## ABSTRACT

WICKER, ANDREW W. Interest-Matching Comparisons using CP-nets. (Under the direction of Professor Jon Doyle.)

The formation of internet-based social networks has revived research on traditional social network models as well as interest-matching, or match-making, systems. In order to automate or augment the process of interest-matching, we follow the trend of qualitative decision theory by using qualitative preference information to represent a user's interests. In particular, a common form of preference statements for humans is used as the motivating factor in the formalization of *ceteris paribus* preference semantics [13]. This type of preference information led to the development of *conditional preference networks* (CP-nets) [6, 5]. This thesis presents a method for the comparison of CP-net preference orderings which allows one to determine a shared interest level between agents. Empirical results suggest that distance measure for preference orderings represented as CP-nets is an effective method for determining shared interest levels. Furthermore, it is shown that differences in the CP-net structure correspond to differences in the shared interest levels which are consistent with intuition. A generalized Kemeny and Snell axiomatic approach for distance measure of strict partial orderings is used as the foundation on which the interest-matching comparisons are based.

**Interest-Matching Comparisons using CP-nets**

by

**Andrew W. Wicker**

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

**Computer Science**

Raleigh, North Carolina

2006

**APPROVED BY:**

| | |
|---|---|
| Dr. Peter R. Wurman | Dr. Robert St. Amant |

Dr. Jon Doyle
Chair of Advisory Committee

*To my parents*

# BIOGRAPHY

Andrew White Wicker was born on December 11, 1981 and raised in Greensboro, North Carolina, just one hour west of North Carolina State University. Upon receiving a Bachelor of Science degree in Computer Science with a minor in Mathematics at North Carolina State University, he decided to continue in the graduate program to pursue his interest in Artificial Intelligence.

## ACKNOWLEDGEMENTS

I would like to thank Dr. Jon Doyle for his time and support throughout the research for this thesis. His attention to detail and highest expectations for research quality pushed me to achieve a level of intellectual development unforeseeable at the beginning of my graduate studies.

# Contents

# List of Figures

# Chapter 1

# Introduction

It is natural for humans to apply new technology to a social context if it is useful in any way. With the advent of the personal computer, and subsequently the internet, users of these tools have discovered both mundane and ingenious ways for facilitating social interaction with other users. It was not long before people began constructing ad-hoc social networks through, for example, mailing lists and forums. These early "social networks" eventually evolved into the more interactive and media-rich social networks that can be seen today (e.g., MySpace.com and FaceBook.com).

Internet-based social networks have become a place where people can go to see and be seen by others. Nearly all of the popular social network sites allow users to upload detailed information about themselves along with photos and other types of multi-media. This helps to create the interactive experience which causes people to continually update their profile, as well as add new friends to their personal social network. This is done typically by querying the database of users for other users that share a common interest. The social network interaction, expansion, and profile updates are what keep users engaged.

## 1.1 Outline of Problem

An increasing amount of work has been done on formalizing what it means for two people, or agents, to share a common interest [1, 9, 14, 16, 22]. This is typically referred to

as interest-making, or match-making. A decision to interact with another person is based strongly on what level of interest is associated with some aspect of that person. These aspects often are a similarity of preference or desires. Although basic notions of interest-matching are not novel ideas, there are numerous approaches to this problem which offer their own advantages and disadvantages [9].

In order to automate or augment the process of interest-matching, it is helpful to develop a model which formalizes the characteristics of a person that are necessary for making this sort of decision. Preference information is a natural way of capturing what it is that a person wants or intends to do. In particular, conditional preference networks (CP-nets) are a natural, compact representation of preference information operating under the *ceteris paribus* semantics [6, 5, 13].

This thesis describes a reasonable method for facilitating interest-matching comparisons between users in a social environment that uses preference information represented as a CP-net. Furthermore, distance measures of orderings are shown to be capable of computing reasonable and intuitive interest levels.

## 1.2 Approach

These on-line social networks are especially intriguing when viewed in the context of multi-agent systems. Each user in the network can be thought of as an individual agent with specific preferences, desires, intentions, and/or goals. The interpretation of a social network environment as a multi-agent environment more easily facilitates formalization of the social interactions using decision-theoretic models.

The social networks now in place have methods through which users are able to expand their network which is based on a level of interest in another user's profile information. A CP-net provides a formal model for representing a user's preferences, which are necessary for making a network expansion decision. It is shown how comparisons can be made between CP-nets of multiple users in an attempt to measure a level of shared interest.

There has been much research conducted on distance measures between orderings over objects [2, 3, 15]. This thesis makes use of a generalized Kemeny-Snell axiomatic approach described by Bogart [2] to form comparisons between CP-net preference orderings. It is shown that the distance function defined by this approach produces reasonable values

when using the preferences represented in CP-nets.

## 1.3 Contributions

This thesis describes a step toward a potential application of CP-nets in a social multi-agent environment. Research on CP-nets has not fully explored this application area. Closely related work on $m$CP-nets focuses on aggregating preference information of multiple agents for group consensus [20], which is in contrast to the individualistic view of agent decision-making that has motivated work on this thesis. By allowing the agents to make decisions based on individual interest-matching comparisons (as opposed to group consensus), they are afforded more freedom in their decision-making process.

A generalized Kemeny-Snell axiomatic approach to measuring distance between strict partial orderings is described [2]. We build on this approach by defining a method for modifications of CP-nets, called *CP-permutations*, which consequently result in modifications to the preference orderings they represent. It is discussed how these modifications could be used to define a much more efficient axiomatic approach to measuring distance between the actual CP-net structure, not the induced preference orderings.

Several problems that are common for interest-matching systems are outlined in [9]. The approach to interest-matching described in this thesis addresses a couple of these problems. First, many interest-matching systems do not allow ranking of interest levels between multiple agents. By using the interest function that is described in this thesis, rankings of interest values are permitted since the values are normalized (i.e., they are independent of the size of the preference orderings being compared). Second, exact matches of outcome domains over which the preferences are specified are often required in interest-matching systems. We describe an induced preference graph expansion method which allows one to make comparisons even if the feature sets (hence, outcome sets) over which the preference orderings are defined are significantly different. This allows one to make partial, or potential interest matches.

An overarching motivation for this thesis is the foundation it lays for extending CP-nets to a social multi-agent environment by defining a formal method for determining shared interest levels between them. This method could be utilized in a multitude of multi-agent applications in which agents must collaborate or coordinate with other agents that

possess similar preferences, desires, or goals. A framework for this sort of application is the next logical step for future work on this thesis topic.

## 1.4 Organization of Thesis

This thesis begins with a discussion of a preferences, in which we describe basic types of relations, preference notation, and the *ceteris paribus* semantics. The *ceteris paribus* semantics is built-upon in Chapter 3 with the introduction of CP-nets. The structure of CP-nets is discussed, along with techniques for reasoning with them. In Chapter 4, we lay out the generalized Kemeny-Snell axiomatic foundation on which the interest-matching comparisons are based. Chapter 5 describes some of the relational and structural properties of CP-nets in a multi-agent environment. Chapter 6 introduces the shared interest function used to make comparisons between CP-nets, and discusses how the generalized Kemeny-Snell axioms apply in the context of preference orderings represented as CP-nets. Several examples are provided in Chapter 7 which show how the interest-matching comparisons are computed when variations between CP-nets are introduced. This thesis concludes with a comparative discussion of related works and future work on this topic.

# Chapter 2

# Preferences

If one wants to define a method through which interest-matching comparisons are made, then it makes sense to use preferences as the comparative information. Preferences are a natural way to express desires, plans, and goals [23]. Furthermore, qualitative preference information is especially easy for humans to state, and less prone to error, in comparison to quantitative utility information [11].

Preference information that has been elicited from a user can be stated over many types of relations. It is important to recognize the differences between these relations. In this chapter, we look at types of relations used to define various types of orderings. This discussion continues with a description of the *ceteris paribus* preference semantics.

## 2.1 Types of Relations

Prior to continuing with a more in-depth look at preference semantics and representation, it is helpful first to take a look at basic relations of preference statements. We emphasize an ordering over the set of outcomes which represents the desirability of an outcome.

Let $O$ be a set of distinct outcomes (or objects). The cardinality of the outcome set $O$ is denoted by $|O|$. The cartesian product $O \times O$, denoted by $O^2$, represents all possible ordered pairs of outcomes in $O$. We can represent each binary relation $R$ on $O$ as a set of

pairs $R \subseteq O^2$. The set notation $(o, o') \in R$ is used instead of the relational notation $oRo'$ to denote a relation between $o$ and $o'$ [19, 18].

**Definition** A relation $R$ on $O$ is *reflexive* if and only if $(o, o) \in R$ for each $o \in O$. A relation $R$ on $O$ is *irreflexive* if and only if $(o, o) \notin R$ for each $o \in O$.

**Definition** A relation $R$ on $O$ is *symmetric* if and only if $(o, o') \in R$ implies $(o', o) \in R$ for each $o, o' \in O$. A relation $R$ on $O$ is *antisymmetric* if and only if $(o, o') \in R$ and $(o', o) \in R$ imply $o = o'$ for each $o, o' \in O$. A relation $R$ on $O$ is *asymmetric* if and only if $(o, o') \in R$ implies $(o', o) \notin R$ for each $o, o' \in O$.

**Definition** A relation $R$ on $O$ is *transitive* if and only if $(o, o') \in R$ and $(o', o'') \in R$ implies $(o, o'') \in R$ for each $o, o', o'' \in O$.

**Definition** A *preorder* is a reflexive and transitive relation.

**Definition** A *partial order* is an antisymmetric preorder.

**Definition** A *strict partial order* is an irreflexive, transitive, and asymmetric relation.

**Definition** An ordering is *complete* if and only if $(o, o') \in R$ or $(o', o) \in R$ for each $o, o' \in O$.

**Definition** A *weak order* is a complete and transitive relation.

There are many other types of orderings defined over these relations that are not mentioned in this thesis. We are concerned primarily with the weak ordering and strict partial ordering. If an ordering type is not specified, or irrelevant in the context of the specific problem, then we will refer to it as simply an ordering.

## 2.2 Preference Notation

Three distinct degrees of preferential ordering are used in the general description of an agent's preferences. Let $o$ and $o'$ be outcomes from a set of outcomes $O$ of executing an action (or, more generally, propositions) on which an agent can place a relation. It is assumed that a complete preorder is placed over $O$ by the relation $\succsim$ [23]. When $o$ is *weakly preferred* to $o'$, we write $o \succsim o'$. We define $o \succ o'$ as *strict preference* of $o$ over $o'$ if $o \succsim o'$ and $o' \not\succsim o$. If $o \succsim o'$ and $o' \succsim o$, we write $o \sim o'$ and define this as *indifference* between $o$ and $o'$.

## 2.3 *Ceteris Paribus* Semantics

Some human preferences are commonly a type of multi-attribute preference that can be stated formally by preference statements operating under the *ceteris paribus* semantics [13]. This type of preference semantics takes an "all else equal" approach when considering alternatives in a decision-making context. Traditionally, decision-theoretic approaches taking advantage of preferences attempt to define an ordering over all outcomes. The problem with this is that most planning systems suffer from a combinatorial explosion with the set of possible outcomes. Enumerating all of these is incredibly inefficient, and sometimes impossible. To avoid this inefficiency, the *ceteris paribus* semantics is used to lift preference statements to preferential patterns that exist over classes of outcomes (or possible worlds).

*Ceteris paribus* preference semantics assumes outcomes are described by a set of binary features $F$. The features in the set $F$ are capable of describing all of the possible classes of outcomes derivable from our decision-making process. We define preferences for outcomes described in terms of features in $F$ over others through *ceteris paribus* preference statements.

Let $o = \langle abc \rangle$ and $o' = \langle \bar{a}bc \rangle$ be outcomes, where $F = \{A, B, C\}$. Assume that we have the following *ceteris paribus* preference rule: $a \succ \bar{a}$. This rule states a preference for $a$ over $\bar{a}$, assuming that all other features in $F$ are held constant. The preference statement is operating under the *ceteris paribus* semantics if it holds true when the truth values for all features other than $a$ and $\bar{a}$ are the same and held constant. Another way of saying this is, every class of outcomes satisfying $a$ is preferred to those satisfying $\bar{a}$ if all other features in these statements are the same and held constant. Given the two outcomes $o$ and $o'$, we would state a preference for $o$ over $o'$, written $o \succ o'$, based on this single *ceteris paribus* rule since the features $B$ and $C$ not mentioned in the rule are the same and held constant in both $o$ and $o'$.

As a less formal example, consider the case of placing preference on airplane flights. There are typically many features to consider when choosing the best, most preferred flight. We specify that cheaper flights are preferred to shorter flights, all else equal. So, we prefer flights that are cheap and not short to flights that are not cheap and short, if all other features relevant to an airplane flight are the same in the alternative outcomes being compared.

We turn our attention now to a graphical representation for preference using a conditional form of the *ceteris paribus* preference semantics.

# Chapter 3

# Conditional Preference Networks

Often it is desirable in automated decision support systems to make decisions using a qualitative specification for preferences, as opposed to a quantitative specification. Of specific interest are *conditional preference networks* (CP-nets), which specify a qualitative and conditional factored representation of preferences in a graphical network [6, 5, 10]. By using conditional *ceteris paribus* preference statements a CP-net exploits independence relations between these preference statements to produce a compact, intuitive, and well-structured graphical model.

## 3.1 Conditional *Ceteris Paribus* Preference Statements

As stated, CP-nets represent conditional preferences operating under the *ceteris paribus* semantics. An example of a conditional preference is provided: "*If I drink coffee, I prefer that it has sugar in it.*" That is, my preference for sugar is dependent on the condition that I am drinking coffee. Applying the semantics of *ceteris paribus* statements, we conclude that the statement about coffee holds true "all else equal". It does not depend on the temperature or type of coffee if it is the same for all coffee under consideration.

The general form of the notation used for these conditional statements is $a$: $b \succ \bar{b}$ (i.e., given $a$, I prefer $b$ to $\bar{b}$). The previously mentioned conditional preference statement about coffee can be represented using the notation of CP-nets as "*Coffee: Sugar* $\succ \neg$*Sugar*".

This notation provides a compact qualitative specification for potentially complex preference statements.

## 3.2 CP-net Structure

When constructing a CP-net, it is necessary first to elicit preference information from the user. This is done by having the user specify influence relations between preference features. The influence relations represent a preference feature independence assumption that is critical to CP-nets and permitted by the use of the conditional *ceteris paribus* preference semantics. This assumption is analogous to the probabilistic independence assumption used in Bayesian networks, however, it is a weaker notion of relations between nodes.

We describe a standard format of the outcomes being used throughout the remainder of this paper. This is done by defining an enumerated finite universe of features denoted by $\mathbf{F} = \langle f_1 \dots f_n \rangle$. The finite universe of features $\mathbf{F}$ contains the features from which we are able to construct all CP-nets used in this thesis. The ordering of the features in the enumeration carries with it no semantical meaning and is used only to permit direct comparison of ordering matrices (see Chapter 6).

Each feature $f$ in a CP-net can be instantiated over a domain of values, $Dom(f)$. For specifying a standard format for outcomes, let $O_{\mathbf{F}} = \prod_{f \in \mathbf{F}} Dom(f)$ be the set of outcomes over $\mathbf{F}$. Each outcome in $O_{\mathbf{F}}$ is a n-tuple, where $f \in \mathbf{F}$ and $|\mathbf{F}| = n$. We denote the outcomes over $F \subset \mathbf{F}$ by $O_F$. Given $O_F$, we say $O_{F'}$ is a set of partial outcomes, where $F' \subset F$. A partial outcome is sometimes referred to as an instantiation. It is worth pointing out that the formal specification of a CP-net allows for arbitrary finite domains, but we assume $|Dom(f)| = 2$ for each $f \in \mathbf{F}$ throughout this paper [6, 5].

As a simple example, let $\mathbf{F} = \{A, B, C\}$. Thus, the set of all outcomes is $Dom(A) \times Dom(B) \times Dom(C)$. For a CP-net $N$, where $F = \{A, C\} \subset \mathbf{F}$, the outcomes will be formatted as $Dom(A) \times Dom(C)$.

We develop formally the *conditional preferential independence* assumption which is exploited in the construction of CP-nets. We define two subsets of features, $X \subset F$ and its complement $Y = F - X$, where $F \subseteq \mathbf{F}$. Let $x_1, x_2 \in O_X$ and $y_1, y_2 \in O_Y$ be instantiations of the feature sets $X$ and $Y$, respectively. Let $x_1y_1$ represent the concatenation of partial outcomes $x_1$ and $y_1$ such that $x_1y_1 \in O_F$ is a complete outcome [6]. We say that $X$

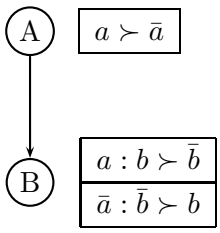


Figure 3.1: A simple CP-net using just two feature nodes

and $Y$ are *preferentially independent* if and only if it is true for all $x_1, x_2, y_1, y_2$ that

$$x_1y_1 \succsim x_2y_1 \iff x_1y_2 \succsim x_2y_2 \tag{3.1}$$

We say that $x_1$ is preferred to $x_2$ if the above relation is true [6, 5]. The notion of *conditional preferential independence* is defined by introducing a third set of features $Z$ such that the nonempty sets $X$, $Y$, and $Z$ partition the set $F$. Given $z \in O_Z$, we say $X$ is *conditionally preferentially independent* of $Y$ if and only if it is true for all $x_1, x_2 \in O_X$ and for all $y_1, y_2 \in O_Y$ that

$$x_1y_1z \succsim x_2y_1z \iff x_1y_2z \succsim x_2y_2z \tag{3.2}$$

The relation stated above is interpreted as saying that $X$ and $Y$ are *conditionally preferentially independent* in the case when $Z$ is instantiated as $z$.

The structure of a CP-net, as considered in this thesis, is defined formally in graph-theoretic terms as a *directed acyclic graph* (DAG). There has been much work on cyclic CP-nets, but they are not considered in this thesis [5]. Each node in a CP-net $N = (F, E)$ is labeled by a single feature $f \in \mathbf{F}$. The parents of a feature node $f$ in a CP-net $N$ are denoted as $Pa(f)$. A feature node $f_j$ is a parent of $f_i$, that is $f_j \in Pa(f_i)$, if there exists a directed edge in the CP-net from $f_j$ to $f_i$. The acyclicity of the CP-nets considered in this paper maintains that inconsistencies will not appear in the feature node relations.

A set of feature influence relations forms a dependency relation on all features. This dependency relation is used to construct the actual CP-net. We start with a simple

example using just two features, $A$ and $B$ [6, 5]. The user specifies that $B$ is *conditionally preferentially dependent* on $A$. Furthermore, the user specifies a preference for $a$ over $\bar{a}$ (i.e., $a \succ \bar{a}$), $b$ over $\bar{b}$ given $a$ (i.e., $a : b \succ \bar{b}$), and $\bar{b}$ over $b$ given $\bar{a}$ (i.e., $\bar{a} : \bar{b} \succ b$). The structure of the basic CP-net that is constructed from this dependency information is depicted in Figure 3.1.

Note that each node in Figure 3.1 is annotated with a *conditional preference table* (CPT). This is similar to the *conditional probability tables* of Bayesian networks. These CPTs provide a compact description of a user's complete preference ordering for domain values of a feature node $f$ over every instantiation of its parent feature nodes, denoted by $O_{Pa(f)}$. So, for feature node $B$ we have both $a : b \succ \bar{b}$ and $\bar{a} : \bar{b} \succ b$ in its CPT.

We call instantiation $u \in O_{Pa(f)}$ a partial outcome since $Pa(f) \subset F$. Each $u \in O_{Pa(f)}$ provides a complete order $\succ_f^u$ over $Dom(f)$ [5]. For example, let $F = \{A, B, C\}$ and let $u = a\bar{b} \in O_{Pa(C)}$, where $Dom(A) = \{a, \bar{a}\}$ and $Dom(B) = \{b, \bar{b}\}$. Then, from Figure 3.2 we have $c \succ_C^u \bar{c}$.

Let $\mathcal{N}_F$ be the set of all CP-nets definable over feature set $F \subseteq \mathbf{F}$ and let $\mathcal{N} = \bigcup_{F \subseteq \mathbf{F}} \mathcal{N}_F$ be the set of all CP-nets.

**Definition** A CP-net $N \in \mathcal{N}_F$ is a directed graph over feature nodes $F$, each node $f \in F$ of which is annotated with a conditional preference table $CPT(f)$. Each $CPT(f)$ associates a complete order $\succ_f^u$ with each $u \in O_{Pa(f)}$.[5]

An example with more detailed preference information is provided. Assume a user has specified a preference for $a$ over $\bar{a}$ (i.e., $a \succ \bar{a}$), $b$ over $\bar{b}$ given $a$ (i.e., $a : b \succ \bar{b}$), $\bar{b}$ over $b$ given $\bar{a}$ (i.e., $\bar{a} : \bar{b} \succ b$), $\bar{c}$ over $c$ given $b$ (i.e., $b : \bar{c} \succ c$), and $c$ over $\bar{c}$ given $\bar{b}$ (i.e., $\bar{b} : c \succ \bar{c}$).

A root node in a CP-net is specified as an unconditioned preference. The feature $A$ will represent the root node in our CP-net for these dependencies since its preference is unconditioned, and thus has no parents. The features $B$ and $C$ will be internal nodes where $A$ is the parent of $B$, and $B$ is the parent of $C$. The graphical structure for the CP-net that is constructed from these dependency relations is depicted in Figure 3.2. The nodes in this CP-net represent the features of interest to the user. Clearly, there is exactly one node for each feature of interest.

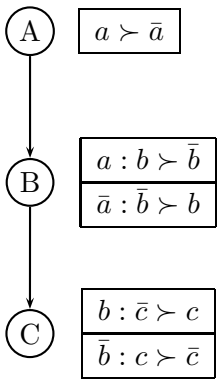


Figure 3.2: A simple CP-net using three feature nodes

## 3.3 CP-net Semantics

Now that a graphical representation of preference information has been constructed we are able to define the semantics of the CP-net to determine a preference ordering derived from the CPTs. The relations in the CPTs define a strict partial ordering over outcomes $O_F$.

Let $\Omega_F$ be the set of all strict partial preference orderings $\omega$ over outcome set $O_F$ and let $\Omega = \bigcup_{F \subseteq \mathbf{F}} \Omega_F$ be the set of all strict partial preference orderings.

We say that a preference ordering $\omega$ satisfies a CP-net if and only if it satisfies the conditional preferences in each of the CPTs. A preference ordering $\omega$ that satisfies a CP-net $N$ is interpreted as the meaning of that CP-net, denoted by $\omega = [\![N]\!]$. So, if a CP-net $N \in \mathcal{N}_F$, then $[\![N]\!] \in \Omega_F$.

In the remainder of this paper, we will denote the inclusion of a preference for $o$ over $o'$ in the ordering $\omega$ by either $(o, o') \in \omega$ or by the subscript notation as in $o \succ_\omega o'$. The later will be used when dealing with large preference orderings since set notation quickly becomes unmanageable. The standard notation for the strict preference relation $\succ$, without a subscript, is used whenever the source of the preference order is clear from the context.

The following preferential ordering satisfies the CP-net in Figure 3.2:

$$ab\bar{c} \succ abc \succ a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ \bar{a}b\bar{c} \succ \bar{a}bc \tag{3.3}$$

The notion of *flipping sequence* can be used to express the semantics of CP-nets.

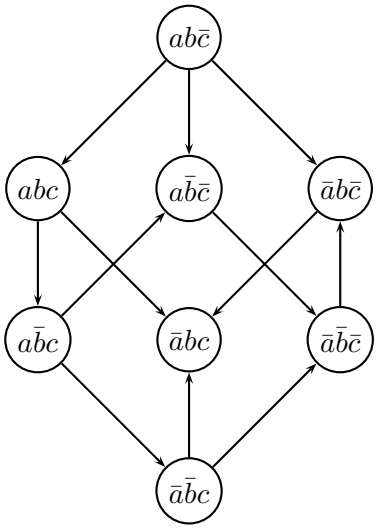


Figure 3.3: Induced preference graph

There are two types of *flipping sequence*: worsening and improving. A *worsening flip* is defined as the change in a single value of an outcome such that the newly created outcome is less preferred than the original. Conversely, an *improving flip* is defined if the newly created outcome is more preferred than the original. A sequence of one of these types of flips is called a *flipping sequence*. This idea forms the mechanism through which reasoning takes place in CP-nets.

Based on the CPTs in Figure 3.2, one might be tempted to state the preference $a\bar{b}\bar{c} \succ \bar{a}\bar{b}c$. However, this preference violates the flipping sequence semantics since $a\bar{b}\bar{c}$ and $\bar{a}\bar{b}c$ differ by more than a single feature value and there does not exist a flipping sequence containing $a\bar{b}\bar{c} \succ \bar{a}\bar{b}c$. Thus, this preference statement does not hold with respect to the CP-net 3.2 and is not included in the preference ordering (3.3) [6]. The specification of the preference ordering is perhaps best described through the preference graph induced by the CP-net it represents.

The CP-net $N$ depicted in Figure 3.2 induces a preference graph $P(N)$ (see Figure 3.3), which is a DAG where the nodes are the outcomes over which the preferences are placed. A directed edge $(o, o')$ is drawn from outcome $o$ to outcome $o'$ in $P(N)$ if and only if $o$ is preferred to $o'$ and $o'$ is a worsening flip of $o$. Now, a flipping sequence defines a strict partial preference ordering $\omega \in \Omega_F$ from the union of all paths of length $n \geq 1$ in $P(N)$,

where $N \in \mathcal{N}_F$. If we do this with the induced preference graph in Figure 3.3, we get the following two preference paths:

$$ab\bar{c} \succ abc \succ a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ \bar{a}b\bar{c} \succ \bar{a}bc \tag{3.4}$$

$$ab\bar{c} \succ abc \succ a\bar{b}c \succ \bar{a}\bar{b}c \succ \bar{a}\bar{b}\bar{c} \succ \bar{a}b\bar{c} \succ \bar{a}bc \tag{3.5}$$

These two preference paths are identical in their preference for all outcomes, excluding $a\bar{b}\bar{c}$ and $\bar{a}\bar{b}c$. Since a preference relation cannot be stated between $a\bar{b}\bar{c}$ and $\bar{a}\bar{b}c$, we say that CP-net 3.2 leaves them *incomparable*.

## 3.4 Reasoning with CP-nets

Let $o = \langle x_1 x_2 \dots x_n \rangle$ and $o' = \langle y_1 y_2 \dots y_n \rangle$ be two outcomes, where $x_i$ and $y_i$ are feature instantiations. Our goal is to determine whether $o' \succ o$ by using the preference information represented in the CP-net. We determine the validity of this preference, and thus define the semantics for CP-nets through finding a worsening or improving flipping sequence [6, 5]. These two differ only in the direction in which the search proceeds, although construction for one might be much more simplified. A worsening flipping search finds a flipping sequence from the more preferred outcome to the less preferred outcome through a sequence of less preferred outcomes. The converse is true for defining an improving flipping search.

We say that a CP-net $N$ entails a preference $o \succ o'$, written $N \models o \succ o'$, if and only if $o \succ o'$ is not violated in any preference ordering that satisfies $N$. In other words, $N \models o \succ o'$ if and only if $(o, o') \in [\![N]\!]$. The following lemma states that entailment, denoted by $\models$, of preferences represented in a CP-net is transitive. This transitivity is easily determined through paths in the induced preference graph. The usefulness of this lemma is stated in Section 6.1 when we describe an ordering matrix construction method using CP-net preference orderings.

**Lemma 3.4.1.** *For a CP-net $N \in \mathcal{N}_F$ and outcomes $o, o', o'' \in O_F$, if $N \models o \succ o'$ and $N \models o' \succ o''$ then $N \models o \succ o''$.[5]*

Both the worsening search and improving search can be viewed as a graphical search. Consider the search for determining whether it is true that $N \models o' \succ o$. For a

worsening search, we define the root node as $o' = \langle y_1 y_2 \dots y_n \rangle$. The children of any node in this graph represent those which are less preferred by flipping exactly one feature value $f$ to $f'$, in accordance with the CPTs, such that $f \succ f'$. Once the entire graph has been properly constructed in accordance with the CPTs, we can determine whether $N \models o' \succ o$ by finding a path in the worsening search graph from root node $o'$ to a node $o$. This test is polynomial in the number of feature nodes for the acyclic (poly)tree-structured CP-nets under consideration [10].

# Chapter 4

# Measuring Similarity

In order to compute interest-matching comparisons between CP-net preference orderings, an interest function and a representation of a CP-net preference ordering are defined. Each of these must preserve the semantics of the original CP-nets. Graph-theoretic isomorphism can tell us that two CP-nets are structurally identical, but does not take into consideration semantic differences captured in the CPTs. In this chapter, we take a look at an axiomatic approach for defining distance functions for strict partial orderings.

## 4.1 An Axiomatic Approach

There has been a tremendous amount of work on comparing basic orderings, or rankings, of qualitative data. A well-known example of this work is that of Kemeny and Snell [15, 17]. Kemeny and Snell lay out a satisfiable set of distance function axioms for comparing weak orderings. There are several advantages to using this approach. First, the axioms define a solid foundation on which to base comparisons. Second, a representation for orderings is provided which conforms to the axioms. Finally, the axiomatic foundation is instrumental in forming theoretical properties within the context of interest-matching comparisons considered in this paper. Because of these benefits, we investigate the Kemeny-Snell axioms as a basis for our comparison measures.

In the context of preference orderings represented in CP-nets, we must generalize

the Kemeny-Snell axioms of weak orderings to be consistent under strict partial orderings (i.e., transitive, asymmetric, and irreflexive relations). The work of Bogart in [2] specifies slight modifications to the Kemeny-Snell axiomatic approach such that these modified axioms generalize the distance function to be consistent over strict partial orderings. These generalized axioms are discussed, and we will refer to these as the generalized Kemeny-Snell axioms.

There are a total of seven generalized Kemeny-Snell axioms that Bogart states are necessary for determining a distance function for comparing strict partial orderings. Let $\omega, \omega', \omega'' \in \Omega_F$ be three strict partial orderings. Let $d_\Omega : \Omega \times \Omega \rightarrow \Re$ be a distance function on strict partial orderings. We say that ordering $\omega'$ is between $\omega$ and $\omega''$, written as $B(\omega, \omega', \omega'')$, if:

1. $(i, j) \in \omega'$ if $(i, j) \in \omega$ and $(i, j) \in \omega''$
2. $(i, j) \in \omega$ or $(i, j) \in \omega''$ if $(i, j) \in \omega'$

Using the set-theoretic notation, we can state these conditions for $B(\omega, \omega', \omega'')$ succinctly by $\omega \cap \omega'' \subseteq \omega' \subseteq \omega \cup \omega''$ [2].

We state the generalized Kemeny-Snell axioms below with the axiom name used by Bogart in parentheses [2]. Axioms $1 - 3$ state the basic properties that all distance functions should satisfy.

**Axiom 1 (D1).** $d_\Omega(\omega, \omega') \geq 0$, *and equality holds iff* $\omega = \omega'$.

**Axiom 2 (D2).** $d_\Omega(\omega, \omega') = d_\Omega(\omega', \omega)$.

**Axiom 3 (D3).** $d_\Omega(\omega, \omega') + d_\Omega(\omega', \omega'') \geq d_\Omega(\omega, \omega'')$, *and equality holds iff* $B(\omega, \omega', \omega'')$.

The next axiom is referred to as the "change of mind" axiom. Basically, two agents that have completely opposite preference orderings should have a maximum distance between them. The complete opposite preference ordering of $\omega$ is denoted by $\omega^{-1} = \{(i, j) | (j, i) \in \omega\}$. Let $\emptyset$ denote the empty strict partial ordering (i.e., the strict partial ordering that makes no comparisons):

**Axiom 4 (C4) (change of mind).** $d_\Omega(\omega, \omega^{-1}) = d_\Omega(\omega, \emptyset) + d_\Omega(\emptyset, \omega^{-1})$

Let $\omega''$ be a preference ordering disjoint from the preference orderings $\omega$ and $\omega'$. If $\omega$ and $\omega'$ differ by the same $\omega''$, then the distances should be equal. This leads to axiom 5:

**Axiom 5 (T5).** *If $\omega''$ is a preference ordering disjoint from $\omega$ and $\omega'$ (i.e., $\omega \cap \omega'' = \emptyset = \omega' \cap \omega''$), then $d_\Omega(\omega, \omega \cup \omega'') = d_\Omega(\omega', \omega' \cup \omega'')$.*

The next axiom states that distance function is invariant with respect to permutations, or relabeling, of the set of outcomes in each ordering. A permutation function $\pi : O_F \to O_F$, as it is used here, is defined as a bijective mapping from a set of outcomes back onto itself. For any preference ordering $\omega$, define $\pi[\omega] = \{(\pi(i), \pi(j)) | (i, j) \in \omega\}$.

**Axiom 6 (P6).** *If $\pi : O_F \to O_F$ is a permutation function, then $d_\Omega(\omega, \omega') = d_\Omega(\pi[\omega], \pi[\omega'])$.*

The seventh and final axiom is simply for specifying a unit of measure.

**Axiom 7 (D7).** *The minimum positive distance is 1.*

Bogart shows that axioms $1 - 7$ are satisfiable by a distance function on the collection of strict partial orderings of a set of outcomes, or on some subset of this collection.

## 4.2 Representation for Comparisons

It is necessary to specify a representation for the orderings we intend to compare that conforms to the generalized Kemeny-Snell axioms in the context of strict partial orderings. First, we define a matrix representation for preference orderings [15, 17, 2, 3]. We call the matrix $M = [m_{ij}]$ an *ordering matrix* [15]. The notation for ordering matrix entries $m_{ij}$ is interpreted as $m_{column,row}$. For an ordering $\omega$, we construct a matrix $M = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 & \text{if } i \text{ is strictly preferred to } j, \\ -1 & \text{if } j \text{ is strictly preferred to } i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

A matrix constructed in this manner will have the following properties:

1. $m_{ii} = 0$ (Irreflexivity)
2. $m_{ij} = -m_{ji}$ (Symmetry)
3. If $m_{ij} \geq 0$ and $m_{jk} \geq 0$, then $m_{ik} \geq 0$; $m_{ik} = 0$ only if $m_{ij} = 0$ and $m_{jk} = 0$. (Transitivity)

Kemeny and Snell state that an *ordering matrix* is in canonical form if the labeling of the matrix columns and rows is consistent with the preference ordering. As an example, let $red \succ green \succ blue$ be an ordering such that *red* is preferred to *green* is preferred to *blue*. Using the definition of the *ordering matrix* for an ordering, we have

$$M = \begin{array}{c} \begin{matrix} \text{r} & \text{g} & \text{b} \end{matrix} \\ \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} \end{array}$$

where the columns, and rows, are labeled as $\langle red, green, blue \rangle$ [15].

Having stated the seven generalized Kemeny-Snell axioms and a satisfying representation for an ordering, we must now define formally the distance function. The existence of this distance function is stated in the following Bogart theorem of the generalized Kemeny-Snell axioms [2]:

**Generalized Kemeny-Snell Theorem 1 (existence theorem).** *There is at most one distance function on the collection of all strict partial orderings over a set of outcomes satisfying axioms* $1 - 7$.

The existence proof of the generalized Kemeny-Snell distance function is a case-based induction proof over the number of outcomes and the reader is referred to [2] for the proof. Let $M = [m_{ij}]$ and $M' = [m'_{ij}]$ be two ordering matrices representing the preference orderings $\omega$ and $\omega'$, respectively. The following Bogart theorem of the generalized Kemeny-Snell axioms defines the unique distance function $d_\Omega$:

**Generalized Kemeny-Snell Theorem 2 (uniqueness theorem).** *The function*

$$d_\Omega(\omega, \omega') = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} |m_{ij} - m'_{ij}| \tag{4.2}$$

*represents the unique distance function satisfying the generalized Kemeny-Snell axioms* $1-7$ *on the collection of all strict partial orderings over a set of outcomes.*

The uniqueness proof is omitted in this thesis and, once again, the reader is referred to [2] for the proof.

## 4.3 A Simple Comparison Example

We will walk through an example of a comparison using the generalized Kemeny-Snell distance function on two orderings. Let $red \succ_\omega green \succ_\omega blue$ and $blue \succ_{\omega'} red \succ_{\omega'} green$ be two complete orderings over the same set of objects $\{red, green, blue\}$. From these orderings, we construct the following *ordering matrices* using the generalized Kemeny-Snell representation. We rearrange the columns and rows of matrix $M'$ so that the columns and rows match the labeling $\langle red, green, blue\rangle$, as in matrix $M$. So, we have

$$M = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} \qquad M' = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

The reason for rearranging the matrix is that we want to be able to make comparisons between corresponding column vectors. If we iterate over the $i^{th}$ column vectors in each matrix, then we know that they correspond to the same object. This rearrangement provides a standard representation form between the two orderings, which more easily facilitates comparisons (see Section 6.2).

Using the distance function $d_\Omega$, we find that

$$d_\Omega(\omega, \omega') = \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3}|m_{ij} - m'_{ij}| = \frac{1}{2}(2+2+4) = 4 \tag{4.3}$$

In the next chapter, we consider basic properties of CP-nets when applied in a social multi-agent environment. Assumptions and restrictions are stated, as well as relations that can be placed between different CP-nets.

# Chapter 5

# CP-nets in a Multi-agent Environment

There has been increasing interest in CP-nets which has led to variations such as UCP-nets [4], which incorporate numerical utility into the CPTs, and TCP-nets [7], which take into account trade-offs by defining relative importance relations. This research on CP-nets, however, has explored minimally the basic properties of CP-nets in multi-agent environments [20]. In this chapter, we look at what assumptions should be made and what properties exist at a structural level between CP-nets in a social multi-agent environment.

## 5.1 Agent Preferences as CP-nets

The motivating factor for the interest-matching comparisons between preferences in CP-nets described in this thesis is the application of CP-nets to social multi-agent environments. Two critical elements of a multi-agent environment are described briefly using CP-nets: agent representation and methods for agent interaction [21].

We consider a multi-agent environment in which each agent has associated with it preferences represented as a CP-net. A multi-agent system often requires some formalism of what it is that an agent would like (or prefer) to do. Representing these preferences as a

CP-net allows us to take advantage of the formal semantics of the CP-net (assuming *ceteris paribus* semantics). Also, the number of outcomes over which *ceteris paribus* preferences are stated is exponential in the number of features. Thus, the compact CP-net representation is a practical option, in terms of space requirements, for representing these human-type preferences within a multi-agent environment.

Decision-theoretic frameworks are applied heavily in multi-agent systems [21]. These have traditionally used purely quantitative information about the agent's preferences and intentions. The introduction of qualitative decision theory (as opposed to traditional decision theory) has caused a marked increase in attempts to define typical agent behavior in qualitative terms [11, 23]. Using a CP-net to represent the preferences of an agent follows this trend.

Building on the influence of qualitative decision theory, we provide a method for computing interest-matching comparisons between the CP-nets of different agents. This method could be used when constructing a decision-theoretic framework in which the agents interact with each other in some manner. The mechanisms through which the agents collaborate, or coordinate, could take advantage of the comparison described in this paper to determine whether another agent has matching interests. The actions of one agent with a matching preference to another agent could be instrumental in helping achieve some goal, or task. The Markov model of a "reasonable man" is one such early attempt by Bogart at describing a decision-making process along these lines based on a weak theory of preference behavior [2]. The remainder of this chapter details some of the properties that these CP-nets should exhibit in a multi-agent environment.

## 5.2 Definition of Terms

We use subscript notation to define a CP-net for agent $i$ as $N_i \in \mathcal{N}_F$ with outcomes $O_F$. The CPTs in $N_i$ are denoted by $CPT_i(f)$ for each $f \in F$. We also denote the feature set for $N_i$ by $F_i$. We define $P(N_i)$ as the preference graph induced by $N_i$. The preference ordering $[\![N_i]\!]$ is denoted by $\omega_i$.

## 5.3 Restrictions and Structural Assumptions

The definition of a CP-net (see Section 3.2) allows for much more generality than is necessary for the types of comparisons considered in this thesis. For example, cycles are not prohibited in the general definition of a CP-net. However, the presence of cycles introduces problems with respect to transitivity and, hence, consistency. Another issue with the general definition is the potentially large domain over which features take on their specific values. There is absolutely nothing that requires the domain to be restricted to a certain number of possible values [6, 5].

We place the following restrictions on those CP-nets used throughout the remainder of this thesis:

1. All feature value assignments are over a binary domain (i.e., for each $f \in \mathbf{F}, |Dom(f)| = 2$)

2. All CP-nets are structured as a *directed acyclic graph* (DAG)

Features are restricted to having binary domains due to the simplicity of comparison measures in the context of interest matching. By restricting the domain of all features to a binary domain, we remove potential complications that can arise when generalizing a comparison measure to a mixed-domain environment. For example, a CP-net (in general) could have one feature with a binary domain and another with a ternary domain.

Most work on CP-nets has focused on acyclic structure, but some work has been carried-out on CP-nets with cycles [5]. There are some interesting theoretical properties to be noted in cyclic CP-nets, but these are generally restricted to certain problems that are not of interest to the area of application in this paper. Thus, the second restriction states that we ignore the issues with cyclic CP-nets by considering only those CP-nets structured as DAGs. By doing this, we ensure that all agents whose CP-net preferences are being compared will behave in a rational manner. Another benefit to this restriction has to do with the known and/or more efficient complexity results for specific types of these structures over more general (cyclic) CP-net structures [10].

## 5.4 Feature-based Case Analysis

The feature sets of two CP-nets divide into four cases. Let $N_i, N_j \in \mathcal{N}$ be our example CP-nets. The feature sets $F_i$ and $F_j$ can be used as the differentiating characteristic such that one of the following cases is satisfied:

1. $F_i = F_j$
2. $F_i \subset F_j$, or $F_j \subset F_i$
3. $|F_i - F_j| > 0$ and $|F_j - F_i| > 0$
4. $|F_i \cap F_j| = 0$

Case 1 is the simplest case, in which the feature sets are identical. In case 2, one feature set contains features not found in the other. Case 3 is for comparisons between two feature sets in which both have features not found in the other.

The fourth case is is satisfied when two CP-nets have no features in common. Since our intention is to compare two different CP-nets and deduce a level of shared interest between the two respective agents, we should have at least one common feature on which to base the comparison. Without this common feature, we are capable only of guessing whether the two agents share a common interest. Thus, if $|F_i \cap F_j| = 0$, then $d_\Omega(\omega_i, \omega_j) = d^\Omega_{max}$.

The nature of the comparison will vary under each of these cases (see Chapter 7). We continue in the next chapter by formalizing the interest-matching comparisons of preference orderings represented as CP-nets. It is shown how the generalized Kemeny-Snell axiomatic approach is applied to this problem.

# Chapter 6

# Comparing Preference Orderings Represented as CP-nets

We turn our attention to the application of the generalized Kemeny-Snell axiomatic approach for measuring distance between preference orderings represented through CP-nets.

The generalized Kemeny-Snell axioms do not require any changes to permit distance computations between preference orderings induced by CP-nets. However, we do need to make clear some of the terminology used in order to remove any ambiguity. We will use the generalized Kemeny-Snell distance function to define a shared interest function. This does not change in any way the properties of the distance function. The following section details an ordering matrix construction method for preference orderings over a set of outcomes that contains incomparable outcomes.

## 6.1 Preference Graph Based Ordering Matrix Construction

In this section, a method is described for constructing an ordering matrix from the preference graph induced by a CP-net, which captures all of the preference information in a strict partial ordering. A benefit to using the induced preference graph is that expansions

of feature sets can be conceptualized more easily using preference graphs. The details of these expansions are discussed in Section 6.6.

Due to the flipping sequence semantics of a preference graph $P(N)$, each edge $e = (o_i, o_j)$ from outcome $o_i$ to outcome $o_j$ in $P(N)$ is interpreted as meaning $o_i \succ o_j$. Recalling Lemma 3.4.1, if we have two edges $e_1 = (o_i, o_j)$ and $e_2 = (o_j, o_k)$, then the transitivity of preference tells us that $o_i \succ o_k$. The path that results from these two edges is defined as a worsening flipping sequence.

We restate equivalently the definition of the generalized Kemeny-Snell ordering matrix representation $M = [m_{ij}]$ over an induced preference graph $P(N)$ using graph-theoretic terminology. We denote a path in $P(N)$ from outcome $o_i$ to outcome $o_j$ as $p_{ij} \in P(N)$:

$$m_{ij} = \begin{cases} 1 & \text{if } p_{ij} \in P(N), \\ -1 & \text{if } p_{ji} \in P(N), \\ 0 & \text{if } p_{ij} \notin P(N) \text{ and } p_{ji} \notin P(N). \end{cases} \tag{6.1}$$

## 6.2 Canonical Form for CP-net Matrix Representation

In the example from Section 4.3, we rearranged the ordering matrix for one of the preference orderings so that the distance function could be computed. However, due to incomparable outcomes, we cannot use simply the preference orderings for the canonical form of the ordering matrices as is done by Kemeny and Snell [15]. In this section, we formalize a canonical form of an ordering matrix which is used to standardize the relation between two ordering matrices of which we intend to compute shared interest.

For some feature $A \in \mathbf{F}$ such that $Dom(A) = \{a, \bar{a}\}$, we say that $a$ is the positive assignment value and $\bar{a}$ is the negative assignment value. For any given feature value assignment, let 0 be the base two numeric interpretation of the negative assignment to that feature and 1 be the base two numeric interpretation of the positive assignment to that feature. This base two numeric interpretation is denoted by the subscript $\sharp$, as in $=_\sharp$ and $<_\sharp$. Using this interpretation for a feature $A$ where $Dom(A) = \{a, \bar{a}\}$, we have $a =_\sharp 1$ and $\bar{a} =_\sharp 0$.

Assume that the outcomes $o_1, o_2 \in O_F$ are in the standard format specified in Section 5.2. A *lexicographic ordering* of two n-tuple outcomes $o_1$ and $o_2$ is defined such that $o_1$ occurs before $o_2$ if and only if the base two numeric interpretation of $o_1$ is less

than that of $o_2$. In other words, $o_1$ occurs before $o_2$, provided that in the first position $k$ (left-to-right) in which they disagree, we have $f_k =_\sharp 0$ in $o_1$ and $f_k =_\sharp 1$ in $o_2$ [8]. For example, if $o = \langle a\bar{b}c \rangle$, then we can interpret this as $o =_\sharp \langle 101 \rangle$, where entry one (left-to-right) corresponds to positive feature value $a \in Dom(A)$, entry two corresponds to negative feature value $\bar{b} \in Dom(B)$, and entry three corresponds to positive feature value $c \in Dom(C)$.

Assume that we have a feature set $F = \{A, B, C\}$. Using the base two numeric interpretation we have described, let $o_1 = \langle a\bar{b}\bar{c} \rangle =_\sharp \langle 100 \rangle$, $o_2 = \langle abc \rangle =_\sharp \langle 111 \rangle$, and $o_3 = \langle \bar{a}\bar{b}c \rangle =_\sharp \langle 001 \rangle$. We say that $\langle o_3, o_1, o_2 \rangle$ is a lexicographic ordering of the three outcomes under the interpretation we have described since $o_3 <_\sharp o_1 <_\sharp o_2$.

**Definition** An ordering matrix is in *canonical form* if and only if the columns, and rows, are labeled as outcomes such that the ordering of these labels satisfies the lexicographic ordering over the outcomes.

An example of an ordering matrix in this canonical form, as described, is straightforward. Let $F = \{A, B\}$ be our feature set and $O_F = \{ab, a\bar{b}, \bar{a}\bar{b}, \bar{a}b\}$ be our outcome set. From the base two numeric interpretation of the feature values, we can state that $\langle \bar{a}\bar{b}, \bar{a}b, a\bar{b}, ab \rangle =_\sharp \langle 00, 01, 10, 11 \rangle$ is a lexicographic ordering of the outcomes in $O_F$. This lexicographic ordering is used in our ordering matrix construction to produce an ordering matrix with the following labels:

$$\begin{array}{c} \\ \bar{a}\bar{b} \\ \bar{a}b \\ a\bar{b} \\ ab \end{array} \begin{array}{c} \begin{array}{cccc} \bar{a}\bar{b} & \bar{a}b & a\bar{b} & ab \end{array} \\ \begin{bmatrix} m_{11} & \dots & & m_{41} \\ \vdots & \ddots & & \vdots \\ & & & \\ m_{14} & \dots & & m_{44} \end{bmatrix} \end{array}$$

If we construct all of our ordering matrices using the canonical form defined, then we ensure that all entries $m_{ij}$ from two ordering matrices (using identical feature sets) will correspond to the same outcomes $o_i$ and $o_j$. The outcome format established by the enumerated universe of features also ensures that adding, removing, or otherwise modifying the CP-net will produce predictable results in the ordering matrix labeling. This point is especially important when feature sets are expanded to allow comparisons, as in cases 2 and 3 mentioned in Section 5.4. The specifics of these expansions are discussed in Section 6.6 and in Chapter 7.

## 6.3 Shared Interest Function

So far in this paper, we have discussed comparison measures in terms of a distance function. For this paper, we wish instead to refer to shared interest as a similarity measure. This serves two purposes: our nomenclature is consistent with the proposed application area, and shared interest provides a normalized measure of the similarity information. A normalized shared interest function is important when comparing the shared interest levels of CP-nets of varying sizes. We will now describe how the shared interest function is constructed.

Consider the distance function for two preference orderings $\omega, \omega' \in \Omega_F$ uniquely defined by the generalized Kemeny-Snell axioms where $n = |O_F|$:

$$d_\Omega(\omega, \omega') = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} |m_{ij} - m'_{ij}| \tag{6.2}$$

Let $\overrightarrow{u_i}$,$\overrightarrow{v_i}$ be the $i^{th}$ column vectors from the ordering matrix representation of $\omega$ and $\omega'$, respectively. These column vectors represent the preferences stated in the respective preference ordering for outcome $o_i$ given each $o_j \in O_F$. Now, define

$$\delta(\overrightarrow{u_i}, \overrightarrow{v_i}) = \sum_{j=1}^{n} |m_{ij} - m'_{ij}| \tag{6.3}$$

This function represents the inner summation of $d_\Omega$. So,

$$d_\Omega(\omega, \omega') = \frac{1}{2} \sum_{i=1}^{n} \delta(\overrightarrow{u_i}, \overrightarrow{v_i}) \tag{6.4}$$

Recalling that we have a ternary matrix representation (i.e., $m_{ij} \in \{-1, 0, 1\}$), we can state the following about $|m_{ij} - m'_{ij}|$:

$$|m_{ij} - m'_{ij}| = \begin{cases} 0 & \text{if } m_{ij} = m'_{ij}, \\ 1 & \text{if } m_{ij} = 0 \text{ and } m'_{ij} \neq 0, \text{ or } m'_{ij} = 0 \text{ and } m_{ij} \neq 0, \\ 2 & \text{if } m_{ij} = -m'_{ij}. \end{cases} \tag{6.5}$$

That is, the ordering matrix entry difference equals 0 if the agents agree on the ordering for outcomes $o_i$ and $o_j$, 1 if exactly one of the agents is unable to state a preference between $o_i$ and $o_j$, and 2 if the agents place an opposing preference between $o_i$ and $o_j$. We note that the maximum value of $|m_{ij} - m'_{ij}|$ is 2, and this occurs whenever $m_{ij} = -m'_{ij}$.

Since our ordering matrix representation requires that $m_{ii} = 0$ for all $i$, we have a maximum of $n-1$ non-zero entries in each of our column vectors. So, by setting $|m_{ij} - m'_{ij}| = 2$ and substracting 2 we get a maximum value $\delta^n_{max}$ for any of the $\delta$ values by:

$$\delta^n_{max} \leq \sum_{j=1}^{n} 2 - 2 = 2n - 2 = 2(n-1) \tag{6.6}$$

The inequality is due to the presence of incomparable outcomes. The equality holds true if and only if all outcomes are comparable. Substituting $\delta^n_{max} = 2(n-1)$ into $d_\Omega$ defines our maximum distance:

$$d^\Omega_{max} = \frac{1}{2}\sum_{i=1}^{n} \delta^n_{max} = \sum_{i=1}^{n}(n-1) = n(n-1) \tag{6.7}$$

This gives us the following definition:

**Definition** $d^\Omega_{max}$ is the *maximum distance* achievable between two preference orderings in $\Omega$ over binary features.

Let $I : \mathcal{N} \times \mathcal{N} \rightarrow [0,1]$ be our shared interest function between two CP-nets. We want $I$ to be strategically equivalent to $d_\Omega$, and normalized [12]. Furthermore, we want the evaluation of our shared interest function to satisfy $I_{min} = 0$ and $I_{max} = 1$. Since $d_\Omega(\omega, \omega') \leq d^\Omega_{max}$, we know that $0 \leq \frac{d_\Omega(\omega,\omega')}{d^\Omega_{max}} \leq 1$. So, we define our shared interest function $I$ as:

$$I(N, N') = 1 - \frac{d_\Omega(\omega, \omega')}{d^\Omega_{max}} \tag{6.8}$$

Thus, $0 \leq I(N, N') \leq 1$, $I_{min} = 0$, and $I_{max} = 1$. This leads us to the following definitions of shared interest between two agents whose preferences are structured as a CP-net.

**Definition** Agents $i$ and $j$ share a *common interest* in each other iff $0 < I(N_i, N_j) \leq 1$.

**Definition** Agents $i$ and $j$ share a *maximum interest* in each other iff $I(N_i, N_j) = I_{max} = 1$.

**Definition** Agents $i$ and $j$ share *no interest* in each other iff $I(N_i, N_j) = I_{min} = 0$.

A maximal shared interest will occur when the two fully-annotated CP-nets are identical. Similarly, no shared interest will exist when the two fully-annotated CP-nets share nothing in common.

There are several benefits to using the normalized shared interest function which

has been described. Many attempts have been made at formalizing an interest-matching (or match-making) system, but most suffer in one way or another from shortcomings, as pointed out in [9]. The most serious of these shortcomings is the lack of a natural ranking in interest-matching systems. The normalized shared interest function we have described allows rankings of shared interest levels between any number of agents.

Many systems are able to determine that two agents are mutually interested in each other (a match) based on only exact matches between the two [9]. If there are any differences between the agents then it is assumed that they can not be a good match. This problem has been addressed through finding potential and partial matches, however, it is often not possible to rank the shared interest levels (or matches) of one agent to several other agents. The interest-matching comparison presented in this thesis allows for rankings of the shared interest levels. Even if the preferences of one agent are specified over a much larger feature set, the normalized shared interest function value will still map to $[0, 1]$.

## 6.4 Computing Shared Interest using CP-nets

We have explained how the generalized Kemeny-Snell is applied to preference orderings represented in CP-nets. The representation on which the generalized Kemeny-Snell distance function operates is applied now in the context of interest-matching comparisons of CP-nets.

**Example 1.** Let $N_1, N_2 \in \mathcal{N}_F$ be two CP-nets between which we intend to measure shared interest. For simplicity, we have defined $N_1$ and $N_2$ over the same feature set $F = \{A, B, C\}$. That is, assume case 1 from Section 5.4 is satisfied. The two CP-nets are shown in Figure 6.1 annotated with their CPTs.

From the CPTs these two CP-nets we construct the respective preference orderings, $\omega_1$ and $\omega_2$. Just as with the updated Kemeny-Snell matrix construction from Section 6.1, we construct the *ordering matrices* using paths from the induced preference graphs. The columns, and rows, of $M_1$ and $M_2$ are labeled in *canonical form* such that the column and row labels are ordered as in $\langle \bar{a}\bar{b}\bar{c}, \bar{a}\bar{b}c, \bar{a}b\bar{c}, \bar{a}bc, a\bar{b}\bar{c}, a\bar{b}c, ab\bar{c}, abc \rangle$. The two ordering matrices

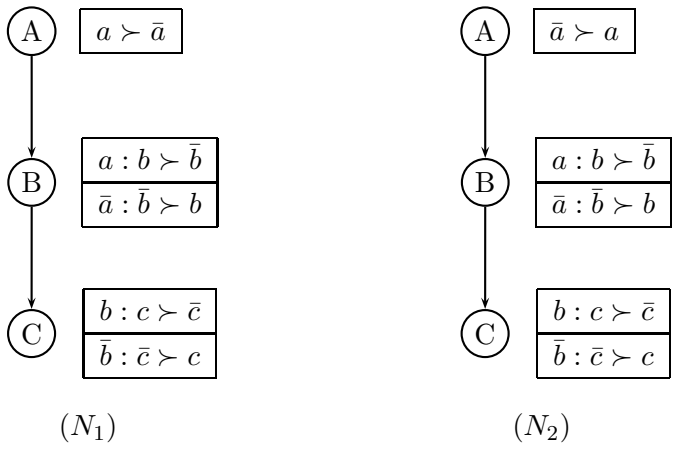


Figure 6.1: Two CP-nets $N_1$ and $N_2$

$M_1$ and $M_2$ are shown below:

$$M_1 = \begin{bmatrix} 0 & -1 & -1 & -1 & 1 & 0 & 1 & 1 \\ 1 & 0 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 0 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 0 & -1 & 1 & 1 \\ 0 & -1 & -1 & -1 & 1 & 0 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 0 & 1 & -1 & -1 & -1 & 0 \\ 1 & 1 & -1 & 0 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 0 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & 0 & 1 \\ 1 & 1 & 0 & 1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

Now, the $i^{th}$ column vectors for $M_1$ and $M_2$ have corresponding labels. Once this has been done, the distance function defined earlier can be used to measure the sum of the distances between each of the column vectors. In effect, the similarity of preference for the outcome specified by the column label is being compared.

We use the distance function $d_\Omega$ to sum over the distances of the preference relations placed over each outcome, and divide by 2 to get the final distance measure. For our example CP-nets $N_1$ and $N_2$, we find that $d_\Omega(\omega_1, \omega_2) = 30$. This measure can be used now to compute the normalized shared interest function value. We note that $d^\Omega_{max} = n(n-1) = 56$, where $n = |O_F| = 8$. We put this in our shared interest function $I$ to get

$$I(N_1, N_2) = 1 - \frac{d_\Omega(\omega_1, \omega_2)}{d^\Omega_{max}} = 1 - \frac{30}{56} = 0.464 \tag{6.9}$$

The example above is intended to provide an overview of the basic techniques being used to make comparisons between the preference relations represented as CP-nets. This is not meant to be a complete outline for the general problem of making interest matching comparisons of CP-nets. Instead, we focused on a well-formed, ideal case which allowed us to demonstrate these techniques and their basic properties without having to deal with the problems associated with other cases. More general cases are discussed in Chapter 7.

## 6.5 CP-net Shared Interest Algorithm

In this section we state formally the CP-net Shared Interest Algorithm which is used to compare two CP-nets and return a value indicating a level of shared interest. In Example 1, we showed how the shared interest between two well-formed CP-nets is computed. The algorithm is generalized in this section to account for all possible cases of CP-net comparisons outlined in Section 5.4.

The shared interest function takes two CP-nets $N \in \mathcal{N}_F$ and $N' \in \mathcal{N}_{F'}$ as input. The restrictions stated in Section 5.3 are assumed to be true of the CP-nets. Using the CPTs, we construct the induced preference graph for each CP-net. Next, we check whether the feature sets are the same for both CP-nets. Of course, it is possible that each set contains features not found in the other. If it is determined that the feature set $F'$ contains features not found in $F$, then we specify these features in the set $V = F' - F$. There are $2^{|V|}$

possible combinations of binary feature values (instantiations) over the set $V$. We define the set $O_V$, which contains all of these combinations. The preference graph $P(N)$ for CP-net $N$ is extended so that it contains $2^{|V|}$ disconnected graphs. Each of these disconnected graphs, which we call $P_{u_k}(N)$, is defined precisely as the original preference graph over $O_F$ except that each outcome $o_j \in O_F$ is now of the form $o_j u_k$, where $u_k \in O_V$ and $u_k$ is unique for each disconnected graph $P_{u_k}(N)$. This same procedure is carried out if it is also determined that $|F - F'| \geq 1$.

The construction of the disconnected preference graphs is both a necessary and sound modification of the original preference graph in order to preserve the flipping sequence semantics of the preference ordering it represents. Each of the disconnected subgraphs, $P_{u_k}(N)$, maintains the preference ordering over the original set of outcomes. By concatenating the same, unique instantiation of the differing features, the flipping sequence remains legally defined. Furthermore, by removing the concatenated feature instantiations from outcomes in $P_{u_k}(N)$ we have preserved the original, unexpanded preference graph for $N$. The preservation of the stated preferences when constructing the expanded preference graph allows reliable comparisons to be made between significantly different CP-nets.

Once the preference graphs have been constructed properly, the ordering matrices are constructed using the path-based method over the preference graphs as mentioned in Section 6.1. The distance function $d_\Omega$ is computed using these ordering matrices. The distance value along with the maximum possible distance, $d^\Omega_{max}$, are used to compute the shared interest function, $I(N, N') = 1 - \frac{d_\Omega(\omega,\omega')}{d^\Omega_{max}}$. This shared interest value is returned.

**Algorithm 1** CP-net Shared Interest Algorithm

**Input:** CP-nets $N$ and $N'$

**Output:** Shared interest level between $N$ and $N'$

Construct induced preference graphs $P(N)$ and $P(N')$

**if** $|F' - F| > 0$ **then**

    Let $V = F' - F$

    Modify $P(N)$ to contain $2^{|V|}$ disconnected preference graphs, call them $P_{u_k}(N)$, such that each $P_{u_k}(N)$ is constructed over outcomes of the form $o_j u_k$ for each $o_j \in O_F$, where $u_k \in O_V$ and $u_k$ is unique for each of the disconnected preference graphs.

**end if**

**if** $|F - F'| > 0$ **then**

    Define $V = F - F'$

    Modify $P(N')$ to contain $2^{|V|}$ disconnected preference graphs, call them $P_{u_k}(N')$, such that each $P_{u_k}(N')$ is constructed over outcomes of the form $o_j u_k$ for each $o_j \in O_{F'}$, where $u_k \in O_V$ and $u_k$ is unique for each of the disconnected preference graphs.

**end if**

Using $P(N)$ and $P(N')$, construct ordering matrices $M$ and $M'$ in canonical form

Compute $d_\Omega(\omega, \omega')$ using $M$ and $M'$

Define $d^\Omega_{max}$ using $n = |O_{F \cup F'}|$

**Return:** $I(N, N') = 1 - \frac{d_\Omega(\omega, \omega')}{d^\Omega_{max}}$

# Chapter 7

# CP-net Variations

The CP-nets used for comparisons will vary in topological structure or in CPT specifications. In this chapter, we provide examples of some of these variations and discuss how they influence the computation of shared interest level. An example is provided for each of these properties along with an inline explanation of the results. It should be noted that none of these variations require changes to the shared interest function. The discussion of examples in this chapter provides the empirical evidence for motivating the structural comparisons discussed in the next chapter.

## 7.1 Defining CP-net Permutations

We start by defining a simple method for permuting CP-nets through modifications to the CPTs, which we call a *CP-permutation*. The construction of a *CP-permutation* function is described below.

Let $N \in \mathcal{N}_F$ be a fully-annotated CP-net. Let $Pref(f) = \{f \succ \bar{f}, \bar{f} \succ f\}$ be the set of all possible preferences for feature $f$. As a step toward defining our *CP-permutation*, we define a method for modifying CPTs in a CP-net. This is done using a flipping function on a CPT row:

**Definition** A *flipping function* is the bijective function $\sigma_f : Pref(f) \rightarrow Pref(f)$ which maps the set of all possible preferences for $f$ back onto itself such that $\sigma_f(\alpha) \neq \alpha$ for each


a flipping function (*a*) not a flipping function (*b*)

Figure 7.1: Example flipping function

| $a : \sigma_B(b \succ \bar{b})$ |
|---|
| $\bar{a} : \bar{b} \succ b$ |

$=$

| $a : \bar{b} \succ b$ |
|---|
| $\bar{a} : \bar{b} \succ b$ |

Figure 7.2: An atomic flipping of one row of CPT(B)

$\alpha \in$ *Pref*$(f)$.

Thus, in Figure 7.1 the flipping function $(a)$ is a flipping function using the definition provided above; however, $(b)$ is not a flipping function. We use the flipping function to define a modification to a single row in a CPT, as well as a modification to the entire CPT. These modifications are called an *atomic flip* and a *CPT flip*, respectively:

**Definition** An *atomic flip*, or *atomic flipping*, is the application of the flipping function $\sigma_f$ to a single row in CPT$(f)$ for some $f \in F$.

Let $N \in \mathcal{N}_F$ be a CP-net, with $F = \{A, B\}$, such that $A \in Pa(B)$ and *Pref*$(B) = \{b \succ \bar{b}, \bar{b} \succ b\}$. Let $\sigma_B$ be the flipping function defined for feature $B$. We say that an atomic flipping $\sigma_B$ has been applied to a row in CPT$(B)$ when we modify, for example, the row $a : b \succ \bar{b}$ to $a : \bar{b} \succ b$. This modification of CPT$(B)$ is depicted in Figure 7.2.

**Definition** A *CPT flip*, or *CPT flipping*, is the modification of each row in CPT$(f)$ according to the flipping function $\sigma_f$. The function $\pi_f : \mathcal{N}_{\mathbf{F}} \rightarrow \mathcal{N}_{\mathbf{F}}$ applies a single *CPT flip* to CPT$(f)$ in a CP-net.

Building on the example of an *atomic flip*, we say that rightmost CPT in Figure 7.3 results from a *CPT flip*, $\pi_B$, of the leftmost CPT$(B)$ since all rows have been modified

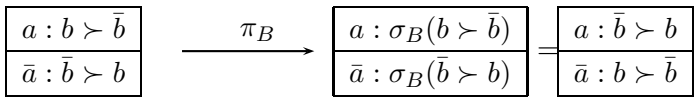


Figure 7.3: CPT-flipping function $\pi_B$

according to $\sigma_B$.

It is simple to note that the CPT flipping function $\pi_f$ can be composed with other CPT flipping functions and applied to the same CP-net. Define a *CP-permutation* function $\Pi = \pi_{f_i} \circ \ldots \circ \pi_{f_m}$ as the composition of CPT flipping functions. If the restriction of $\Pi$ to feature $f$, denoted by $\Pi \restriction f$, is such that $\Pi \restriction f = \pi_f$, then we say that CPT($f$) is flipped in $\Pi$. Furthermore, if CPT($f$) is not flipped in $\Pi$, then $\Pi \restriction f = id$, where $id$ denotes the identity function, meaning that CPT($f$) is unaffected by $\Pi$. A *CP-permutation* function $\Pi$ and CPT flipping function $\pi_f$ are the same operationally when we have $\Pi = \pi_f$ (i.e., $\Pi$ is composed of only one CPT flipping function).

**Definition** A *CP-permutation* function $\Pi = \pi_{f_i} \circ \ldots \circ \pi_{f_m}$ is the composition of *CPT flipping* functions $\pi_f$.

A simple identity property $\Pi(\Pi(N)) = N$ holds for the function $\Pi$. The identity definition follows intuition due to the nature of $\sigma_f$. By definition, the application of $\Pi$ to a CP-net $N$ will result in a CP-net $N'$. Applying $\Pi$ once again for $N'$ simply flips the CPTs back to the original CP-net $N$.

## 7.2 Relation between Shared Interest Values

The following example will expand on Example 1 by considering the relation of one shared interest measure to another and discuss what is the relation to feature importance. A level of relative feature importance is assigned through the explicit relation of the feature nodes in the graph by interpreting $f_j \in Pa(f_i)$ as meaning that feature $f_j$ is more important to the user than $f_i$. The purpose is to motivate what we mean when we say that an agent whose preferences are represented in a CP-net shares a higher level of interest with another

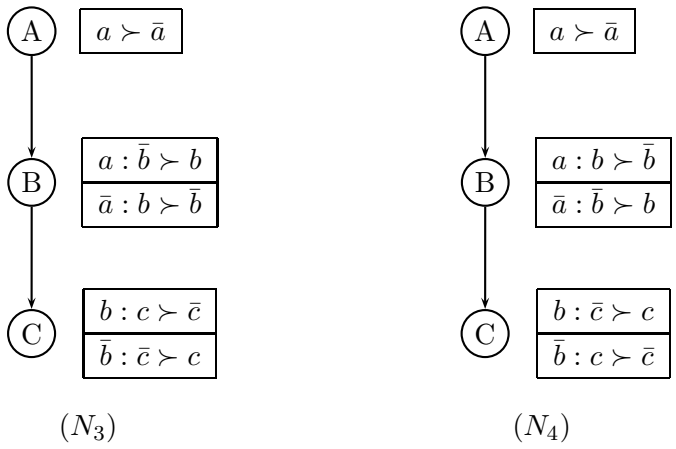


Figure 7.4: Two CP-nets $N_3$ and $N_4$

agent than some other agents. We continue to use the CP-nets $N_1$ and $N_2$ from Example 1.

**Example 2.** Let $N_3, N_4 \in \mathcal{N}_F$ be two CP-nets over the same feature set $F = \{A, B, C\}$ from Example 1. These CP-nets are shown in Figure 7.4. We want to compare $N_1$ to each of $N_2$, $N_3$, and $N_4$. We have applied CPT flipping functions to $N_1$ such that $N_2 = \pi_A(N_1)$, $N_3 = \pi_B(N_1)$, and $N_4 = \pi_C(N_1)$. Thus, we have CPT-flipped the preference for the most important feature in $N_2$, second most important feature in $N_3$, and the least important feature in $N_4$, where relative feature importance over $\{A, B, C\}$ is the same for all of the CP-nets. Intuition tells us that two agents that have opposing preferences on a feature that is most important to each of them will have less shared interest in each other than two agents that disagree only in a less important feature. We will now test this intuition on our example problem with the four CP-nets.

The induced preference graphs $P(N_3)$ and $P(N_4)$, and the preference orderings $\omega_3$ and $\omega_4$, are constructed from the CPTs of each CP-net. From these preference graphs, we

construct the respective ordering matrices, $M_3$ and $M_4$, in canonical form:

$$M_3 = \begin{bmatrix} 0 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 1 & 1 & 0 & 1 \\ -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & 0 & -1 & -1 \\ -1 & -1 & -1 & 0 & 1 & 1 & 0 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & -1 & 0 \end{bmatrix}$$

$$M_4 = \begin{bmatrix} 0 & 1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & 0 & -1 & -1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & -1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ -1 & 0 & -1 & -1 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 1 & 0 \end{bmatrix}$$

Using the generalized Kemeny-Snell distance function, we find that $d_\Omega(\omega_1, \omega_2) = 30$, $d_\Omega(\omega_1, \omega_3) = 18$, and $d_\Omega(\omega_1, \omega_4) = 10$. The shared interest function $I$ is computed using $d^\Omega_{max} = n(n-1) = 56$, where $n = |O_F| = 8$.

$$I(N_1, N_2) = 1 - \frac{d_\Omega(\omega_1, \omega_2)}{d^\Omega_{max}} = 1 - \frac{30}{56} = 0.464 \tag{7.1}$$

$$I(N_1, N_3) = 1 - \frac{d_\Omega(\omega_1, \omega_3)}{d^\Omega_{max}} = 1 - \frac{18}{56} = 0.678 \tag{7.2}$$

$$I(N_1, N_4) = 1 - \frac{d_\Omega(\omega_1, \omega_4)}{d^\Omega_{max}} = 1 - \frac{10}{56} = 0.821 \tag{7.3}$$

Returning to our initial statement about the expected results, we can see that it is indeed consistent with our intuition since:

$$I(N_1, N_2) < I(N_1, N_3) < I(N_1, N_4) \tag{7.4}$$

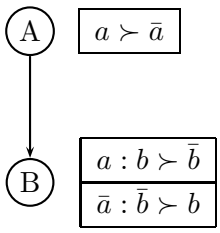


Figure 7.5: CP-net $N_5$

Thus, we conclude that agent 1 would be most interested in agent 4, given also the preferences of agent 2 and agent 3. Note that this does not mean that agent 1 is not at all interested in the other two agents. The usefulness of these shared interest values of agent 1 with agents 2 and 3 would depend on the specific application and implementation details.

## 7.3 Feature Subset

We turn our attention now to case 2 mentioned in Section 5.4. An example is provided, along with discussion of the method used.

**Example 3.** Consider CP-net $N_5$ from Figure 7.5, where $F_5 = \{A, B\}$. We want to compare $N_5$ to $N_2$ used in Examples 1 and 2. This comparison satisfies the condition for case 2 from Section 5.4 since $F_5 \subset F_2$.

From the CPTs of $N_5$, we construct the induced preference graph shown in Figure 7.6. The preference graph represents the following preference ordering, which contains no incomparable outcomes:

$$ab \succ_{\omega_5} a\bar{b} \succ_{\omega_5} \bar{a}\bar{b} \succ_{\omega_5} \bar{a}b \tag{7.5}$$

Since the feature sets are not the same, we cannot make the comparisons using only the methods presented earlier. We specify a method for expanding $P(N_5)$ to include those features in $V = \{C\} = F_2 - F_5$. Since $N_5$ does not make any preference statement about those features not in $F_5$, $P(N_5)$ is modified by constructing $2^{|V|}$ disconnected preference graphs, $P_u(N_5)$, such that for all $o_j^5 \in O_{F_5}$ we have one subgraph for $o_j^5 c$ and another subgraph for $o_j^5 \bar{c}$, where $u \in \{c, \bar{c}\} = O_V$. We distinguish the expanded preference ordering from the original preference ordering by using $\succ^e$.

The agent's preferences do not change over the original outcomes when the ex-

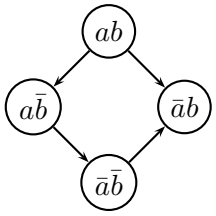


Figure 7.6: Induced preference graph $P(N_5)$

panded feature value assignments are concatenated. This is because the value assignment for the new feature(s) is the same over the entire preference ordering. The semantics from the CP-net of interpreting a preference ordering as flipping sequence remains true with the expanded preference orderings.

It is important to note that the preference ordering represented in an expanded preference graph is not defined in an agent's CP-net. A CP-net contains those features over which an agent has placed a preference. Since the agent has not stated preferences for the features in the expanded feature set, we can not define the preferences in the expanded preference graph through the CP-net. Essentially, the expanded preference graph represents the original (unexpanded) preferences of an agent in the presence of other features.

These incomparable orderings, which are defined through the disconnected preference graphs and not the actual CP-net $N_5$, are:

$$abc \succ^{e}_{\omega_5} a\bar{b}c \succ^{e}_{\omega_5} \bar{a}\bar{b}c \succ^{e}_{\omega_5} \bar{a}bc \tag{7.6}$$

$$ab\bar{c} \succ^{e}_{\omega_5} a\bar{b}\bar{c} \succ^{e}_{\omega_5} \bar{a}\bar{b}\bar{c} \succ^{e}_{\omega_5} \bar{a}b\bar{c} \tag{7.7}$$

In the general case, we would construct $2^{|V|}$ disjoint preference graphs for $N_i$ when comparing $N_i$ to $N_j$, where $F_i \subset F_j$ and $V = F_i - F_j$. The CP-net flipping semantics forces us to state the preferences over $F_5 \cup \{C\}$ as disjoint orderings for $c$ and $\bar{c}$ since no preference has been stated explicitly in the CP-net $N_5$ for feature $C$. Since the value assignment for feature $C$ is the same and held constant for each of the orderings, we maintain the integrity of our original preference orderings $\omega_5$.

Now that we have shown how to expand our preference ordering to include those features over which a preference is not stated, the ordering matrix is constructed so that the distance function can be applied and shared interest comparisons computed. Recall that the construction of the ordering matrix specifies that an entry $m_{ij} = 0$ if $i$ and $j$ represent

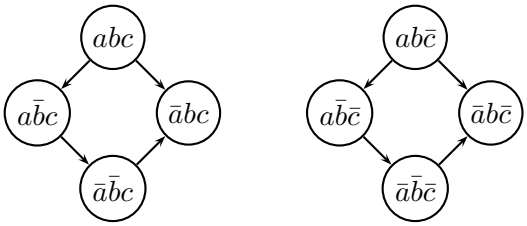


Figure 7.7: Expanded (disconnected) preference graph from $P(N_5)$

incomparable outcomes. This has been restated in Section 6.1 in terms of path-based connectivity. The use of paths to define preference and incomparability from a preference graph carries over naturally when discussing disconnected preference graphs.

The ordering matrix $M_5'$ is constructed in canonical form over the outcomes defined through the expanded feature set $F_5 \cup \{C\}$.

$$M_5' = \begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ -1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Having constructed the ordering matrix using the disconnected preference graph we are able to make the comparison between $M_2$ and $M_5'$ in the standard manner by computing the shared interest function $I$ using the distance function $d_\Omega$ over $M_2$ and $M_5'$. Once again, the shared interest function $I$ is computed using $d^\Omega_{max} = n(n-1) = 56$, where $n = |O_{F_5 \cup F_2}| = 8$.

$$I(N_2, N_5) = 1 - \frac{d_\Omega(\omega_2, \omega_5)}{d^\Omega_{max}} = 1 - \frac{31}{56} = 0.446 \tag{7.8}$$

This example has detailed the steps for forming comparisons between two CP-nets satisfying case 2 from Section 5.4. The general form of these steps has been provided in

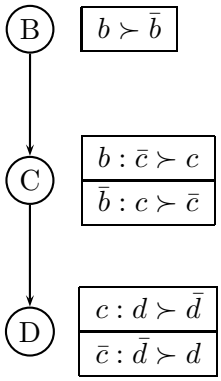


Figure 7.8: CP-net $N_6$

this example. These can now be applied to any case 2 interest-matching problem, and they can easily be extended to case 3 comparisons.

## 7.4 Variations in both Feature Sets

In the previous section, we looked at how comparisons are made between two CP-nets where one feature set is a subset of the other. It is unsafe to make the assumption that we will always have two feature sets such that $F_i \subseteq F_j$. Thus, we discuss case 3 from Section 5.4 and show how computing the shared interest level between two CP-nets of this form is a simple extension from those satisfying case 2.

**Example 4.** Let $N_5 \in \mathcal{N}_{F_5}$ and $N_6 \in \mathcal{N}_{F_6}$ be two CP-nets such that $|F_6 - F_5| > 0$ and $|F_5 - F_6| > 0$. That is, each of the two feature sets has at least one feature not in the other. The CP-net $N_5$ is the same as in Figure 7.5 and $N_6$ is shown in Figure 7.8.

We use the CPTs of each CP-net to construct the preference graph. The preference graph for $N_5$ is shown in Figure 7.6 and for $N_6$ in Figure 7.9.

Since the feature sets for $N_5$ and $N_6$ satisfy $|F_6 - F_5| > 0$ and $|F_5 - F_6| > 0$, both preference graphs will need to be expanded. We consider first the expanded preference graph for $N_5$. Just as in Example 3, we define the set of features that do not appear in $F_5$. Let $V_5 = F_6 - F_5 = \{C, D\}$. So, we will need to expand $P(N_5)$ to $2^{|V_5|}$ disconnected graphs; one for each $u \in O_{V_5} = \{cd, \bar{c}\bar{d}, \bar{c}d, c\bar{d}\}$. This expanded preference graph, with four disconnected subgraphs, is shown in Figure 7.10.

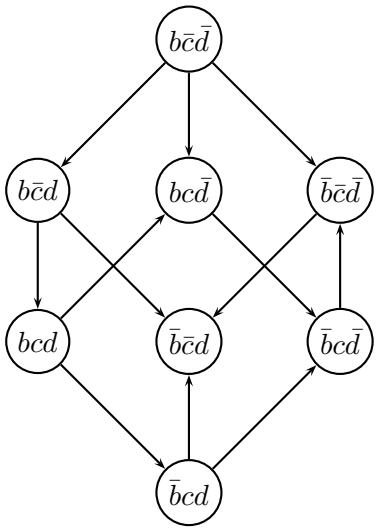


Figure 7.9: Induced preference graph for $N_6$

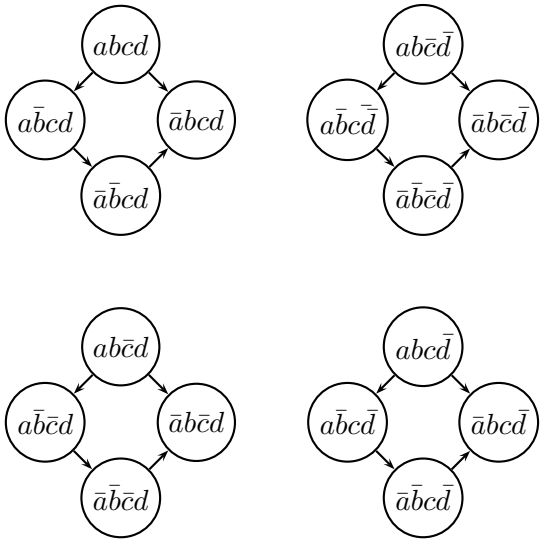


Figure 7.10: Another expanded (disconnected) preference graph from $P(N_5)$

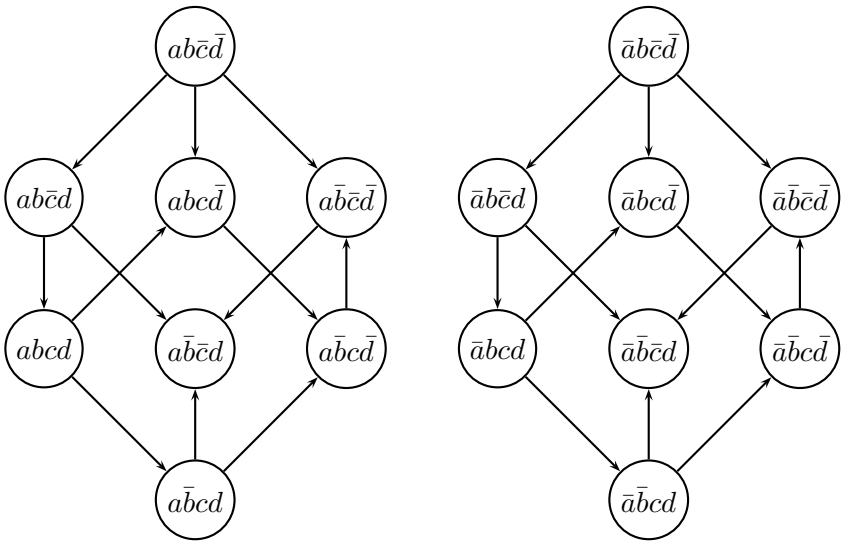


Figure 7.11: Expanded (disconnected) preference graph for $N_6$

The preference graph for $N_6$ is expanded in a similar manner as with $N_5$. Let $V_6 = F_5 - F_6 = \{A\}$ be the set of features not in $F_6$. We expand $P(N_6)$ to two disconnected subgraphs. The expanded preference graph is shown in Figure 7.11.

Now that we have constructed both expanded preference graphs over the same set of outcomes $O_{F_5 \cup F_6}$, we are able to construct the ordering matrix for each in canonical form. The computation of the shared interest level between $N_5$ and $N_6$ is carried out in the same manner as in all other cases from this point forward. With $n = |O_{F_5 \cup F_6}| = 16$, we have $d^{\Omega}_{max} = 240$. The distance function gives us $d_{\Omega}(\omega_5, \omega_6) = \frac{140}{2} = 70$. So,

$$I(N_5, N_6) = 1 - \frac{d_{\Omega}(\omega_5, \omega_6)}{d^{\Omega}_{max}} = 1 - \frac{70}{240} = 0.708 \tag{7.9}$$

## 7.5 Topological Differences

All of the examples so far have assumed that the topological structure of the CP-nets is identical. We can not make this assumption in general since, even over the exact same set of features, dependency relations can be significantly different. Fortunately, computing the shared interest level between CP-nets with different topological structures is precisely the same as with any other case. In this section, examples are provided to show this similarity.

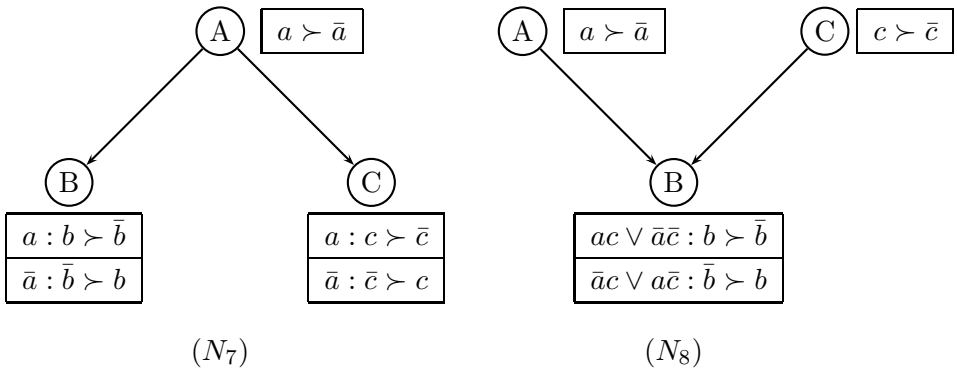


Figure 7.12: CP-nets $N_7$ and $N_8$

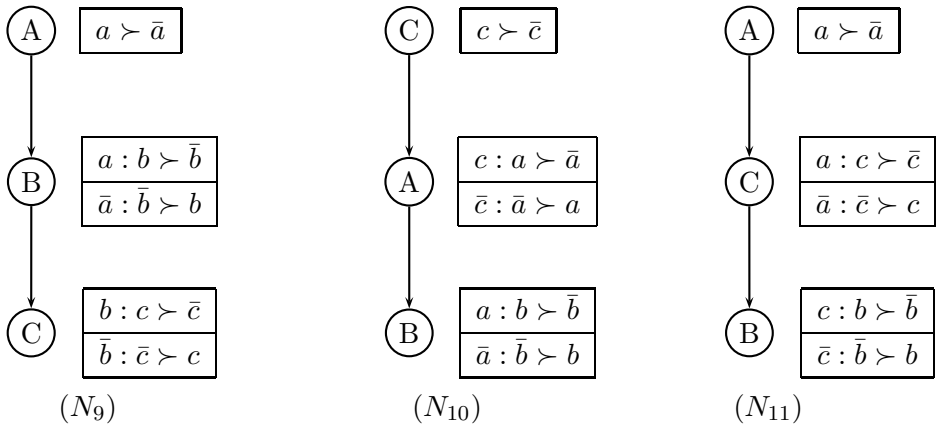


Figure 7.13: CP-nets $N_9$, $N_{10}$, and $N_{11}$

Consider CP-net $N_5$ from Figure 7.5, where $F_5 = \{A, B\}$. Assume that we extend the CP-net by adding another feature to the feature set so that $F = \{A, B, C\}$. If we assume that the relative importance among the original features in $N_5$ holds true, then there are five possible placements of our new node (assuming a connected graph) to create a new CP-net. We could place the new feature $C$ as a child of $B$, a parent of $A$, between $A$ and $B$, a root node parent of $B$, or a leaf node child of $A$. An annotated CP-net corresponding to each of these placements is provided in Figures 7.12 and 7.13.

We would like to identify any properties with respect to shared interest comparisons that can be specified for these variations, which could be topologically different (e.g.,

$N_9$ and $N_7$). It is conjectured that shared interest level decreases as the number of feature nodes influenced by the additional feature node increases. We interpret influence in this context as the number of descendants (i.e., not just immediate children) of the added feature node $f$, denoted by $Ch(f)$.

**Example 5.** Let $N_7, N_8, N_9, N_{10}, N_{11} \in \mathcal{N}_F$ be the CP-nets shown in Figures 7.12 and 7.13. Each of these CP-nets induces a preference graph. Using the paths in the preference graphs, we are able to define the ordering matrix for each.

$$M_7 = \begin{bmatrix} 0 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & -1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & -1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

$$M_8 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & -1 & -1 & 0 & 1 & 0 & 1 \\ -1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 0 & 0 & 1 & 0 & 1 \\ -1 & 0 & -1 & 0 & 0 & 1 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 & 0 & -1 & 1 \\ -1 & 0 & -1 & 0 & 1 & 1 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

$$M_9 = \begin{bmatrix} 0 & -1 & -1 & -1 & 1 & 0 & 1 & 1 \\ 1 & 0 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 0 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 0 & -1 & 1 & 1 \\ 0 & -1 & -1 & -1 & 1 & 0 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

$$M_{10} = \begin{bmatrix} 0 & 1 & -1 & 0 & -1 & 1 & -1 & 1 \\ -1 & 0 & -1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 0 & 1 & -1 & 1 & -1 & 1 \\ 0 & 1 & -1 & 0 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & 0 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

$$M_{11} = \begin{bmatrix} 0 & -1 & -1 & -1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 0 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 0 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 0 & 1 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 & 0 & -1 & 1 \\ 0 & -1 & -1 & -1 & 1 & 1 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

We want to determine how shared interest level varies between $N_5$ and each of $N_7$, $N_8$, $N_9$, $N_{10}$, $N_{11}$ as the number of feature nodes influenced by the additional feature node $C$ increases. We use the generalized Kemeny-Snell distance function to compute the shared

interest levels between $N_5$ and each of the other CP-nets. As with the other examples using $N_5$, we define the ordering matrix $M_5'$ from the expanded (disconnected) preference graph and have $d_{max}^{\Omega} = 56$:

$$I(N_5, N_7) = 1 - \frac{d_{\Omega}(\omega_5, \omega_7)}{d_{max}^{\Omega}} = 1 - \frac{14}{56} = 0.75 \tag{7.10}$$

$$I(N_5, N_8) = 1 - \frac{d_{\Omega}(\omega_5, \omega_8)}{d_{max}^{\Omega}} = 1 - \frac{16}{56} = 0.714 \tag{7.11}$$

$$I(N_5, N_9) = 1 - \frac{d_{\Omega}(\omega_5, \omega_9)}{d_{max}^{\Omega}} = 1 - \frac{15}{56} = 0.732 \tag{7.12}$$

$$I(N_5, N_{10}) = 1 - \frac{d_{\Omega}(\omega_5, \omega_{10})}{d_{max}^{\Omega}} = 1 - \frac{23}{56} = 0.589 \tag{7.13}$$

$$I(N_5, N_{11}) = 1 - \frac{d_{\Omega}(\Omega_5, \Omega_{11})}{d_{max}^{\Omega}} = 1 - \frac{20}{56} = 0.643 \tag{7.14}$$

With the shared interest function values in hand, we turn our attention to a topological analysis of the CP-net in order to explain the resulting shared interest ordering: $I(N_5, N_7) > I(N_5, N_9) > I(N_5, N_8) > I(N_5, N_{11}) > I(N_5, N_{10})$.

The feature node $C$ has been placed in various locations of each CP-net used in this example. In $N_7$, the feature $C$ is a leaf child node of feature node $A$ and, thus, does not directly influence the value for any features. Feature $B$ is the only feature in $N_8$ that is influenced by feature $C$. In $N_9$, the feature $C$ is a leaf child node of feature node $B$. For $N_{10}$, feature $C$ influences feature $A$, as well as $B$, and interpreted as the most important feature in the CP-net. In $N_{11}$, feature $C$ is placed as both a child of $A$ and a parent of $B$.

It is clear that the more influence that feature $C$ exerts on the CP-net, the less interesting that CP-net becomes for $N_5$. This is closely related to feature modifications in Example 2 based on importance. It is desirable to formalize the components that define the influence of the added feature.

Let $Pa(C)$ be the set of parents feature $C$ and $Desc(C)$ be the set of all feature nodes contained in the subtree rooted at feature node $C$. Based on the example provided, it is clear that shared interest level drops as $|Desc(C)|$ increases. Although feature importance of those feature nodes in the parent set $Pa(C)$ seems to have some effect with cases where $Ch(C)$ is held constant, it quickly becomes negligible once $|Desc(C)|$ is increased. This observation is stated in the following conjecture:

**Conjecture 1.** *If $N_i$ and $N_j$ are CP-nets created from CP-net $N$ by the addition of some feature $f$ and no other modifications take place, then $I(N, N_i) > I(N, N_j)$ whenever $|Desc_j(f)| > |Desc_i(f)|$.*

Basic tree structure properties (i.e., parents and children) have been specified in this conjecture as being factors in this type of shared interest function computation, based on empirical results. While it may be possible to prove this conjecture, or some variation of it, using standard CP-nets, the formal treatment of trade-offs in TCP-nets could be instrumental in a complete proof. Also, some notion of a probability distribution over the possible entries for CPTs might help to generalize this further. The proof of this conjecture is left for future work.

## 7.6 A Structural Approach to Distance Measure

The examples discussed in this chapter all have illustrated how variations in the CP-net structure affect the distance measure between the induced preference orderings. The intention was to show that the shared interest function is consistent with intuition when two CP-nets differ in specific ways. However, through these examples we have revealed a method for lifting the distance function to the actual CP-net structure.

A significant problem with the current approach to measuring distance between preference orderings induced by CP-nets is that the computation takes exponential time and space with respect to the number feature nodes in the CP-net. A CP-net with $n$ feature nodes will induce a preference ordering over $2^n$ outcomes. This preference ordering is represented in a $2^n \times 2^n$ ordering matrix. Clearly, this is not a very efficient method, and seriously limits the applicability in most practical (i.e., space and time limited) situations.

We now seek to define a distance function that takes polynomial time with respect to the number of feature nodes in the CP-net.

A distance function on CP-nets should be strategically equivalent to the distance function on the induced preference orderings. Let $d_{\mathcal{N}} : \mathcal{N} \times \mathcal{N} \rightarrow \Re$ be the distance function over CP-nets. The metric $d_{\mathcal{N}}$, intuitively, should satisfy $d_{\mathcal{N}}(N, N') \leq d_{\mathcal{N}}(N, N'')$ whenever $d_\Omega([\![N]\!], [\![N']\!]) \leq d_\Omega([\![N]\!], [\![N'']\!])$.

We have determined a candidate distance metric over the CP-net graphs. This

metric is formalized in terms of edge and node additions to a CP-net graph. Refer to CP-nets $N$ in graph-theoretic terms as pairs $(F, E)$. The addition of a single disconnected node to a CP-net graph corresponds to a distance of 1. This disconnected node represents a single bit of information and, thus, increases the distance by 1. Formally, if $f \cap F = \emptyset$ and $N' = (F \cup f, E)$, then $d_{\mathcal{N}}(N, N') = 1$. In a similar manner, the addition of an edge between two existing nodes in a CP-net corresponds to a distance of 1. Assume a new edge $(f, f')$ is added to a CP-net from source feature node $f$ to destination feature node $f'$. The entries in CPT($f'$) are duplicated for each instantiation $u \in O_f$ of the source feature node $f$. This new edge effectively splits the CPT of the destination node, thus representing an increase in distance by 1. Formally, if $(f, f') \cap E = \emptyset$, $f, f' \in F$, and $N' = (F, E \cup (f, f'))$, then $d_{\mathcal{N}}(N, N') = 1$.

Although the edge and node additions can be used to construct any CP-net graph, we need to use the notion of flipping rows in the CPTs in order to construct all annotated CP-nets. Using atomic flippings, we say that a single atomic flipping of a row in a CPT corresponds to a distance of 1. Now, we are able to construct all fully-annotated CP-nets through combinations of edge additions, node additions, and atomic flippings.

One issue with this distance metric over CP-net graphs is that the feature importance is not taken into consideration when applying atomic flippings. We have shown in Section 7.2 that flipping the CPT of more important features has a greater impact on distance. It is conjectured that a weight can be used to represent the feature importance so that an atomic flipping corresponds to a distance greater than or equal to 1, depending on the feature importance weight.

We have not yet shown that this candidate metric over CP-nets is consistent with the metric over preference orderings. We leave the formalization of an efficient distance function over the CP-nets to future work. By showing that the generalized Kemeny-Snell distance function of preference orderings induced by CP-nets is consistent with intuition, this thesis has laid the groundwork for defining a compatible distance function over CP-nets through, non-exclusively, either of the two methods discussed above.

# Chapter 8

# Related Work

The idea of interest-matching, or match-making, is by no means a new idea. There have been numerous attempts to model some aspect of human behavior that allows reliable interest-matching comparisons to be made between people. In this chapter, we will take a comparative look at some other noteworthy examples of this sort of work.

## 8.1 Ontology-based Approach

A more recent example of work on interest-matching systems is that of Koh [16]. The approach to interest-matching (or match-making, as used by Koh) is based on an ontology which represents the user's interests. A probabilistic interpretation of the edges in the ontology concept hierarchy is used to define the interests of a persons as a probability distribution. The Kullback-Leibler distance function is applied to the interest probability distribution of different users to perform interest-matching between them.

The use of an ontology to represent the interests of a person is consistent with many other attempts to develop interest-matching systems. There have been many successful applications of ontology-based interest-matching systems, however, the probabilistic approach has some potential flaws. It is very difficult for a human to assign a probabilistic value to an interest, or preference. Slight probabilistic disagreements in an ontology subtree of two users could be negligible in real-life, but result in different interest values. Any

system built on top of this sort of approach will accumulate this error. The qualitative approach to preference specification used in this thesis avoids this problem altogether.

## 8.2 *m*CP-nets

An overarching goal of this thesis was to lay the groundwork for a potential extension of CP-nets to a multi-agent environment which involved social-behavior. The only other known attempt to extend CP-nets to a multi-agent environment is that of *m*CP-nets [20]. With *m*CP-nets, each agent specifies its preferences through a partial CP-net. The partial CP-net of each agent in the system is then used to determine a group consensus through the application of a social welfare function. This appears to be a promising extension of the CP-net model, however, group consensus is not always desirable in multi-agent decision making applications.

It may be preferred that each agent make its own decision independent of the preferences, or decisions, of other agents. The interest-matching comparison described in this thesis allows for a more individualistic view of agent decision making. Bogart discusses briefly in [2] a Markov model of a "reasonable man". This model is based off the interpretation of changing from one ordering to another as a probability value that is inversely proportional to the distance between them. Although the discussion is very brief, it does suggest a potential application of the interest-matching comparisons described in this thesis to Markov decision processes that could influence an agents plan or collaboration. An exploration of this potential application is reserved for future work.

# Chapter 9

# Concluding Remarks

It has become clear in recent years that the need for interest-matching methods will continue to grow. Social networking applications and the introduction of the semantic web both are examples of areas that require some notion of interest-matching. Many attempts have been made at defining shared interest levels between different agents. The most common examples are those using ontologies to define what it is in which an agent is interested. Another, increasingly popular method is to use an agent's preferences or desires to represent it's interests.

This thesis has described a method for interest-matching comparisons using preference orderings represented in a CP-net. The CP-nets have been shown to be compact way for representing some preferences operating under the *ceteris paribus* semantics [6]. The human-type nature of the *ceteris paribus* preference semantics thus makes the CP-net a natural choice for representing the information necessary for performing interest-matching between human agents.

A form of the Kemeny-Snell axiomatic approach to measuring distance of preference orderings has been used which generalizes the approach on weak orders to strict partial orders [2]. A shared interest function has been defined which builds on the generalized Kemeny-Snell distance function. Empirical results suggest that the shared interest function provides reasonable interest-matching comparisons that are consistent with the intuition of what these values should be.

A refinement of the (generalized) Kemeny and Snell axiomatic approach for dis-

tance measures of preference orderings has been described. A lexicographic ordering matrix construction is specified now in a method that maintains a canonical form across all possible preference orderings, including those resulting from feature set expansions.

A significant benefit to using the approach described in this thesis is that the shared interest values can be ranked independent of the size of the CP-nets. This shared interest function is essentially a normalization of the distance function between the preference orderings induced by the CP-nets. When comparing a CP-net to other CP-nets that are of varying sizes, we are able to define a shared interest value ranking among all of the comparisons. It is noted in [9] that this is a common problem in many other interest-matching systems.

## 9.1 Future Work and Open Problems

The results achieved represent a step toward developing a formal framework in which these interest-matching comparisons can be used to implement more reliable social agent applications. This framework would encompass the entire process of selecting other agents, computing a potential level of shared interest, and using the results to step-forward a decision-making process of an agent based on it's preferences or desires.

There are some open problems that remain with the interest-matching comparison approach described in this thesis. First, it is desirable to state formally how the placement of an additional feature node in a CP-net changes the shared interest function value (see Section 7.4). Although the shared interest function value seems to decrease as the number of child nodes of the new feature node increases (i.e., as the influence of the new node increases), this has yet to be proven.

Second, the computational complexity of the interest-matching comparison method restricts significantly the applicability of this approach. This is due to the exponential size of the preference orderings, with respect to the number of features, that are represented in the CP-nets. Thus, it is desirable to state these shared interest comparisons in terms of the structural differences of the CP-nets themselves, as opposed to the potentially huge preference orderings they represent. This would permit a polynomial time algorithm with respect to the number of feature nodes in the CP-nets. This type of approach likely will require a substantial reworking of the distance axioms on CP-net structures and an initial

attempt at this is described in Section 7.6. It was the goal of this thesis to show a reasonable and intuitive method for interest-matching comparisons; not find the most efficient method possible. The problems mentioned form a topic for future work on this approach.

# Bibliography

[1] Alon Altman and Moshe Tennenholtz. On the axiomatic foundations of ranking systems. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, Scotland, UK*, pages 917–922, July 30-August 5, 2005.

[2] Kenneth P. Bogart. Preference structures I: Distances between transitive preference relations. *Journal of Mathematical Sociology*, 3(1):49–67, 1973.

[3] Kenneth P. Bogart. Preference structures II: Distances between asymmetric relations. *Journal of Applied Mathematics*, 29(2):254–262, September 1975.

[4] Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. UCP-Networks: A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI), Seattle, Washington, USA*, pages 56–64, August 2-5, 2001.

[5] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research (JAIR)*, 21:135–191, 2004.

[6] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI), Stockholm, Sweden*, pages 71–80, July 30-August 1 1999.

[7] Ronen I. Brafman and Carmel Domshlak. Introducing variable importance tradeoffs

into CP-nets. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI), Edmonton, Alberta, Canada*, pages 69–76, August 1-4, 2002.

[8] Richard A. Brualdi. *Introductory Combinatorics*. Prentice Hall, 3 edition, 1999.

[9] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. A system for principled matchmaking in an electronic marketplace. In *Proceedings of the Twelfth International World Wide Web Conference (WWW), Budapest, Hungary*, pages 321–330, May 20-24, 2003.

[10] Carmel Domshlak and Ronen I. Brafman. CP-nets - Reasoning and consistency testing. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR), Toulouse, France*, pages 121–132, April 22-25, 2002.

[11] Jon Doyle and Richmond H. Thomason. Background to qualitative decision theory. *AI Magazine*, 20(2):55–68, 1999.

[12] Jon Doyle and Michael P. Wellman. Modular utility representation for decision-theoretic planning. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems (AIPS), College Park, Maryland, USA*, pages 236–242, June 15-19, 1992.

[13] Jon Doyle and Michael P. Wellman. Representing preferences as *ceteris paribus* comparatives. In *Working Notes of the AAAI Spring Symposium on Decision-Theoretic Planning*, March 1994.

[14] Leonard N. Foner. Yenta: A multi-agent, referral-based matchmaking system. In *Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, California, USA*, pages 301–307, February 5-8, 1997.

[15] John G. Kemeny and J. Laurie Snell. *Mathematical Models in the Social Sciences*. MIT Press, 1962.

[16] Waikit Koh and Lik Mui. An information theoretic approach to ontology-based interest matching. In *Proceedings of the Second Workshop on Ontology Learning (IJCAI '01), Seattle, Washington, USA*, August 4, 2001.

[17] Fred S. Roberts. *Discrete Mathematical Models with Applications to Social, Biological, and Environmental Problems*. Prentice-Hall, 1976.

[18] Fred S. Roberts. *Measurement Theory with Applications to Decisionmaking, Utility, and the Social Sciences*. Addision-Wesley, 1979.

[19] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill, 4 edition, 1999.

[20] Francesca Rossi, Kristen B. Venable, and Toby Walsh. mCP-nets: Representing and reasoning with preferences of multiple agents. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence and Sixteenth Conference on Innovative Applications of Artificial Intelligence, San Jose, California, USA*, pages 729–734, July 25-29, 2004.

[21] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2 edition, 2003.

[22] Michael Terry, Elizabeth D. Mynatt, Kathy Ryall, and Darren Leigh. Social net: Using patterns of physical proximity over time to infer shared interests. In *Extended Abstracts of the 2002 Conference on Human Factors in Computing Systems (CHI), Minneapolis, Minnesota, USA*, pages 816–817, April 20-25, 2002.

[23] Michael P. Wellman and Jon Doyle. Preferential semantics for goals. In *Proceedings of the Ninth National Conference on Artificial Intelligence, Anaheim, California, USA*, pages 698–703, 1991.

# Appendices

# Appendix A

# Notation Definitions

$\succsim$ : weak preference

$\succ$ : strict preference

$\sim$ : indifference

$\succ^e$ : strict preference over extended outcomes

$=_\sharp$, $<_\sharp$ : base two numeric interpretation relations

$\{A, B, C, \dots\}$ : a set of features

$\{a, \bar{a}, b, \bar{b}, \dots\}$ : a set of feature value assignments

$\mathbf{F}$ : the enumerated finite universe of features

$\mathcal{N}$ : the set of all CP-nets

$N_i$ : the CP-net for agent $i$

$F_i$ : the feature set for CP-net $N_i$

$E_i$ : the directed edge set for CP-net $N_i$

$CPT_i(f)$ : the conditional preference table for feature $f$ in CP-net $N_i$

$Pa(f)$ : the set of parents of feature $f$

$Desc(f)$ : the set of all feature nodes in the subtree rooted at feature node $f$

$P(N_i)$ : the preference graph induced by CP-net $N_i$

$p_{ij}$ : a path from outcome $i$ to outcome $j$ in a preference graph

$O_F$ : the set of all outcomes over $F$

$\Omega_F$ : the set of all strict partial orderings over $O_F$

$\omega_i$ : the preference ordering for agent $i$ induced by $N_i$

$[\![N_i]\!]$ : the meaning of CP-net $N_i$, where $[\![N_i]\!] = \omega_i$

$M_i$ : the ordering matrix representation for CP-net $N_i$

$\sigma_f$ : flipping function for feature $f$

$\pi_f$ : CPT-flipping function for feature $f$

$\Pi$ : *CP-permutation* function

# Appendix B

# Application Source Code

## B.1 Overview

The application source code listed in this appendix is not meant to be a full implementation of the interest-matching comparisons described in this thesis. Instead, only straight-forward interest-matching comparisons are computed between two CP-nets over the same feature set. The intention was to use this application as a research aid since shared interest function computations become unmanageable to compute manually once the feature set grows beyond three or four features.

The C# programming language was utilized using the Microsoft® Visual Studio® 2005 application development environment. This decision of programming language was somewhat arbitrary, and based loosely on the desire to use a more modern programming language. Implementation and usage details are provided in the documentation within the source code. The XML documentation format was used to facilitate easy extraction.

## B.2 CPNet class

```
/*
 * <summary>
 * author: Andrew Wicker <awwicker@unity.ncsu.ed>
```

```
 * </summary>
 */

using System;
using System.Collections.Generic;
using System.Text;

namespace CPNets {
    /*
     * <summary>
     * Class definition of a CP-net
     * </summary>
     */
    class CPNet
    {
        private int[,] orderingMatrix;  // ordering matrix representation as array
        private int numberOfEdges;
        private List<Object> nodes = new List<Object>();

        /*
         * <summary>
         * Constructor method
         * </summary>
         * <param name="preferenceOrdering">the preference ordering input by the user.</param>
         * <param name="cpnetEdgeSet">the edge set description of this CP-net graph.</param>
         */
        public CPNet(String preferenceOrdering, String cpnetEdgeSet)
        {
            ParseEdgeSet(cpnetEdgeSet);
            orderingMatrix = new int[GetNumberOfOutcomes(), GetNumberOfOutcomes()];
            ParseOrdering(preferenceOrdering);
        }

        /*
         * <summary>
         * Parses the edge set for this CP-net input by the user.
         * </summary>
         * <param name="cpnetEdgeSet">this CP-net edge set input by user.</param>
         */
        public void ParseEdgeSet(String cpnetEdgeSet)
        {
            Char[] delimiters = new char[] { ' ' };
```

```
    numberOfEdges = 0;
    foreach (String e in cpnetEdgeSet.Split(delimiters))
    {
        // assumes format e = (A,B)
        Feature fOne = new Feature(e[1].ToString());
        Feature fTwo = new Feature(e[3].ToString());
        fOne.AddChild(fTwo);
        fTwo.AddParent(fOne);
        AddNode(fOne);
        AddNode(fTwo);
        numberOfEdges++;
    }
}

/*
 * <summary>
 * Sets ordering matrix
 * </summary>
 * <param name="orderingMatrix">the ordering matrix in canonical form for this
 *  CP-net.</param>
 */
public void SetOrderingMatrix(int[,] orderingMatrix)
{
    this.orderingMatrix = orderingMatrix;
}

/*
 * <returns>the ordering matrix for this CP-net.</returns>
 */
public int[,] GetOrderingMatrix()
{
    return orderingMatrix;
}

/*
 * <returns>number of features in this CP-net.</returns>
 */
public int GetNumberOfFeatures()
{
    return this.nodes.Count;
}
```

```
/*
 * <returns>the number of features in this CP-net.</returns>
 */
public int GetNumberOfOutcomes()
{
    return (int)Math.Pow(2, this.nodes.Count);
}

/*
 * <param name="m2">the ordering matrix for another CP-net being compared
 *  to this one.</param>
 * <returns>the Kemeny-Snell distance value between this CP-net ordering matrix
 * and the ordering matrix m2 of another CP-net.</returns>
 */
public float Distance(int[,] m2)
{
    float d = 0;
    // compute the k-s distance function between orderingMatrix and m2
    for (int i = 0; i < GetNumberOfOutcomes(); i++)
    {
        for (int j = 0; j < GetNumberOfOutcomes(); j++)
        {
            d += Math.Abs(orderingMatrix[i, j] - m2[i, j]);
        }
    }
    return d / 2;
}

/*
 * <param name="cTwo">another CP-net to which this one is being compared</param>
 * <returns>the interest value between this CP-net and another CP-net <c>cTwo</c>.</returns>
 */
public float Interest(CPNet cTwo)
{
    float d = Distance(cTwo.orderingMatrix);
    // use distance function to compute interest
    float dmax = GetNumberOfOutcomes() * (GetNumberOfOutcomes() - 1);
    return 1 - (d / dmax);
}

/*
 * <summary>
```

```
 * Parses the ordering over outcomes that are input as a <c>String</c>
 * delimited by spaces. The space delimiter can easily be changed to anything else.
 * </summary>
 * <param name="preferenceOrdering">the preference ordering input by the user.</param>
 */
public void ParseOrdering(String preferenceOrdering)
{
    Outcome[] mOrdering = new Outcome[GetNumberOfOutcomes()];
    Char[] delimiters = new char[] { ' ' };
    int i = 0;
    foreach (String oInput in preferenceOrdering.Split(delimiters))
    {
        Outcome o = new Outcome(oInput, GetNumberOfFeatures());
        mOrdering[i] = o;
        i++;
    }
    for (i = 0; i < GetNumberOfOutcomes(); i++)
    {
        for (int j = 0; j < GetNumberOfOutcomes(); j++)
        {
            // assign the ordering matrix values according to canonical form
            if (WorseningFlip(mOrdering[i], mOrdering[j]) && (i < j))
            {
                orderingMatrix[mOrdering[i].GetIntegerValue(), mOrdering[j].GetIntegerValue()] = -1;
                orderingMatrix[mOrdering[j].GetIntegerValue(), mOrdering[i].GetIntegerValue()] = 1;
            }
        }
    }
    //compute the transitive closure
    for (i = 0; i < GetNumberOfOutcomes(); i++)
    {
        for (int j = 0; j < GetNumberOfOutcomes(); j++)
        {
            for (int k = 0; k < GetNumberOfOutcomes(); k++)
            {
                if ((orderingMatrix[i, j] == 1) && (orderingMatrix[j, k] == 1))
                {
                    orderingMatrix[i, k] = 1;
                    orderingMatrix[k, i] = -orderingMatrix[i, k];
                }
                else if ((orderingMatrix[i, j] == -1) && (orderingMatrix[j, k] == -1))
                {
```

```
                    orderingMatrix[i, k] = -1;
                    orderingMatrix[k, i] = -orderingMatrix[i, k];
                }
            }
        }
    }
}

/*
 * <summary>
 * Determines whether or not the preference <c>oOne > oTwo</c> is
 *  a worsening flip.
 * </summary>
 * <param name="oOne">an outcome.</param>
 * <param name="oTwo">an outcome.</param>
 * <returns>true if <c>oOne > oTwo</c> is a legal worsening flip;
 *          false otherwise.</returns>
 */
public Boolean WorseningFlip(Outcome oOne, Outcome oTwo)
{
    int i = 0;
    int diff = 0;

    while ((i < oOne.oArray.Length) && (i < oTwo.oArray.Length))
    {
        if (oOne.oArray[i].GetIntegerValue() != oTwo.oArray[i].GetIntegerValue())
        {
            diff++;
        }
        i++;
    }
    if (diff == 1)
    {
        return true;
    }
    else
    {
        return false;
    }
}

/*
```

```
 * <summary>
 * Outputs the ordering matrix for this CP-net.
 * </summary>
 */
public void OutputOrderingMatrix()
{
    String row = "";
    for (int i = 0; i < GetNumberOfOutcomes(); i++)
    {
        for (int j = 0; j < GetNumberOfOutcomes(); j++)
        {
            if (orderingMatrix[i, j] == -1)
            {
                row += " " + orderingMatrix[i, j];
            }
            else
            {
                row += "  " + orderingMatrix[i, j];
            }
        }
        Console.WriteLine(row);
        row = "";
    }
}

/*
 * <summary>
 * Adds a node to the graph.
 * </summary>
 * <param name="f">a feature.</param>
 */
public void AddNode(Object f)
{
    Boolean exists = false;
    // check node f is already in the list of nodes
    for (int i = 0; i < nodes.Count; i++)
    {
        if (((Feature)nodes[i]).GetFeature().Equals(((Feature)f).GetFeature()))
        {
            exists = true;
        }
    }
```

```
            if (!exists)
            {
                nodes.Add(f);
            }
        }

        /*
         * <returns>returns the number of nodes in this graph.</returns>
         */
        public int GetNumberOfNodes()
        {
            return this.nodes.Count;
        }

        /*
         * <returns>returns the number of edges in this graph.</returns>
         */
        public int GetNumberOfEdges()
        {
            return this.numberOfEdges;
        }
    }
}
```

## B.3 Feature class

```
/*
 * <summary>
 * author: Andrew Wicker <awwicker@unity.ncsu.ed>
 * </summary>
 */

using System;
using System.Collections;
using
System.Collections.Generic;
using System.Text;

namespace CPNets {
```

```
/*
 * <summary>
 * Feature class definition
 * </summary>
 */
class Feature
{
    private String feature;
    private FeatureAtom posAtom;
    private FeatureAtom negAtom;
    private List<Object> parentNodes = new List<Object>();
    private List<Object> childNodes = new List<Object>();

    /*
     * <summary>
     * Constructor method.
     * </summary>
     */
    public Feature(String feature)
    {
        this.feature = feature.ToUpper();
        posAtom = new FeatureAtom(feature.ToLower());
        negAtom = new FeatureAtom("-" + feature.ToLower());
    }

    /*
     * <returns>this feature name.</returns>
     */
    public String GetFeature()
    {
        return this.feature;
    }

    /*
     * <summary>
     * Sets the feature name.
     * </summary>
     * <param name="feature">the new feature name.</param>
     */
    public void SetFeature(String feature)
    {
        this.feature = feature.ToUpper();
```

```
        }

        /*
         * <returns>the positive value assignment for this feature.</returns>
         */
        public FeatureAtom GetPosAtom()
        {
            return this.posAtom;
        }

        /*
         * <returns>the negative value assignment for this feature.</returns>
         */
        public FeatureAtom GetNegAtom()
        {
            return this.negAtom;
        }

        /*
         * <summary>
         * Adds a parent node to this node.
         * </summary>
         * <param name="f">a feature.</param>
         */
        public void AddParent(Object f)
        {
            this.parentNodes.Add(f);
        }

        /*
         * <summary>
         * Adds a child node to this node.
         * </summary>
         * <param name="f">a feature.</param>
         */
        public void AddChild(Object f)
        {
            this.childNodes.Add(f);
        }
    }
}
```

## B.4 FeatureAtom class

```
/*
 * <summary>
 * author: Andrew Wicker <awwicker@unity.ncsu.ed>
 * </summary>
 */

using System;
using System.Collections.Generic;
using System.Text;

namespace CPNets {
    /*
     * <summary>
     * Class definition for a feature atom, i.e., a feature value assignment.
     * </summary>
     */
    class FeatureAtom
    {
        private Boolean value;
        private String atom;

        /*
         * <summary>
         * Constructor method.
         * </summary>
         * <param name="atom">a single feature value assignment.</param>
         */
        public FeatureAtom(String atom)
        {
            this.atom = atom;
            if (atom.Length == 1)
            {
                value = true;
            }
            else
            {
                value = false;
            }
        }
```

```
        /*
         * <returns>the string value of this feature atom.</returns>
         */
        public String GetAtomValue()
        {
            return this.atom;
        }

        /*
         * <returns>the boolean value of this feature atom.</returns>
         */
        public Boolean GetValue()
        {
            return this.value;
        }



        /*
         * <returns>the [1,0] integer value of this feature atom.</returns>
         */
        public int GetIntegerValue()
        {
            if (value == true)
            {
                return 1;
            }
            else
            {
                return 0;
            }
        }
    }
}
```

## B.5 Outcome class

```
/*
 * <summary>
 * author: Andrew Wicker <awwicker@unity.ncsu.ed>
```

```
 * </summary>
 */

using System;
using System.Collections.Generic;
using System.Text;

namespace CPNets {
    /*
     * <summary>
     * Class definition for an outcome defined over the feature set of a CP-net.
     * </summary>
     */
    class Outcome
    {
        private String o;               // String representation of this outcome
        private int numberOfFeatures;
        public FeatureAtom[] oArray;    // array representation of this outcome

        /*
         * <summary>
         * Constructor method.
         * </summary>
         */
        public Outcome(String o, int numberOfFeatures)
        {
            this.o = o;
            this.numberOfFeatures = numberOfFeatures;
            oArray = new FeatureAtom[numberOfFeatures];
            ParseOutcome(o);
        }

        /*
         * <summary>
         * Parses the string representation of the outcome <c>o</c>.
         * </summary>
         * <param name="o">an outcome.</param>
         */
        public void ParseOutcome(String o)
        {
            int i = 0;
            for (int j = 0; j < o.Length; j++)
```

```
            {
                if (o[j] == '-')
                {
                    oArray[i] = new FeatureAtom(String.Concat(o[j], o[j + 1]));
                    j++;
                }
                else
                {
                    oArray[i] = new FeatureAtom(o[j] + "");
                }
                i++;
            }
        }

        /*
         * <returns>the integer value for the bit interpretation of this outcome.</returns>
         */
        public int GetIntegerValue()
        {
            int value = 0;
            for (int i = 0; i < oArray.Length; i++)
            {
                value += (int)Math.Pow(2, (oArray.Length - i) - 1) * oArray[i].GetIntegerValue();
            }
            return value;
        }
    }
}
```

## B.6 Program class

```
/*
 * <summary>
 * author: Andrew Wicker <awwicker@unity.ncsu.ed>
 * </summary>
 */

using System;
using System.Collections.Generic;
```

```
using System.Text;

namespace CPNets {
    class Program
    {
        /*
         * <summary>
         * The main method for the CPNets program. The user is prompted to input the edge set
         * and preference ordering for two CP-nets. The shared interest value between these two
         * CP-nets is returned, along with the ordering matrix for each. The edge set should be
         * input using standard notation delimited by a single space. For example, <c>(A,B) (B,C)</c>.
         * The preference orderings should be input using the character '-' to denote a negative
         * feature value and the outcomes in the ordering should be single space delimited.
         * For example, <c>ab a-b -a-bc -ab</c>. If there are incomparable outcomes, they should
         * be listed in the ordering just as they are induced by the CPTs. The incomparability
         * will be determined when parsed by the program.
         * </summary>
         * <param name="args">the console arguments.</param>
         */
        public static void Main(string[] args)
        {
            // Enter information for CP-net C1
            Console.WriteLine("Enter the C1 directed edge set:");
            Console.WriteLine("ex.) (A,B) (B,C) (B,D)");
            String cOneEdgeSet = Console.ReadLine();
            Console.WriteLine("Enter C1 preference ordering (including incomparables):");
            Console.WriteLine("ex.) ab a-b -a-b -ab");
            String cOnePreferenceOrdering = Console.ReadLine();
            CPNet cOne = new CPNet(cOnePreferenceOrdering, cOneEdgeSet);
            cOne.OutputOrderingMatrix();

            // Enter information for CP-net C2
            Console.WriteLine("Enter the C2 directed edge set:");
            Console.WriteLine("ex.) (A,B) (B,C) (B,D)");
            String cTwoEdgeSet = Console.ReadLine();
            Console.WriteLine("Enter C2 preference ordering (including incomparables):");
            Console.WriteLine("ex.) ab a-b -a-b -ab");
            String cTwoPreferenceOrdering = Console.ReadLine();
            CPNet cTwo = new CPNet(cTwoPreferenceOrdering, cTwoEdgeSet);
            cTwo.OutputOrderingMatrix();
            Console.WriteLine("I(C1,C2) = " + cOne.Interest(cTwo));
            Console.ReadLine();
```

```
        }
    }
}
```