A formal framework for handling conflicting desires

Leila Amgoud

Institut de Recherche en Informatique de Toulouse (IRIT)
118, route de Narbonne, 31062 Toulouse, France
amgoud@irit.fr
http://www.irit.fr/recherches/RPDMP/persos/Leila/index.html

Abstract. This paper combines ideas from argumentation [1, 8] with desires and planning rules, in order to give a formal account of how consistent sets of intentions can be obtained from a conflicting set of desires. We show how conflicts may arise between desires and we resolve them. We argue that the set of desires can be clustered in three categories: i) the *intentions* of the agent, ii) the *rejected desires* and iii) the *desires in abeyance*. Finally, we show that the use of argumentation with desires is different from the usual kind of argumentation with default rules.

1 Introduction

An increasing number of software applications are being conceived, designed, and implemented using the notion of autonomous agents. These applications vary from email filtering, through electronic commerce, to large industrial applications. In all of these disparate cases, however, the notion of autonomy is used to denote the fact that the software has the ability to decide for itself which goals it should adopt and how these goals should be achieved.
Different architectures have emerged as candidates for studying these agent-based systems [2–4, 6, 12, 10, 11]. One of these architectures regards the system as a rational agent adopting certain *mental* attitudes: the beliefs (B), the desires (D) and the intentions (I). An agent can have contradictory desires. However, its intentions are a coherent subset of desires which the agent is committed to achieve.
In [5], the authors explored principles governing rational balance among an agent's beliefs, goals, actions and intentions. In [10] Rao and Georgeff showed how different rational agents can be modeled by imposing certain conditions on the persistence of an agent's beliefs, desires or intentions (the BDI model). In decision theory, Pearl [9] illustrated how planning agents are provided with goals – defined as desires together with commitments – and charged with the task of discovering (or performing) some sequence of actions to achieve those goals.
Most of formalizations are sophisticated enough to handle many aspects of BDI agents. However, they do not show how agent's intentions are calculated from the whole set of its desires. In other words, it is not clear how an agent chooses

a subset of its possibly contradictory desires.
Inspired from work on argumentation theory, we present in this paper a framework for handling contradictory desires. We will show that the problem can be formulated exactly as in argumentation theory and then we show that we can use all the techniques of argumentation to resolve it.
We suppose that an agent is equipped with a base of desires, a base of plans to carry out in order to achieve the desires (we are not interested in the way in which these plans are generated), and finally a knowledge base. Then we show how conflicts can emerge between desires, how to formalize these conflicts and finally how to solve them to obtain the intentions of the agent. Let's illustrate our purposes by the following example.

Example 1 *X is an agent who has the two following desires:*

1. *To go on a journey to central Africa. (jca)*
2. *To finish a publication before going on a journey. (fp)*

Let's suppose that the following plans to achieve the above desires are generated:

$$\begin{cases} t \wedge vac \rightarrow jca \\ w \rightarrow fp \\ ag \rightarrow t \\ fr \rightarrow t \\ hop \rightarrow vac \\ dr \rightarrow vac \end{cases}$$

with: t = "to get the tickets", vac = "to be vaccinated", w = "to work", ag = "to pass to the agency", fr = "to have a friend who may bring the tickets", hop = "to go to a hospital", dr = "to go to a doctor".

For example, the rule $t \wedge vac \rightarrow jca$ means that to go on a journey in central Africa, the agent X should get tickets and should be vaccinated. The rule $w \rightarrow fp$ expresses that the agent should work in order to finish his paper. To get tickets, the agent can either pass to an agency or ask a friend of him to get them. Similarly, to be vaccinated, the agent has the choice between going to a doctor *or* going to a hospital. *In these two last cases, the agent has two plans for each action.*
Note that getting tickets *and* being vaccinated *become two* sub-desires *of the agent with each one having its own plans.*
In addition to the set of plans and the set of desires, the agent may have also another base containing its knowlege and some integrity constraints. In our example, we have:

$$\begin{cases} w \rightarrow \neg ag \\ w \rightarrow \neg dr \end{cases}$$

These two rules means that if the agent works, he can neither pass to an agency nor go to a doctor. Consequently, the plans of its two initial desires are conflicting.

Of course, it would be ideal if all the desires can become intentions. As our example illustrates, this is not always the case. In this paper we will answer to the following questions: which desire will become an intention of the agent, and with which plan*?*

2 Argumentation frameworks

Argumentation is a reasoning model based on the construction of arguments and counter-arguments (or defeaters) followed by the selection of the most acceptable of them.

In Dung's work [7, 8], an argumentation framework is defined as a pair consisting of a set of arguments and a binary relation representing the defeasibility relation between arguments. Here, an argument is an abstract entity whose role is only determined by its relation to other arguments. Then its structure and its origin are not known.

Definition 1 ([8]). *An* argumentation framework *is a pair* $< \mathcal{A}, \mathcal{R} >$ *where* $\mathcal{A}$ *is a set of arguments and* $\mathcal{R}$ *is a binary relation representing a defeasibility relationship between arguments, i.e.* $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$. *(A, B)* $\in \mathcal{R}$ *or equivalently "*$A\ \mathcal{R}\ B$*" means that the argument A defeats the argument B. We also say that A and B are in conflict.*
An argumentation framework is finitary *iff for each argument A there are finitely many arguments which defeat A.*

Definition 2. *Let* $< \mathcal{A}, \mathcal{R} >$ *be an argumentation framework, and* $S \subseteq \mathcal{A}$.

- *S is* conflict-free *iff* $\nexists\ A, B \in S$ *such that* $A\ \mathcal{R}\ B$.
- *S* defends *A iff* $\forall\ B \in \mathcal{A}$, *if* $B\ \mathcal{R}\ A$ *then* $\exists\ C \in S$ *such that* $C\ \mathcal{R}\ B$.

Each defeasibility relation leads to an argumentation framework. Defeating arguments can in turn be defeated by other arguments so we need to define a notion of the *status* of arguments. This notion of *status* is the central element of any argumentation framework. Its definition takes as input the set of all possible arguments and their mutual relations of defeat, and produces as output a division of arguments into three classes of arguments:

- The class of *acceptable arguments*. They represent the "good" arguments. In the case of handling inconsistency in knowledge bases, for example, the formulas supported by such arguments will be inferred from the base.
- The class of *rejected arguments*. They are those arguments defeated by acceptable arguments. Such arguments would not be considered in the process of inference from a knowledge base, for example.
- The arguments which are neither acceptable nor rejected are gathered in the so-called class of *arguments in abeyance*.

Note that to define the rejected arguments and the arguments in abeyance of a given argumentation framework, we first need to determine the set of acceptable arguments of that framework. For that purpose Dung suggested several semantics: *basic semantics* and *preferred semantics*.

Definition 3. *Let* $< \mathcal{A}, \mathcal{R} >$ *be an argumentation framework, and* $S \subseteq \mathcal{A}$.

- S *is a* preferred extension *iff* S *is maximal (for set inclusion), conflict-free and it defends all its elements.*
- S *is a* basic extension *iff* S *is the least fixpoint of the function* $\mathcal{F}(S) = \{A \in \mathcal{A} — A$ *is defended by* $S\}$.

3 Basic definitions

In this section, we present the basic notions of a framework for handling an agent's desires: a logical language, a desire, an action, a realization tree of a desire and finally defeasibility between actions.

3.1 Logical language

Let $\mathcal{L}$ be a propositional language. $\vdash$ denotes classical inference and $\equiv$ denotes logical equivalence. Each agent is equipped with three bases $<\mathcal{D}, \mathcal{P}, \Sigma>$ such that:

- $\mathcal{D}$ contains formulas of $\mathcal{L}$. The elements of $\mathcal{D}$ represent the initial desires of the agent. For example, an agent may have the following desires: *to finish a publication*, *to go to a dentist*, etc... Note that the set $\mathcal{D}$ may be inconsistent. This means that an agent is allowed to have contradictory desires.
- $\mathcal{P}$ is considered as a base of plans. It contains formulas having the form $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow h$ where $\varphi_1, \ldots, \varphi_n$, h are literals of $\mathcal{L}$. Such a formula means that to achieve h, the agent should realize $\varphi_1, \ldots, \varphi_n$.
- Σ contains formulas of $\mathcal{L}$. They represent the knowledge of the agent and some integrity constraints.

Example 2 *In example 1, the agent has the following bases:*

$$\mathcal{D} = \{jca, fp\},\ \mathcal{P} = \left\{ \begin{array}{cl} t \wedge vac & \rightarrow jca \\ w & \rightarrow fa \\ ag & \rightarrow t \\ fr & \rightarrow t \\ hop & \rightarrow vac \\ dr & \rightarrow vac \end{array} \right.$$

$$\text{and } \Sigma = \left\{ \begin{array}{l} w \rightarrow \neg ag \\ w \rightarrow \neg dr \end{array} \right.$$

3.2 The notion of desire

A desire is either an element h of $\mathcal{D}$ or a sub-desire of h. In example 1, the desire of going on a journey to central Africa has two sub-desires which are:

- getting the tickets
- being vaccinated

Let's note that the sub-desires are not in the base $\mathcal{D}$. Formally:

Definition 4 (Desire). *A* desire *is:*

- *a formulae $h \in \mathcal{D}$.*
- *a formulae h such that $\exists \varphi_1 \ldots \wedge \varphi_n \rightarrow h \in \mathcal{P} \cup \Sigma$*
- *a formulae h such that $\exists \varphi_1 \wedge h \ldots \wedge \varphi_n \rightarrow h' \in \mathcal{P}$. In this case, h is called* a sub-desire *of h'. The function Subdesire(h') returns the set $\{h', \varphi_1, \ldots \varphi_n\}$ of the sub-desires of h'. Note that each desire is a sub-desire of itself.*

Example 3 *In example 1, the agent has two desires (jca, fa) and several sub-desires like: w, tb, vac, ag, fr, hop and dr.*

3.3 The notion of action

As noted above, a desire may have a plan to achieve it. We bring the two notions together in a new notion of *action*.

Definition 5 (Action). *An* action *is a pair $a = < h, H >$ such that:*

- *h is a desire.*
- *If $\exists \varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow h \in \mathcal{P} \cup \Sigma$ then $H = \{\varphi_1, \ldots, \varphi_n\}$ else $H = \emptyset$.*

The function $Desire(a) = h$ returns the desire of an action a and the function $Plan(a) = H$ returns its plan.
We denote by $\aleph$ the set of all actions which may be constructed from the triple $<\mathcal{D}, \mathcal{P}, \Sigma>$.

Remark 1 –

- *If the set H (the plan) is empty, this means that the desire has not yet gotten a plan to achieve it or the desire is* atomic *which means that it does not need any plan to be achieved.*
- *A desire may have several plans to be achieved. In this case, we have as many actions for that desire as plans.*

Example 4 *In example 1, we have the following actions: $a_1 = <jca, \{t, vac\}>$, $a_2 = <fa, \{w\}>$ and $a_3 = <w, \emptyset>$. Action a_3 means that:*

1. *The desire of working is atomic or,*
2. *The plan of working is not given yet.*

The realization of an action, *ie* the execution of its plan, requires in certain situations the decomposition of this plan. Each element of the initial plan gives place to a new action with a new plan. We call this kind of actions *sub-actions*. Formally:

Definition 6 (Sub-action). *Let a_1 and a_2 be two actions of $\aleph$. a_1 is a* sub-action *of a_2 iff $Desire(a_1) \in Plan(a_2)$. In other words, $Desire(a_1)$ is a sub-desire of $Desire(a_2)$.*

Example 5 *In example 1, the action* $a_1 = <jca, \{t, vac\}>$ *has the following sub-actions:* $a_{11a} = <t, \{ag\}>$, $a_{11b} = <t, \{fr\}>$, $a_{12a} = <vac, \{dr\}>$, *and* $a_{12b} = <vac, \{hop\}>$

An action may have consequences.

Definition 7 (Action's consequences). *Let* $a \in \aleph$.

$$Consequence(a) = \{\phi_i : such that Desire(a) \cup Plan(a) \cup \Sigma \vdash \phi_i\}.$$

Example 6 *In example 1, the action* $a_2 = <fp, \{w\}>$ *has the following consequences:* $\{\neg ag, \neg dr, fp, w\}$

3.4 Conflicting actions

After a small study undertaken on the various conflicts which may exist between actions (desires), we think that there are four great families of conflicts. In fact, two actions a_1 and a_2 may be conflicting for one of the following reasons:

- *desire-desire* conflict, *ie* $Desire(a_1) \cup Desire(a_2) \cup \Sigma \vdash \bot$
- *plan-plan* conflict, *ie* $Plan(a_1) \cup Plan(a_2) \cup \Sigma \vdash \bot$.
- *consequence-consequence* conflict, *ie* $Consequence(a_1) \cup Consequence(a_2) \vdash \bot$.
- *plan-consequence* conflict, *ie* $Plan(a_1) \cup Consequence(a_2) \vdash \bot$ or $Plan(a_2) \cup Consequence(a_1) \vdash \bot$.

Example 7 *Let's consider an agent equipped with the following bases:* $\mathcal{D} = \{fp, b, s\}$, $\Sigma = \{b \rightarrow \neg ws\}$ *and*

$$\mathcal{P} = \begin{cases} ws \wedge wd \rightarrow fp \\ \quad c \quad \rightarrow b \\ \quad \neg c \quad \rightarrow s \end{cases}$$

with:

ws = *"to work Saturday"*
wd = *"to work Sunday"*
fp = *"to finish the paper"*
c = *"to take the car"*
b = *"to go to the beach Saturday"*
s = *"to make savings"*

$\aleph$ *contains the following actions:*

- $a_1 = <fp, \{ws, wd\}>$.
- $a_2 = <b, \{c\}>$.
- $a_3 = <s, \{\neg c\}>$.

There exists a plan-plan *conflict between* a_2 *and* a_3. *The two actions have contradictory plans here.*
The consequences of action a_2 *are* ($Consequence(a_2) = \{ps, \neg ws\}$). *There exists a plan-consequence conflict between* a_2 *and* a_1. *There exists also a consequence-consequence conflict between* a_1 *and* a_2. *Contrary to the conflict between* a_2 *and* a_3, *the plans of* a_1 *and* a_2 *seemed compatible, the conflict emerges from the consequences of the plans.*

These different kinds of conflicts between actions are brought together in a unique relation of *conflict* defined as follows:

Definition 8 (Conflict). *Let a_1 and a_2 be two actions of $\aleph$. a_1* conflicts *with a_2 iff:* $\{Desire(a_1),\ Desire(a_2)\} \cup Plan(a_1) \cup Plan(a_2) \cup \Sigma \vdash \bot$.

Example 8 *In example 1, a_{11a} = $<t,\ \{ag\}>$ conflicts with a_2 = $<fp,\ \{w\}>$. Indeed, $Plan(a_{11a}) \cup \Sigma \vdash \{\neg w\}$ and $Plan(a_2) = \{w\}$, thus a_{11a} and a_2 are incompatible.*

Property 1. The *conflicts* relation is symmetrical. However, it is neither *reflexive* nor *transitive*.

Example 9 *Let's consider an agent with a base of desires $\mathcal{D} = \{h_1, h_2, h_3\}$ and a base of plans:*

$$\mathcal{P} = \begin{cases} a & \rightarrow h_1 \\ b & \rightarrow h_2 \\ \neg a \wedge \neg b & \rightarrow h_3 \end{cases}$$

Let's suppose that $\Sigma = \emptyset$. a_1, a_2 and a_3 are actions such that: $a_1 = <h_1,\ \{a\}>$, $a_2 = <h_2,\ \{b\}>$ and $a_3 = <h_3,\ \{\neg a,\ \neg b\}>$.

a_1 conflicts with a_3 and a_3 conflicts a_2. However, a_1 does not conflicts with a_2.

Remark 2 *An action may conflict with itself.*

Let's consider the following example:

Example 10 *Let $\mathcal{D} = \{\neg a'\}$, $\mathcal{P} = \{a \wedge b \wedge c \rightarrow \neg a'\}$ and $\Sigma = \{a \rightarrow a'\}$. The action $< \neg a', \{a, b, c\} >$ conflicts with itself.*

3.5 Tree of realization

In example 1, the action a_2 does not attack action a_1. However, it attacks one of its sub-actions such as a_{11a}. In this case, we cannot say that the two desires $Desire(a_1)$ and $Desire(a_2)$ are realizable in the the same time.
Thus, to check if a given desire is realizable, the corresponding action and all its sub-actions must be taken into account. It is then necessary to introduce a new notion of a *tree of realization* of a given desire.
A *tree of realization* of a given desire d is an *AND* tree. Its nodes are actions and its arcs represent the sub-action relationship. The root of the tree is an action for the desire d.
It is an AND tree because all the sub-actions of a given action must be carried out. When for the same desire, there are several plans to carry it out, only one plan is considered. Formally:

Definition 9. *A* tree of realization *g of a desire h is a finite tree such that:*

- *$< h, H >$ is the root of the tree.*

- *A node* $< h', \{\varphi_1, \ldots, \varphi_n\} >$ *has exactly* n *children* [1] $< \varphi_1, H'_1 >, \ldots, < \varphi_n, H'_n >$.
- *The leaves of the tree are atomic actions.*

The function $Root(g) = h$ *returns the desire of the root. The function* $Nodes(g)$ *returns the set of all the actions of the tree* g.
$\mathcal{G}(\aleph)$ *denotes the set of all the trees of realization that we can build from the set* $\aleph$.

Remark 3 *By analogy with argumentation theory, a realization tree can compared to an argument. A realization tree is built to achieve a given desire whereas an argument is built to support a given conclusion. A conclusion may be supported by several arguments and similarly, a desire can have several trees of realization. This case arises when one of its sub-desires has several plans.*

Example 11 *In example 1, the desire "jca" whose action is* $a_1 = <jca, \{t, vac\}>$ *has four trees of realization as shown in figure 1.*

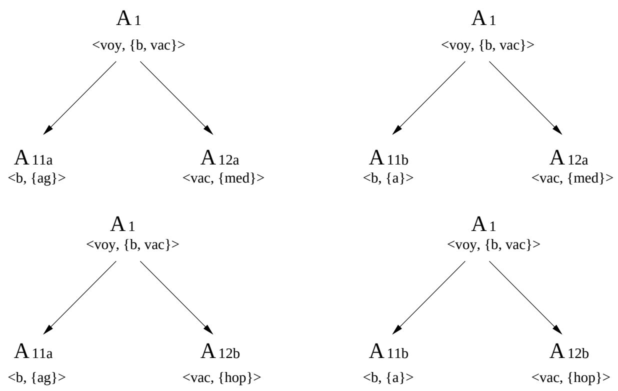

Fig. 1. Trees of realization

Since the actions may be conflicting, two realization trees may then be conflicting too. That relation between realization trees will be called "Attack" relation.

Definition 10 (Attack). *Let* g_1, $g_2 \in \mathcal{G}(\aleph)$. g_1 *attacks* g_2 *iff* $\exists a_1 \in Nodes(g_1)$ *and* $\exists a_2 \in Nodes(g_2)$ *such that* a_1 *conflicts* a_2.

Property 2. The relation attack is *symmetrical.*

[1] If a desire has several plans to carry it out, only one is considered in a tree.

Let's now consider the following example:

Example 12 *Let's suppose an agent equipped with the following bases:* $\mathcal{D} = \{c\}$, $\Sigma = \emptyset$ *and* $\mathcal{P} = \{a \wedge b \rightarrow c, e \rightarrow a, d \wedge g \rightarrow b, \neg e \rightarrow g, x \rightarrow d\}$.
The realization tree of the desire c contains the following actions: $< c, \{a, b\} >$, $< a, \{e\} >$, $< b, \{d, g\} >$, $< g, \{\neg e\} >$, $< d, \{x\} >$, $< e, \emptyset >$, $< \neg e, \emptyset >$ *and* $< x, \emptyset >$.
It is clear that $< a, \{e\} >$ *conflicts* $< g, \{\neg e\} >$.

This example shows clearly that a realization tree of a desire can attack itself. Such trees are said *self-attacked realization trees.*
They can be compared to self-defeating arguments. Like conclusions supported only by self-defeating arguments cannot be inferred from a knowledge base (for example), it is obvious that a desire whose trees of realization are all inconsistent is a *rejected* one. This means it is impossible to carry out such desire.

4 A formal system for handling desires

From the preceding definitions, we can now introduce a formal system for handling conflicting desires of an agent.

Definition 11 (System for handling desires). *Let's consider a triple* $<\mathcal{D}, \mathcal{P}, \Sigma>$.
A system for handling desires (SHD) is a pair $< \mathcal{G}(\aleph)$, *Attack* $>$ *such that* $\mathcal{G}(\aleph)$ *is a set of realization trees and Attack is a binary relation representing the defeasibility relation between the realization trees (Attack* $\subseteq \mathcal{G}(\aleph) \times \mathcal{G}(\aleph)$*).*

As in argumentation theory, we partition the set $\mathcal{G}(\aleph)$ into three cateories of realizations trees:

- The class of *acceptable realization trees.* They represent the *good plans* to achieve their corresponding desires. That desires will become the intentions of the agent.
- The class of *rejected realization trees.* They are those attacked by acceptable realization trees.
- The class of *realization trees in abeyance* which gathers the realization trees which are neither acceptable nor rejected.

In what follows we give the different semantics of "acceptable realization trees". We first start by giving new definitions of *conflict-free* and *defence* in the context of conflicting desires.

Definition 12 (Conflict-free). *Let* $< \mathcal{G}(\aleph), Attack >$ *be a SHD and* $S \subseteq \mathcal{G}(\aleph)$.
S is conflict-free *iff:* $\nexists$ g_1 *and* g_2 *in S such that* g_1 *attacks* g_2. *It follows that if S is conflict-free then* $\forall g_i \in S$, g_i *is not self-attacked.*

As we can see, the notion of conflict-free is very similar to the one in argumentation theory. However, it is not the case for the notion of defence. The semantics of a realization tree is a complete plan to achieve a desire and our aim is to achieve a maximum of desires. The idea is if a given desire d_1 can be achieved with a plan g_1 then if another plan g_2 for the same desire attacks a plan g_3 of another desire d_2, we will accept g_3 to enable the agent to achieve its two desires. Formally:

Definition 13 (Defence). *Let $< \mathcal{G}(\aleph), Attack >$ be a SHD, $S \subseteq \mathcal{G}(\aleph)$ and $g \in \mathcal{G}(\aleph)$. S* defends *g iff $\forall\ g' \in \mathcal{G}(\aleph)$ s.t g' attacks g $\exists\ g$" $\in S$ s.t Desire(g") = Desire(g').*

Preferred extensions

Using the new definition of defence, we show the following property:

Property 3. Each SHD has at least one preferred extension.

Let S_1, ..., S_n be the different preferred extensions of a SHD. For a given set of trees, the function $Desires(S_i) = \{h \in \mathcal{D}$ such that $\exists\ g \in S_i$ and $Root(g) = h\}$ returns the different desires of these trees.

Property 4. Let $< \mathcal{G}(\aleph)$, $Attack >$ be a SHD and S_1, ..., S_n the corresponding extensions. The sets $Desires(S_1)$, ..., $Desires(S_n)$ are not always maximal (for set inclusion).

Let's consider the follwing example:

Example 13 *In example 1, we can construct 5 realization trees. Four of them correspond to the desire jca. Let's note them g_1, g_2, g_3, g_4. The fifth one corresponds to the desire fp and it is denoted by g_5.*
In this case, we have exactly two extensions:

- $S_1 = \{g_1, g_2, g_3, g_4\}$ *with* $Desires(S_1) = \{jca\}$
- $S_2 = \{g_4, g_5\}$ *with* $Desires(S_2) = \{jca, fp\}$

Note that $Desires(S_1) \subseteq Desirs(S_2)$.

The goal is to carry out the maximum of desires of the agent. In the preceding example, the agent can carry out its two desires in extension S_2. Thus among all the preferred extensions, we keep only those which carry out a maximum of desires (for set inclusion). Let denote them by S_i, ..., S_j.

Example 14 *In example 1, the two desires of the agent become intentions. However, to achieve these intentions, the agent should use the plans of the trees $\{g_4, g_5\}$.*

Preferred extensions give us different sets of desires which may be achieved together. There is no preference between the extensions.

Basic extension

The basic extension characterises the set of acceptable realization trees by a function $\mathcal{F}$ that returns for each set of realization trees the ones that are defended by that set.

Theorem 1. –

- *Let* $< \mathcal{G}(\aleph), Attack >$ *be a SHD. The function* $\mathcal{F}$ *is* monotonic *(for set inclusion). In other terms, if* S, $S' \subseteq \mathcal{G}(\aleph)$ *such that* $S \subseteq S'$ *then* $\mathcal{F}(S) \subseteq \mathcal{F}(S')$.
- *The function* $\mathcal{F}$ *has a least fixpoint:* $\mathcal{S}$
- $\mathcal{S}$ *is the basic extension of the SHD.*

When the system SHD is finitary (each realization tree is attacked by a finite number of realization trees), the function $\mathcal{F}$ is *continuous*. In that case, the least fixpoint of $\mathcal{F}$ can be obtained by iterative applications of the function $\mathcal{F}$ to the emptyset.

Theorem 2. *Let* $< \mathcal{G}(\aleph), Attack >$ *be a SHD.*

- $< \mathcal{G}(\aleph), Attack >$ *is finitary.*
- $\mathcal{F}$ *is continuous.*
- *The least fixpoint of* $\mathcal{F}$ *is:* $\mathcal{S} = \bigcup_{i \geq 1} F^i(\emptyset) = \mathcal{C} \cup \bigcup_{i \geq 1} F^i(\mathcal{C}))$. $\mathcal{C}$ *denotes the set of* non-attacked *realization trees.*

Unlike in argumentation theory, the function $\mathcal{F}$ does not preserve the conflict-free property. In other words, if S is conflict-free, it is not always the case for $\mathcal{F}(S)$. This phenomenon occurs in a very particular case when the two defeasible trees, let's say g_1 and g_2, are both other alternatives of their corrsponding desires. In this case, the second alternatives are both in the set S.

Theorem 3. –

- *The set* $\mathcal{S}* = \mathcal{S} \setminus \{g, g' \in \mathcal{S}$ *s.t* g *attacks* $g'\}$ *is conflict-free.*
- *The intentions of the agent are gathered in the set* $\mathcal{I} = \{Root(g_i)$ — $g_i \in \mathcal{S}*\}$

5 Conclusion

This paper introduces a formal model for computing the intentions of an agent from its set of possibly contradictory desires. We showed how and when desires can conflict. We formalized this concept of conflicts and then we showed how to solve them inspired from work on argumentation theory.
More work, of course, remains to be done in this area. Particularly, we shall study the meaning of other semantics in argumentation like stable extension and complete extensions in our context. In [1], the authors have proposed a proof theory testing whether a given argument is acceptable (using the basic semantics). We shall use those results to test if a given realization tree is acceptable and consequently its desire is an intention.

We shall also enrich the model by introducing preferences between the desires. This will help to refine the classification of the desires by leaving less desires in abeyance. We can imagine two sources of preferences. The first one is the agent itself. This means that an agent can have preferences over its set of desires $\mathcal{D}$. In this case, if there is a conflict between two actions, we keep the one whose desire is mostly preferred by the agent. The second source of preferences is *argumentation*. In this case, two actions a_1 and a_2 can be in conflict but one of them can have a good reason (*argument*) to be carried out. We are currently investigating these matters.

References

1. L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. 34:197–216, 2002.
2. M. Bratman. *Intentions, plans, and practical reason.* Harvard University Press, Massachusetts., 1987.
3. M. Bratman, D. Israel, and M. Pollack. *Plans and resource bounded reasoning.*, volume 4. Computational Intelligence., 1988.
4. P. R. Cohen and H. J. Levesque. Intention is choice with commitment. In *Artificial Intelligence*, volume 42, 1990.
5. P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In *In P. R. Cohen, J. Morgan and M. E. Pollack, eds. Intentions in communication*, pages 221–256, 1990.
6. J. Doyle. *Rationality and its role in reasoning.*, volume 8. Computational Intelligence., 1992.
7. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI'93*, pages 852–857, 1993.
8. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.
9. J. Pearl. From conditional ought to qualitative decision theory. In *Proceedings of UAI'93*, pages 12–20, 1993.
10. A. S. Rao and M. P. Georgeff. Modeling rational agents within a bdi architecture. In *Proceedings of KR'91*, 1991.
11. A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In *Proceedings of KR'92*, 1992.
12. A. S. Rao and M. P. Georgeff. Bdi agents: from theory to practice. In *Proceedings of the 1st International Conference on Multi Agent Systems*, pages 312–319, 1995.