UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

Argumentation based preference aggregation

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Patricio Mosse

Director: Patrice Buche, Madalina Croitoru y Jérôme Fortin

Codirector: Ricardo Rodríguez

Buenos Aires, 2012

AGREGACIÓN DE PREFERENCIAS BASADA EN ARGUMENTACIÓN

El objetivo de este proyecto de investigación es agregar las preferencias de distintas partes interesadas utilizando argumentación en una Fase de Negociación. Esto es realizado en el contexto de un proyecto específico, explicado en esta tesis.

La agregación de las preferencias de distintas partes interesadas puede derivar en conflictos debido a las diferencias en las necesidades de cada uno. Acá es donde puede utilizarse argumentación para resolver conflictos, ya que puede aparecer nueva información que conduzca a cambios en las preferencias, incluso al agregado o remoción de preferencias existentes.

El concepto de Fase de Negociación se introduce para permitir que las partes interesadas revisen sus preferencias, discutan, argumenten e ingresen nueva información. Como resultado de esta fase, las preferencias pueden modificarse para resolver conflictos, o podemos descubrir que un conflicto no puede resolverse mediante argumentación, conduciendo a otras soluciones como la división de un escenario para manejar las elecciones por separado.

Keywords: Preferencias, Argumentación, Negociación, EcoBioCap, Multi-Agente.

ARGUMENTATION BASED PREFERENCE AGGREGATION

The objective of this research project is to aggregate the preferences of different stakeholders using argumentation in a Negotiation Phase. This is done in the context of a specific project, which is explained in this thesis.

The aggregation of several stakeholders' preferences may incur into different conflicts due to differences in the needs of each stakeholder. This is where argumentation can be used in order to solve conflicts, as new information may appear and lead to changes in the preferences, even adding new preferences or removing old ones.

The concept of Negotiation Phase is introduced in order to allow stakeholders to review their preferences, discuss, argue, and enter new information. As a result of this phase, preferences may be changed in order to solve conflicts, or we may find out that a conflict is not solvable by argumentation, leading to other solutions such as splitting a scenario in order to manage different choices separately.

Keywords: Preferences, Argumentation, Negotiation, EcoBioCap, Multi-Agent.

ACKNOWLEDGEMENTS

I would like to express my very great appreciation to the University of Buenos Aires and the Faculty of Natural and Exact Sciences, for the high quality, public and free education I received, and to INRIA, for the opportunity of travelling to France for six months in order to do an internship and my Master Thesis.

I would like to convey my deep gratitude to Professor Patrice Buche, Professor Jérôme Fortin, Associate Professor Madalina Croitoru and Professor Ricardo Rodríguez, my research supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work. I would also like to acknowledge Souhila Kaci and Alain Gutierrez for their useful comments.

Special thanks to Bruno Paiva Lima da Silva for helping me settle in Montpellier, and to Sergio Romano, Alejandro Nuñez and Herman Schinca, who accompanied me during my career in Argentina and helped me continue.

Finally, I wish to thank my family (Ino Mosse, Marta Mosse, Matias Mosse, Veronica D'Amato, Lila Mosse, Alexis Mosse, Jonas Mosse, Veronica Satz, Francisco Mosse, Leticia Monastiridis, Valentin Mosse, Lucas Mosse, Cecilia Escudero, Benjamin Mosse, Michel Mosse, Candela Mosse, William Glover, Charly Mosse and Billy Mosse) and friends (Sebastian Zenobi, Martin Belingheri, Federico Luna, Bruno Binora, Leonardo Bauer, Sabrina Abolsky, Julia Dvorkin, Martina Lewin, Cristina Bergalio, Juan Zotalis, Sebastian Koutsovitis and Lucas Carra) for their support and encouragement throughout my studies, specially Sebastián Uchitel, Francisco Mosse and Matías Mosse, who made me choose this career.

A mis viejos y a mis abuelos.

CONTENTS

1. INTRODUCTION

The research question of this Thesis is 'how to aggregate non independent preferences using argumentation based negotiation'.

The application of this Thesis is the European project EcoBioCap (ECOefficient BIOdegradable Composite Advanced Packaging) [1].

We have proposed a preference logic and a negotiation phase based on argumentation to address the above questions and implement it. The implementation was evaluated by experts from the EcoBioCap project.

1.1 State of the Art

A decision support system (DSS) can facilitate and enhance the group decision making process. Such a DSS must typically *aggregate* the preferences of multiple entities in the group in order to recommend a final decision. Argumentation forms a natural way of encoding reasons as to why some action should be taken (or avoided), and several researchers have recently focused on the problem of preference aggregation within an argumentation framework [2, 9, 10].

While assuming the existence of preferences, the logical semantics of these preferences is typically not fully described, and the modelling choices made are not explained. Since the chosen semantics need to satisfy the requirements of the application at hand, defining such preference logic semantics accordingly is an important task.

In this thesis we instantiate an argumentation system designed to aggregate the preferences of multiple parties where preference information is defined in terms of expertise knowledge on the topic.

Our system is based on the popular ASPIC+ model [12]; the use of this model as one of our foundations allows us to ensure that it adheres to several desiderata, including the rationality postulates described in [4]. ASPIC+ provides an abstract model of argument (though far less abstract than the one described in [6]), and allows for different logics to be embedded within it. Part of our work is therefore to describe one such logic. In this work we introduce a logic of preferences which we use within our argument framework, and describe how arguments and argument related concepts such as attacks can be obtained. We also discuss how to consider different viewpoints elicitation within our framework, and in particular, how the viewpoints of an expert can be represented. Finally, we propose a global methodology to detect and solve conflicts and get to the desired output.

The language we described is based on [14, 3] which describes the semantics of preferences between propositions.

[1] http://www.ecobiocap.eu

1.2 EcoBioCap Project

The motivation and context for this work comes from the argumentation based decision support system that is being built for the European project EcoBioCap (ECOefficient BIOdegradable Composite Advanced Packaging). The aim of EcoBioCap is to provide customizable, ecoefficient, biodegradable packaging to EU consumers. Figure 1.1 illustrates the workflow of the project. The choice of food packaging depends on several physical factors that arise from interactions between the packaging and the food at the molecular, nanoscopic, microscopic and macroscopic levels. The project's industrial partners liase with research laboratories in order to ensure the feasibility of the research oriented packaging material produced. Finally, a decision support system is needed in order to choose between the feasible packaging based on several preferences of EcoBioCap stakeholders.

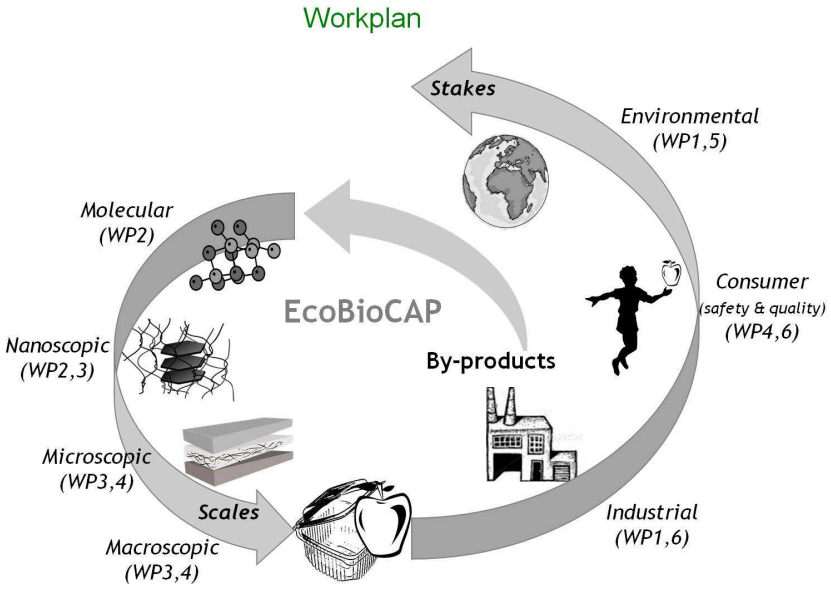

Fig. 1.1: The EcoBioCap workpackage cycle

One of the project's core tasks involves carrying out a strategic analysis of stakeholder requirements in terms of food quality and safety but also cost, technical and environmental impact. The aim is to provide the right inputs to the further steps of the project with the development of modelling and decision support tools.

The EcoBioCap decision support system consists of several components, as shown in Figure 1.2. The fresh food database captures the gas exchange properties for each potential packaged foodstuff. The packaging database plays a similar role, identifying the oxygen and carbon dioxide permeability as well as other properties (e..g transparency) of different packing materials. The third and final datastore tracks stakeholders preferences, which are defined in terms of positive and negative preferences. Negative preferences correspond to constraints since they specify what values or objects have to be rejected (i.e. those which do not satisfy the constraints). Positive preferences correspond to wishes to specify which objects are more desirable to others.

The decision support workflow operates as follows:

1. Information from the stakeholder preferences and fresh food database is used as input to a simulator, which computes the ideal permeability properties of a packaging for a given food type.

2. The permeability data, together with information from the packaging database and stakeholder preferences, is used to run a multi criteria query, from which a list of possible packagings, ranked from best to worst, can be identified.

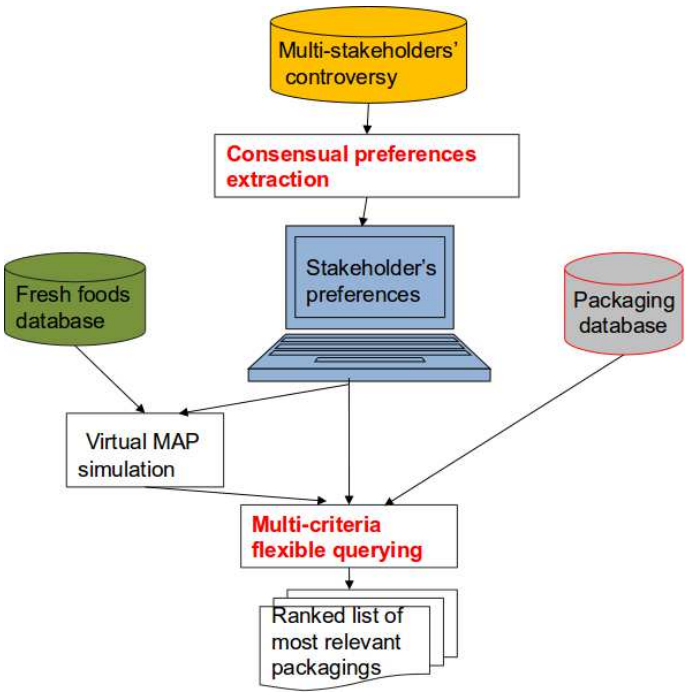

Fig. 1.2: Decision Support System within EcoBioCap

The aggregation of different stakeholder preferences into a consistent preference set must occur before the system begins running.

Note that for stakeholder preference elicitation and their corresponding arguments a series of questionnaires have been sent to each EcoBioCap actor (package makers, cheese makers and consumers from different European countries). The criteria on which the questions were focusing were transparency, price, nano particle presence, biodegradability, etcetera. The objective was to collect information about the interest and needs of the stakeholders. As previously mentioned, the main results of this survey forms part of the initial specifications which are the basis of the experimental trials on the development of biodegradable packaging materials. In the next subsection we give an aggregation example of a real-world scenario of such preferences expressed by stakeholders.

1.3 Real-world scenario and aggregation example

In this subsection we describe a complete aggregation example of a real-world scenario. We consider two specific stakeholders [2]: Expert 1, a sheep cheese maker, and Expert 2, a multi-national dairy products corporation.

The aim of the following example is to give an insight of the kind of preferences stakeholders have, how to aggregate them and how the arguments behind each two preferences values can be used in order to solve a conflict.

To be able to devise the best packaging for cheese, the two stakeholders Expert 1 and Expert 2 were asked for their preferences. In order to do this, a technical survey was sent to each of them. The survey was built in order to understand their current packaging methods and their expectations for a new packaging system. It was divided between different sections: 'Description of the new packaging material' (we asked them about the material, the type of machine, the type of filling method, mechanical properties, the method of labelling, etc., in order to be able to use this information in case of a conflict), 'Storage and handling of the foods to be packed' (we asked them about the conditions during handling, the quality and safety criterion, etc.), 'Competitive/benchmarking packaging materials/systems' (we asked them about the strengths and weaknesses of their competitive brands), 'Customer food requirements' (we asked them about some key customer requirements related to the use of by-products, raw materials for PHA production and other preferences), 'Ranking criterion for decision (packaging choice)' (we asked them to rank the list of preferred criterion by decreasing importance) and 'Other limitations'.

Once we had the results of the surveys, we went through them extracting the preferences and the arguments supporting them manually. This could be done automatically in a future development of the project. Here we present the arguments for four criteria: 'Color', 'Shelf Life', 'Type of Machine' and 'Type of Labelling':

1. Color

   (a) Expert 1 says that "my first options for color are green or mix and match as that is the way customers know us. If that is not possible then I could have a transparent package to allow them to see the product."

   (b) Expert 2 says that "I would like to use mix and match because that is how the customers know us. In case that is not possible, we prefer transparent over green as green is being used by other brands".

2. Shelf Life

   (a) Expert 1 claims that "the shelf life must be at least of 40 days as it is the current shelf life of packed cheese and the new material must permit the same or extend it".

[2] Please note that due to confidentiality requirements, the names of the experts are enclosed and their statements have been modified.

(b) Expert 2 notes that “the shelf life must be of 70 days as it is a supermarket constraint”.

3. Type of Machine

(a) Expert 1 claims that “I prefer hand made packaging over Horizontal form-fill-seal as hand made is our current way of packaging and the Horizontal form-fill-seal is available but is being used for another type of cheese, and using that machine implicates we must label the products using a sticker attached. We cannot use Vertical form-fill-seal as we don’t have that machine”.

(b) Expert 2 notes that “I prefer to use Vertical form-fill-seal machine or Horizontal form-fill-seal over hand made because when using hand made packaging the atmosphere in the head-space can’t be controlled so the cheese quality can’t be guaranteed. Between the machines we prefer Vertical form-fill-seal over Horizontal form-fill-seal as it is the only machine available at the moment”.

4. Type of Labelling

(a) Expert 1 notes that “I prefer direct printing over sticker attached as the sticker can be easily removed by the customer”.

(b) Expert 2 claims that “I prefer sticker attached over direct printing as it is easier to use when changing customer, it permits resealability at home and the cost is low”.

For the aggregation of preferences we need to identify all possible conflicts and solve them. The desired output of this process is to obtain the most preferred values for each criterion.

Let’s take the criteria one by one and see how can we aggregate the preferences:

1. Color:

   For this criterion, the possible values are ‘Green’, ‘Mix and Match’ and ‘Transparent’. For Expert 1 the only options are ‘Green’ and ‘Mix and Match’, due to a marketing wish, so he won’t be accepting ’Transparent’ as an option. Expert 2 expressed he would like to have ’Mix and Match’, and if that is not possible then he prefers ‘Transparent’ over ‘Green’. As we can see, ’Mix and Match’ is the most preferred value for Expert 1 and is also one of the wanted values for Expert 2, so that value is the result of the aggregation of preferences for this criterion.

2. Shelf Life:

   For this criterion, the possible values are the natural positive numbers. As Expert 1 claims that the shelf life must be of at least 40 days, and Expert 2 requested a shelf life of 70 days, we can aggregate the preferences and have 70 or plus as the result.

3. Type of Packaging:

   For this criterion, the possible values are ‘Horizontal form-fill-seal’ (‘HFFS’), ‘Vertical form-fill-seal’ (‘VFFS’) and ‘Hand Made’. Expert 1 said that ‘VFFS’ is not applicable for him, and that he prefers ‘Hand Made’ over ‘HFFS’. On the other hand, Expert 2 expressed a preference for ‘VFFS’ over ‘HFFS’, and ‘HFFS’ over ‘Hand Made’.

   ‘VFFS’ is not applicable for Expert 1, so even it is the most preferred option for Expert 2, we are not going to consider that value. Then, we look at the next most preferred option. As we can see, Expert 1 prefers ‘Hand Made’ over ‘HFFS’ while Expert 2 prefers the contrary. Now, as we have an unsolvable conflict, we need to look at the argument supporting the preferences and start a negotiation phase. We recall the arguments:

   (a) Expert 1 claims that “I prefer hand made packaging over Horizontal form-fill-seal as hand made is our current way of packaging and the Horizontal form-fill-seal is available but is being used for another type of cheese, and using that machine implicates we must label the products using a sticker attached. We cannot use Vertical form-fill-seal as we don’t have that machine”.

   (b) Expert 2 notes that he prefers to use “Horizontal form-fill-seal over hand made because when using hand made packaging the atmosphere in the head-space can’t be controlled so the cheese quality can’t be guaranteed”.

   Expert 1 prioritizes the organization, while Expert 2 is concerned about quality. By looking at the arguments, we also find out an implication: using ‘HFFS’ implies using ‘Sticker Attached’ as a ‘Type of Labelling’, and for that criterion Expert 1 prefers ‘Direct Printing’ because the sticker could be easily removed by customers, so it is a marketing wish.

   The negotiation phase is started with the two opposed arguments, a) and b). Expert 1 and 2 discuss in order to enter new information and arguments to help solve the conflict. In this example the arguments expressed are not true in the real world, but we will assume they are for simplicity reasons.

   Expert 1 says that “with the new policies it is very easy to control the atmosphere in the head-space when using hand made packaging”.

   Expert 2 doesn’t have any other arguments as now he is convinced. The argumentation finishes, Expert 2 changes his mind, and the conflict is solved. ‘Hand Made’ is the chosen value for this criterion. As the implication had ‘HFFS’ as the hypothesis, it is not followed in this case.

4. Type of Labelling:

For this criterion, the possible values are ‘Sticker Attached’ and ‘Direct Printing’. Expert 1 prefers ‘Direct Printing’ due to a marketing wish while Expert 2 prefers ‘Sticker Attached’ because of the cost and because it is easy to change it for any customer. As this is another unsolvable conflict, we start a new negotiation phase. Expert 2 argues that with a special glue customers won’t be able to remove the sticker. Expert 1 then changes his mind, solving the conflict. The result of the aggregation for this criterion then is ‘Sticker Attached’.

The result of the aggregation of preferences is ’Mix and Match’ for Color, 70 for Shelf Life, ’Hand Made’ for Type of Machine and ’Sticker Attached’ for Type of Labelling.

1.4 Organization of this document

Our main objective is to automatize this process once the preferences have been elicited. In the example shown in this section, conflicts were solved by using a negotiation phase. Another possibility could have been ranking the arguments. For example, determining that Quality is more important than Organization or Marketing. This corresponds to the future work of this project, as in this document we consider that the categories of the arguments are all equal, and will be discussed in the conclusions (Chapter 4, page 41). Negotiation phases based on argumentation help us to update the preferences and solve conflicts by obtaining new information from the stakeholders.

For the theoretical research of this project we have used ASPIC to instantiate our argumentation framework. Later in the implementation of the prototype system we haven’t use it, to focus in the conflicts and problems that may appear during the use of a negotiation phase, and for simplicity reasons.

This document is structured as follows. In this chapter we have presented the state of the art and explained the need of an argumentation based negotiation phase in the context of the EcoBioCap project. In Chapter 2 we introduce the underlying logical language in which preferences are expressed, we describe its syntax and semantics to talk about arguments, and the formalization of our methods. This is followed by Chapter 3 which describes the implementation of the prototype system. Finally, in Chapter 4 we present some conclusions and perspectives for future work.

2. METHODOLOGY

The problem we have has a defined input and output. Our input is a set of preferences for each stakeholder and for each criterion. We also have a set of implications (for example choosing 'HFFS' for the criterion 'Type of Machine' implicates choosing also 'Sticker Attached' for the criterion 'Labelling'). The output of the system is a consensual preference value for each criterion. By consensual, we understand that there are no conflicts among the preferences of the actors.

The output can be extended in the future in order to return not one but a set of ranked preferred values for each criterion (for example, for the 'Color' criterion we could say that 'Green' and 'Transparent' are the values that don't present conflicts, and that 'Green' is preferred to 'Transparent').

In order to obtain the desired output, we are going to:

1. Aggregate the preferences of all stakeholders into a single set.
2. Detect conflicts among them automatically.
3. For each conflict, use a Negotiation Phase and argue for resolving it, if possible.
4. If it is not possible to solve a conflict, solve it by splitting the options into different scenarios.
5. Return the consensual preference value for each criterion.

To explain these steps we will formalize some concepts.

2.1 Modalization

Let $S = \{s_1, ..., s_n\}$ be a set of stakeholders.

The set of stakeholders we consider in this example are Expert 1 and Expert 2. Thus, $S = \{\text{Expert 1, Expert 2}\}$.

Let $C = \{c_1, ..., c_k\}$ be a set of criteria. For every criterion $c \in C$, V_c denotes the set of possible values the criterion c can take.

In our application, the set of criteria is:

$C = \{\text{color, shelf life, type of machine, labelling }\}$, where the values associated to these criteria are:

- $V_{color} = \{\text{green, transparent, mix \& match}\}$
- $V_{shelflife} = \mathbb{N}^+$
- $V_{typeofmachine} = \{\text{hffs, vffs, hand made}\}$

- $V_{labelling} = \{\text{direct printing, sticker attached}\}$

One way to obtain preferences from the stakeholders is to ask them to rank the values of each criterion, and to use an utility function afterwards.

Let us consider an infinite discrete set $V = \{+, ++, +++, ...\}$ and a preorder $<$ defined on V. We also consider two special elements in V: "always" and "impossible". We define V^+ as $V^+ = V \cup \{always, impossible\}$. We extend the preorder $<$ on V^+ such that: $impossible < + < ++ < +++ < ... < always$.

Observation 1: These two new elements are needed in order to express impossible constraints, that is, values that are not wanted by the stakeholders, and positive constraints, that is, values that are the only ones eligible by the stakeholders and that make all the rest impossible. As a result of the aggregation of preferences, we are never going to choose a value that is "impossible" for a stakeholder. If we cannot aggregate a criterion for that reason we will have to split the scenario in order to manage all the possibilities.

For each criterion $c \in C$ and stakeholder $s \in S$ we define an utility function $utility_c^s : V_c \to V^+$. This function expresses the preference relation between the values for one criterion and one stakeholder, and the values that are either positive or negative constraints.

We define a preorder $\prec_c^s$ on the values v of each criterion $c \in C$ for each stakeholder $s \in S$: $\forall v_1, v_2 \in V_c$, $(v_1, v_2) \in \prec_c^s$ iff $utility_c^s(v_1) < utility_c^s(v_2)$.

Please note that according to the previous definition, we can map the elements of V_c for each stakeholder $s \in S$ onto as many *always* and *impossible* images as needed. We will use the following notation: $NA_c^s = \{v \in c \text{ such that } utility_c^s(v) = impossible\}$ and $W_c^s = \{v \in c \text{ such that } utility_c^s(v) = always\}$. In the following, for simplicity of notation, we will only use the $\prec_c^s$ preorder.

For simplicity reasons, we note $v_1 \prec_c^s v_2$ if $(v_1, v_2) \in \prec_c^s$. Let A be a set of values ($A \subseteq V_C$). We note $A \prec_c^s v_2$ if $\forall e \in A, e \prec_c^s v_2$, and we note $v_2 \prec_c^s A$ if $\forall e \in A, v_2 \prec_c^s e$.

Notation 1: When there is no possible confusion, we can suppress the stakeholder and the criterion in the notation and write $\prec$ directly.

Let us consider 'color' criterion and its set of possible values: $V_{\text{color}} = \{\text{green, transparent, mix \& match}\}$. We abbreviate 'green' by 'g', 'transparent' by 't' and 'mix & match' by 'mm'.

1. $W_{\text{color}}^{\text{Expert 1}}$: g, mm.
2. $\prec_{\text{color}}^{\text{Expert 1}}$: t $\prec \{g, mm\}$.
3. $\prec_{\text{color}}^{\text{Expert 2}}$: g $\prec$ t $\prec$ mm.

Let us consider 'shelf life' criterion and its set of possible values: $V_{\text{shelf life}} = \mathbb{N}^+$.

4. $W_{\text{shelf life}}^{\text{Expert 1}}$: 70.
5. $W_{\text{shelf life}}^{\text{Expert 2}}$: 40.

Let us consider 'type of machine' criterion and its set of possible values: $V_{\text{type of machine}} = \{\text{hffs, vffs, hand made}\}$. We abbreviate 'hand made' by 'h'.

6. $NA^{\text{Expert 1}}_{\text{type of machine}}$: vffs.
7. $\prec^{\text{Expert 1}}_{\text{type of machine}}$:vffs $\prec$ hffs $\prec$ h.
8. $\prec^{\text{Expert 2}}_{\text{type of machine}}$: h $\prec$ hffs $\prec$ vffs.

Observation 2: Please note that this criterion is related to 'labelling' criterion due to an implication (restriction) found in the arguments, we will talk about this in the preference's aggregation process.

Let us consider 'labelling' criterion and its set of possible values: $V_{\text{labelling}} = \{\text{direct printing, sticker attached}\}$. We abbreviate 'direct printing' by 'dp' and 'sticker attached' by 'sa' for simplicity of notation.

1. $\prec^{\text{Expert 1}}_{\text{labelling}}$: dp $\prec$ sa .
2. $\prec^{\text{Expert 2}}_{\text{labelling}}$: sa $\prec$ dp .

Observation 3: In the above, we always have at least a non rejected value for each criterion, e.g. $\forall s \in S, \forall c \in C, NA^s_c \subsetneq V_c$.

Aggregating the above preferences of each stakeholder involves detecting inconsistencies between preference orders.

We also consider implications that we may find among the preferences, expressing that choosing a value for a criterion forces the selection of another value for another criterion:

Let $I =<$ c1, v1, c2, v2, a1 $>$ be an implication, where c1 is the hypothesis' criterion, v1 is the hypothesis' value, c2 is the conclusion's criterion, v2 is the conclusion's value, and a1 is the stakeholders' argument.

If there are no inconsistencies the preference aggregation is straightforward. On the other hand, if there are inconsistencies to be solved, then we can use the arguments of the stakeholder, which support the preferences (each pair of values has an associated argument). These arguments have been elicited via technical surveys sent to every stakeholder.

Observation 4: We can only choose one value for each criterion for the aggregation of preferences, although this could be extended in the future for this project.

Lets recall the 'type of machine' criterion discussed above. We have an unsolvable conflict when we find out that Expert 1 prefers 'Hand Made' over 'HFFS' and Expert 2 prefers the contrary. Let us take another look at the arguments behind the preferences from above:

1. Expert 1 claims that "I prefer hand made packaging over Horizontal form-fill-seal as hand made is our current way of packaging and the Horizontal form-fill-seal is available but is being used for another type of cheese, and using that machine implicates we must label the products using a sticker attached. We cannot use Vertical form-fill-seal as we don't have that machine".
2. Expert 2 notes that "I prefer to use Vertical form-fill-seal machine or Horizontal form-fill-seal over hand made because when using hand made packaging the atmosphere in the head-space can't be controlled so the cheese quality can't be guaranteed. Between the machines we prefer Vertical form-fill-seal over Horizontal form-fill-seal as it is the only machine available at the moment".

We find out that there is an implication: the selection of 'HFFS' for the 'type of machine' criterion implies the selection of 'sticker attached' for the 'labelling' criterion. By analysing the arguments we detect that Expert 1 prioritizes Organization, while Expert 2 is concerned about Quality. Also, choosing 'HFFS' implies that we would be harming Marketing.

In order to solve this conflict we need to consider which criterion is more important for us. In this case, Quality is more important, so we choose 'HFFS' for the 'type of machine' criterion and therefore 'sticker attached' for the 'labelling' criterion.

This same conflict can be solved by using a Negotiation Phase. This will be explained later in this chapter.

2.2 Preference Conflicts

Preference conflicts always confront two different values of the same criterion. They also face two different stakeholders and two preferences. Each stakeholder has an initial argument to defend his value (or attack the opposite value).

Let $X = \{\langle \text{v1, s1, a1}\rangle, \langle \text{v2, s2, a2}\rangle, c\}$ be a conflict. v1 represents the first value in conflict, s1 the stakeholder defending it, and a1 the argument or reason he presents. v2 represents the second value in conflict, s2 the stakeholder defending it, and a2 his argument. c is a criterion.

$X = \{\langle \text{Green, Expert 1, Argument 1}\rangle, \langle \text{Transparent, Expert 2, Argument 2}\rangle, \text{Color}\}$.

In Section 3.4, page 27, we explain how to detect conflicts within the preferences automatically, and we present several examples to clarify our methods.

In order to formalize conflicts, we need to define a preference language. Language $\mathscr{L}$ is explained in the next subsection.

2.3 Representing Preferences

In this section we introduce a logical language able to represent the kind of user preferences expressed above. Notably we are interested in a propositional logic based language on which we

define a preference operator. This will allow to perform reasoning both from a classical propositional view point but also from a preference view point.

We start by defining the syntax of our language, the semantics, and then we try to infer basic axioms such as transitivity. We prove that under normal assumptions the transitivity does not hold and therefore we need to further study and constrain this language in order to be able to produce a sound and complete set of axioms that characterize it. This last aspect is very important since it will be needed in order to automatically infer preferences conflicts and therefore be able to set the basis for an argumentation system based on such conflicts and attacks.

In Section 2.3.2 we explain how to enforce transitivity in a model, using maximal centred models [7].

2.3.1 The Syntax of $\mathscr{L}$

In the following we inductively define the syntax of the language $\mathscr{L}$ we consider as the basis of our argumentation framework.

Let $\mathscr{L}$ be the language generated from a set of propositional symbols $\mathscr{PS}$ together with the connectives $\wedge, \neg, \succeq, \not\succeq$ and $\preceq$ as follows:

- $Prop(\mathscr{PS})$ is the set of formulae classically defined in propositional logic over connectors $\wedge, \neg$.
- If $A \in Prop(\mathscr{PS})$ then $A \in \mathscr{L}$.
- If $A, B \in Prop(\mathscr{PS})$ then $A \succeq B \in \mathscr{L}$ and $A \not\succeq B \in \mathscr{L}$.
- If $A, B \in \mathscr{L}$ then $A \wedge B \in \mathscr{L}$.
- If $A, B \in Prop(\mathscr{PS})$ then $(A \preceq B) \in \mathscr{L}$ and $\neg(A \not\succeq B) \in \mathscr{L}$.

We also make use of the classical equivalences:

- $A \vee B = \neg(\neg A \wedge \neg B)$
- $A \rightarrow P = \neg A \vee B$
- $A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$.

2.3.2 The Semantics of $\mathscr{L}$

Having described the syntax of $\mathscr{L}$, we now define its semantics. We define a model as $M \subseteq 2^{\mathscr{PS}}$. In order to assume transitivity, we make the strong assumption of maximal centred models [7]. Let $\mathscr{M}$ be a set of models. An interpretation is defined with the help of a set of models $\mathscr{M}$ and a given total order $\geq$ over $\mathscr{M} : \geq \subseteq M \times M$. We write $M_1 \geq M_2$ iff $(M_1, M_2) \in \geq$ and $M_1 \not\geq M_2$ iff $(M_1, M_2) \notin \geq$. We associate the total order with a utility function over the propositions within a model: $\mu : 2^{\mathscr{PS}} \mapsto \mathbb{R}$. Given such a function, $\geq$ is defined as follows : $M_1 \geq M_2$ if and only if $\mu(M_1) \geq \mu(M_2)$.
Given a model M, we define the satisfaction relation $\models$ according to a model as follows:

- $\forall A \in Prop(\mathscr{P}\mathscr{S}), M \models A$ iff $A \in M$.
- $\forall A \in Prop(\mathscr{P}\mathscr{S}), M \models \neg A$ iff $A \notin M$.
- $\forall A, B \in Prop(\mathscr{P}\mathscr{S}), M \models A \wedge B$ iff $M \models A$ and $M \models B$.

Now, let $(\mathscr{M}, \geq)$ be an interpretation. The satisfaction relation $\models$ and the unsatisfaction relation $\not\models$ are defined on $(\mathscr{M}, \geq)$ as follows:

- $\forall a \in \mathscr{P}\mathscr{S}$, $(\mathscr{M}, \geq) \models a$ iff $\forall M \in \mathscr{M}$, $a \in M$, otherwise $(\mathscr{M}, \geq) \not\models a$.

- $\forall A, B \in Prop(\mathscr{P}\mathscr{S})$ $(\mathscr{M}, \geq) \models A \succeq B$ iff $\forall M_1, M_2 \in \mathscr{M}$ satisfying:
  - $M_1 \models A \wedge \neg B$
  - $M_2 \models B \wedge \neg A$
  - M_1, M_2 coincide for all elements of $\mathscr{P}\mathscr{S} \setminus VAR(A) \cup VAR(B)$

  then $M_1 \geq M_2$
  otherwise, $(\mathscr{M}, \geq) \not\models A \succeq B$

- $\forall A, B \in Prop(\mathscr{P}\mathscr{S})$ $(\mathscr{M}, \geq) \models A \not\succeq B$ iff $\exists M_1, M_2 \in \mathscr{M}$ satisfying:
  - $M_1 \models A \wedge \neg B$
  - $M_2 \models B \wedge \neg A$
  - M_1, M_2 coincide for all elements of $\mathscr{P}\mathscr{S} \setminus VAR(A) \cup VAR(B)$

  then $M_2 \geq M_1$
  otherwise, $(\mathscr{M}, \geq) \not\models A \not\succeq B$.

- $\forall A, B \in Prop(\mathscr{P}\mathscr{S})$ $(\mathscr{M}, \geq) \models (A \preceq B)$ iff $\forall M_1, M_2 \in \mathscr{M}$ satisfying:
  - $M_1 \models A \wedge \neg B$
  - $M_2 \models B \wedge \neg A$
  - M_1, M_2 coincide for all elements of $\mathscr{P}\mathscr{S} \setminus VAR(A) \cup VAR(B)$

  then $M_2 \geq M_1$
  otherwise, $(\mathscr{M}, \geq) \not\models (A \preceq B)$.

- $\forall A, B \in \mathscr{L}$, $(\mathscr{M}, \geq) \models A \wedge B$ iff $(\mathscr{M}, \geq) \models A$ and $(\mathscr{M}, \geq) \models B$
  otherwise, $(\mathscr{M}, \geq) \not\models A \wedge B$.
- $\forall A \in \mathscr{L}$, $(\mathscr{M}, \geq) \models \neg A$ iff $(\mathscr{M}, \geq) \not\models A$
  otherwise, $(\mathscr{M}, \geq) \not\models \neg A$.

2.3.3 Inference Rules

ASPIC+ encodes a logic within an argumentation framework by utilising its proof theory represented as a set of inference rules. We must therefore identify the set of inference rules that are valid in $\mathscr{L}$.

Given a set $\mathscr{F}$ of formulae on $\mathscr{L}$, we say that $C \in \mathscr{L}$ is the consequence of $\mathscr{F}$ (denoted $\mathscr{F} \vdash C$) if and only if for all interpretations $(\mathscr{M}, \geq)$ such that for every $F \in \mathscr{F}$, $(\mathscr{M}, \geq) \models F$ it is the case that $(\mathscr{M}, \geq) \models C$.

We assume the standard properties of associativity and distributivity over $\wedge$ and $\neg$, as in classical propositional logic, and again include the standard abbreviations for implication ($\rightarrow$), and disjunction ($\vee$). We then denote by $RS^{\mathscr{P}\mathscr{S}}$ the set of classical inference rules in propositional logic. Language $\mathscr{L}$ also admits the following valid rules:

$$
\begin{aligned}
&RS_1 : (\phi \preceq \psi) \vdash (\phi \not\succeq \psi) \\
&RS_2 : \neg(\phi \not\succeq \psi) \vdash (\psi \succeq \phi) \\
&RS_3 : (\phi \succeq \psi) \wedge (\psi \succeq \sigma) \vdash (\phi \succeq \sigma) \\
&RS_4 : (\neg\phi \succeq \phi) \wedge (\psi \rightarrow \phi) \vdash (\neg\psi \succeq \psi) \\
&RS_5 : (\phi \succeq \neg\phi) \wedge (\psi \rightarrow \phi) \vdash (\psi \succeq \neg\psi)
\end{aligned}
$$

RS_1 can be shown true by noting that the left hand side can only be true if either $M_1 < M_2$, or $(M_1, M_2) \notin \geq$ (i.e. they are incomparable). Both of these situations satisfy the requirement imposed by the right hand side of the rule. The proof for RS_2 is analogous to this. RS_3 follows directly from the transitivity of the model preference relation $\geq$ (we make the strong assumption here that the model set is maximal centered, see [7] for more details). The last two rules can be trivially proven from the definition of the set of models satisfying the preference operator.

2.3.4 Contrariness function on $\mathscr{L}$

Given the inference rules above we define a contrariness function $cf : \mathscr{L} \rightarrow 2^{\mathscr{L}}$ as below. Intuitively this function denotes the mutual exclusivity between two formulae within a given $\mathscr{M}$. This function will be used in the next section to define the attack between two arguments in our argumentation system.
For all "classical" propositional formulae ϕ:

- $cf(\phi) = \{\neg\phi\}$
- $cf(\neg\phi) = \{\phi\}$.

For all other formulae in $\mathscr{L}$:

- $cf((a \preceq b)) = \{(a \succeq b)\}$,
- $cf((a \succeq b)) = \{(b \succeq a), (a \preceq b), (a \not\succeq b)\}$
- $cf((a \not\succeq b)) = \emptyset$
- $cf(\neg(a \not\succeq b)) = \{(a \succeq b), (b \preceq a), (b \not\succeq a)\}$

2.3.5 Not closed model sets

The semantics shown above has been previously introduced in [5] where the strong assumption of maximal centered models [7] has been made. We will prove that transitivity cannot hold anymore if this assumption is lifted. As a consequence most of the inference rules defined in [5] can no longer be inferred. We present in the following the general version of this language and define a much poorer contrariness function used for mapping our language onto ASPIC+ (similar to [5]).

Property 1: The transitivity relation is not valid in the defined logic: $\forall a,b,c \in \mathscr{L}$, the satisfaction relation does not hold for all interpretations $(\mathscr{M},\succeq)$. So:

$$(\mathscr{M},\succeq) \nvDash (a \succeq b) \wedge (b \succeq c) \rightarrow a \succeq c$$

Proof: We are going to build a counterexample. Let $(\mathscr{M},\succeq)$ be the following interpretation over the symbols $\mathscr{PS} = \{a,b,c\}$:

$$\mathscr{M} = \{M1, M2\}.$$

Where $M1 = \{a,b\}, M2 = \{c,b\}$, and $M2 \succeq M1$.
Let us show that:

1. $(\mathscr{M},\succeq) \vDash (a \succeq b)$
2. $(\mathscr{M},\succeq) \vDash (b \succeq c)$
3. $(\mathscr{M},\succeq) \nvDash (a \succeq c)$

1. $(\mathscr{M},\succeq) \vDash (a \succeq b) iff$
   $\forall M3, M4 \in \mathscr{M}$
   if $M3 \vDash a \wedge \neg b$
   $M4 \vDash \neg a \wedge b$
   and M3 and M4 coincide for all other elements of $\mathscr{PS}$
   then $M3 \succeq M4$.

   $\nexists M \in \mathscr{M} \diagup M \vDash a \wedge \neg b$.
   So the previous statement is always true.

2. Same kind of reasoning.

3. $(\mathscr{M},\succeq) \vDash (a \succeq c) iff$
   $\forall M3, M4 \in \mathscr{M}$
   if $M3 \vDash a \wedge \neg c$
   $M4 \vDash \neg a \wedge c$
   and M3 and M4 coincide for all other elements of $\mathscr{PS}$
   then $M3 \succeq M4$.

   $M1 \vDash a \wedge \neg c$.
   $M2 \vDash c \wedge \neg a$.
   M1 and M2 coincide for all other elements of $\mathscr{PS}$.

But $M2 \succeq M1$.
So this is a counterexample because:
$(\mathscr{M}, \succeq) \models (a \succeq b) \wedge (b \succeq c) but (\mathscr{M}, \succeq) \not\models (a \succeq c)$

□

Property 2: The following two rules are not valid in the defined logic. $\forall a, b \in \mathscr{L}$, the following satisfaction relations do not hold for all interpretations $(\mathscr{M}, \succeq)$:

$$(\mathscr{M}, \succeq) \not\models (\neg a \succeq a) \wedge (b \rightarrow a) \rightarrow (\neg b \succeq b)$$

$$(\mathscr{M}, \succeq) \not\models (a \succeq \neg a) \wedge (b \rightarrow a) \rightarrow (b \succeq \neg b)$$

For all "classical" propositional formulae ϕ:

- $cf(\phi) = \{\neg\phi\}$
- $cf(\neg\phi) = \{\phi\}$.

For all other formulae in $\mathscr{L}$:

- $cf((a \succeq b)) = \{(b \succeq a)\}$

In the next subsection we define the argumentation system allowing 1) conflict detection based on the contrariness function and 2) for the negotiation phase.

2.4 Argumentation

Let us present some basic definitions at work in Dung's theory of argumentation [6]. We restrict them to finite argumentation frameworks.

Definition 1 (argumentation framework): An argumentation framework is a pair (A, R) where A is a finite set of arguments and $R \subseteq A \times A$.

For example: $AF = (\{a, b, c\}, \{(a, b), (b, c)\})$

Clearly enough, the set of finite argumentation frameworks is a proper subset of the set of Dung's finite argumentation frameworks, where every argument can be attacked by finitely many arguments. The definition above clearly shows that a finite argumentation framework is nothing but a finite, labelled digraph.

A second important notion is the notion of absence of conflicts. Intuitively, two arguments should not be considered together whenever one of them attacks the other one.

Definition 2 (conflict-free sets): Let $AF = \langle A, R \rangle$ be a finite argumentation framework. A subset S of A is conflict-free if and only if for every $a, b \in S$, we have $(a, b) \notin R$.

For example: $AF = (\{a, b, c\}, \{(a, b), (a, c)\}).S = \{(b, c)\}$

Definition 3 (acceptable sets): Let $AF = \langle A, R \rangle$ be a finite argumentation framework. An argument $a \in A$ is acceptable w.r.t. a subset S of A if and only if for every $b \in A$ s.t. $(b,a) \in R$, there exists $c \in S$ s.t. $(c,b) \in R$. A set of arguments is acceptable w.r.t. S when each of its elements is acceptable w.r.t. S.

Requiring the absence of conflicts and the form of autonomy captured by self-acceptability leads to the notion of admissible set.

Definition 4 (admissible sets): Let $AF = \langle A, R \rangle$ be a finite argumentation framework. A subset S of A is admissible if and only if S is conflict-free and it is acceptable w.r.t. S.

The significance of the concept of admissible sets is reflected by the fact that every extension of an argumentation framework under the standard semantics introduced by Dung (preferred, stable, complete and grounded extensions) is an admissible set, satisfying some form of optimality:

Definition 5 (extensions): Let $AF = \langle A, R \rangle$ be a finite argumentation framework.

- A subset S of A is a preferred extension of AF if and only if it is maximal w.r.t. $\subseteq$ among the set of admissible sets for AF.
- A subset S of A is a stable extension of AF if and only if it is conflict-free and for every argument a from $A \setminus S$, there exists $b \in S$ s.t. $(b,a) \in R$.
- A subset S of A is a complete extension of AF if and only if it is admissible and it coincides with the set of arguments acceptable w.r.t. itself.
- A subset S of A is a grounded extension of AF if and only if it is the least element w.r.t. $\subseteq$ among the complete extensions of AF.

Our system is instantiated on top of the popular ASPIC+ model [12]; the use of this model as one of our foundations allows us to ensure that it adheres to several desiderata, including the rationality postulates described in [4]. ASPIC+ provides an abstract model of argument (though far less abstract than the one described in [6]), and allows for different logics to be embedded within it. Part of our work is therefore to describe one such logic.

2.5 ASPIC+

Following [12], we define an argumentation system as the tuple $\mathscr{AS} = (\mathscr{L}, cf, \mathscr{R}, \geq)$, where:

- $\mathscr{L}$ is the logical language explained in Section 2.3.1;
- cf is the *contrariness function* from $\mathscr{L}$ to $2^{\mathscr{L}}$ as defined in Section 2.3;
- $\mathscr{R} = \mathscr{R}_s \cup \mathscr{R}_d$ the set of inference rules ($\mathscr{R}_s$ are the strict inference rules and $\mathscr{R}_d$ are the defeasible inference rules) within $\mathscr{L}$ defined cf. Section 2.3 as follows:
  - $\mathscr{R}_s = \{RS_1, ..., RS_5\} \cup RS^{\mathscr{P}\mathscr{S}}$;

- $\geq$ a preference ordering over defeasible rules, which we ignore for now ($\geq = \emptyset$).

In what follows, for a given argument, the function Prem returns all the formulas of K (called premises) used to build the argument, Conc returns its conclusion, Sub returns all its sub-arguments, DefRules returns all the defeasible rules of the argument and, finally, TopRule returns the last inference rule used in the argument. K_n is a set of (necessary) axioms. Intuitively, arguments cannot be attacked on their axiom premises.

Given a knowledge base $\mathscr{K} \subseteq \mathscr{L}$, we define an argument A as per Definition 3.6 of [12], requiring that $\phi, \neg\phi$ cannot be inferred from either $\texttt{Conc}(A)$ or $\texttt{Prem}(A)$.

An argument A on the basis of a knowledge base $\mathscr{K}$ is:

1. ϕ if $\phi \in \mathscr{K}$ with $\texttt{Prem}(A) = \{\phi\}$; $\texttt{Conc}(A) = \phi$; $\texttt{Sub}(A) = \{\phi\}$; $\texttt{Rules}(A) = \emptyset$; $\texttt{TopRule}(A) =$ undefined.
2. $A_1, \ldots A_n \rightarrow / \Rightarrow \psi$ if $A_1, \ldots, A_n$ are arguments such that there is a strict or defeasible rule $\texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \rightarrow / \Rightarrow \psi$ in $\mathscr{R}_s/\mathscr{R}_d$.
   - $\texttt{Prem}(A) = \texttt{Prem}(A_1) \cup \ldots \cup \texttt{Prem}(A_n)$
   - $\texttt{Conc}(A) = \psi$,
   - $\texttt{Sub}(A) = \texttt{Sub}(A_1) \cup \ldots \cup \texttt{Sub}(A_n) \cup \{A\}$
   - $\texttt{DefRules}(A) = \texttt{DefRules}(A_1) \cup \ldots \cup \texttt{DefRules}(A_n) \cup \{\texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \rightarrow / \Rightarrow \psi\}$
   - $\texttt{TopRule}(A) = \texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \rightarrow / \Rightarrow \psi$

ASPIC+ defines three types of attacks:

1. An *undercutting attack* from argument A to argument B on some $B' \in \texttt{Sub}(B)$ occurs iff $\texttt{Conc}(A) \in cf(B')$ and B' is of the form $B''_1, \ldots, B''_n \rightarrow \phi$.
2. A *rebutting attack* from A on argument B occurs iff $\texttt{Conc}(A) \in cf(\phi)$ for some $B' \in \texttt{Sub}(B)$ of the form $B''_1, \ldots, B''_n \rightarrow \phi$. If $\texttt{Conc}(A)$ is a contrary of ϕ then A contrary-rebuts B.
3. A *undermining attack* occurs from argument A to B iff $\texttt{Conc}(A) \in cf(\phi)$ for some $\phi \in \texttt{Prem}(B) \backslash K_n$. If $\texttt{Conc}(A)$ is a contrary of ϕ or $\phi \in K_a$ then A contrary-undermines B.

The notion of an attack is then extended to the notion of a defeat via two concepts:

1. Argument A successfully rebuts argument B if A rebuts B on B' and A contrary-rebuts B'.
2. A successfully undermines B if A undermines B on ϕ and A contrary-undermines B.

Then argument A defeats B iff no premise of A is an issue and A undercuts or successfully rebuts or successfully undermines B. If A defeats B and B does not defeat A, then A strictly defeats B.

We can now map from ASPIC to a Dung argumentation framework $\langle Arg, Def \rangle$ by defining the set of arguments Arg from the definition of arguments above, and defining Def according to the

defeat notion defined above.

In [1], Amgoud analyses the ASPIC+ framework and argues that it suffers from five weaknesses: i) its logical formalism is ill-defined, ii) it may return undesirable results, iii) it builds on some counter-intuitive assumptions, iv) it violates some rationality postulates, and v) it allows counter-intuitive instantiations. These arguments are later rebutted in [11] by Prakken and Modgil. The analysis of this information and its impact on this project is part of future work and won't be discussed yet.

2.6 Negotiation Phase

We introduce the idea of a Negotiation Phase to help us decide between two opposite preferences in conflict. When stakeholders express their preferences, they also present a reason sustaining them, which we consider as the argument behind the preference. Therefore, every conflict presents two arguments attacking each other. The idea behind that is that two opposed preferences represent the starting point of a discussion. So, we assume that the arguments supporting them attack each other. When the negotiation phase start, we label each argument with a letter an build a first argumentation framework with both arguments attacking each other.

During the negotiation phase, we let stakeholders enter new information, arguments and attacks, and we calculate Dung's preferred extensions in every step. Whenever we have a single preferred extension including only one of the original arguments, we let the user finish the argumentation. The preference supported by the argument that lost is turned around, in order to support the value it was attacking before. The preference supported by the argument that won, stays there.

If the conflict is not solvable by argumentation and no stakeholder can convince the other one (if the argumentation framework always presents more than one preferred extension) then there is no possible solution but split the scenario. This means that the whole system (preferences, stakeholders, arguments) is duplicated in order to have two equal systems. After that, in one of them one of the opposed preferences is removed, and in the other one, the second preference is removed. This let us work in parallel with both possibilities and not disappoint anyone of the stakeholders.

Splitting scenarios can lead to many parallel scenarios, as they can grow exponentially. Because of this reason, in the first implementation of the prototype system we are only going to let the user keep one of the scenarios and the other ones will be discarded.

Fig. 2.1: Argumentation Framework created for the two arguments supporting the preferences in conflict. The circles represent argument and the arrows represent attacks. The arrow goes from the attacking argument to the attacked argument.

In Figure 2.1 we can see an argumentation framework just created. It presents two arguments that

support two opposed preferences (one stakeholder prefers Sticker Attached over Direct Printing while the other one prefers the contrary).

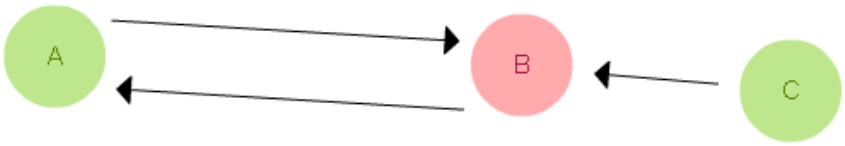

Fig. 2.2: Argumentation Framework with a new argument added in the Negotiation Phase. The preferred extension includes one of the original arguments and not the other one, so we could finish the argumentation at this point if there are no new arguments.

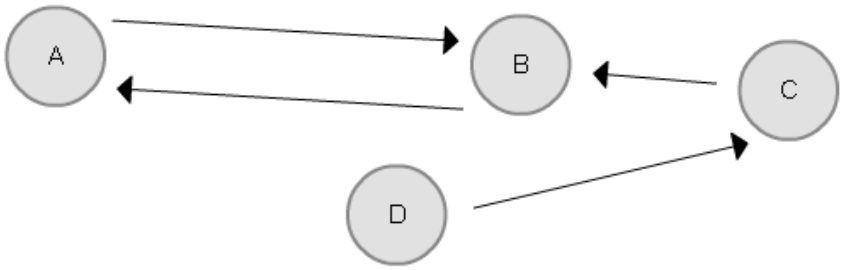

Fig. 2.3: Argumentation Framework with two new arguments added in the Negotiation Phase. There are two preferred extensions, therefore we cannot finish the argumentation at this point. If there are no new arguments we are going to need to split the scenario in order to manage all the possible values as output for this criterion.

In Figure 2.2, the argumentation framework has been updated with a new argument, attacking argument B. It now presents a single preferred extension, which contains argument A and C and not argument B. Thus, we are able to finish the negotiation phase unless stakeholders want to add new arguments.

In Figure 2.3, the argumentation framework has been updated again with another argument. As we now have two preferred extensions, we need either to wait for new arguments or split the scenario and manage both preferences separately in parallel.

2.7 Most Preferred Value

The system finds the most preferred value for each criterion and scenario when there are no more conflicts, if possible. If it is not possible, we let the user choose the value manually.

It is important to notice that for the algorithm in this phase, we take advantage of the lack of information about preferences. If one or more stakeholders didn't provide information about their preference between two values, but another stakeholder did, then we will assume they all have the same preference.

The algorithm also discards specific values for the election of the most preferred value depending on the implications that involve them.

The system tries to choose automatically the most preferred value for each criterion that doesn't have any conflicts within their preferences, as soon as the user enters the preferences information and conflicts are detected. This is done in order of appearance of the criteria.

Every time the user solves a conflict, if the system detects that a criterion is not involved in any conflict anymore, it tries to select the most preferred value for it, if possible. If not, it will be chosen by the user later. We will provide more details in Section 3.8, page 33.

In Chapter 3 we present the prototype system we have developed and describe in detail the algorithms and decisions we have taken.

3. IMPLEMENTATION

In order to test the usage of a negotiation phase in the argumentation based preference aggregation, we have developed a prototype system. This prototype was implemented in Java.

The first implementation of this prototype includes a simple user interface, as a more complex and dynamic user interface is intended to be implemented in a second version of the prototype.

The objective of the prototype is to allow the user to aggregate the preferences of all the stakeholders, automatically detect conflicts within them, and provide him with tools in order to solve them. For example, it simulates Negotiation Phases between the stakeholders so they can enter new information, preferences and take decisions.

This implementation is based on the theoretical work on ASPIC+, but is much simpler as our intention was to detect problems that may occur during negotiation phases.

The work-flow of the prototype system can be divided into the following phases:

1. Data Entry
2. Global Preferences Representation
3. Cycles
4. Preference Conflicts
5. Not Applicable & Wanted Values
6. Conflict Resolution
7. Argumentation Framework
8. Most Preferred Value
9. Decision's Information

Those phases are now described and explained.

3.1 Data Entry

This is the first phase of the system and its objective is to allow the user to enter all the information elicited in the technical surveys. In this step the user can load the data of the criteria, possible values, implications, stakeholders and preferences.

In the first version of the prototype this phase is not available, and instead the information is loaded from different input files, which represent different test cases. The format of the input files is fully described in Appendix C. Here we describe this phase as if the system had an interface to load the information manually.

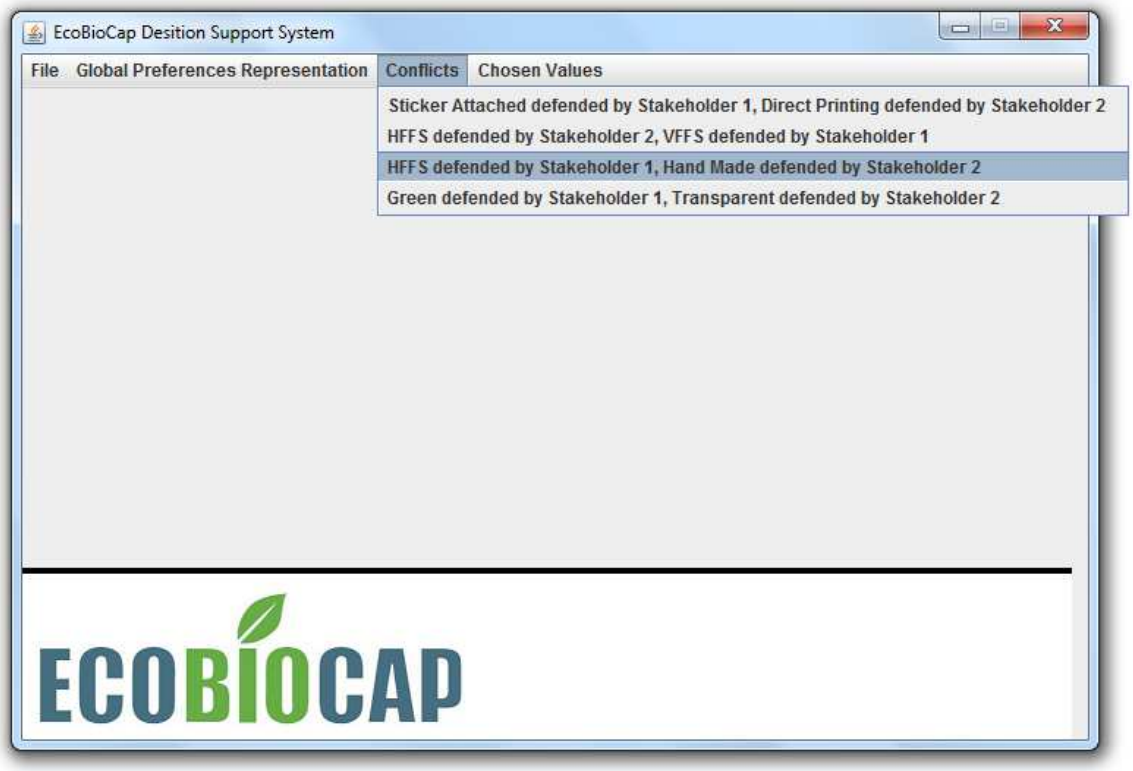

Fig. 3.1: Screenshot of the first version of the prototype, displaying the conflicts that were found.

3.1.1 Criteria

During this phase the user enters the information of the criteria into the system. For each criterion he must specify a name and the list of possible values. The system assigns an unique identifier for each criterion, and it doesn't allow two criteria with the same name.

3.1.2 Stakeholders

In this phase the user enters the information of the stakeholders into the system. This includes the information of their preferences as elicited in the technical surveys. For each stakeholder the user enters a name, the preferences, the Wanted values and the Not Applicable values.

The user is provided with a screen in order to do that. For each stakeholder he will be able to select each criterion at a time in order to specify which values are 'Wanted' and which ones are 'Not Applicable', and also to select a pair of values, one to be the Less Preferred and one to be the Most Preferred.

If for one stakeholder and criterion a value appears in the Wanted list, it cannot appear in the Not Applicable list and vice versa. Also, the Not Applicable list cannot include all the criterion's values. Finally, the stakeholders' preferences cannot determine a cycle as that would be inconsistent.

3.1.3 Implications

Implications have an hypothesis and a conclusion, which are values from different criteria. They also have the list of stakeholders that mentioned the implication, which is not used at the moment in any algorithm and is only for information purposes. Implications are useful to determine the interrelation between the criteria.

The user can now enter the information of the implications into the system. He can also add implications during the Negotiation Phase. If the user enters a new implication that already exists for another stakeholder, then that implication is updated and the new stakeholder is added into the list. This way, we avoid having duplicated implications.

To avoid an inconsistent Knowledge Base, we don't allow a stakeholder to enter implications that determine a cycle between the criteria. After entering all implications, an algorithm is run to detect this kind of cycles. If any, the user is asked to remove some implications until their implications do not determine a cycle between the criteria.

We can have cycles between the criteria due to the aggregation of the implications of different stakeholders, for example (the arrow goes from the hypothesis to the conclusion):

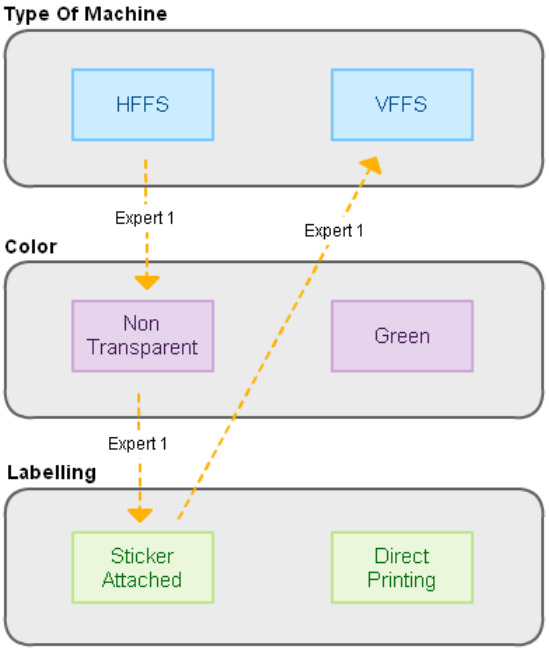

Fig. 3.2: We say we have a cycle between the implications when we can start from one criterion and go back to the same criterion by following the implications between the values, for the same stakeholder. In this case we have a cycle for Expert 1, so his information is inconsistent and we don't allow it.

1. Expert 1 says: With 'HFFS' it is impossible to have a transparent material. Also it is a marketing constraint to use Sticker Attached with no transparent packages as ink is not nice in this kind of packages.

2. Expert 2 says: When we use Sticker Attached we need to use 'VFFS' as it is the only machine that simplifies the process of adding the sticker.

So, as we can find this kind of cycles, algorithms have validations to avoid infinite loops. For example, if we select 'HFFS', then the first implication forces us to select 'Non Transparent'. Then, the second implication forces us to select 'Sticker Attached'. Finally, we don't follow the last implication as 'Type Of Machine' criterion has a chosen value already.

3.2 Global Preferences Representation

In this step the user is ready to generate the Global Preferences Representation. Although in this version of the prototype everything is represented in a text format, the idea for the second version of the system is to represent them with a rich and dynamic graph in the screen. This includes implications and 'Not Applicable' and 'Wanted' values. Each value will be represented as a vertex and each edge will represent a preference from one value over another one.

To represent that two stakeholders have the same preference (for example both of them prefer 'Transparent' over 'Green') we are going to use a single edge between the two values, with the arguments of both stakeholders attached to the edge. Another option could have been to use a directed multi-graph, but we preferred to have all the information together as many stakeholders can participate in the same Negotiation Phase in order to solve a specific conflict.

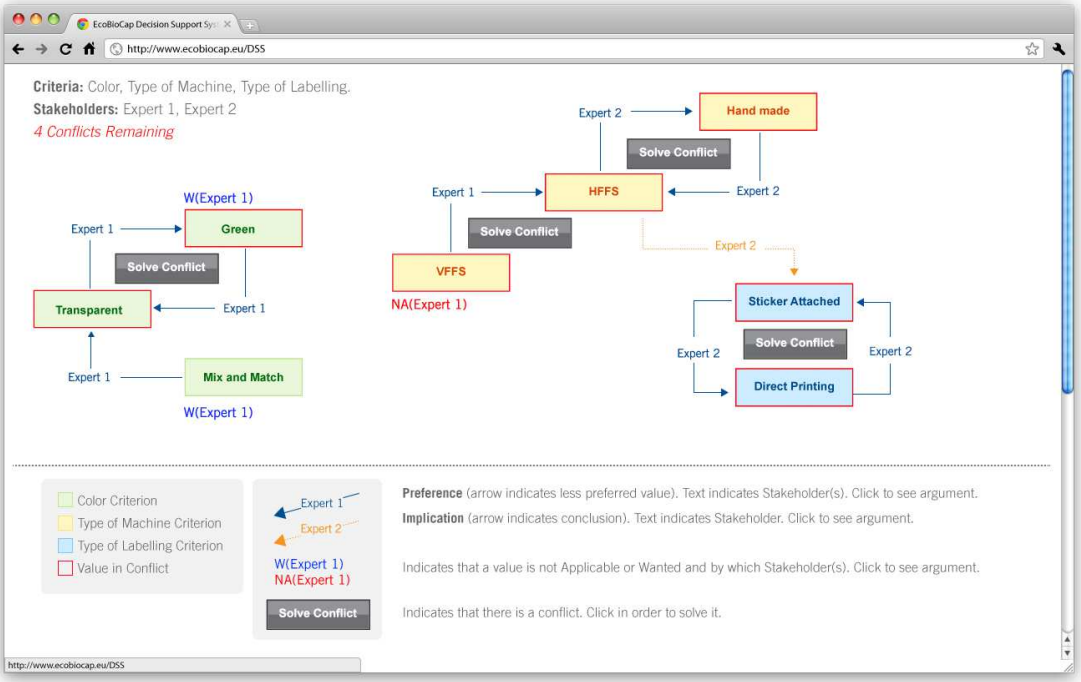

Fig. 3.3: Sketch of Global Preferences Representation.

In Figure 3.3 we can see a sketch of how the Global Preferences Representation could look after implementing a rich user interface for the system. Different colors are used for values of different criteria. Blue arrows represent preferences, and they go from the most preferred value to the less preferred one. Yellow arrows represent implications. Each value show the list of stakeholders that marked it as Not Applicable or Wanted, if any.

3.3 Cycles

Finding cycles within the preferences is very important as it will be used for conflict detection, in a process explained in the following subsection. It is important to notice that we want to show the user absolutely all conflicts within a set of preferences, to let him decide which conflicts he prefers to solve first (solving one conflict may then automatically solve another one, as preferences are changed in this process).

In order to do this we first tried using JGraphT [1]. This framework already provides algorithms for cycle detection [2], but those algorithms were not useful for us as they were returning the cycles with the maximum number of nodes only, and not any smaller cycles inside them.

After analysing several libraries we found a set of Java classes that provided an algorithm for detecting all elementary cycles in a graph with the algorithm of Johnson [13]. In order to do this, it searches for strong connected components, using the algorithm of Tarjan [8]. The constructor gets an adjacency-list of a graph. Based on this graph, it gets a nodenumber s, for which it calculates the subgraph, containing all nodes $\{s, s + 1, ..., n\}$, where n is the highest nodenumber in the original graph (for example it builds a subgraph with all nodes with higher or same nodenumbers like the given node s). It returns the strong connected component of this subgraph which contains the lowest nodenumber of all nodes in the subgraph.

The author of this implementation is Frank Meyer [3], and it was released in March 2009.

3.4 Preference Conflicts

Now conflicts can be detected automatically. In order to do that, the alogirthm analyses all cycles found within the preferences and determines if they represent a conflict or not, as not all cycles are considered as a conflict. Let us illustrate this with some examples.

Notation 2: In the following examples, vertices represent values, and edges represent preferences. Each edge goes from the most preferred value to the less preferred value, and the list of stakeholders holding that preference is notated in the edge.

In Figure 3.4 we can see that there is a cycle between two values, Green and Transparent. As the edges correspond to different stakeholders, we detect a valid conflict between them.

[1] http://www.jgrapht.org
[2] http://jgrapht.org/javadoc/org/jgrapht/alg/CycleDetector.html
[3] web@normalisiert.dot.de

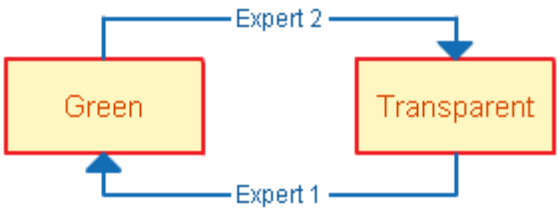

Fig. 3.4: Conflict between Green and Transparent. The arrow goes from the Most Preferred value to the Less Preferred value.

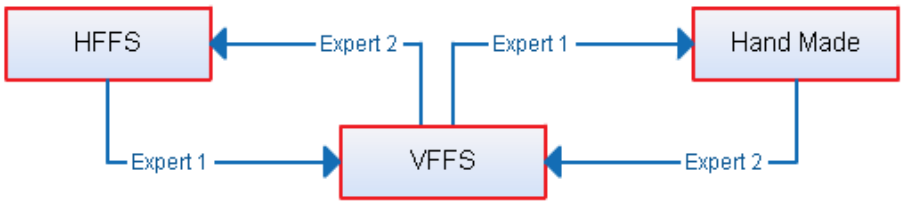

Fig. 3.5: Conflict between HFFS and VFFS, and conflict between VFFS and Hand Made.

Fig. 3.6: Cycle between three values that can be collapsed into a cycle between two values. Therefore, we have a conflict between HFFS and Hand Made.

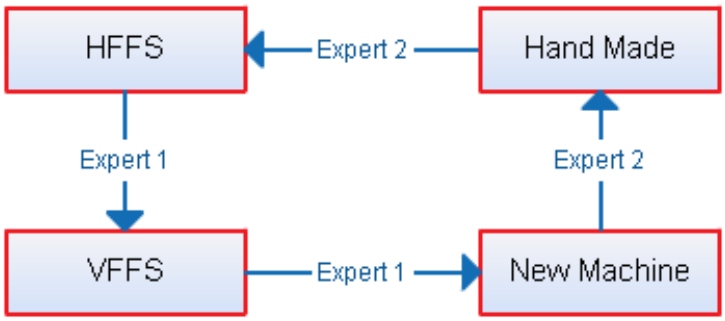

Fig. 3.7: Cycle between four values that can be collapsed to a cycle between two values. Therefore, conflict between HFFS and New Machine.

In Figure 3.5 we can see two different cycles, and again their edges correspond to different stakeholders. Therefore, we detect two valid conflicts, one between HFFS and VFFS, and another one between VFFS and Hand Made.

In Figure 3.6 we see a cycle between three values. In order to inherit a conflict we need to collapse two edges into one, to create a new cycle between two values. As the edges between HFFS and

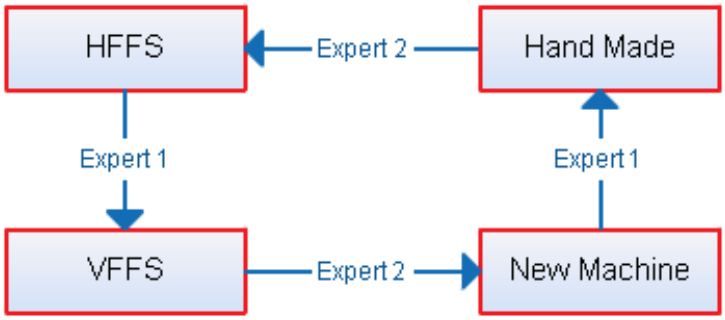

Fig. 3.8: No conflict as we cannot collapse edges to obtain a cycle between two values. The user needs to choose a value manually.

VFFS, and between VFFS and Hand Made both correspond to Expert 1, they can be collapsed into a single edge between HFFS and Hand Made. Therefore, we have a conflict between them as there is an edge between Hand Made and HFFS sustained by Expert 2.

In Figure 3.7 we see a cycle between four values. In this case we can collapse the edge between HFFS and VFFS with the edge between VFFS and New Machine, and we can also collapse the edge between New Machine and Hand Made with the one between Hand Made and HFFS. Therefore, we have a conflict between HFFS and New Machine.

In Figure 3.8 there is not a valid conflict as we cannot collapse consecutive edges corresponding to the same stakeholder. Therefore, we cannot transform the cycle into a cycle between only two values.

The information of 'Wanted' and 'Not Applicable' values is not used at this moment in the algorithms to define conflicts.

The user can decide after detecting conflicts what to do with the nodes marked as 'Not Applicable' or 'Wanted'.

Note: Algorithm for Conflict Detection

The pseudocode for this algorithm and its explanation are available in Appendix D.

3.5 Not Applicable & Wanted Values

Values can be not applicable for a stakeholder but the most preferred one for many others. Because of this, in this implementation we are not taking any automatic decision respecting those values. Instead, we let the user decide what to do with them. They can either remove the Not Applicable tag from a value (and this is then done for all the stakeholders that marked it as Not Applicable) or either remove the value completely from the system.

Users can also decide to remove the Wanted tag from a value, this option is available at the same moment. They cannot remove those values though.

3.6 Conflict Resolution

The user then selects the option to solve a specific conflict, as conflicts need to be resolved one at a time. The Negotiation Phase starts.

Let's focus in the conflict between VFFS and Hand Made in the second example from the previous section:

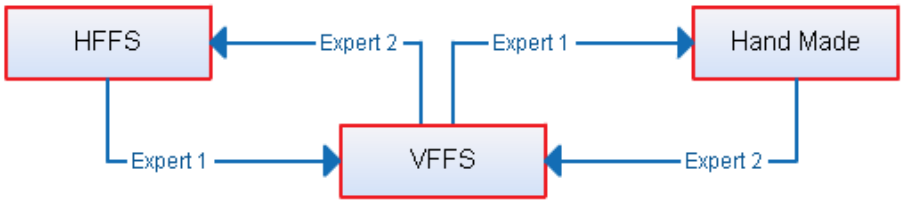

Fig. 3.9: Conflict between HFFS and VFFS, and conflict between VFFS and Hand Made.

Expert 1 says: We prefer to use a machine because when using hand made packaging the atmosphere in the head-space can't be controlled so the cheese quality can't be guaranteed (Argument A).

Expert 2 says: We prefer Hand Made as it is our current way of packaging and VFFS is available but being used for another cheese (Argument B).

So, we have one conflict due to two opposite preferences, supported by two arguments. We call Expert 1's argument as argument A, and Expert 2's argument as argument B. The Argument's Graph is generated:

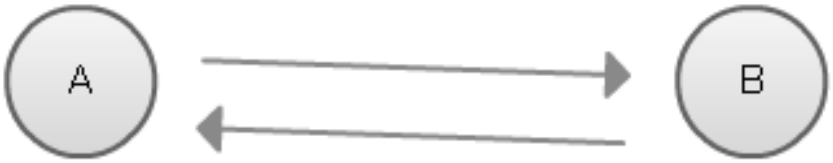

Fig. 3.10: Argumentation Framework

The user is now allowed to enter new arguments into the graphs as attacks to other arguments.

Also, the user can add new implications into the system. Implications are considered in a later phase by the algorithms (when a criterion is analysed in order to select the chosen value), but as in this phase new information can appear, it is a good moment to allow this. In a future implementation, the user will be allowed to add new criteria and preferences.

The user adds new arguments, for example:

A: We prefer Hand Made as it is our current way of packaging and VFFS is available but being used for another cheese.

B: We prefer to use a machine because when using hand made packaging the atmosphere in the head-space can't be controlled so the cheese quality can't be guaranteed.

C: There are new procedures that allow to control the atmosphere in the head-space when using hand made.

D: It is very expensive to train all the employees so they can follow these new procedures.

E: Actually the training is very fast and the implementation too.

In order to see the attacks, refer to Figure 3.11.

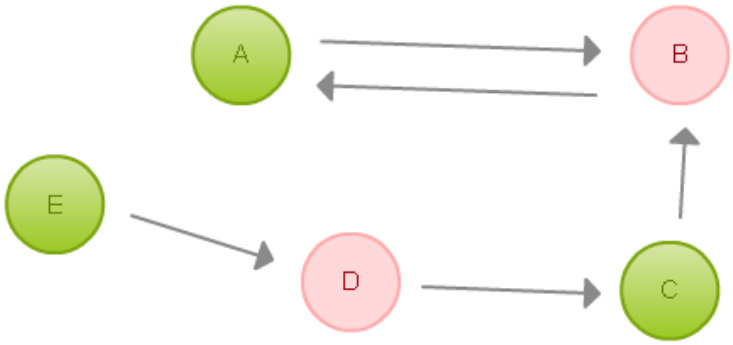

Fig. 3.11: Argumentation Framework

Every time an argument is added, the preferred extensions are calculated. The user is able to proceed to the next step when there is one preferred extension only, and containing one of the original arguments (A or B).

In this case, argument A wins, so the preference supported by argument B changes its direction and the cycle is broken.

The user can enter arguments that attack any other argument. For example, we could add 5 more arguments to attack argument B from different points of view.

Argument D can lead us in a future implementation to add a new criterion called 'Cost' and new preferences. At the moment we will only consider it as an argument to solve this conflict.

If the Negotiation Phase fails because it is impossible to determine which arguments are more important, then the user is able to split the Global Preferences Graph into two scenarios. As in the first implementation this can lead to an exponential number of scenarios, the system will show the user the different options, and after the user chooses one scenario, he will continue working with that one only.

In the above example, one scenario contains the preference supported by argument A and not the preference supported by argument B. The second scenario contains the preference supported by

argument B and not the preference supported by argument A.

Now let's see the example from Figure **??**:

Fig. 3.12: Conflict between HFFS and Hand Made.

So we have one conflict between Expert 1 and Expert 2. Under the closed model set assumption we can assume that transitivity can be applied to preferences. Then we could join both preferences of Expert 1 and infer that he prefers 'HFFS' over 'Hand Made', and support this new preferences using both arguments supporting the two original preferences. So now we have a new preference that is in conflict with Expert 2's preference. We can start a new Negotiation Phase.

After the Negotiation Phase, either Expert 1 can win, or Expert 2 can win. If Expert 1 wins, then Expert 2's preference changes its direction, and the chosen value is 'Hand Made'. If Expert 2 wins, then both Expert 1's preferences are removed and the new preference added in this phase changes its direction, so 'HFFS' becomes the chosen value.

Even if we have a cycle between four or more nodes, if we can apply transitivity to infer new preferences, we can transform them into smaller cycles and solve them in the way mentioned in this section.

In Figure 3.13 we can see a sketch of how the Negotiation Phase could look in the prototype system after developing a rich user interface.

3.7 Argumentation Framework

In order to create argumentation frameworks and find preferred extensions in our implementation, we used an external Java framework called JavaDungAF [4] from the Argumentation Research Group called ARG:dundee [5].

This is an open source framework composed of a few classes that implements the admissible semantics. JavaDungAF's implementation of the admissible semantics essentially comprises three stages:

[4] http://arg.dundee.ac.uk/dungomatic/docs
[5] http://www.arg.dundee.ac.uk

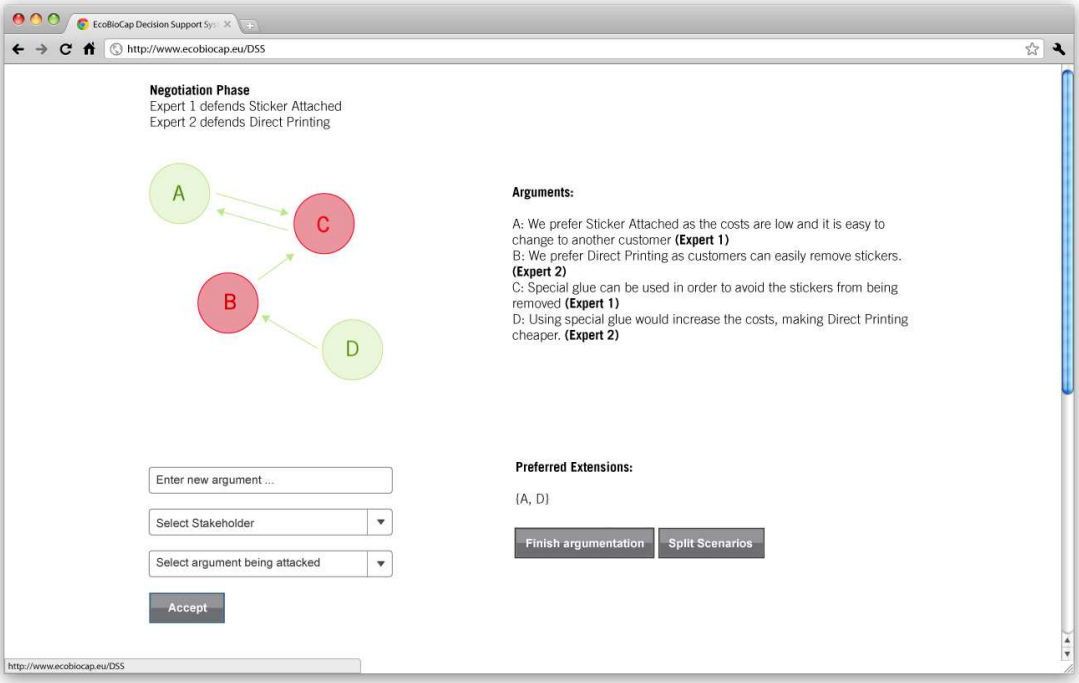

Fig. 3.13: Sketch of a Negotiation Phase.

1. The first stage finds the defence-sets around each argument. An argument's defence-sets are those sets of arguments which (i) include the argument; and (ii) are admissible; and (iii) subsume no other set satisfying (i) and (ii) [15].

2. The second stage uses the set of all defence-sets to find the preferred extensions.

3. The third stage uses the preferred extensions to find the remaining (nonminimal, non-maximal) non-empty admissible sets, if any exist.

While the second and third stages are straightforward, the first stage is less so. DungAF uses a simplified and slightly modified version of Vreeswijk's algorithm to determine the defence-sets around an argument [15]. Vreeswijk's algorithm was designed to generate labelled defence-sets1, and includes a slight error; JavaDungAF's version is simplified in that it generates merely defence-sets, and is modified to address the error.

This set of classes is used in our prototype system in order to generate an argumentation framework for the negotiation phase, with the initial two arguments and attacks. It provides simple methods to add new arguments, attacks, and calculate the preferred extensions.

3.8 Most Preferred Value

When there are no more conflicts for a criterion and scenario, the system chooses the most preferred value, if possible.

As we introduced in Section 2.7, we take advantage of the lack of information about preferences. If one or more stakeholders didn't provide information about their preference between two values, then we assume they can agree with another stakeholder who expressed his preference. This rule is only used to discard some values at the beginning of the algorithm. We say a value is eligible to be the most preferred one only if there are no other values marked as more preferred, regardless the number of stakeholders holding the preferences. After getting the list of eligible values, we don't use this rule anymore and just analyse the information of Not Applicable and Wanted tags, apart from the number of stakeholders defending a value.

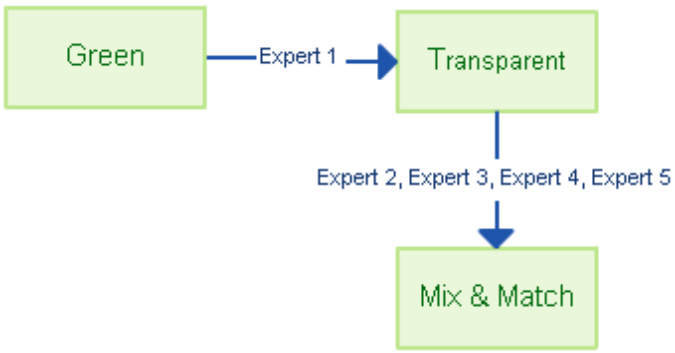

Fig. 3.14: Green is the most preferred value, even though it is preferred by only one stakeholder. We interpret that as the rest of the stakeholders didn't express their preference regarding Green and Transparent, they can agree with Expert 1 about his preference.

In Figure 3.14 we see an example of a criterion with three possible values. Green is considered the most preferred one, even though it is preferred to Transparent only by one stakeholder, while Transparent is preferred to Mix & Match by four stakeholders. We interpret that stakeholders who don't express a preference between two values may agree with the preference express by other stakeholders. Therefore, Experts 2-5 can agree with Expert 1 about his preference.

We also need to check for which criteria we can choose the most preferred value automatically. Even when not all cycles are considered conflicts, we are not going to run the algorithm on criteria that have cycles within their preferences. This decision is because we think those criteria just need more information before we can determine which values are in conflict. In the future we would like to run a new negotiation phase for these cases in order to get that information, but in this first implementation we are going to ask the user to choose the most preferred value manually.

In Figure 3.15 we see another example of a criterion with three possible values, and preferences that determine a cycle. Even though this is not considered as a conflict, as we cannot collapse the edges because they correspond to different stakeholders, this could be a conflict if we had more information. Without more preferences expressed here, we cannot determine which ones are the preferences in conflict.

For the rest of the criteria, we need to check which values are eligible to be the most preferred one. We discard a value when it has implications and their conclusions correspond to criteria which already have a chosen value, and at least one of those chosen values is different than the one implied

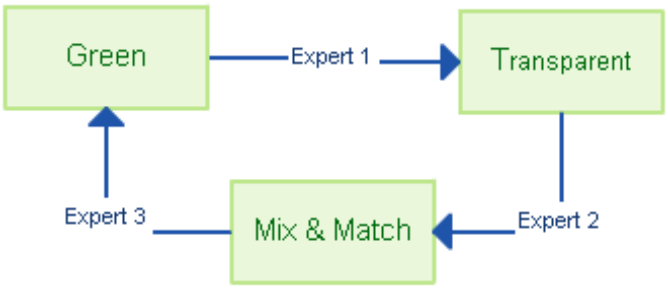

Fig. 3.15: We cannot choose the most preferred value automatically even though this cycle is not considered as a conflict yet.

in the conclusion of an implication.

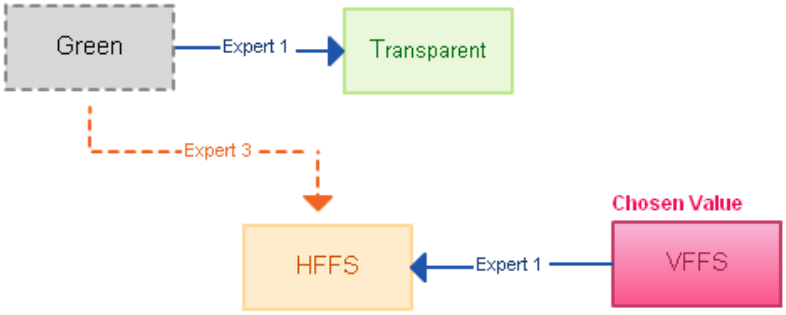

Fig. 3.16: VFFS is the chosen value for the Type Of Machine criterion. As Green implies HFFS, when choosing the most preferred value for Color that value is going to be disabled.

In Figure 3.16 we see two criteria: 'Color' and 'Type Of Machine'. 'Type Of Machine' already has a chosen value, which is 'VFFS'. We can also see that 'Green' implies 'HFFS'. Therefore, when choosing the most preferred value for 'Color', 'Green' is going to be disabled as it implies a value that is different from the already chosen one.

$Green \rightarrow HFFS \Leftrightarrow \neg HFFS \rightarrow \neg Green.$

It is important to note which order is used by the system to automatically choose the most preferred value. As soon as the information of the preferences is loaded and conflicts are detected for the first time, the system tries to choose the most preferred value automatically for the criterion that are not involved in any conflict. This is done in order of appearance of each criterion. If the system finds out that it needs to set a value for another criterion automatically due to an implication, it first informs the user and ask him if he wants to proceed or not. In case of a negative answer, no values are chosen for the first criterion and the related ones. Apart from this, every time a conflict is solved, if the criterion doesn't have conflicts any more, the system tries to find the most preferred value again, following the same idea. At the end, the user is able to set the most preferred value manually for the remaining criteria.

The algorithm also discards a value from the list of eligible ones if their are four grades of implications. In other words, it is discarded if it implies a second one, which implies a third one, which

implies a fourth one, regardless the criteria involved. The idea behind this is that we don't want to set automatically so many values because we would be loosing a lot of information. We think it is okay to have some values chosen automatically but more that four grades is not good because of the loss of information.

Let $I = \{I_1, I_2, I_3, I_4\}$ be a set of implications, where:

$I_1 =<$ Color, Green, Type Of Machine, HFFS, Argument 1 $>$.
$I_2 =<$ Type Of Machine, HFFS, Type Of Labelling, Sticker Attached, Argument 2 $>$.
$I_3 =<$ Type Of Labelling, Sticker Attached, Shelf Life, 40, Argument 3 $>$.
$I_4 =<$ Shelf Life, 40, Price, 5, Argument 4 $>$.

When choosing the most preferred value for the 'Color' criterion we are going to discard 'Green' as it implies a value which implies a third one, which implies a fourth one.

From the list of eligible values, we keep the ones which have the lower number of stakeholders in their Not Applicable list. From these, we keep the ones that have the higher number of stakeholders in their Wanted list. Finally, we pick the one with the higher number of stakeholders in the preferences that involve them directly (as most preferred values). If there is a draw between two or more values, then we ask the user to choose the most preferred value manually as we cannot take a decision automatically.

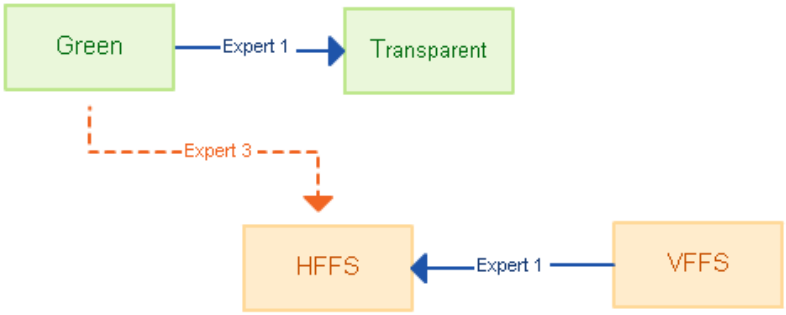

Fig. 3.17: If we choose Green for Color, then HFFS will be automatically chosen, even though it is not the most preferred value for Type Of Machine.

After choosing a value for a criterion, if there are implications involved, the respective values are chosen for the related criteria. In Figure 3.17 we see two criteria: 'Color' and 'Type Of Machine'. As 'Green' implies 'HFFS', if we choose that value for Color, then 'HFFS' will be selected for 'Type Of Machine'.

In Figure 3.18 we see a criterion with four possible values. In order to decide which one is the most preferred, we keep the ones with no incoming arrows only, 'Green' and 'Blue', because no other values are more preferred than them. As none of them are Wanted or Not Applicable, we count the stakeholders that prefer them directly to another value. In this case 'Green' is preferred

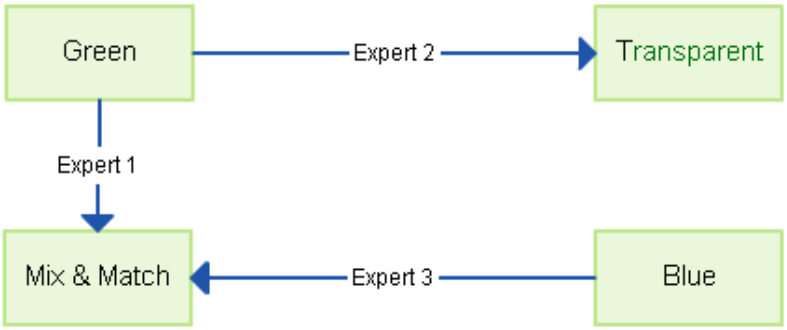

Fig. 3.18: Both Green and Blue don't have incoming arrows, but we choose Green as it is preferred by more stakeholders.

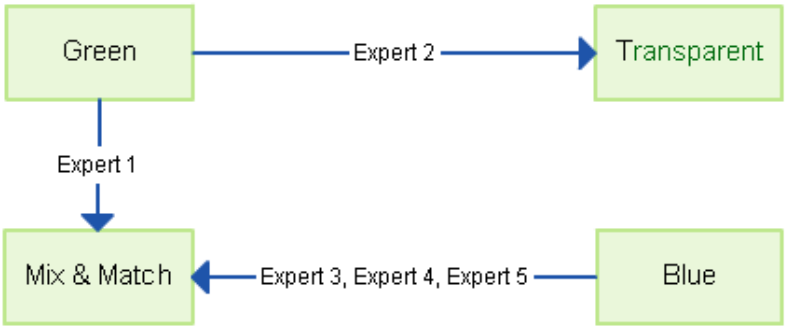

Fig. 3.19: Both Green and Blue don't have incoming arrows, but we choose Blue as it is preferred by more stakeholders.

Fig. 3.20: Green is the chosen value even though Transparent is wanted by Expert 2, because Transparent has an incoming arrow, which means that there is another value more preferred by a stakeholder.

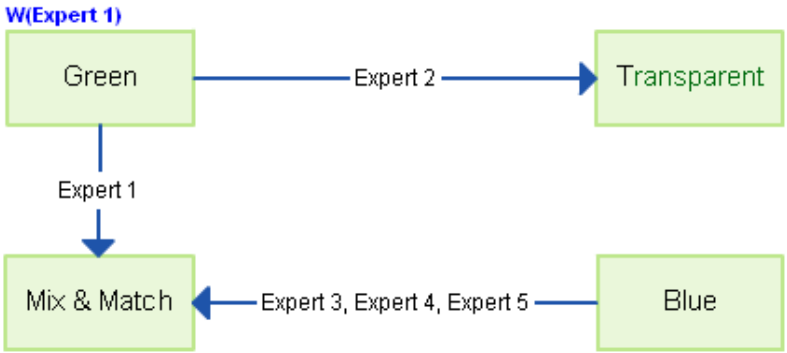

Fig. 3.21: Both Green and Blue don't have incoming arrows. Blue is preferred by more stakeholders, but Green has the higher number of stakeholders in its Wanted list, therefore it is the chosen value.

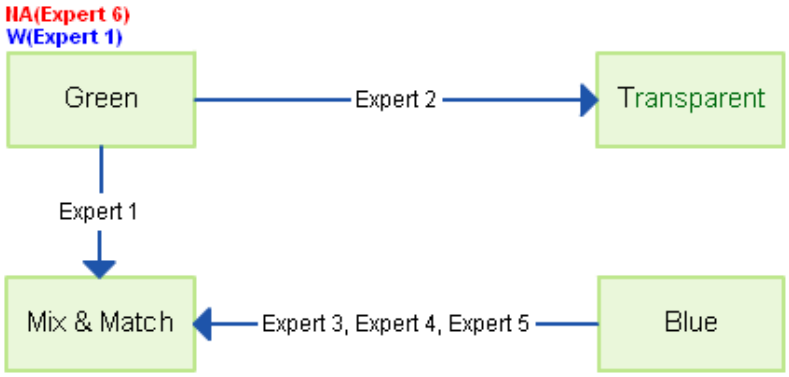

Fig. 3.22: Both Green and Blue don't have incoming arrows. Green has the higher number of stakeholders in its Wanted list, but Blue has the lower number of stakeholders in its Not Applicable list, therefore it is the chosen one.

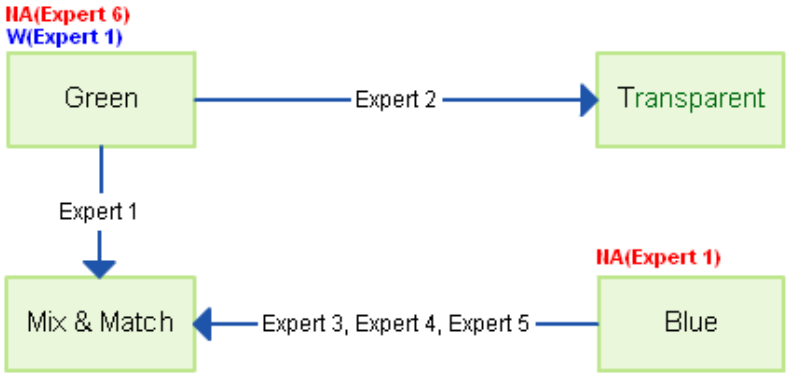

Fig. 3.23: Both Green and Blue don't have incoming arrows and equal number of stakeholders in their Not Applicable list. Therefore, we choose Green as it has the higher number of stakeholders in its Wanted list.

by two stakeholders and 'Blue' only by one, therefore 'Green' is the chosen one for this criterion.

This is not the case of the criterion in Figure **??**, where 'Blue' is preferred for more stakeholders and therefore it is the chosen value.

In Figure 3.21 the criterion is the same one, but 'Green' has a stakeholder in his Wanted list, therefore we don't count the stakeholders supporting 'Green' and 'Blue' as we choose 'Green' directly.

In Figure 3.22 we have the same situation but 'Green' has a stakeholder also in his Not Applicable list. As 'Blue' doesn't have stakeholders in his Not Applicable list, we choose it for this criterion, without counting the stakeholders or analysing the Wanted information.

In the case of Figure 3.23, both 'Green' and 'Blue' have a stakeholder also in their Not Applicable list. Therefore, we analyse the Wanted information and choose 'Green' as the most preferred one.

Note: Algorithm for Most Preferred Value

The pseudocode for this algorithm and its explanation are available in Appendix **??**.

3.9 Decision's Information

During the execution of the system, all the information is stored in the Log. There, the user can find all the decisions that were taken, either automatically by the system, or manually by him. If a scenario was splitted or new arguments were added, all this will be presented in the Log.

Apart from this log, in the screen were the chosen values are presented, the user can also see the reasons. A value may have been chosen manually, or automatically because of an implication, or automatically by the system after the user solved all the conflicts.

After choosing a value for each criterion, the result is presented to the user. This includes the Decision Tree, which shows what happened during the execution. All new arguments added by the user, attacks, new implications, split of scenarios, etcetera, are displayed in order to show what happened. This can be a graphical or a text representation.

In a future implementation where multiple parallel scenarios will be allowed, the information about the scenarios will also be presented in a tree, with many nodes showing all the scenarios and resolutions. In this implementation, a single tree is presented with the single scenario.

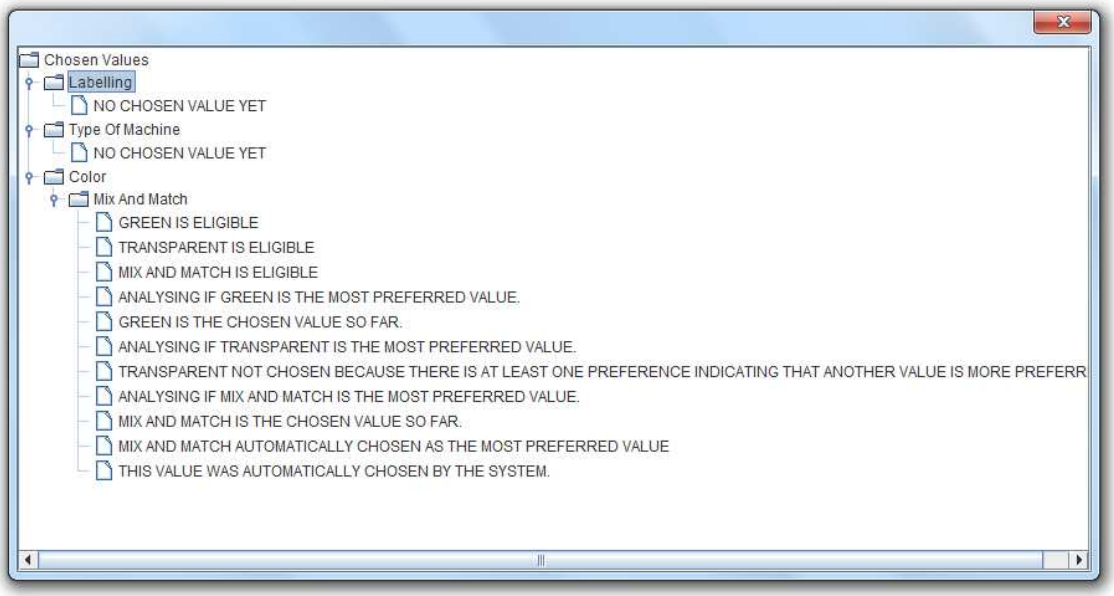

Fig. 3.24: Example of the screen that displays the chosen values for a scenario and the reasons.

3.10 Evaluation

Shown to experts.

4. CONCLUSIONS & PERSPECTIVES

The main contribution of the paper lies in the presentation of a real aggregation scenario and the use of preference logic and negotiation phase based in argumentation for addressing this concrete problem.

In order to use the ASPIC argumentation model we needed to define a preference logic according to the scenario at hand. After defining the syntax and semantics of this logic we then proved that transitivity does not hold if certain assumptions are not made, limiting the inference capabilities of our argumentation system.

We also developed a prototype system and validated it in terms of decision support for EcoBioCap.

4.1 Future Work

The current and future work directions are numerous:

1. We are interested in analysing the possible weaknesses of ASPIC mentioned by Amgoud in [1] and rebutted by Prakken and Modgil in [11].

2. We think it is possible to infer a ranking between the criteria directly from the arguments. For example, we could find constraints in the stakeholders' preferences. That raking could help us to decide between different values in some situations.

3. In the current implementation, implications goes from one value from one criterion, to another value in a different criterion. We are interested in extending this concept in order to be able to talk about groups of values implications groups of values.

4. We would like to develop a richer user interface for the prototype system, to display the graphs of preferences and arguments in a better format. Also, we would like to make it a multi user application. All this is important to have a dynamic interesting application in order to encourage discussions.

5. Not Applicable and Wanted tags are used in the algorithms for choosing the most preferred value for a criterion. Despite that, we would like to analyse other ways of using that information, considering the number of stakeholders wanting or discarding a value.

6. For the EcoBioCap projects, customers from different countries were asked for there preferences. We would like to use this information in the future, as hundreds of customer may express opposite preferences than the ones expressed by the cheese or package makers. Also, we are interested in defining the strength of the preferences of each part.

7. For the conflict detection we are using a direct graph to store values and preferences. In other words, if many stakeholders hold a preference, a single edge is created. We would like to use a multi-graph instead to allow multiple edges between pair of values and analyse the impact in the performance and behaviour of the conflict detection algorithms.

8. For creating the algorithms we didn't focus in their complexity or performance. We would like to review them and update them if necessary.

9. At the moment, a single value is returned for each criterion and scenario. We are interested in returning a ranked list of preferred values instead, in order to provide more information to the Decision Support System.

10. Whenever we have a cycle that is not yet considered a conflict, we would like to run a Negotiation Phase. That way we could retrieve more information to help to determine which ones are the values in conflict.

APPENDIX

A. ABSTRACT DATA TYPE

A.1 TAD preferences

1. CriterionName: String
2. Wanted: Set(value)
3. NA: Set(value)
4. Pref: Set(Preference)

For example:

$<'Color',\{'Green','Mix\&Match'\},\{\},\{\{'Green','Transparent'\},\{'Mix\&Match','Transparent'\}\}>$

A.2 TAD preference

1. PreferredValue: value
2. LessPreferredValue: value
3. Argument: String

For example:

$<'Transparent','Green'$,'Because customers want to see the cheese' $>$ implies that Transparent is preferred over Green.

A.3 TAD value

1. Name: String

This is a value for a criterion, for example: HFFS, Transparent, 'Sticker Attached'. Values must be unique and cannot be repeated in different criteria.

A.4 TAD stakeholder

1. StakeholderName: String
2. Pref: Set(preferences)

For the preferences, we are going to have one item for each criterion.

A.5 TAD implication

1. Hyphotesis: value
2. Conclusion: value
3. StakeholderNameList: List(String)
4. Argument: String

For example: $<'Transparent','HFFS','Expert1'$, 'There is a dependency between the machine and the color' $>$ means that selecting 'Transparent' implies also selecting 'HFFS', for Expert 1, because of the reason mentioned.

A.6 TAD criterion

1. CriterionName: String
2. PossibleValues: Set(value)

For example: $<'Color''Transparent','Green','Mix\&Match'>$

A.7 TAD ecobiocapdata

1. StakeHolderList: Set(stakeholder)
2. CriterionList: Set(criterion)
3. ImplicationList: Set(implication)

B. OBJECT MODEL

In order to see the Object Model, please refer to the annexed document ‘ObjectModel.pdf’. That document presents the class diagram with their relations and attributes.

C. INPUT FILES

The input file for the prototype system is a XML file, with the format described in Figure C.1. As we can see there, we first request the list of criteria with their possible values, each one with an ID and Name.

After that, we expect the list of stakeholders. Each stakeholder must have an ID, a Name, and a list of Preferences. Each Preferences must have a unique Criterion ID, a list of Not Applicable values, a list of Wanted values and a list of Preference.

Each Preference presents the Preferred Value, the Less Preferred Value and the ID of an argument.

After this, we expect the list of Implications, each of them with the Conclusion Criterion ID, the Conclusion Value ID, the Hyphotesis Criterion ID, the Hyphotesis Value ID and the list of Stakeholders' IDs.

Finally, we need to complete the total list of Arguments, each of them with an ID, the text and the Stakeholder's ID that mentioned it.

```
▼<EcoBioCapData>
  ▼<CriterionList>
    ▼<Criterion>
       <Id/>
       <Name/>
      ▼<PossibleValueList>
        ▼<Value>
           <Id/>
           <Name/>
         </Value>
        ▼<Value>
           <Id/>
           <Name/>
         </Value>
       </PossibleValueList>
     </Criterion>
   </CriterionList>
  ▼<StakeholderList>
    ▼<Stakeholder>
       <Id/>
       <Name/>
      ▼<PreferencesList>
        ▼<Preferences>
           <CriterionId/>
          ▼<NotApplicableList>
            ▼<NotApplicable>
               <Id/>
               <ArgumentId/>
             </NotApplicable>
           </NotApplicableList>
          ▼<WantedList>
            ▼<Wanted>
               <Id/>
               <ArgumentId/>
             </Wanted>
           </WantedList>
          ▼<PreferenceList>
            ▼<Preference>
               <PreferredValue/>
               <LessPreferredValue/>
               <ArgumentId/>
             </Preference>
            ▼<Preference>
               <PreferredValue/>
               <LessPreferredValue/>
               <ArgumentId/>
             </Preference>
           </PreferenceList>
         </Preferences>
       </PreferencesList>
     </Stakeholder>
   </StakeholderList>
  ▼<ImplicationList>
    ▼<Implication>
       <ConclusionCriterionId/>
       <ConclusionValueId/>
       <HyphotesisCriterionId/>
       <HyphotesisValueId/>
      ▼<StakeholderList>
         <Id/>
         <Id/>
       </StakeholderList>
     </Implication>
   </ImplicationList>
  ▼<ArgumentList>
    ▼<Argument>
       <Id/>
       <ArgumentText/>
       <StakeholderId/>
     </Argument>
    ▼<Argument>
       <Id/>
       <ArgumentText/>
       <StakeholderId/>
     </Argument>
   </ArgumentList>
</EcoBioCapData>
```

Fig. C.1: XML Input File.

D. CONFLICT DETECTION ALGORITHM

This algorithm analyses a cycle and determines if we can infer one or more conflicts from it. We assume that we already have the list of cycles within the preferences of one single criterion, and that the cycle is between three or more vertices (as we are using a different and much more simpler algorithm when the cycle is between two nodes).

We can infer a conflict from a cycle only when we can collapse the edges of the cycle until it has only two vertices. To be able to collapse two edges, they must be consecutive and be supported by the same stakeholder (as an edge can be supported by many stakeholders, we only require that at least one stakeholder is repeated for both edges). This is because we assume that if a stakeholder prefers A to B and B to C, then by transitivity he prefers A to C.

The idea behind this algorithm is that a cycle has a conflict only when it can be divided into two consecutive paths, one supported by one stakeholder, and the other one supported by a different one. Please notice that this algorithm has some complexity because we may find different ways of splitting a cycle like that, by looking at the different stakeholders supporting the edges.

What we are going to do is to start from one edge, read its supporters, and for each of them go through the rest of the edges trying to determine a point where the previous edges are supported by one supporter and the following by a new one. It is important to notice that if there is a conflict, then no matter where is the starting point of the two parts, we can start the analysis from any edge, as at least one of its supporters must be supporting one of the parts. If none of the supporters of an edge are supporting one of the two parts, then we are not going to have a conflict.

So, during the algorithm we keep some variables in order to know if we are in the possible first path, or if we moved into the second one, and the list of potential supporters for the second one. As we start our analysis from one specific edge and we repeat the process for each one of its supporters, then if there is a conflict it is going to be between the current supporters and the ones supporting the second path.

Between lines 1 and 2 we get the first edge and its supporters. Then we ran a for in order to determine for each of them if we can divide the cycle into two paths, and make him the supporter of the first one, and find one or more that supports the second one. Inside the for, the current stakeholder will be called the OriginalStakeholder.

From lines 4 to 6 we initialize some variables. We assume we have a conflict and try to prove that we don't. We create two variables to store the list of supporters for the second path. With are going to work with the first one in order to determine which supporters are not there anymore when we move to the next edge, and we need the second one to store the supporters in order to access them when creating the conflict definitively.

Then, we go through the rest of the edges in the for starting at line 7. From lines 8 to 11 we get a new edge and its supporter list and we create a variable that indicates if in this new edge we can still find the OriginalStakeholder or not (line 13).

From lines 13 to 17 we analyse if we moved to the second path or not. If the OriginalStakeholder is not there, and we didn't move to the second path before, then we probably just moved to it. Therefore, we add the stakeholders from the current edge into the two supporter lists.

From lines 18 to 44 we analyse another option: if we already are in the second path, then we need to see which ones of the supporters found in the first edge of the second path are still here. From lines 19 to 32 we use the supporter list to see if anyone of them is also in this edge, and we are going to keep only the ones that we find. This is very useful as if at any edge we cannot find any more supporters from the supporter list, then we cannot infer a conflict from the cycle as it cannot be divided into two consecutive paths support by different stakeholders.

From line 33 to 36 we analyse that exactly. If we don't have any more supporters and the OriginalStakeholder is not present, then we don't have a conflict. It is important to ask for the OriginalStakeholder too as we could have moved back to the first path. Remember we started the analysis from a random edge, so we could have gone to the second path and back to the first one. This is analysed between lines 37 and 39.

Finally, between lines 40 and 43 we add our current supporters to the other supporter list, in order to have that ready for the moment we actually need to create a conflict. The creation of the conflict is not shown here, but it is a simple process where we set which stakeholders support each part, which vertices are involved in each part, etcetera.

```
input : List<Vertex> cycle
output: Conflicts detected if any.
1  edge1 ⟵ GetEdgesBetweenVertices(cycle[0], cycle[1])
2  stakeholderList1 ⟵ edge1.GetStakeholderList()
3  for originalStakeholderId ∈ stakeholderList1 do
4      isConflict ⟵ true, secondHalf ⟵ false
5      vertexFirstPartStarts, vertexSecondPartStarts ⟵ 0
6      supporterList, finalSupporterList ⟵ ∅
7      for i ← 1 to cycle.size() do
8          previousVertex ⟵ cycle[i]
9          nextVertex ⟵ cycle[i + 1]%cycle.size()
10         edge ⟵ GetEdgesBetweenVertices(previousVertex, nextVertex)
11         stakeholderList ⟵ edge.GetStakeholderList()
12         originalPresent ⟵ stakeholderList.Contains(originalStakeholderId)
13         if originalPresent == false and secondHalf == false then
14             secondHalf ⟵ true, vertexSecondPartStarts ⟵ i
15             supporterList.addAll(stakeholderList)
16             finalSupporterList.addAll(stakeholderList)
17         end
18         else if secondHalf then
19             anySupporters ⟵ false
20             for j ← 0 to supporterList.size() do
21                 supporter ⟵ supporterList[j]
22                 if supporter > 0 then
23                     supporterStillThere ⟵ false
24                     for stakeholderId ∈ stakeholderList do
25                         if supporter == stakeholderId then
26                             supporterStillThere, anySupporters ⟵ true
27                             break
28                         end
29                     end
30                     if !supporterStillThere then supporterList.set(j, 0)
31                 end
32             end
33             if secondHalf and !anySupporters and !originalPresent then
34                 isConflict ⟵ false
35                 break
36             end
37             else if secondHalf and !anySupporters and originalPresent then
38                 vertexFirstPartStarts ⟵ i
39             end
40             else if secondHalf and anySupporters then
41                 finalSupporterList.clear()
42                 finalSupporterList.addAll(supporterList)
43             end
44         end
45         if isConflict then CreateConflict()
46     end
47 end
```

Algorithm 1: Conflict Detection

E. CHOSEN VALUE ALGORITHM

This algorithm tries to determine the most preferred value for a Criterion. It is ran when there are no more conflicts within its preferences. In the input we receive a list of eligible values, this are the values that don't have any problems regarding the implications in which they are involved. The details about this is explained in Section 3.8.

In line 1 we return 0 if we don't have eligible values, as we cannot choose the most preferred value automatically. In line 2 we also return 0 if there are cycles within the preferences. The reason for this is also explained in Section 3.8.

From line 3 to 7 we initialize some variables. We are going to use them to keep the information of the most preferred value so far in every step of the algorithm. The idea is to go through all the eligible values, and compare each one with the best one so far. As the criterion to choose a value over another one depends on the number of direct supporters (these are the stakeholders that prefer the value over another one directly, with no transitivity), the stakeholders in the Wanted and Not Applicable lists, we are going to have variables for all these factors.

In line 8 we start a for to iterate through the eligible values. In line 9 we determine if there exists any preference that indicate that another value is more preferred than the one we are analysing, even by a single stakeholder. Is this is the case, we move to the next value.

From line 11 to 13 we get the information we need from the value we are analysing and in line 14 we create a variable that will indicate if this value is better than the most preferred one so far. This is determined between lines 15 and 29:

Lines 15 to 17 chooses the value if the number of Not Applicable is lower than the current one, lines 18 to 20 discard the value if the number of Not Applicable is higher than the current one, lines 21 to 22 chooses the value if the number of Wanted is higher than the current one (at this moment we know that the number of Not Applicable is equal), lines 24 to 26 discards the value if the number of Wanted is lower than the current one, and lines 27 to 29 chooses the value if the number of direct supporters is higher than the current one.

Lines 30 to 32 determine that the value we are analysing is equally good as the chosen one so far. This last case means that we cannot determine the most preferred value automatically and therefore we return 0.

From lines 33 to 39 we update the most preferred value so far with the information of the value we were analysing, if it is better.

As the variable chooseThisValue is initialized in false, some steps that set it again to false are not necessary, but we decided to keep the algorithm this way to make it easier to understand, as we are not worried about performance issues.

```
input  : int criterionId, List<int> eligibleValues
output: The chosen value for the criterion, if it could be determined.
1  if eligibleValues.isEmpty() then return 0
2  if GetCyclesFroCriterion.size() > 0 then return 0
3  currentChosenValue ⟵ 0
4  currentStakeholderCount ⟵ 0
5  moreThanOneMatch ⟵ false
6  currentWantedBy ⟵ 0
7  currentNotApplicableBy ⟵ GetStakeholderList().size() + 1
8  for valueId ∈ eligibleValues do
9      anotherValueMorePreferred ⟵ IsAnotherValueMorePreferred(valueId)
10     if anotherValueMorePreferred == false then
11         wantedBy ⟵ GetWantedIdList(valueId).size()
12         notApplicableBy ⟵ GetNotApplicableIdList(valueId).size()
13         stakeholderCount ⟵ GetStakeholderCount(valueId)
14         chooseThisValue ⟵ false
15         if notApplicableBy < currentNotApplicableBy then
16             chooseThisValue ⟵ true
17         end
18         else if notApplicableBy > currentNotApplicableBy then
19             chooseThisValue ⟵ false
20         end
21         else if wantedBy > currentWantedBy then
22             chooseThisValue ⟵ true
23         end
24         else if wantedBy < currentWantedBy then
25             chooseThisValue ⟵ false
26         end
27         else if stakeholderCount > currentStakeholderCount then
28             chooseThisValue ⟵ true
29         end
30         else if stakeholderCount == currentStakeholderCount then
31             moreThanOneMatch ⟵ true
32         end
33         if chooseThisValue then
34             moreThanOneMatch ⟵ false
35             currentChosenValue ⟵ valueId
36             currentStakeholderCount ⟵ stakeholderCount
37             currentWantedBy ⟵ wantedBy
38             currentNotApplicableBy ⟵ notApplicableBy
39         end
40     end
41 end
42 if moreThanOneMatch then return 0
43 return currentChosenValue
```

Algorithm 2: Chosen Value

BIBLIOGRAPHY

[1] Leila Amgoud. Five weaknesses of aspic+ (regular paper). In Greco Salvatero, editor, *International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU), Italy, 09/07/2012-13/07/2012*, pages 122–131, http://www.springer.com, juillet 2012. Springer Berlin / Heidelberg.

[2] Leila Amgoud and Srdjan Vesic. A new approach for preference-based argumentation frameworks. *Annals of Mathematics and Artificial Intelligence*, 63(2):149–183, December 2011.

[3] Meghyn Bienvenu, Jérôme Lang, and Nic Wilson. From preference logics to preference languages, and back. In *KR*, 2010.

[4] Martin Caminada and Leila Amgoud. An axiomatic account of formal argumentation. In *AAAI*, pages 608–613, 2005.

[5] Madalina Croitoru, Jerome Fortin, and Nir Oren. Arguing with preferences in ecobiocap. In *Proceedings of the 4th International Conference on Computational Models of Argument*, page to appear, 2012.

[6] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence Journal*, 77:321–357, 1995.

[7] Sven Ove Hansson. What is ceteris paribus preference? *Journal of Philosophical Logic*, 25(3):pp. 307–332, 1996.

[8] Donald B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4(1):77–84, 1975.

[9] Souhila Kaci and Leendert van der Torre. Preference-based argumentation: Arguments supporting multiple values. *Int. J. Approx. Reasoning*, 48(3):730–751, aug 2008.

[10] Sanjay Modgil. Reasoning about preferences in argumentation frameworks. *Artif. Intell.*, 173(9-10):901–934, June 2009.

[11] H. Prakken and S. Modgil. Clarifying some misconceptions on the aspic+ framework. *Computational Models of Argument. Proceedings of COMMA 2012*, 2012.

[12] Henry Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2):93–124, 2011.

[13] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[14] Johan van Benthem, Patrick Girard, and Olivier Roy. Everything else being equal: A modal logic for *ceteris paribus* preferences. *J. Philosophical Logic*, 38(1):83–125, 2009.

[15] Gerard A. W. Vreeswijk. An algorithm to compute minimally grounded and admissible defence sets in argument systems. In *Proceedings of the 2006 conference on Computational Models of Argument: Proceedings of COMMA 2006*, pages 109–120, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.