On the Use of an ATMS for Handling Conflicting Desires

Leila Amgoud and **Claudette Cayrol**

Institut de Recherche en Informatique de Toulouse
118, route de Narbonne
31062 Toulouse, France
{amgoud, ccayrol}@irit.fr

Abstract

This paper presents a revised version of a framework proposed in (Amgoud 2003) which computes consistent sets of *intentions* from a conflicting set of *desires* and a set of *beliefs*. That framework enables us to restate the problem of computing intentions in the context of argumentation theory. Indeed, interacting arguments are interpreted as competing plans for achieving some desire, or conflicting plans for achieving different desires.
Another important contribution of this paper is to present an ATMS-based proof theory for that framwork. Indeed, we show that the different concepts defined and used in (Amgoud 2003) can be restated taking advantage of the well-known Assumption-based Truth Maintenance System (de Kleer 1986a; 1986b).

Keywords: Autonomous agents, BDI, ATMS.

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 (Bratman 1987; Bratman, Israel, & Pollack 1988; Cohen & Levesque 1990a; Doyle 1992; Rao & Georgeff 1991; 1992; 1995). 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) (the BDI architecture). An agent can have contradictory desires. However, its intentions are a coherent subset of desires, which the agent is committed to achieve.
In (Cohen & Levesque 1990b), Cohen and Levesque explored principles governing rational balance among an agent's beliefs, goals, actions and intentions. In (Rao & Georgeff 1991), 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. In decision theory, Pearl (Pearl 1993) 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.
In (Amgoud 2003), Amgoud has presented a framework for handling contradictory desires. That framework computes consistent sets of *intentions* from a conflicting set of *desires* and a set of *beliefs*. Another novelty of that framework is that it restates the problem of computing intentions in the context of argumentation theory. Indeed, interacting arguments are interpreted as competing plans for achieving some desire, or conflicting plans for achieving different desires.

The aim of this paper is to propose a slightly revised version of the above framework and to present an ATMS-based proof theory for the obtained framework. Indeed, we show that the different concepts defined and used in (Amgoud 2003) can be restated taking advantage of the well-known Assumption-based Truth Maintenance System (de Kleer 1986a; 1986b). In fact, restating argument computation as an ATMS allows one to use existing ATMS algorithms to build a framework for generating intentions from desires.

This paper is organized as follows: We start by presenting an illustrative example of an agent having two desires which are apparently not simultaneously achievable. In the next two sections, we describe the revised formal framework previously introduced in (Amgoud 2003). We then introduce the basic concepts of an Assumption-based Truth Maintenance System (ATMS).

In the next section, we show how to encode the problem of computing the set of intentions as a one of computing labels, nogoods and contexts. The last section is devoted to some concluding remarks and some perspectives.

Example

Let us consider 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)

In addition to the desires, the agent is supposed to have beliefs on the way of achieving a given desire:

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

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

For example, the rule $t \wedge vac \rightarrow jca$ means that the agent believes that if he gets tickets and he is vaccinated then he will be able to go on a journey in central Africa. The rule $w \rightarrow fp$ expresses that the agent believes that if he works then he will be able to finish his paper. To get tickets, the agent can either visit 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 the hospital*. In these two last cases, the agent has two ways to achieve the same desire.
An agent may have also another kind of beliefs representing integrity constraints. In our example, we have:

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

These two rules mean that the agent believes that if he works, he can neither visit an agency nor go to a doctor. Obviously, in this example, some ways of achieving the initial desires are conflicting.

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

Basic definitions

In this section, we present the framework previously introduced by Amgoud in (Amgoud 2003) for handling conflicting desires.

Logical language

Let $\mathcal{L}$ be a propositional language. $\vdash$ denotes classical inference and $\equiv$ denotes logical equivalence. An agent is supposed to be equipped with a base $\mathcal{D}$ of desires, a belief base Σ containing the 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 base $\mathcal{C}$ of integrity constraints.

- $\mathcal{D}$ contains literals 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.
- Σ contains rules 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 the agent believes that if he realizes $\varphi_1, \ldots, \varphi_n$ then he will be able to achieve h.
- $\mathcal{C}$ contains formulas of $\mathcal{L}$. They represent a kind of integrity constraints.

Example 1 *The agent who wants to go on a journey to Central Africa has the following bases:*

$$\mathcal{D} = \{jca, fp\},\ \Sigma = \begin{cases} t \wedge vac & \rightarrow & jca \\ w & \rightarrow & fp \\ ag & \rightarrow & t \\ fr & \rightarrow & t \\ hop & \rightarrow & vac \\ dr & \rightarrow & vac \end{cases}$$

and $\mathcal{C} = \begin{cases} w & \rightarrow & \neg ag \\ w & \rightarrow & \neg dr \end{cases}$

The notion of desire/sub-desire

A desire is any element of $\mathcal{D}$. A desire h may have sub-desires. In example 1, the desire of going on a journey to central Africa has two sub-desires which are: ”getting the tickets” and ”being vaccinated”. The sub-desire ”getting the tickets” has itself the two following sub-desires: ”having a friend who may bring the tickets” and ”visiting an agency”.

Definition 1 (Desire/Sub-desire) *Let us consider an agent equipped with the bases* $<\mathcal{D}, \Sigma, \mathcal{C}>$.

1. $\mathcal{D}$ *is the set of the* desires *of the agent.*
2. $Sub\mathcal{D}$ *is the set of the* sub-desires *of the agent: A literal* $h' \in Sub\mathcal{D}$ *iff there exists a rule* $\varphi_1 \wedge h' \ldots \wedge \varphi_n \rightarrow h \in \Sigma$ *with* $h \in \mathcal{D}$ *or* $h \in Sub\mathcal{D}$. *In that case,* h' *is a sub-desire of* h.

The notion of partial plan

As noted above, an agent may have one or several ways to achieve a given desire. We bring the two notions together in a new notion of *partial plan*.

Definition 2 (Partial plan) *A* partial plan *is a pair* a = $<h, H>$ *such that:*

- h *is a desire or a sub-desire.*
- $H = \{\varphi_1, \ldots, \varphi_n\}$ *if there exists a rule* $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow h \in \Sigma$, $H = \emptyset$ *otherwise.*

The function $Desire(a) = h$ *returns the desire or sub-desire of a partial plan* a *and the function* $Support(a) = H$ *returns the support of the partial plan.* $\aleph$ *will gather all the partial plans that can be built from* $<\mathcal{D}, \Sigma, \mathcal{C}>$.

Remark 1 *A desire may have several partial plans.*

Remark 2 *Let* $a = < h, H >$ *be a partial plan. Each element of the support* H *is a sub-desire of* h.

Definition 3 *A partial plan* $a = < h, H >$ *is* elementary *iff* $H = \emptyset$.

Remark 3 *If there exists an elementary partial plan for a desire* h*, it means that the agent knows how to achieve directly* h.

Example 2 *In example 1, we have several partial plans. For example:* a_1 = *<jca, {t, vac}>*, a_{11a} = *<t, {ag}>*, a_{11b} = *<t, {fr}>*, a_{12a} = *<vac, {dr}>*, a_{12b} = *<vac, {hop}>*, a_2 = *<fp, {w}>* *and* a_{21} = *<w,* $\emptyset$ *>.*

The notion of complete plan

A partial plan shows the actions that should be performed in order to achieve the corresponding desire (or sub-desire). However, the elements of the support of a given partial plan are considered as sub-desires that must be achieved at their turn by another partial plan. The whole way to achieve a given desire is called a *complete plan*. A *complete plan* for a given desire d is an *AND* tree. Its nodes are partial plans and its arcs represent the sub-desire relationship. The root of the tree is a partial plan for the desire d. It is an AND tree because all the sub-desires of d must be considered. When for the same desire, there are several partial plans to carry it out, only one is considered in a tree. Formally:

Definition 4 (Complete plan) *A* complete plan g *for a desire* h *is a finite tree such that:*

- $h \in \mathcal{D}$.
- *The root of the tree is a partial plan <h, H> .*
- *A node <h', {*φ_1*, ...,* φ_n*}> has exactly* n *children* $<\varphi_1, H'_1>, \ldots, <\varphi_n, H'_n>$ *where* $<\varphi_i, H'_i>$ *is a partial plan for* φ_i.
- *The leaves of the tree are elementary partial plans.*

The function $Root(g) = h$ *returns the desire of the root. The function* $Nodes(g)$ *returns the set of all the partial plans of the tree* g. $\mathcal{G}$ *denotes the set of all the complete plans that can be built from the triple* $<\mathcal{D}, \Sigma, \mathcal{C}>$. *The function* $Leaves(g)$ *returns the set of the leaves of the tree* g.

Example 3 *In example 1, the desire* jca *has four complete plans (*g_1*,* g_2*,* g_3*,* g_4*)* g_1 = {*<jca,* $\{t, vac\}$*>, <t,* $\{ag\}$*>, <ag,* $\emptyset$*>, <vac,* $\{hop\}$*>, <hop,* $\emptyset$*>*}, g_2 = {*<jca,* $\{t, vac\}$*>, <t,* $\{ag\}$*>, <ag,* $\emptyset$*>, <vac,* $\{dr\}$*>, <dr,* $\emptyset$*>*}, g_3 = {*<jca,* $\{t, vac\}$*>, <t,* $\{fr\}$*>, <fr,* $\emptyset$*>, <vac,* $\{dr\}$*>, <dr,* $\emptyset$*>*} *and* g_4 = {*<jca,* $\{t, vac\}$*>, <t,* $\{fr\}$*>, <fr,* $\emptyset$*>, <vac,* $\{hop\}$*>, <hop,* $\emptyset$*>*}*. whereas the desire* fp *has only one complete plan* g_5 *with* g_5 = {*<jca,* $\{w\}$*>, <w,* $\emptyset$*>*}.

The notion of Conflicts

In (Amgoud 2003), it has been shown that partial plans may be conflicting for several reasons.These different kinds of conflicts are brought together in a unique relation of *conflict* defined as follows:

Definition 5 (Conflict) *Let* a_1 *and* a_2 *be two partial plans of* $\aleph$. a_1 conflicts *with* a_2 *iff:* $\{Desire(a_1), Desire(a_2)\} \cup Support(a_1) \cup Support(a_2) \cup \mathcal{C} \cup \Sigma \vdash \perp$.

Example 4 *In example 1,* a_{11a} = *<t, {ag}> conflicts with* a_2 = *<fp,* $\{w\}$*>. Indeed, Support(*a_{11a}*)* $\cup\, \mathcal{C} \vdash \{\neg w\}$ *and Support(*a_2*) =* $\{w\}$.

More generally, a set of partial plans may be conflicting.

Definition 6 *Let* $S \subseteq \aleph$. S *is conflicting iff* $\bigcup_{a \in S} (\{Desire(a)\} \cup Support(a)) \cup \mathcal{C} \cup \Sigma \vdash \perp$.

Since partial plans may be conflicting, two complete plans may be conflicting too.

Definition 7 (Attack) *Let* g_1*,* $g_2 \in \mathcal{G}$. 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 with* a_2.

Note that a complete plan may attack itself. In this case, the corresponding desire is not achievable.
More generally we are interested in sets of complete plans such that there is no conflict between their nodes. Formally:

Definition 8 (Conflict-free) *Let* $S \subseteq \mathcal{G}$. S *is* conflict-free *iff* $\bigcup_{g \in S} [\bigcup_{a \in\, Nodes(g)} (Support(a) \cup \{Desire(a)\})] \cup \mathcal{C} \cup \Sigma \nvdash \perp$.
If $S = \{g\}$*, then we say that the complete plan* g *is conflict-free.*

We can show easily that any conflict-free set of complete plans does not contain two conflicting (in the sense of the relation Attack) elements. Formally:

Proposition 1 *Let $S \subseteq \mathcal{G}$. If S is conflict-free then $\nexists$ g_1 and g_2 in S such that g_1 attacks g_2.*

Proof *This result follows directly from Definition 5, Definition 7 and Definition 8.* ■

The converse is generally false as shown by the following example:

Example 5 *X is an agent equipped with the following bases: $\mathcal{D}$ = $\{a, b, c\}$, $\mathcal{C}$ = $\{b' \wedge c' \rightarrow \neg a\}$ and Σ =*

$$\begin{cases} a' & \rightarrow & a \\ b' & \rightarrow & b \\ c' & \rightarrow & c \end{cases}$$

There are three complete plans (g_1, g_2, g_3) one for each desire g_1 = {<a, $\{a'\}$>, <a', $\emptyset$>}, g_2 = {<b, $\{b'\}$>, <b', $\emptyset$>} and g_3 = {<c, $\{c'\}$>, <c', $\emptyset$>}. It is easy to check that $\nexists$ i, j such that g_i attacks g_j. However, the constraint given in $\mathcal{C}$ implies that the set S = $\{g_1, g_2, g_3\}$ is not conflict-free.

From the definition of the notion of conflict-free, the following result can be showed.

Proposition 2 *Let $S \subseteq \mathcal{G}$. S is conflict-free iff $\bigcup_{g \in S}$ $Leaves(g)$ is conflict-free.*

Proof **(⇒)** *Obvious from the definition of conflict-free.*
(⇐) *Assume that $\bigcup_{g \in S}$ $Leaves(g)$ is conflict-free.*
Let g_i $\in$ $\mathcal{G}$ and let E_i denote $\{desire(a) | a \in Leaves(g_i)\}$. From the definition of a complete plan, we have:
$E_i \cup \Sigma \vdash Support(a) \cup \{desire(a)\}$ for each a $\in$ $Nodes(g_i)$
$\bigcup_{g \in S}$ $Leaves(g)$ is conflict-free means that: $\bigcup_i$ $E_i \cup \Sigma \cup \mathcal{C} \nvdash \perp$.
Then $\bigcup_{g \in S} \bigcup_{a \in Nodes(g)}$ $\{desire(a)\} \cup Support(a) \cup \mathcal{C} \cup \Sigma \nvdash \perp$. ■

The following example shows that we can find a complete plan which is not conflict-free even if it does not attack itself.

Example 6 *X is an agent equipped with the following bases: $\mathcal{D}$ = $\{d\}$, $\mathcal{C}$ = $\{b' \wedge c' \rightarrow \neg a\}$ and Σ =*

$$\begin{cases} a' & \rightarrow & a \\ b' & \rightarrow & b \\ c' & \rightarrow & c \\ a \wedge b \wedge c & \longrightarrow & d \end{cases}$$

There is a unique complete plan for d which does not attack itself and whose set of nodes is conflicting: g = {<d, $\{a, b, c\}$>, <a, $\{a'\}$>, <a', $\emptyset$>, <b, $\{b'\}$>, <b', $\emptyset$>, <c, $\{c'\}$>, <c', $\emptyset$>}

Obviously a desire which has no conflict-free complete plan will be called *unachievable*. This means it is impossible to carry out such a desire.

Definition 9 (Unachievable desire) *A desire d is* unachievable *if $\nexists$ $g \in \mathcal{G}$ s.t $Root(g)$ = d and g is conflict-free.*

A formal system for handling desires

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

Definition 10 (System for handling desires) *Let's consider a triple <$\mathcal{D}$, Σ, $\mathcal{C}$>. The pair <$\mathcal{G}$, $Attack$> will be called a system for handling desires (SHD).*

A SHD has the same features as an argumentation framework (Amgoud & Cayrol 2002). Inspired by previous work on argumentation theory, we will define acceptable sets of complete plans. Then we will be able to partition the set $\mathcal{G}$ into three categories:

- The class of *accepted complete plans*. They represent the *good plans* to achieve their corresponding desires. These desires will become the intentions of the agent.
- The class of *rejected complete plans*. These are the self-attacked ones.
- The class of *complete plans in abeyance* which gathers the complete plans which are neither acceptable nor rejected.

We give below the semantics of "acceptable sets of complete plans".

Definition 11 *Let <$\mathcal{G}$, $Attack$> be a SHD and $S \subseteq \mathcal{G}$. S is an* acceptable *set of complete plans iff:*

- *S is conflict-free.*
- *S is maximal (for set inclusion).*

Example 7 *In example 1, there are four complete plans (g_1, g_2, g_3, g_4) for the desire "going on a journey to central africa" and exactly one complete plan g_5 for the desire "finishing the paper". Moreover, g_5 attacks g_1, g_2 and g_3. We have exactly two acceptable sets of complete plans:*

- $S_1 = \{g_1, g_2, g_3, g_4\}$
- $S_2 = \{g_4, g_5\}$

Note that if S is an acceptable set of complete plans, $\{Root(g) | g \in S\}$ is a set of desires which can be achieved together. An intention set is a subset of $\{Root(g) | g \in S\}$ which the agent commits to carry out.

Definition 12 (Intentions) *A set of desires $\mathcal{I} \subseteq \mathcal{D}$ is an intention set iff there exists an acceptable set of complete plans S such that $\mathcal{I}$ = $\{Root(g) | g \in S\}$.*

Example 8 *In example 1, $\mathcal{I}_1 = \{jca\}$ is achievable both with plans of S_1 and S_2. Contrastedly, $\mathcal{I}_2 = \{jca, fp\}$ is achievable only with the plans of S_2.*

The purpose of an agent is to achieve a maximal subset of $\mathcal{D}$. Consequently, he will choose maximal intention sets. In the above example, he will choose the set $\mathcal{I}_2$.

Assumption-based Truth Maintenance System: ATMS

In this section we present the basic concepts of an ATMS which forms the backbone of our proof theory.
Introduced by De Kleer (de Kleer 1986a; 1986b), an ATMS mainly has to manage interdependencies in a knowledge base. It may be used as an interface between an inference engine and the associated knowledge base. In that case, each inference performed by the program is sent to the ATMS which must in turn determine which data may be believed in or not, according to the available knowledge.
The specific mechanisms of an ATMS are based upon the distinction between two sets of different data (represented by propositional variables), that are supposed to be defined at the beginning: *assumptions* and *non-assumptions*. The basic idea is that the assumptions are the primitive data from which all other data can be derived. They are the parameters which characterize the different situations worth-considering for a given problem, that problem being described by a set of clauses which are called the *justifications*.
In the following, $\mathcal{P}$ denotes a set of propositional symbols. $\mathcal{A} \cup \mathcal{NA}$ denotes a partition of $\mathcal{P}$ with elements of $\mathcal{A}$ called *assumptions* and elements of $\mathcal{NA}$ are non-assumptions. $\mathcal{BJ}$ denotes a set of clauses built from $\mathcal{P}$.

Example 9 *Consider as an example the following base of justifications, using upper-case symbols to denote assumptions:*

$$\left\{\begin{array}{lcl} Ab & \rightarrow & c \\ B & \rightarrow & b \\ C & \rightarrow & b \\ Db & \rightarrow & \end{array}\right.$$

In this example, $\mathcal{A}$ = {A, B, C, D} and $\mathcal{NA}$ = {b, c}. The clause $Ab \rightarrow c$ means that under the assumption A, if b is true then c also is true. The clause $Db \rightarrow$ means that under the assumption D, b is false.

The basic notion of an ATMS is that of environment which is any set of assumptions. Formally:

Definition 13 *A set of assumptions $E \subseteq \mathcal{A}$ is called an* environment.

An environment may be incoherent with respect to the base $\mathcal{B}J$ of clauses.

Definition 14 (Environment) *An environment E is* incoherent *with respect to $\mathcal{B}J$ iff $\mathcal{BJ} \cup E \vdash \perp$.*
An environment E is coherent *iff it is not incoherent.*

The main purpose of an ATMS is to answer two questions. The first is as follows: "May a given conjunction of assumptions be true, if we want to satisfy the base of justifications?". This corresponds to the concept of *nogood* in an ATMS. A *nogood* is a minimal (for set-inclusion) set of assumptions such that the assumptions it contains cannot be all taken as true (i.e. valuated by true) with respect to the base of justifications. So, a set of assumptions is inconsistent with the base of justifications if and only if it contains a nogood.

Definition 15 (Nogood) *Let $\mathcal{BJ}$ be a set of clauses. A* nogood *is a minimal (for set-inclusion) incoherent environment.*

Let's illustrate the above definition on the base of clauses introduced in example 9.

Example 10 (Example continued) *In the above base, there are two nogoods: $\{B, D\}$ and $\{C, D\}$.*

The second important question to which an ATMS should give an answer is the following one: "On which set of assumptions may some assertion be believed, if we take into account the justifications?". This corresponds to the concept of *label* in an ATMS: The *label* of a proposition is a complete and minimal collection of environments (sets of assumptions) so that if the base of justifications is satisfied and each assumption of the environment is taken as true, then the proposition is true. Minimal means that there are no two environments in the label such that one contains the other. Complete means that each environment that makes the proposition true includes at least one environment of the label.

Definition 16 (Label) *Let $\mathcal{BJ}$ be a set of clauses. Let d be an element of $\mathcal{P}$. Let E be a coherent environment. E $\in$ Label(d) w.r.t. $\mathcal{BJ}$ iff $\mathcal{BJ} \cup E \vdash d$ and no $E' \subseteq E$ satisfies $\mathcal{BJ} \cup E' \vdash d$.*
Moreover, the label is complete *in the sense that each environment E such that $\mathcal{BJ} \cup E \vdash d$ contains at least one element of Label(d).*

Example 11 (Example continued) *As said in example 9, the set $\mathcal{NA}$ contains two symbols: b and c. According to $\mathcal{BJ}$, $Label(b) = \{\{B\}, \{C\}\}$ and $Label(c) = \{\{A, B\}, \{A, C\}\}$.*

An ATMS is also able to return for any set of assumptions, the data which are true under those assumptions. This is called the *context* of that set of assumptions.

Definition 17 (Context) *Let E be a coherent environment. $Context(E)$ = $\{d|E$ contains an environment of Label(d) $\}$.*

Example 12 (Example continued) *In example 9, $Context(\{B, C\})$ = $\{b\}$.*

An ATMS-based proof theory

As said before, the aim of this paper is to propose a proof theory of the framework presented in previous sections. The proof theory is based on an ATMS. In what follows we introduce that proof theory.
As noted above, an ATMS handles two kinds of data: *assumptions* and *non-assumptions*. A parallel can be established between the fundamental concepts of an ATMS and the different notions defined in the framework presented in the third section. In fact, the elementary partial plans will play the role of assumptions. The leaves of a conflict-free complete plan for a given desire d will be obtained by an environment in the label of a data associated with d after an appropriate coding. We will show also that sets of conflicting elementary partial plans will correspond to nogoods.

An ATMS represents data with only *positive literals*. So we need to encode the three bases $\mathcal{D}$, Σ and $\mathcal{C}$ into three new bases $\mathcal{D}'$, Σ' and $\mathcal{C}'$ using an appropriate procedure. In fact, the coding should produce clauses which are tractable by an ATMS. For example, in order to eliminate a negative literal $\neg x$, we introduce a new symbol nx. Then we replace elsewhere $\neg x$ by nx and we add the two following clauses $x \wedge nx \rightarrow$ and $\rightarrow x \vee nx$ to the base $\mathcal{C}'$. Note that disjunctions are handled in an ATMS (see (de Kleer 1988) for more details).
In the rest of this paper, we will use only the modified bases $\mathcal{D}'$, Σ' and $\mathcal{C}'$ such that $\mathcal{C}'$ contains only negative clauses. $\aleph'$ and $\mathcal{G}'$ will denote the associated sets of partial and complete plans. Moreover, the set of assumptions $\mathcal{A}$ = {h such that <h, H> $\in \aleph'$ and H = $\emptyset$} and the set of justifications $\mathcal{BJ}$ = $\Sigma' \cup \mathcal{C}'$.

Example 13 *In example 1, $\mathcal{A}$ = $\{w, fr, ag, dr, hop\}$ and $\mathcal{BJ}$ = $\Sigma \cup \mathcal{C}$. Note that in this example, we don't need to encode the bases.*

The following property shows clearly that the notion of conflict-free in our framework is strongly related to the notion of coherence of an environment in an ATMS.

Property 1 *Let $E \subseteq \mathcal{A}$. E is a coherent environment iff $\{$<h, $\emptyset$>, h $\in E\}$ is a conflict-free set of partial plans.*

Proof *The proof follows directly from Definition 6 and Definition 14.* ■

We can show also that the leaves of a conflict-free complete plan for a given desire d will be obtained by an environment in the label of a data associated with d.

Property 2 *Let $d \in \mathcal{D}'$ and let $E \subseteq \mathcal{A}$.*
$E \cup \mathcal{BJ} \vdash d$ iff $\exists$ g complete plan for d such that $\{Desire(a), a \in Leaves(g)\}$ = E.

Proof

($\Rightarrow$) *Let $E \subseteq \mathcal{A}$ such that $E \cup \mathcal{BJ} \vdash d$.*
There exists $E' \subseteq E$, $E' \in Label(d)$. The labels are computed from ($\mathcal{A}$, $\mathcal{BJ}$). $Label(d)$ is updated by taking into account a justification for d, of the form $\phi_1 \wedge \ldots \wedge \phi_n \rightarrow d$, and the previously computed $Label(\phi_1)$, ..., $Label(\phi_n)$. Negative clauses are only considered for testing the coherence of an environment, and they are all in $\mathcal{C}'$.
So, from Definition 4, there exists a complete plan g for d such that $\{desire(a), a \in Leaves(g)\}$ = E'. The plan is built recursively from $d, \{\phi_1, \ldots, \phi_n\}$ and a plan for each ϕ_i.

($\Leftarrow$) *Let g be a complete plan for d. Let E = $\{desire(a), a \in Leaves(g)\}$. Each non terminal node of g corresponds to a justification in Σ'. So from the definition of a complete plan, we have $E \cup \Sigma' \vdash d$ then $E \cup \mathcal{BJ} \vdash d$.* ■

In our framework, we have shown that some desires are not achievable because their complete plans are not conflict-free. We can show that such desires have an empty label.

Property 3 *Let $d \in \mathcal{D}'$. d is unachievable iff $Label(d)$ = $\emptyset$.*

Proof

($\Rightarrow$) *Let $d \in \mathcal{D}'$ such that d is unachievable. That means that $\nexists$ $g \in, \mathcal{G}$, g is conflict-free, g is a plan for d.*
Let us suppose that $Label(d)$ is not empty. Let $E \in Label(d)$. Then $E \cup \mathcal{BJ} \vdash d$. By property 2, there exists a complete plan g for d such that E = $\{desire(a), a \in Leaves(g)\}$.
E is a coherent environment, so by property 1, $Leaves(g)$ is a conflict-free set of plans. Then by proposition 2, g is conflict-free, which is in contradiction with the initial assumption "d is unachievable". So $Label(d)$ must be empty.

($\Leftarrow$) *Let $d \in \mathcal{D}'$ such that $Label(d)$ = $\emptyset$. Let us suppose that there exists g a conflict-free complete plan for d. Let E = $\{desire(a), a \in Leaves(g)\}$.*
By property 2, $E \cup \mathcal{BJ} \vdash d$. By property 1 and proposition 2, E is coherent. So there exists $E' \in Label(d)$, $E' \subseteq E$, which is in contradiction with the assumption $Label(d)$ = $\emptyset$. So d is unachievable. ■

From the above results, we show that there is a correspondence between the label of a desire d and its conflict-free complete plans. However, we obtain only minimal complete plans in the following sense.

Definition 18 *Let $g \in \mathcal{G}$. g is a minimal plan for d iff $\nexists$ $g' \in \mathcal{G}$ such that $g' \neq g$, $Root(g')$ = d and $Leaves(g') \subset Leaves(g)$.*

Proposition 3 *Let $d \in \mathcal{D}'$. $Label(d) \neq \emptyset$ then $Label(d)$ = $\{E_1, \ldots, E_n\}$ with E_i = $\{Desire(a)$, a $\in Leaves(g_i)\}$ and g_1, $\ldots$, g_n are the minimal conflict-free complete plans of d.*

Proof *It follows directly from the definition of Label(d) and the above results.* ■

Example 14 *Let $\mathcal{D}$ = $\{d\}$, $\mathcal{C}$ = $\emptyset$ and Σ = $\{x \wedge y \wedge z \rightarrow u$, $u \rightarrow d$, $x \wedge y \rightarrow v$, $v \rightarrow d\}$. There are two conflict-free complete plans g_1 and g_2 for d $Leaves(g_1)$ = $\{< x, \emptyset >, < y, \emptyset >, < z, \emptyset >\}$, $Leaves(g_2)$ = $\{< x, \emptyset >, < y, \emptyset >\}$. g_2 is the unique minimal complete plan for d. In the ATMS setting, $\mathcal{A}$ = {x, y, z}, $\mathcal{BJ}$ = Σ and $Label(d)$ = $\{\{x, y\}\}$.*

Example 15 *In example 1, there are four minimal conflict-free complete plans for the desire jca: g_1, g_2, g_3 and g_4. In the ATMS setting, $Label(jca)$ = $\{\{ag, hop\}$, $\{ag, dr\}$, $\{fr, hop\}$, $\{fr, dr\}\}$. Concerning the desire fp, there is a unique conflict-free complete plan g_5 and exactly one environment in the label of fp: $Label(fp)$ = $\{\{w\}\}$.*

In addition to labels, an ATMS computes also the nogoods which represent the sets of assumptions which cannot be considered together as true. In our application, a nogood represents the set of directly achievable desires that cannot be performed together. We will denote by $\mathcal{NGS}$ = $\{N_1, \ldots, N_n\}$ the set of all the nogoods that can be computed from <$\mathcal{A}$, $\mathcal{BJ}$>.

Example 16 *In example 1, there are two nogoods: N_1 = $\{ag, w\}$ and N_2 = $\{w, dr\}$.*

From the set of nogoods, an ATMS also enables to compute the so-called *maximal coherent environments*. In the following, $T_1, \ldots, T_n$ will denote all the maximal coherent environments computed from <$\mathcal{A}$, $\mathcal{BJ}$>.

Proposition 4 •

- *Let $S_1, \ldots, S_n$ be the acceptable sets of complete plans. $\forall$ i, $\{Desire(a)$ s.t. a $\in \bigcup_{g \in S_i} Leaves(g)\}$ is a maximal coherent environment.*
- *Conversely, for each maximal coherent environment T_j, $\exists$ S_j (an acceptable set of complete plans) such that $\{Desire(a)$ s.t a $\in \bigcup_{g \in S_j} Leaves(g)\}$=$T_j$.*

Proof

(⇒) *Let S_i be an acceptable set of complete plans. Let E_i denote $\{Desire(a), a \in \cup_{g \in S_i} Leaves(g)\}$. S_i is conflict-free, so by proposition 2, $\cup_{g \in S_i} Leaves(g)$ is conflict-free and according to property 1, E_i is a coherent environment.*
If E_i is not maximal coherent, there exists $h \in \mathcal{A}$ and $h \notin E_i$ such that $E_i \cup \{h\}$ is coherent. By property 1, $\cup_{g \in S_i} Leaves(g) \cup < h, \emptyset >$ is conflict-free.
Let us consider S'_i = S_i $\cup$ $\{< a, \emptyset >\}$. S'_i $\subseteq$ $\mathcal{G}$, S'_istrictly contains S_i. $\cup_{g \in S'_i} Leaves(g)$ = $\cup_{g \in S_i} Leaves(g) \cup < h, \emptyset >$ is conflict-free. So by proposition 2, S'_i is conflict-free, which is in contradiction with the assumption "S_i is maximal conflict-free".

(⇐) *Let T_j be a maximal coherent environment. Let X_j denote {<h, $\emptyset$>, h $\in T_j$}. By property 1, X_j is conflict-free. X_j is a set of complete plans (each elementary partial plan is a complete plan). So either X_j is a maximal conflict-free set of plans, and then X_j = S_j, or there exists S_j an acceptable set of plans such that X_j $\subseteq S_j$, $X_j \neq S_j$ ($X_j \subset S_j$).*
If X_j = S_j then $\{Desire(a), a \in \cup_{g \in S_i} Leaves(g)\}$ = T_j.
If $X_j \subseteq S_j$ and $X_j \neq S_j$, then X_j = $\cup_{g \in X_j} Leaves(g)$ $\subset$ $\cup_{g \in S_j} Leaves(g)$. The above inclusion is strict since each plan in X_j is elementary. Then T_j $\subset$ $\{Desire(a), a \in \cup_{g \in S_j} Leaves(g)\}$.
So, by the first part of proposition 4, T_j is strictly included in a maximal coherent environment, which is in contradiction wth the assumption "T_j is maximal coherent". ■

In the following, we show that the intention sets can be computed through the context of each maximal coherent environment T_i.

Proposition 5 *I is an intention set iff $\exists$ T_i such that I = $Context(T_i) \cap \mathcal{D}'$.*

Proof *Due to proposition 4, it is sufficient to prove the following result:*
Let S_i be an acceptable set of complete plans. Let T_i = $\{Desire(a) | a \in \cup_{g \in S_i} Leaves(g)\}$ be the corresponding maximal coherent environment. $d \in \mathcal{D}' \cap Context(T_i)$ iff $\exists g \in S_i$, Root(g) = d.

(⇒) *Let $d \in \mathcal{D}' \cap Context(T_i)$. It means that $d \in \mathcal{D}'$ and there exists $E_i \in Label(d)$, $E_i \subseteq T_i$. By proposition 3, there exists g complete conflict-free plan for d with E_i = $\{Desire(a) | a \in Leaves(g)\}$. By definition of T_i, we are sure that $g \in S_i$, and we have $Root(g)$ = d.*

(⇐) *Let $d \in \mathcal{D}'$ such that $\exists g \in S_i$, $Root(g)$ = d. We may assume that g is minimal. Indeed, if g is not minimal, wa can find g' minimal such that $Root(g)$ = $Root(g')$ = d and $Leaves(g') \subset Leaves(g)$. Moreover, $g' \in S_i$: $\cup_{g \in S_i \cup \{g'\}} Leaves(g)$ = $\cup_{g \in S_i} Leaves(g)$. So S_i*

$\cup \{g'\}$ is conflict-free. S_i is maximal conflict-free, so $g' \in S_i$.
So, $\exists g$ minimal conflict-free complete plan for d, $g \in S_i$. By proposition 3, $Label(d)$ contains $E_i = \{Desire(a), a \in Leaves(g)\}$. Since $g \in S_i$, $E_i \subseteq T_i$. And by definition of a context, $d \in Context(T_i)$. ■

Example 17 *In example 1, there are two maximal coherent environments: $T_1 = \{w, fr, hop\}$ and $T_2 = \{ag, fr, hop, dr\}$. $Context(T_1) = \{w, fr, hop, fp, jca\}$ $Context(T_2) = \{ag, fr, dr, hop, jca\}$. There are two intention sets $I_1 = \{fp, jca\}$ and $I_2 = \{jca\}$.*

Conclusion and perspectives

In this paper, we have presented a framework which computes the intentions of an agent from its set of possibly contradictory desires. A link is then established between that framework and an ATMS-based computation procedure. However, that framework has some limits due to the use of a poor language. In fact, elements of Σ and those of $\mathcal{C}$ are represented in the same way and handled also in the same way by the ATMS. However, the meaning of a rule $a \rightarrow b$ in Σ is: "if a is achieved then b can be achieved" but it is not "if a is achieved then b will be actually achieved". In the following, we illustrate this point on a variant of example 1.

Example 18 *Let $\mathcal{D} = \{jca\}$, $\mathcal{C} = \{ag \rightarrow \neg visit\}$ and $\Sigma = \{t \wedge vac \rightarrow jca, hop \rightarrow visit, ag \rightarrow t, hop \rightarrow vac\}$. The rule $hop \rightarrow visit$ means that if the agent goes to a hospital, then he can visit a friend of him. The constraint $ag \rightarrow \neg visit$ means that if the agent goes to an agency then he will not have enough time to visit his friend in the hospital.*
According to our framework, the desire jca of the agent cannot be achieved because of the rule $hop \rightarrow visit$ which induces a conflict with the base $\mathcal{C}$. However, the agent can achieve its desire if he goes to the hospital in order to be vaccinated, witout visiting its friend.

An extension of the framework proposed in this paper consists in solving the problem stated in the above example.
In this paper, we compute sets of intentions. We would like to be able to answer the question "whether a given desire can be an intention of the agent" without computing all the sets. For that purpose, we will define a proof theory.
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 partial plans, we keep the one whose desire is mostly preferred by the agent. The second source of preferences is *argumentation*. In this case, two partial plans 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.

Acknowledgments

This work was supported by the Commission of the European Communities under contract IST-2004-002307, "Argumentation Service Platform with Integrated Components".

References

Amgoud, L., and Cayrol, C. 2002. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence* 34:197–216.

Amgoud, L. 2003. A formal framework for handling conflicting desires. In *Proceedings of the 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'2003*, 552–563.

Bratman, M.; Israel, D.; and Pollack, M. 1988. Plans and resource bounded reasoning. *Computational Intelligence.* 4:349–355.

Bratman, M. 1987. *Intentions, plans, and practical reason.* Harvard University Press, Massachusetts.

Cohen, P. R., and Levesque, H. J. 1990a. Intention is choice with commitment. In *Artificial Intelligence*, volume 42.

Cohen, P. R., and Levesque, H. J. 1990b. Rational interaction as the basis for communication. In *In P. R. Cohen, J. Morgan and M. E. Pollack, eds. Intentions in communication*, 221–256.

de Kleer, J. 1986a. An assumption-based tms. *Artificial Intelligence* 28:127–162.

de Kleer, J. 1986b. Extending the atms. *Artificial Intelligence* 28:163–196.

de Kleer, J. 1988. A general labeling algorithm for atms. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI)*, 188–192.

Doyle, J. 1992. *Rationality and its role in reasoning*, volume 8. Computational Intelligence.

Pearl, J. 1993. From conditional ought to qualitative decision theory. In *Proceedings of UAI'93*, 12–20.

Rao, A. S., and Georgeff, M. P. 1991. Modeling rational agents within a bdi architecture. In *Proceedings of KR'91.*

Rao, A. S., and Georgeff, M. P. 1992. An abstract architecture for rational agents. In *Proceedings of KR'92.*

Rao, A. S., and Georgeff, M. P. 1995. Bdi agents: from theory to practice. In *Proceedings of the 1st International Conference on Multi Agent Systems*, 312–319.