer on top of our decision making technique, in these situations.

Decision Making Technique Variability. The empirical evaluation of our technique showed that it is able to make a choice on behalf of the users as good as that made by a human domain expert. However, the conducted user study indicated that, even though our technique indicates good recommendations and helps users to make choices, it is not *always* able to make the right choice on their behalf, preventing them from delegating tasks to a system, which is our ultimate goal. As our decision making technique has variable parts (e.g. modifier scale, functions and weights), which were instantiated in this thesis after running the technique with different alternatives, it is future work to improve results by exploring this variability — either experimentally or by proposing individual-specific approaches — and investigating other user-centric principles to be adopted. Machine learning techniques can be adopted to instantiate these variables, as well as using fuzzy logics in the modifier scale to improve the interpretation of expressive speech acts and rates.

Inter-agent Decision Making. In this thesis, we have considered decision making in the context of a single agent. However, decisions can also be made in situations in which: (i) a single decision maker takes into account preferences of other agents, e.g. if a decision maker wants to decide at which hotel to stay, and one of her preferences is to stay at the same hotel as her colleague, then the preferences of her colleague should be also taken into account; and (ii) multiple decision makers must make a joint decision. Our goal is to extend our approach in order to address these situations, and this involves explicitly representing third-party preferences, reasoning about them and explaining decisions made in these scenarios.

Mixed Initiative Decision Making. Our approach receives as input a set of preferences and a set of available options and give as output a choice with an associated explanation. The automation of decision making is helpful as this task requires humans to demand high amounts of cognitive effort. However, as our evaluation showed, our technique is not always able to make an adequate choice. Besides improving our technique, better results can be achieved by adopting a mixed initiative approach: an initial set of preferences is given, which can be later refined according to the choice presented and explanations given. Moreover, this initial set of preferences can be obtained by mining preferences from historical data.

In summary, the work presented in this thesis advances work on automated decision making in three main directions: preference representation, preference reasoning and explanation generation. Clearly there is still much to do in order to make concrete our vision of having agents able to make decisions on behalf of users in a multi-agent scenario, but our work consists of a significant step towards this vision.

Bibliography

[Adomavicius and Tuzhilin 2005] ADOMAVICIUS, G.; TUZHILIN, A.. **Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions**. Knowledge and Data Engineering, IEEE Transactions on, 17(6):734–749, june 2005.

[Agrawal and Wimmers 2000] AGRAWAL, R.; WIMMERS, E. L.. **A framework for expressing and combining preferences**. In: PROCEEDINGS OF THE 2000 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, p. 297–306, New York, NY, USA, 2000. ACM.

[Amgoud et al. 2008] AMGOUD, L.; DIMOPOULOS, Y. ; MORAITIS, P.. **Making decisions through preference-based argumentation**. In: KR, p. 113–123, 2008.

[Austin 1975] AUSTIN, J. L.. **How to Do Things with Words: Second Edition (William James Lectures)**. Harvard University Press, 1975.

[Ayres and Furtado 2007] AYRES, L.; FURTADO, V.. **Owlpref: Uma representação declarativa de preferências para web semântica**. In: ANAIS DO XXVII CONGRESSO DA SBC, p. 1411–1419, Rio de Janeiro, Brazil, July 2007. SBC.

[Back 2006] BACH, K.. **Speech acts and pragmatics**. In: THE BLACKWELL GUIDE TO THE PHILOSOPHY OF LANGUAGE, chapter 8, p. 147–167. Wiley-Blackwell, 2006.

[Basili et al. 1986] BASILI, V. R.; SELBY, R. W. ; HUTCHENS, D. H.. **Experimentation in software engineering**. IEEE Trans. Softw. Eng., 12(7):733–743, 1986.

[Basili and Rombach 1988] BASILI, V. R.; ROMBACH, H. D.. **The tame project: towards improvement-oriented software environments**. Software Engineering, IEEE Transactions on, 14(6):758–773, 1988.

[Bistarelli et al. 1997] BISTARELLI, S.; MONTANARI, U. ; ROSSI, F.. **Semiring-based constraint satisfaction and optimization**. J. ACM, 44:201–236, March 1997.

[Bistarelli et al. 2010] BISTARELLI, S.; PINI, M. S.; ROSSI, F. ; VENABLE, K. B.. **From soft constraints to bipolar preferences: modelling framework and solving issues**. J. Exp. Theor. Artif. Intell., 22:135–158, June 2010.

[Bohanec and Rajkovič 1993] BOHANEC, M.; RAJKOVIČ, V.. **Knowledge-based explanation in multi-attribute decision making**. In: Nagel, S., editor, COMPUTER AIDED DECISION ANALYSIS: THEORY AND APPLICATIONS, p. 189–204. Quorum Books, 1993.

[Borzsonyi et al. 2001] BÖRZSÖNYI, S.; KOSSMANN, D. ; STOCKER, K.. **The skyline operator**. In: PROCEEDINGS OF THE 17TH INTERNATIONAL CONFERENCE ON DATA ENGINEERING, p. 421–430, Washington, DC, USA, 2001. IEEE Computer Society.

[Boutilier et al. 2001] BOUTILIER, C.; BACCHUS, F. ; BRAFMAN, R. I.. **Ucp-networks: A directed graphical representation of conditional utilities**. In: PROCEEDINGS OF THE 17TH CONFERENCE IN UNCERTAINTY IN ARTIFICIAL INTELLIGENCE, UAI '01, p. 56–64, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[Boutilier et al. 2004] BOUTILIER, C.; BRAFMAN, R. I.; DOMSHLAK, C.; HOOS, H. H. ; POOLE, D.. **Cp-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements**. J. Artif. Int. Res., 21(1):135–191, 2004.

[Brafman et al. 2006] BRAFMAN, R. I.; DOMSHLAK, C. ; SHIMONY, S. E.. **On graphical modeling of preference and importance**. J. Artif. Int. Res., 25:389–424, March 2006.

[Braun and Clarke 2006] BRAUN, V.; CLARKE, V.. **Using thematic analysis in psychology**. Qualitative Research in Psychology, 3(2):77–101, 2006.

[Brafman and Domshlak 2008] BRAFMAN, R. I.; DOMSHLAK, C.. **Graphically structured value-function compilation**. Artif. Intell., 172:325–349, February 2008.

[Buchanan and Shortliffe 1984] BUCHANAN, B. G.; SHORTLIFFE, E. H.. **Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley series in artificial intelligence)**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[Carenini and Moore 2006] CARENINI, G.; MOORE, J. D.. **Generating and evaluating evaluative arguments**. Artif. Intell., 170:925–952, August 2006.

[Chen and Pu 2010] CHEN, L.; PU, P.. **User evaluation framework of recommender systems**. In: PROCEEDINGS OF 2010 WORKSHOP ON SOCIAL RECOMMENDER SYSTEMS (SRS'10) AT THE ACM INTERNATIONAL CONFERENCE ON INTELLIGENT USER INTERFACES (IUI'10), New York, NY, USA, 2010. ACM.

[Chomicki 2003] CHOMICKI, J.. **Preference formulas in relational queries**. ACM Trans. Database Syst., 28:427–466, December 2003.

[Chomicki et al. 2005] CHOMICKI, J.; GODFREY, P.; GRYZ, J. ; LIANG, D.. **Skyline with presorting: Theory and optimizations**. In: Klopotek, M.; Wierzchon, S. ; Trojanowski, K., editors, INTELLIGENT INFORMATION PROCESSING AND WEB MINING, volumen 31 de **Advances in Intelligent and Soft Computing**, p. 595–604. Springer Berlin / Heidelberg, 2005.

[Cormen et al. 2001] CORMEN, T. H.; STEIN, C.; RIVEST, R. L. ; LEISERSON, C. E.. **Introduction to Algorithms**. McGraw-Hill Higher Education, 2nd edition, 2001.

[Domshlak et al. 2003] DOMSHLAK, C.; ROSSI, F.; VENABLE, K. B. ; WALSH, T.. **Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques**. In: PROCEEDINGS OF THE 18TH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, p. 215–220, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[Domshlak and Joachims 2007] DOMSHLAK, C.; JOACHIMS, T.. **Efficient and non-parametric reasoning over user preferences**. User Modeling and User-Adapted Interaction, 17(1-2):41–69, 2007.

[Domshlak 2008] DOMSHLAK, C.. **A snapshot on reasoning with qualitative preference statements in ai**. In: Maier, G.; Rammerstorfer, F. G.; Salençon, J.; Schrefler, B.; Serafini, P.; Riccia, G.; Dubois, D.; Kruse, R. ; Lenz, H.-J., editors, PREFERENCES AND SIMILARITIES, volumen 504 de **CISM International Centre for Mechanical Sciences**, p. 265–282. Springer Vienna, 2008.

[Dung 1995] DUNG, P. M.. **On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games**. Artif. Intell., 77:321–357, September 1995.

[Dyer 2005] DYER, J. S.. **Maut: Multiattribute utility theory**. In: Jose Figueira, Salvatore Greco, M. E., editor, MULTIPLE CRITERIA

DECISION ANALYSIS: STATE OF THE ART SURVEYS, chapter 7, p. 265–295. Springer Science + Business Media, Inc., Boston, 2005.

[El-Beltagy et al. 1999] EL-BELTAGY, S. R.; RAFEA, A. A. ; SAMEH, A. H.. **An agent based approach to expert system explanation**. In: PROCEEDINGS OF THE TWELFTH INTERNATIONAL FLORIDA ARTIFICIAL INTELLIGENCE RESEARCH SOCIETY CONFERENCE, p. 153–159. AAAI Press, 1999.

[Engel and Wellman 2008] ENGEL, Y.; WELLMAN, M. P.. **Cui networks: a graphical representation for conditional utility independence**. J. Artif. Int. Res., 31:83–112, January 2008.

[Friedrich 2004] FRIEDRICH, G.. **Elimination of spurious explanations**. In: PROCEEDINGS OF THE 16TH EUREOPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE, ECAI'2004, p. 813–817. IOS Press, 2004.

[Gelain et al. 2010] GELAIN, M.; PINI, M. S.; ROSSI, F.; VENABLE, K. B. ; WILSON, N.. **Interval-valued soft constraint problems**. Annals of Mathematics and Artificial Intelligence, 58:261–298, April 2010.

[Glaser 1992] GLASER, B. G.. **Emergence vs. Forcing: Basics of Grounded Theory Analysis**. Sociology Press, Mill Valley, USA, 1992.

[Hansson 1990] HANSSON, S. O.. **Defining "good" and "bad" in terms of "better"**. Notre Dame Journal of Formal Logic, 31(1):136–149, 1990.

[Hansson 1996] HANSSON, S. O.. **What is ceteris paribus preference?** Journal of Philosophical Logic, 425:307–332, 1996.

[Hansson 2001] HANSSON, S. O.. **The Structure of Values and Norms (Cambridge Studies in Probability, Induction and Decision Theory)**. Cambridge University Press, 2001.

[Holland and Kießling 2004] HOLLAND, S.; KIESSLING, W.. **Situated preferences and preference repositories for personalized database applications**. In: Atzeni, P.; Chu, W.; Lu, H.; Zhou, S. ; Ling, T. W., editors, CONCEPTUAL MODELING (ER 2004), volumen 3288 de **Lecture Notes in Computer Science**, p. 511–523. Springer Berlin / Heidelberg, 2004.

[Jensen 2001] JENSEN, F. V.. **Bayesian Networks and Decision Graphs**. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[Junker 2004] JUNKER, U.. **Quickxplain: Preferred explanations and relaxations for over-constrained problems**. In: PROCEEDINGS OF THE NINETEENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, SIXTEENTH CONFERENCE ON INNOVATIVE APPLICATIONS OF ARTIFICIAL INTELLIGENCE (AAAI), p. 167–172, San Jose, California, USA, 2004. AAAI Press / The MIT Press.

[Junker 2008] JUNKER, U.. **Recent advances in constraints**. chapter Preference-Based Problem Solving for Constraint Programming, p. 109–126. Springer-Verlag, Berlin, Heidelberg, 2008.

[Keeney 1944] KEENEY, R. L.. **Value-focused thinking – A Path to Creative Decisionmaking**. Harvard University Press, London, England, 1944.

[Keeney and Raiffa 1976] KEENEY, R. L.; RAIFFA, H.. **Decisions with Multiple Objectives: Preferences and Value Tradeoffs**. Wiley series in probability and mathematical statistics. John Wiley & Sons, Inc, New York, 1976.

[Kießling 2002] KIESSLING, W.. **Foundations of preferences in database systems**. In: PROCEEDINGS OF THE 28TH INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB '02, p. 311–322. VLDB Endowment, 2002.

[Klein and Shortliffe 1994] KLEIN, D. A.; SHORTLIFFE, E. H.. **A framework for explaining decision-theoretic advice**. Artif. Intell., 67:201–243, June 1994.

[Koutrika and Ioannidis 2006] KOUTRIKA, G.; IOANNIDIS, Y.. **Personalization of queries based on user preferences**. In: Bosi, G.; Brafman, R. I.; Chomicki, J. ; Kießling, W., editors, PREFERENCES: SPECIFICATION, INFERENCE, APPLICATIONS, número 04271 em Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[Labreuche 2011] LABREUCHE, C.. **A general framework for explaining the results of a multi-attribute preference model**. Artif. Intell., 175:1410–1448, May 2011.

[Lacave and Diez 2004] LACAVE, C.; DIEZ, F. J.. **A review of explanation methods for heuristic expert systems**. Knowl. Eng. Rev., 19:133–146, June 2004.

[Lichtenstein and Slovic 2006] LICHTENSTEIN, S.; SLOVIC, P.. **The construction of preference**. Cambridge University Press, New York, USA, 2006.

[McGeachie and Doyle 2008] MCGEACHIE, M.; DOYLE, J.. **Utility functions for ceteris paribus preferences**. Computational Intelligence, 20(2):158–217, 2004.

[McNee et al. 2006] MCNEE, S. M.; RIEDL, J. ; KONSTAN, J. A.. **Being accurate is not enough: how accuracy metrics have hurt recommender systems**. In: CHI '06 EXTENDED ABSTRACTS ON HUMAN FACTORS IN COMPUTING SYSTEMS, CHI EA '06, p. 1097–1101, New York, NY, USA, 2006. ACM.

[McSherry 2005] MCSHERRY, D.. **Explanation in recommender systems**. Artif. Intell. Rev., 24:179–197, October 2005.

[Meseguer et al. 2006] MESEGUER, P.; ROSSI, F. ; SCHIEX, T.. **Soft constraints**. In: Rossi, F.; van Beek, P. ; Walsh, T., editors, HANDBOOK OF CONSTRAINT PROGRAMMING, p. 281–328. Elsevier, 2006.

[Modgil 2009] MODGIL, S.. **Reasoning about preferences in argumentation frameworks**. Artif. Intell., 173:901–934, June 2009.

[Nakatsu 2006] NAKATSU, R. T.. **Explanatory power of intelligent systems**. In: Gupta, J. N. D.; Forgionne, G. A. ; Mora T., M., editors, INTELLIGENT DECISION-MAKING SUPPORT SYSTEMS, Decision Engineering, p. 123–143. Springer London, 2006.

[Nunes et al. 2010a] NUNES, I.; BARBOSA, S. ; LUCENA, C.. **An end-user domain-specific model to drive dynamic user agents adaptations**. In: PROCEEDINGS OF THE 22TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING (SEKE 2010), p. 509–514, San Francisco, USA, 7 2010. KSI.

[Nunes et al. 2010b] NUNES, I.; BARBOSA, S. D. J. ; DE LUCENA, C. J. P.. **Increasing users' trust on personal assistance software using a domain-neutral high-level user model**. In: PROCEEDINGS OF THE 4TH INTERNATIONAL CONFERENCE ON LEVERAGING APPLICATIONS OF FORMAL METHODS, VERIFICATION, AND VALIDATION - VOLUME PART I, ISoLA'10, p. 473–487, Berlin, Heidelberg, 2010. Springer-Verlag.

[Nunes et al. 2012a] NUNES, I.; MILES, S.; LUCK, M. ; LUCENA, C.. **User-centric preference-based decision making (extended abstract)**. In: ELEVENTH INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS'12), 2012.

[Nunes et al. 2012b] NUNES, I.; MILES, S.; LUCK, M. ; LUCENA, C.. **User-centric principles in automated decision making**. In: 21ST BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE (SBIA 2012), 2012.

[Nunes et al. 2012c] NUNES, I.; MILES, S.; LUCK, M. ; LUCENA, C.. **Investigating explanations to justify choice**. In: J. Masthoff et al., editor, 20TH INTERNATIONAL CONFERENCE ON USER MODELING, ADAPTATION AND PERSONALIZATION (UMAP 2012), volumen 7379 de **LNCS**, p. 212–224, Montreal, Canada, July 2012. Springer-Verlag.

[Nunes et al. 2013] NUNES, I.; MILES, S.; LUCK, M.; BARBOSA, S. ; LUCENA, C.. **A user explanation technique to justify choices**. In: 18TH INTERNATIONAL CONFERENCE ON INTELLIGENT USER INTERFACES (IUI 2013), 2013. Submitted.

[O'Sullivan et al. 2007] O'SULLIVAN, B.; PAPADOPOULOS, A.; FALTINGS, B. ; PU, P.. **Representative explanations for over-constrained problems**. In: PROCEEDINGS OF THE 22ND NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE - VOLUME 1, p. 323–328. AAAI Press, 2007.

[Papadimitriou et al. 2011] PAPADIMITRIOU, A.; SYMEONIDIS, P. ; MANOLOPOULOS, Y.. **A generalized taxonomy of explanations styles for traditional and social recommender systems**. Data Mining and Knowledge Discovery, p. 1–29, mar 2011.

[Patel-Schneider et al. 2004] PATEL-SCHNEIDER, P. F.; HAYES, P. ; HORROCKS, I.. **OWL web ontology language semantics and abstract syntax section**. Technical report, W3C, 2004.

[Payne et al. 1988] PAYNE, J. W.; BETTMAN, J. R. ; JOHNSON, E. J.. **Adaptive strategy selection in decision making**. Journal of Experimental Psychology: Learning, Memory, and Cognition, 14(3):534–552, 1988.

[Petticrew and Roberts 2006] PETTICREW, M.; ROBERTS, H.. **Systematic Reviews in the Social Sciences**. Blackwell Publishing, 2006.

[Pu and Chen 2007] PU, P.; CHEN, L.. **Trust-inspiring explanation interfaces for recommender systems**. Knowledge-Based Systems, 20:542–556, 2007.

[Rahwan and Simari 2009] RAHWAN, I.; SIMARI, G.. **Argumentation in Artificial Intelligence**. Springer, 2009.

[Said et al. 2009] SAID, A. F.; RAFEA, A.; EL-BELTAGY, S. R. ; HASSAN, H.. **Automatic generation of explanation for expert systems implemented with different knowledge representations**. WTOS, 8:55–64, January 2009.

[Schiaffino and Amandi 2004] SCHIAFFINO, S.; AMANDI, A.. **User - interface agent interaction: personalization issues**. Int. J. Hum.-Comput. Stud., 60(1):129–148, 2004.

[Schwartz 2005] SCHWARTZ, B.. **The paradox of choice: Why more is less**. Harper Perennial, 2005.

[Schubert et al. 2010] SCHUBERT, M.; FELFERNIG, A. ; MANDL, M.. **Fastxplain: conflict detection for constraint-based recommendation problems**. In: PROCEEDINGS OF THE 23RD INTERNATIONAL CONFERENCE ON INDUSTRIAL ENGINEERING AND OTHER APPLICATIONS OF APPLIED INTELLIGENT SYSTEMS - VOLUME PART I, IEA/AIE'10, p. 621–630, Berlin, Heidelberg, 2010. Springer-Verlag.

[Searle 1969] SEARLE, J.. **Speech acts : an essay in the philosophy of language**. Cambridge University Press, 1969.

[Shafir et al. 1998] SHAFIR, E.; SIMONSON, I. ; TVERSKY, A.. **Reason-based choice**. In: PREFERENCE, BELIEF AND SIMILARITY, p. 937–962. MIT Press, 1998.

[Shankar and Musen 1999] SHANKAR, R.; MUSEN, M.. **Justification of automated decision-making: medical explanations as medical arguments**. Proceedings of the AMIA Symposium, p. 395–399, 1999.

[Siberski et al. 2006] SIBERSKI, W.; PAN, J. Z. ; THADEN, U.. **Querying the semantic web with preferences**. In: Cruz, I.; Decker, S.; Allemang, D.; Preist, C.; Schwabe, D.; Mika, P.; Uschold, M. ; Aroyo, L., editors, THE SEMANTIC WEB - ISWC 2006, volumen 4273 de **Lecture Notes in Computer Science**, p. 612–624. Springer Berlin / Heidelberg, 2006.

[Simon 1955] SIMON, H. A.. **A behavioral model of rational choice**. The Quarterly Journal of Economics, 69(1):99–118, 1955.

[Simonson and Tversky 1992] SIMONSON, I.; TVERSKY, A.. **Choice in context: Tradeoff contrast and extremeness aversion**. Journal of Marketing Research, 29(3):281–295, 1992.

[Tapucu et al. 2008] TAPUCU, D.; CAN, O.; BURSA, O. ; UNALIR, M. O.. **Metamodeling approach to preference management in the semantic**

web. In: 4TH MULTIDISCIPLINARY WORKSHOP ON ADVANCES IN PREFERENCE HANDLING (M-PREF 2008), p. 116–123, Chicago, Illinois, USA, July 2008. IAAA.

[Tintarev and Masthoff 2007] TINTAREV, N.; MASTHOFF, J.. **A survey of explanations in recommender systems**. In: PROCEEDINGS OF THE 2007 IEEE 23RD INTERNATIONAL CONFERENCE ON DATA ENGINEERING WORKSHOP, p. 801–810, Washington, DC, USA, 2007. IEEE Computer Society.

[Tintarev and Masthoff 2011] TINTAREV, N.; MASTHOFF, J.. **Designing and evaluating explanations for recommender systems**. In: Ricci, F.; Rokach, L.; Shapira, B. ; Kantor, P. B., editors, RECOMMENDER SYSTEMS HANDBOOK, p. 479–510. Springer US, 2011.

[Toulmin 2003] TOULMIN, S. E.. **The Uses of Argument**. Cambridge University Press, July 2003.

[Tversky 1972] TVERSKY, A.. **Elimination by aspects: a theory of choice**. Psychological Review, 79(4):281–299, 1972.

[Tversky 1996] TVERSKY, A.. **Contrasting rational and psychological principles of choice**. In: Zeckhauser, R. J.; Keeney, R. L. ; Sebenius, J. K., editors, WISE CHOICES : GAMES, DECISIONS, AND NEGOTIATIONS, p. 5–21. Harvard Business School Press, 1996.

[Vapnik 1998] VAPNIK, V. N.. **Statistical Learning Theory**. Wiley-Interscience, 1998.

[Wierenga 2008] WIERENGA, B.. **Handbook of Marketing Decision Models**. Springer Publishing Company, Incorporated, 1st edition, 2008.

[Wordsworth 1992] WORDSWORTH, J. B.. **Software Development with Z: A Practical Approach to Formal Methods in Software Engineering**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1992.

[Ye and Johnson 1995] YE, L. R.; JOHNSON, P. E.. **The impact of explanation facilities on user acceptance of expert systems advice**. MIS Q., 19:157–172, June 1995.

[Zanker and Ninaus 2010] ZANKER, M.; NINAUS, D.. **Knowledgeable explanations for recommender systems**. In: WEB INTELLIGENCE AND INTELLIGENT AGENT TECHNOLOGY (WI-IAT), 2010 IEEE/WIC/ACM

INTERNATIONAL CONFERENCE ON, volumen 1, p. 657–660, 31 2010-sept. 3 2010.

[von Neumann and Morgenstern 1944] NEUMANN, J. V.; MORGENSTERN, O.. **Theory of Games and Economic Behavior**. Princeton University Press, 1944.

A
Questionnaire: Preference Expression

This appendix presents the questionnaire used in our study. The participant had the option of answering it either in English or in Portuguese. If participants tried to go back in the questionnaire, they were notified that changes would not be stored.

A.1
Introduction: Survey about User Preferences

The purpose of this survey is to collect data that helps in understanding the user preferences expression. The survey is completely anonymous and all information collected will be used solely for statistical analysis within the context of this study. The survey has four steps and the estimated time for answering the survey is around 20 minutes.

Please, click on the image below to start the survey in English:

A.2
Part I: User Data

- Age: a positive integer;
- Gender: a value from {Male, Female};
- Country: a value from a provided list of countries;
- City: a string;
- Working/Studying Field: a string;
- How many laptops have you already had (including current ones)? a value from {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10+};
- If you had(have) at least one laptop, did you yourself choose it(them)? a value from {All of them, Most of Them, Some of them, A few, No one};
- How do you rate your knowledge about which computer features to consider when buying a laptop? a value from {Expert, Advanced, Intermediate, Beginner, No knowledge};

A.3
Part II: Preference Specification

Suppose that you want to **buy a new laptop** and somebody is going to buy it for you. You are going to specify all preferences and restrictions of this person, who will buy the laptop for you with no further communication after the initial specification.

We present below a simple example of a preference specification on the flight domain.

Example

1. I like to minimize the price, I always pay promotional fares.
2. I don't like making connections.
3. I prefer the shortest flying time as possible, as long as I have at least one hour to make connections.
4. Flying time and number of connections are more important to me than the price.

Please, write down below the specification that you would provide to this person so he/she can buy the **laptop** for you.

[text area in which participants write their specification]

A.4
Part III: Options Selection

Now, let's assume that all laptops available for you are the ones listed below. Please, indicate which laptop you would choose. You can rank up to five options (at least one is required).

If you notice, during this selection process, that your previously specified preferences are incomplete, please do not go back. You will have the chance to review your preferences in the next (and last) step.

1. Option 1: {laptop list};
2. Option 2: {laptop list};
3. Option 3: {laptop list};
4. Option 4: {laptop list};
5. Option 5: {laptop list}.

Laptop Catalog

Use the catalog below to choose your laptop options. In the selection boxes above, laptops are identified by their SKU number. In addition, laptop names are also displayed. Instead of selecting a laptop manually, you may also click on the select button of the chosen laptop, which will be selected in the first empty select box.

The following actions can be performed in the catalog:

– *Sort*: laptops can be ordered according to the value selected in the box "Sort by;"

– *Filter*: different filters (price range, brand, ...) can be added or removed, when the filter links are clicked;

– *Show laptop details*: by clicking on the laptop name, a new window is opened with the specification of the selected laptop; and

– *Compare laptops*: you can selected 2 or 3 laptops to be compared. After selecting the laptops, click on the "Compare" button, and a new window will be opened with a comparison table.

Obs. All prices are in American dollars.

Available options were given as shown in Figure A.1.

A.5
Part IV: Preference Specification Review

After choosing the laptops from the previous page, would you have specified your preferences and restrictions in a different way? If so, please make the necessary modifications in your specification. Please, note that you do not know about the available laptop options while making this specification.

Initial Preference Specification

[initial preference specification provided by the participant]

Chosen Laptops

You may click on the name of the laptops to see their details.

[the up to five laptops chosen by the participant]

Reviewed Preference Specification

[text area in which participants write their reviews specification – it is initialised with the initial specification]

Figure A.1: Laptop Options.

B
Z Specification

In this appendix, we present a formal specification of our preference model presented in Chapter 3, using the Z notation (Wordsworth 1992).

B.1
Ontology Metamodel

$$Boolean ::= TRUE \mid FALSE$$

$$[Number, Character, String, Date]$$

$$\begin{array}{|l} \hline EnumerationValue \\ name : String \\ \hline \end{array}$$

$$\begin{array}{|l} \hline Enumeration \\ name : String \\ values : \mathbb{P}\, EnumerationValue \\ \hline \end{array}$$

$$\begin{aligned} PrimitiveType ::= {} & booleanT \mid characterT \mid stringT \\ \mid {} & numericT\langle\!\langle Boolean \times Number \times Number \rangle\!\rangle \\ \mid {} & dateT \mid enumeration\langle\!\langle Enumeration \rangle\!\rangle \end{aligned}$$

B.1.1
Concept and Attributes

$$[Adjective, ScaleValue]$$

$$\begin{array}{|l} \hline Scale \\ name : String \\ values : \mathrm{seq}_1\, ScaleValue \\ \hline \end{array}$$

$$
\begin{array}{|l}
\textit{AttributeT} \\
\hline
name : String \\
scales : \mathbb{P}\, Scale \\
\hline
\end{array}
$$

$$
ConcreteAttribute ::= naturalAttribute\langle\!\langle AttributeT \rangle\!\rangle \mid constructedAttribute\langle\!\langle AttributeT \rangle\!\rangle
$$

$$
\begin{array}{|l}
\textit{ProxyAttribute} \\
\hline
AttributeT \\
contreteAttributes : \mathbb{P}_1\, ConcreteAttribute \\
\hline
\end{array}
$$

$$
Attribute ::= proxyAttribute\langle\!\langle ProxyAttribute \rangle\!\rangle \mid concreteAttribute\langle\!\langle ConcreteAttribute \rangle\!\rangle
$$

$$
\begin{array}{|l}
\textit{Concept} \\
\hline
name : String \\
attributes : \mathbb{P}\, Attribute \\
adjectives : \mathbb{P}\, Adjective \\
\hline
(\forall\, pa : ProxyAttribute \mid proxyAttribute\ pa \in attributes \bullet \\
\quad (\forall\, ca : ConcreteAttribute \mid ca \in pa.contreteAttributes \bullet \\
\quad\quad concreteAttribute\ ca \in attributes)) \\
\hline
\end{array}
$$

$$
\forall\, concept, other : Concept \bullet \\
\quad concept.attributes \cap other.attributes = \emptyset
$$

$$
Type ::= primitive\langle\!\langle PrimitiveType \rangle\!\rangle \mid composite\langle\!\langle Concept \rangle\!\rangle
$$

$$
\begin{array}{|l}
superConcept : Concept \pfun Concept \\
\end{array}
$$

$$
\begin{array}{|l}
conceptAttributes : Concept \pfun \mathbb{P}\, Attribute \\
\hline
\forall\, concept : Concept \bullet \\
\quad (\neg\, (\exists\, super : Concept \bullet \\
\qquad super = superConcept(concept)) \land \\
\qquad (conceptAttributes(concept) = concept.attributes)) \lor \\
\quad ((\exists\, super : Concept \bullet \\
\qquad super = superConcept(concept)) \land \\
\qquad (conceptAttributes(concept) = \\
\qquad\quad concept.attributes \cup \\
\qquad\quad conceptAttributes(superConcept(concept))))
\end{array}
$$

$$
\begin{array}{|l}
attType : Attribute \fun Type
\end{array}
$$

$$
\begin{array}{|l}
validAttRef : \mathrm{seq}\, Attribute \times Attribute \fun Boolean \\
\hline
\forall\, attSeq : \mathrm{seq}\, Attribute \bullet \\
\quad \forall\, att : Attribute \bullet \\
\qquad ((\#attSeq = 0) \land \\
\qquad\quad (validAttRef(attSeq, att) = TRUE)) \lor \\
\qquad ((\#attSeq \neq 0) \land \\
\qquad\quad (\exists_1\, c : Concept \bullet \\
\qquad\qquad att \in conceptAttributes(c) \land \\
\qquad\qquad attType(last\ attSeq) = composite\ c) \land \\
\qquad\quad (validAttRef(front\ attSeq, last\ attSeq) = TRUE))
\end{array}
$$

$$
\begin{array}{|l}
\hline
AttRef \\
attribute : Attribute \\
context : \mathrm{seq}\, Attribute \\
\hline
validAttRef(context, attribute) = TRUE \\
\hline
\end{array}
$$

B.1.2 Instantiation

$$
\begin{aligned}
Literal ::= {} & bool\langle\!\langle Boolean \rangle\!\rangle \mid numeric\langle\!\langle Number \rangle\!\rangle \\
\mid {} & char\langle\!\langle Character \rangle\!\rangle \mid string\langle\!\langle String \rangle\!\rangle \\
\mid {} & date\langle\!\langle Date \rangle\!\rangle \mid enum\langle\!\langle EnumerationValue \rangle\!\rangle
\end{aligned}
$$

ConceptInstance

$$name : String$$
$$concept : Concept$$

$$\forall\, conceptInstance, other : ConceptInstance \bullet$$
$$\quad conceptInstance.name \neq other.name \lor$$
$$\quad conceptInstance = other$$

$$Instance ::= literal\langle\!\langle Literal \rangle\!\rangle \mid conceptInstance\langle\!\langle ConceptInstance \rangle\!\rangle$$

$$validAttType : Attribute \times Instance \rightarrow Boolean$$

$$\forall\, att : Attribute \bullet$$
$$\quad (\forall\, instance : Instance \bullet$$
$$\quad\quad (\forall\, ci : ConceptInstance \mid instance = conceptInstance\ ci \bullet$$
$$\quad\quad\quad attType(att) = composite\ ci.concept) \lor$$
$$\quad\quad (\forall\, lit : Literal \mid instance = literal\ lit \bullet$$
$$\quad\quad\quad (\forall\, b : Boolean \mid lit = bool\ b \bullet$$
$$\quad\quad\quad\quad attType(att) = primitive\ booleanT) \lor$$
$$\quad\quad\quad (\forall\, n : Number \mid lit = numeric\ n \bullet$$
$$\quad\quad\quad\quad (\forall\, b : Boolean \bullet (\forall\, lb, ub : Number \bullet$$
$$\quad\quad\quad\quad\quad attType(att) =$$
$$\quad\quad\quad\quad\quad\quad primitive\ (numericT\ (b, lb, ub))))) \lor$$
$$\quad\quad\quad (\forall\, c : Character \mid lit = char\ c \bullet$$
$$\quad\quad\quad\quad attType(att) = primitive\ characterT) \lor$$
$$\quad\quad\quad (\forall\, s : String \mid lit = string\ s \bullet$$
$$\quad\quad\quad\quad attType(att) = primitive\ stringT) \lor$$
$$\quad\quad\quad (\forall\, d : Date \mid lit = date\ d \bullet$$
$$\quad\quad\quad\quad attType(att) = primitive\ dateT) \lor$$
$$\quad\quad\quad (\forall\, ev : EnumerationValue \mid lit = enum\ ev \bullet$$
$$\quad\quad\quad\quad (\forall\, e : Enumeration \bullet$$
$$\quad\quad\quad\quad\quad attType(att) = primitive\ (enumeration\ e) \land$$
$$\quad\quad\quad\quad\quad ev \in e.values))))$$

ConceptState

$$ConceptInstance$$
$$slots : Attribute \rightarrow Instance$$

$$\text{dom}\ slots = conceptAttributes(concept) \land$$
$$(\forall\ att : Attribute \mid att \in \text{dom}\ slots \bullet$$
$$\quad validAttType(att, slots(att)) = TRUE)$$

B.2
Propositional Formulae

$$ComparisonOperator ::= equal \mid not\_equal \mid less \mid less\_equal$$
$$\mid\ greater \mid greater\_equal$$

QualifiedConcept

$$concept : Concept$$
$$adjective : Adjective$$

$$adjective \in concept.adjectives$$

$$inSeq : ScaleValue \times \text{seq}\ ScaleValue \rightarrow Boolean$$

$$\forall\ sv : ScaleValue \bullet$$
$$\quad (\forall\ seqSv : \text{seq}\ ScaleValue \bullet$$
$$\quad\quad (\#seqSv = 0 \Leftrightarrow inSeq(sv, seqSv) = FALSE) \lor$$
$$\quad\quad (\#seqSv \neq 0 \land$$
$$\quad\quad\quad ((sv = head\ seqSv \Leftrightarrow inSeq(sv, seqSv) = TRUE)$$
$$\quad\quad\quad \lor (inSeq(sv, seqSv) = inSeq(sv, tail\ seqSv)))))$$

AttributeScaleSpecification

$$
\begin{array}{l}
attRef : AttRef \\
scaleValue : ScaleValue \\
\hline
(\forall\, attT : AttributeT \mid \\
\quad concreteAttribute\,(naturalAttribute\ attT) = \\
\quad\quad attRef.attribute \lor \\
\quad concreteAttribute\,(constructedAttribute\ attT) = \\
\quad\quad attRef.attribute \bullet \\
\quad (\exists\, scale : Scale \mid scale \in attT.scales \bullet \\
\quad\quad inSeq(scaleValue, scale.values) = TRUE)) \lor \\
(\forall\, pa : ProxyAttribute \mid \\
\quad proxyAttribute\ pa = attRef.attribute \bullet \\
\quad (\exists\, scale : Scale \mid scale \in pa.scales \bullet \\
\quad\quad inSeq(scaleValue, scale.values) = TRUE))
\end{array}
$$

AttributeValueSpecification

$$
\begin{array}{l}
attRef : AttRef \\
op : ComparisonOperator \\
instance : Instance \\
\hline
validAttType(attRef.attribute, instance) = TRUE
\end{array}
$$

$$
\begin{array}{rcl}
AtomicFormula & ::= & qualifiedConcept\langle\!\langle QualifiedConcept\rangle\!\rangle \\
& \mid & attributeScale\langle\!\langle AttributeScaleSpecification\rangle\!\rangle \\
& \mid & attributeValue\langle\!\langle AttributeValueSpecification\rangle\!\rangle
\end{array}
$$

$$
\begin{array}{rcl}
PropForm & ::= & atomic\langle\!\langle AtomicFormula\rangle\!\rangle \\
& \mid & not\langle\!\langle PropForm\rangle\!\rangle \\
& \mid & or\langle\!\langle PropForm \times PropForm\rangle\!\rangle \\
& \mid & and\langle\!\langle PropForm \times PropForm\rangle\!\rangle
\end{array}
$$

B.3
Preference Metamodel

B.3.1
Preference

$$OptimizationType ::= minimization \mid maximization$$

$$Goal ::= attributeGoal\langle\langle AttRef \times OptimizationType\rangle\rangle$$

$$\begin{aligned} Constraint ::= & \; constraintPreference\langle\langle PropForm\rangle\rangle \\ \mid & \; intervalPreference\langle\langle PropForm\rangle\rangle \\ \mid & \; aroundPreference\langle\langle PropForm\rangle\rangle \end{aligned}$$

$$\begin{aligned} & \forall\, intPref : Constraint \bullet \\ & \quad (\forall\, pa : PropForm \mid \\ & \qquad intPref = intervalPreference\; pa \bullet \\ & \qquad (\forall\, avs1, avs2 : AttributeValueSpecification \bullet \\ & \qquad\quad pa = and\,(atomic\,(attributeValue\; avs1), \\ & \qquad\qquad atomic\,(attributeValue\; avs2)) \land \\ & \qquad\quad avs1.attRef = avs2.attRef \land \\ & \qquad\quad ((avs1.op = less) \lor (avs1.op = less\_equal)) \land \\ & \qquad\quad ((avs2.op = greater) \lor (avs2.op = greater\_equal)))) \end{aligned}$$

$$\begin{aligned} & \forall\, aroPref : Constraint \bullet \\ & \quad (\forall\, pa : PropForm \mid \\ & \qquad aroPref = aroundPreference\; pa \bullet \\ & \qquad (\forall\, avs : AttributeValueSpecification \bullet \\ & \qquad\quad pa = atomic\,(attributeValue\; avs) \land \\ & \qquad\quad avs.op = equal)) \end{aligned}$$

$$DontCare ::= dontCareAttribute\langle\langle AttRef\rangle\rangle$$

$$
\begin{aligned}
PrefTarget &::= enumValue\langle\langle EnumerationValue\rangle\rangle \\
&\mid concept\langle\langle Concept\rangle\rangle \mid instance\langle\langle Instance\rangle\rangle \\
&\mid constraintTarget\langle\langle Constraint\rangle\rangle \\
&\mid value\langle\langle Value\rangle\rangle
\end{aligned}
$$

$$
\begin{aligned}
LikertScale &::= best \mid very\_good \mid good \mid neutral \mid bad \\
&\mid very\_bad \mid worst
\end{aligned}
$$

$$
\begin{aligned}
ExpressiveSpeechAct &::= prefer \mid need \mid desire \mid avoid \mid like \\
&\mid want \mid accept \mid require \mid love \mid hate
\end{aligned}
$$

$$
\begin{aligned}
ClassificatoryStat &::= ratingStat\langle\langle PrefTarget \times LikertScale\rangle\rangle \\
&\mid qualifyingStat\langle\langle PrefTarget \times ExpressiveSpeechAct \\
&\qquad \times Boolean\rangle\rangle
\end{aligned}
$$

$$
\begin{aligned}
ComparativeStat &::= orderStat\langle\langle PrefTarget \times PrefTarget \times Boolean\rangle\rangle \\
&\mid indifferentStat\langle\langle \mathbb{P}\, PrefTarget\rangle\rangle
\end{aligned}
$$

$$
\begin{aligned}
PreferenceStat &::= classificatoryStat\langle\langle ClassificatoryStat\rangle\rangle \\
&\mid comparativeStat\langle\langle ComparativeStat\rangle\rangle
\end{aligned}
$$

$$
\begin{aligned}
PreferenceT &::= statement\langle\langle PreferenceStat\rangle\rangle \\
&\mid goal\langle\langle Goal\rangle\rangle \\
&\mid constraint\langle\langle Constraint\rangle\rangle \\
&\mid dontCare\langle\langle DontCare\rangle\rangle
\end{aligned}
$$

Preference

$$
\begin{aligned}
&condition : PropForm \\
&decisionContext : PropForm \\
&preference : PreferenceT
\end{aligned}
$$

B.3.2 Priority

$$
\begin{aligned}
PriorityT &::= attributePriority\langle\langle AttRef \times AttRef \times Boolean\rangle\rangle \\
&\mid attributeIndifference\langle\langle \mathbb{P}\, AttRef\rangle\rangle \\
&\mid preferencePriority\langle\langle Preference \times \mathbb{Z}\rangle\rangle
\end{aligned}
$$

Priority

$condition : PropForm$

$decisionContext : PropForm$

$priority : PriorityT$

C
Questionnaire: Survey on Reasons for Choice

This appendix presents the questionnaire used in our study of arguments adopted by people to justify a decision (Chapter 8). Participants had the option of answering it either in English or in Portuguese. If participants tried to go back in the questionnaire, they were notified that changes would not be stored.

C.1
Introduction: Survey on Reasons for Choice

The purpose of this survey is to collect data that helps in understanding the **reasons for a choice**. The survey is completely anonymous and all information collected will be used solely for analysis within the context of this study. The survey has three steps and the estimated time for answering the survey is around 10 minutes.

Please, click on the image below to start the survey in English:

C.2
Part I: User Data

- Age: a positive integer;
- Gender: a value from {Male, Female};
- Country: a value from a provided list of countries;
- City: a string;
- Working/Studying Field: a string.

*All fields are mandatory.

C.3
Part II: Option Selection

Assume you are going to spend holidays in New York with a close friend (you do not mind sharing a bed with him/her). You were given the following hotel options, from which you have to choose one. Which would you choose?

NB1. Prices are in American dollars.

NB2. Provided options are based in real data, but details were changed for the purposes of this study.

– *Chosen Option*: {Hotel 91, Econo Lodge Times Square, The Hotel at Times Square, Comfort Inn Times Square, Renaissance New York Hotel 57};

Available options were given as shown in Figure C.1.

C.4
Part III: Reasons for your Choice

Could you please tell us why you chose "*chosen hotel*", and why you rejected the other options? Assume that this **justification must be good enough so that it can be used to convince your friend that your choice is the best one**. Please, use full sentences.

– Why did you [*accept*/*reject*] "Hotel 91"?
[text area in which participants write their explanation]
– Why did you [*accept*/*reject*] "Econo Lodge Times Square"?
[text area in which participants write their explanation]
– Why did you [*accept*/*reject*] "The Hotel at Times Square"?
[text area in which participants write their explanation]
– Why did you [*accept*/*reject*] "Comfort Inn Times Square"?
[text area in which participants write their explanation]
– Why did you [*accept*/*reject*] "Renaissance New York Hotel 57"?
[text area in which participants write their explanation]

Available options are presented again (Figure C.1).

	Hotel 91 **2 Stars** 91 East Broadway, Lower East Side, NY 10002 New York	Econo Lodge Times Square **3 Stars** 302 West 47th Street, Midtown, NY 10036 New York	The Hotel at Times Square **3 Stars** 59 West 46th Street, Midtown, NY 10036 New York	Comfort Inn Times Square **3 Stars** 129 West 46th St, Midtown, NY 10036 New York	Renaissance New York Hotel 57 **4 Stars** 130 East 57th Street, Midtown, NY 10022 New York
Distance from Times Square	Map	Map	Map	Map	Map
Price (per night)	**$97.3**	**$129.99**	**$134.99**	**$144.99**	**$219.0**
Check-in	From 15:00	15:00 - 23:30	15:00 - 00:00 hours	15:00 - 00:00 hours	From 15:00 hours
Check-out	Until 11:00	Until 12:00	Until 12:00 hours	Until 12:00 hours	Until 12:00 hours
Internet	Wired internet is available in the entire hotel and is free of charge.	Wi-fi is available in the entire hotel and is free of charge. Wired internet is available in the hotel rooms and is free of charge.	Wi-fi is available in the entire hotel and is free of charge.	Wi-fi is available in the entire hotel and is free of charge.	Wired internet is available in public areas and is free of charge. Wired internet is available in the hotel rooms and costs USD 16.95 per 24 hours
Parking	Public parking is possible at a location nearby (reservation is not needed) and costs USD 32 per day.	Public parking is possible at a location nearby (reservation is not needed) and costs USD 30 per day.	Public parking is possible at a location nearby (reservation is not needed) and costs USD 35 per day.	Public parking is possible at a location nearby (reservation is not needed) and costs USD 23 per day.	Private parking is possible on site (reservation is not needed) and costs USD 55 per day. Public parking is possible at a location nearby (reservation is not needed) and costs USD 55 per day.
24-hour Front Desk	X	X	X	X	X
Express Check-in/Check-out		X	X		X
Luggage Storage	X	X	X	X	X
Elevator	X	X	X	X	X
Bar			X		X
Restaurant					X
Laundry	X		X	X	X
Fitness Center			X	X	X
Business Center		X	X	X	X
Room Service					X
Breakfast in the Room					X
Room Type	Queen Room	Superior Queen Room	Queen Room	Queen Room	Queen Room
Breakfast Included		X	X	X	X
Safety Deposit Box		X	X	X	X
Air Conditioning	X	X	X	X	X
Heating	X	X	X	X	X
Iron		X	X	X	X
Hairdryer	X	X	X	X	X
Bath or Shower	X	X	X	X	X
Toilet	X	X	X	X	X
Alarm Clock or Wake up Service	X	X	X	X	X
Telephone	X	X	X	X	X
TV	X	X	X	X	X
Cable TV	X	X	X	X	X
Radio	X	X	X	X	X
Tea/Coffee Maker		X	X	X	X
Seating Area			X		X
Work Desk	X	X	X	X	X
Refrigerator		X			
Room size	19.0	21.0	21.0	22.0	24.0

Figure C.1: Hotel Options.

D
Application for Evaluating our Proposed Approach

This appendix presents the application used in our study to evaluate our approach and compare explanation generation techniques (Chapter 10).

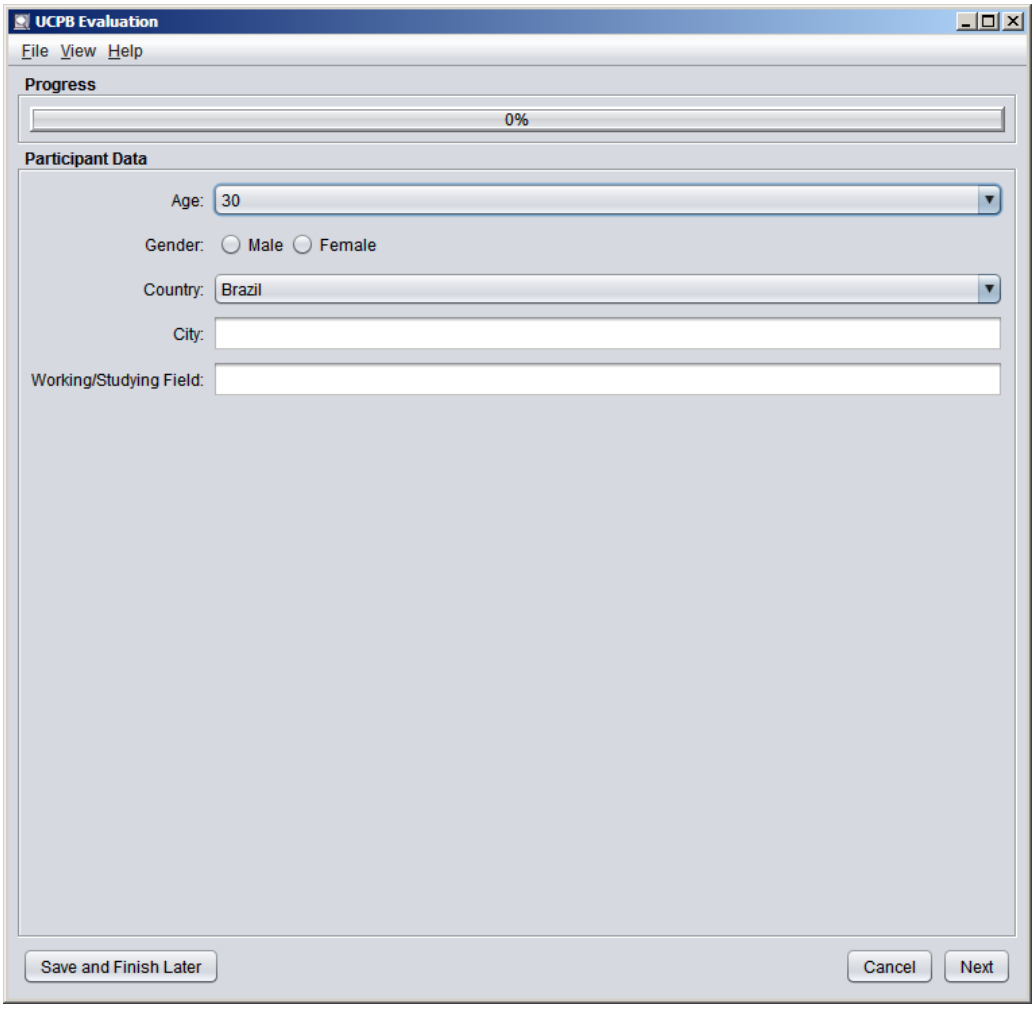
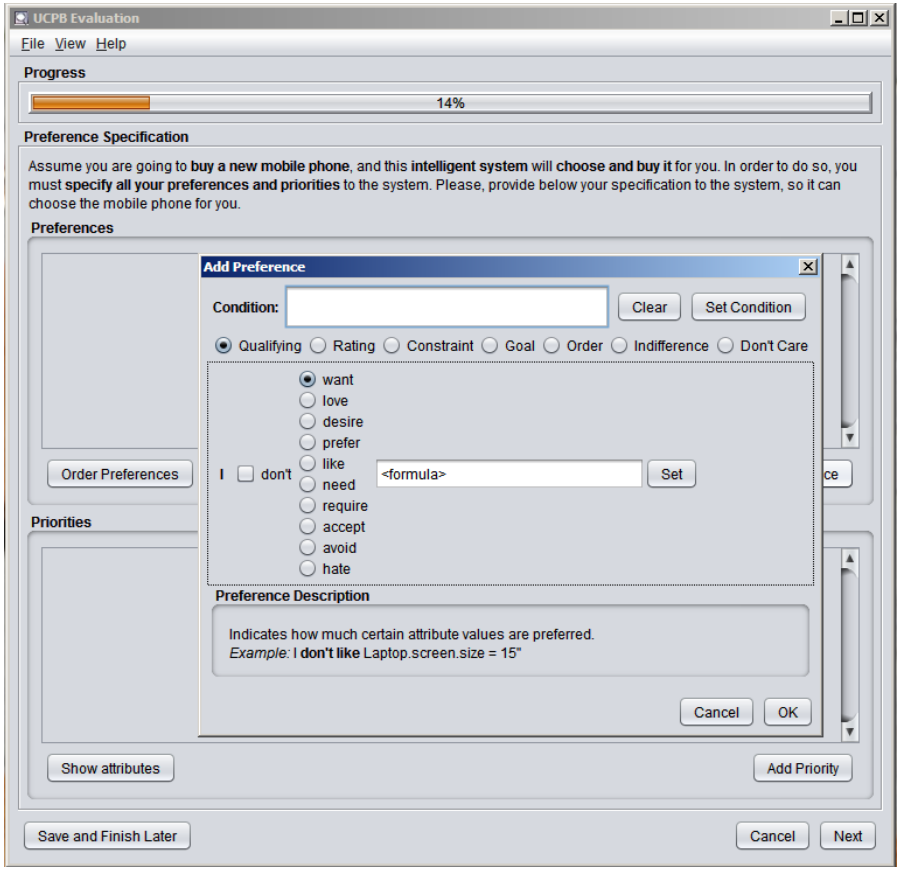
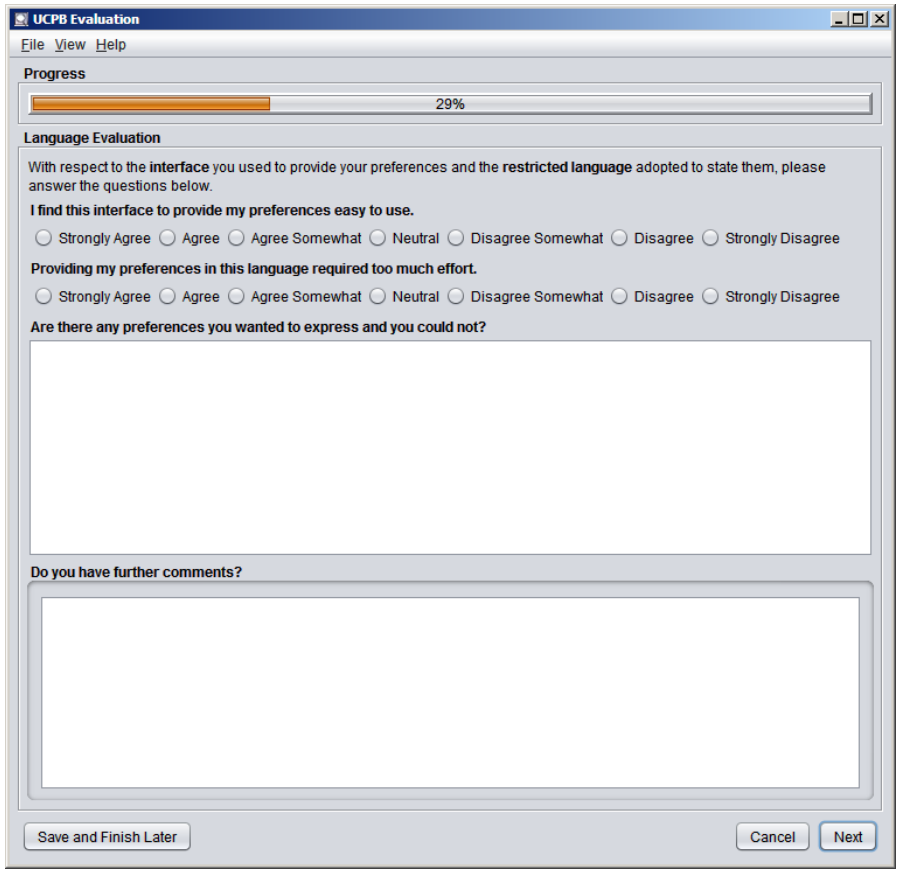


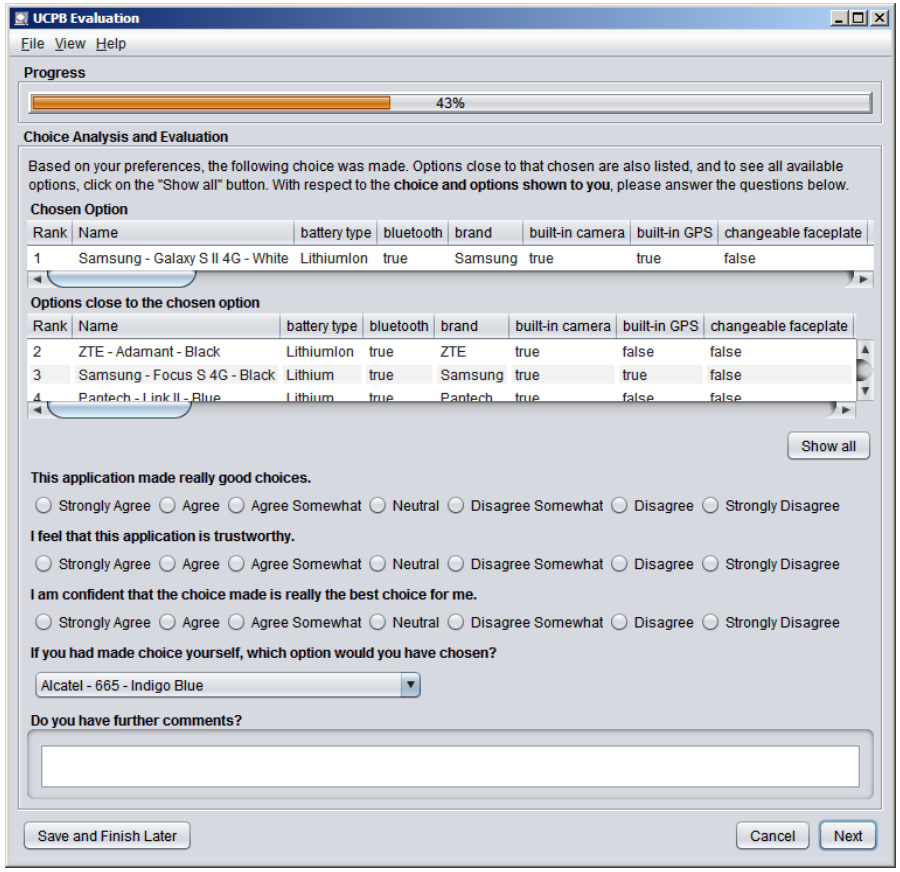
Figure D.1: Participant Data.



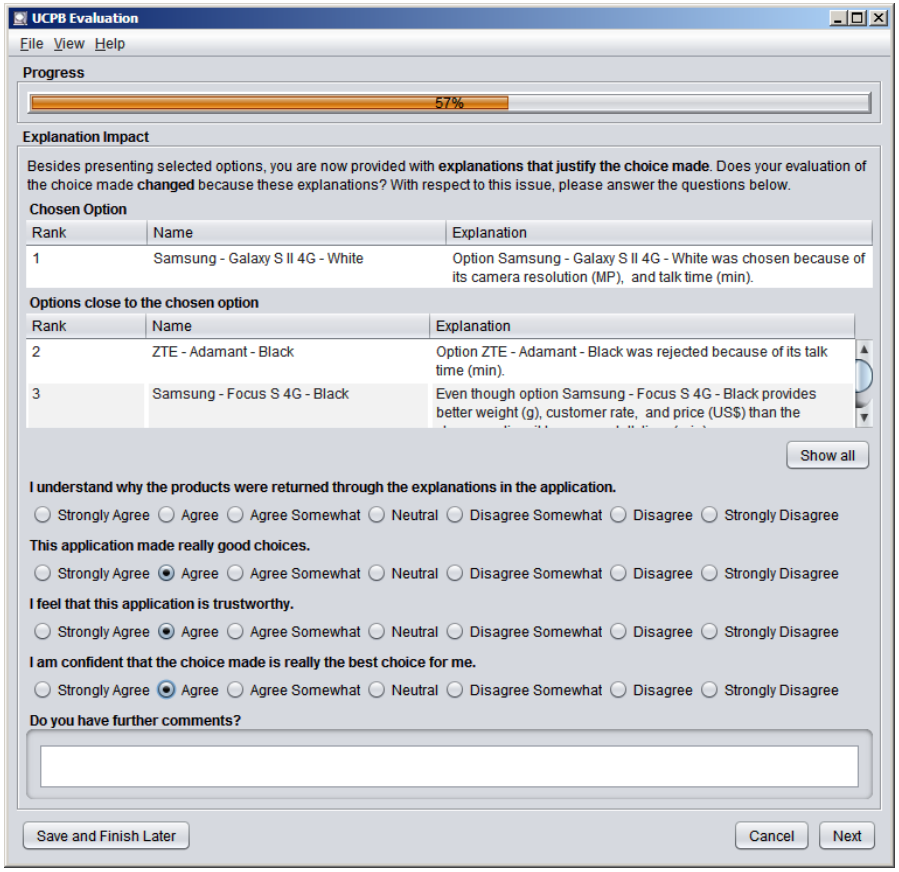
D.2(a): Preferences.



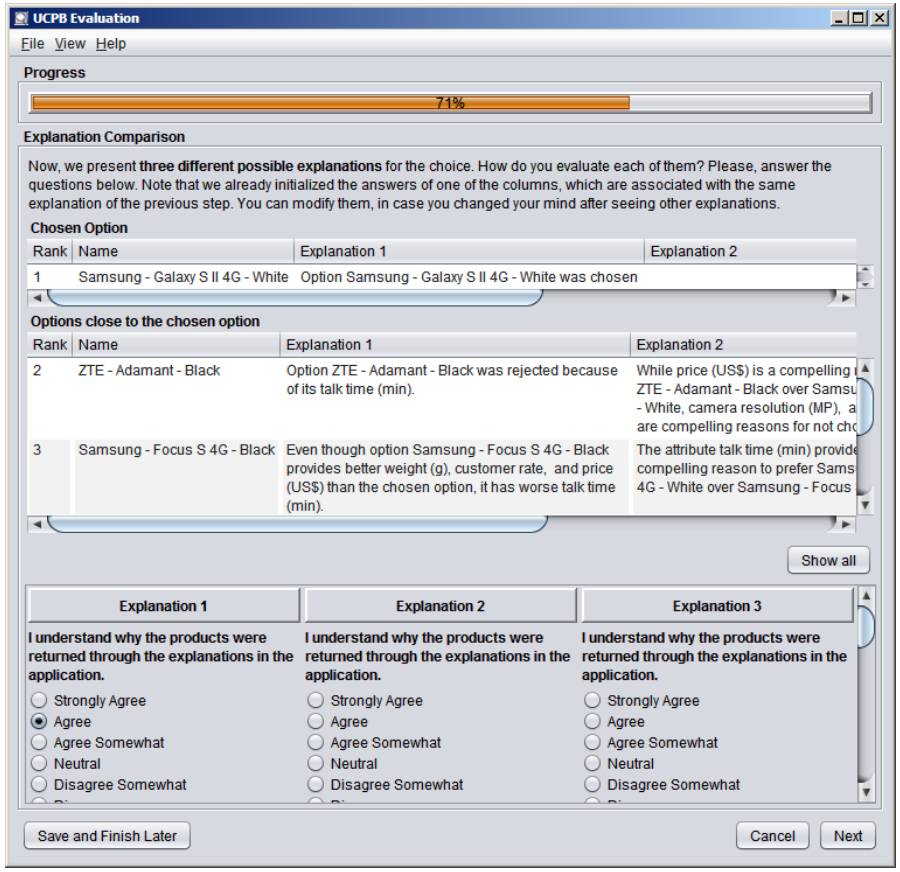
D.2(b): Preference Language Evaluation.



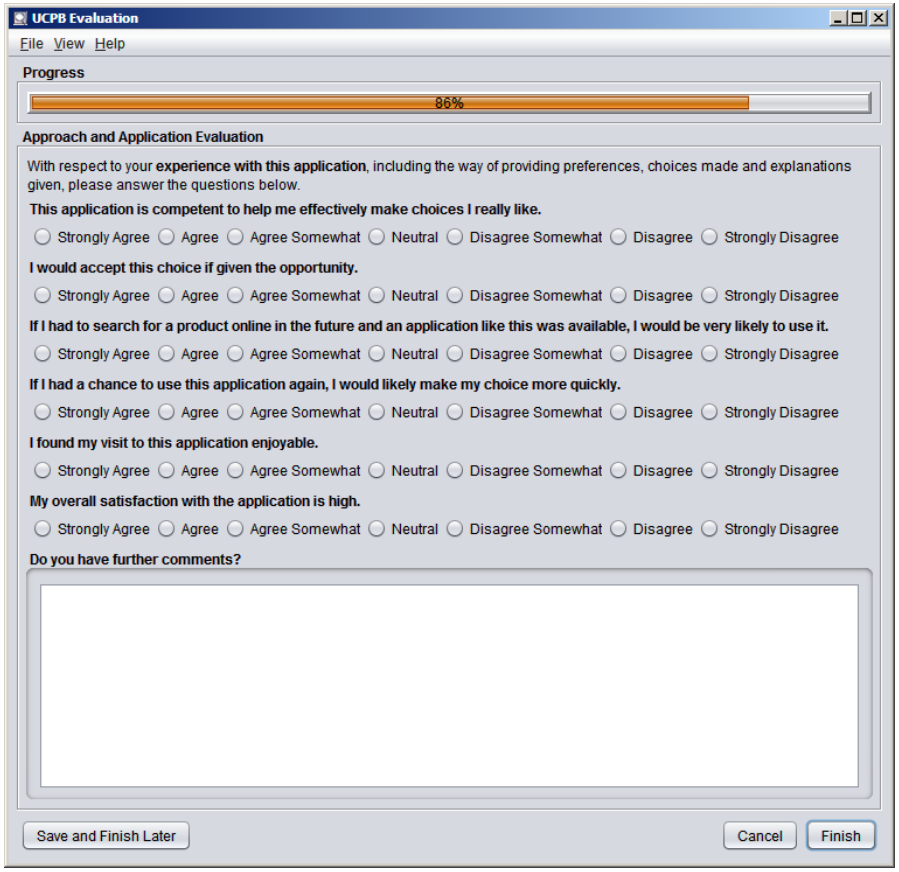
D.2(c): Choice Analysis and Evaluation.



D.2(d): Explanation Impact.



D.2(e): Explanation Comparison.



D.2(f): Approach Evaluation.