# Calcul des intentions d'agent à partir de ses désirs

Leila Amgoud * Andréas Herzig * David Mercier *
amgoud@irit.fr herzig@irit.fr dmercier@irit.fr

*Institut de Recherche en Informatique de Toulouse (IRIT)
118, route de Narbonne
31062 Toulouse Cédex – FRANCE

**Résumé :**
Une des architectures les plus célèbres pour les systèmes orientés agent est l'architecture BDI. Elle considère le système comme étant un agent rationnel ayant certaines attitudes *mentales* : les croyances (B), les désirs (D) et les intentions (I).
Un agent peut avoir des désirs contradictoires. Cependant, ses intentions sont un sous-ensemble cohérent des désirs que l'agent s'engage à réaliser.
Dans cet article, nous présentons un système qui calcule à partir des désirs d'un agent l'ensemble de ses intentions. Nous montrons comment les conflits peuvent surgir entre les désirs, comment formaliser ces conflits et enfin comment les résoudre.

**Mots clés :** Désirs, Intentions, Conflits.

**Abstract:**
One of the most famous architectures for agent-oriented system is the BDI one. It views the system as a rational agent having certain *mental attitudes* of belief, desire and intention (BDI). An agent can have conflicting desires. However, its intentions is a consistent subset of the desires that the agent is committed to achieve.
Inspired from work on argumentation [1, 7], in this paper we present a formal framework for handling conflicting desires. We show how conflicts may arise between desires and we give a flavour way to resolve them. Finally 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*.

**Key words :** Desires, Intentions, Conflicts.

## 1 Introduction

Plusieurs applications logicielles ont été conçues et implémentées en utilisant la notion d'agent autonome. Ces applications varient du commerce électronique à de larges applications industrielles. Dans tous ces cas disparates, la notion d'autonomie est employée pour dénoter le fait que le logiciel a la capacité de décider, par lui-même, quels buts il devra adopter et comment ces buts devront être réalisés.
Différentes approches ont émergé comme candidates à l'étude des systèmes orientés agent [2, 3, 4, 6, 12, 10, 11]. Une de ces architectures considère le système comme un agent rationnel ayant certaines attitudes *mentales* : les croyances (B), les désirs (D) et les intentions (I). Un agent peut avoir des désirs contradictoires. Cependant, ses intentions sont un sous-ensemble cohérent de désirs que l'agent s'engage à réaliser.
Dans [5], les auteurs ont exploré des principes régissant l'équilibre rationnel parmi les croyances, les buts, les actions et les intentions d'un agent. Dans [10] on a montré comment différents agents rationnels peuvent être modelisés en imposant certaines conditions sur la persistance des croyances, des désirs ou des intentions d'un agent (le modèle BDI). En théorie de la décision, Pearl [9] a illustré comment des agents de planification sont équipés de buts – définis en tant que désirs avec des engagements – et sont chargés de la tâche de découvrir (ou d'exécuter) des actions dans un certain ordre pour réaliser ces buts. Dans [8], les auteurs se sont intéressés à l'étude des différents conflits pouvant exister entre des désirs.
La plupart des formalisations sont assez sophistiquées pour manipuler beaucoup d'aspects des agents BDI. Cependant, elles ne montrent pas comment des intentions sont calculées à partir de l'ensemble des désirs. En d'autres termes, il n'est pas clair comment un agent choisit un sous-ensemble de ses désirs.
Nous présentons dans cet article un système pour traiter des désirs contradictoires. Nous supposons qu'un agent est doté d'une base de désirs, d'une base de plans pour réaliser ces derniers (nous ne nous intéressons pas à la manière dont ces plans ont été générés), et enfin d'une base de connaissances contenant les éventuelles contraintes d'intégrité et des connaissances sur l'environnement. Ensuite nous montrons comment des conflits peuvent surgir entre les désirs, comment formaliser ces conflits et enfin comment les résoudre pour obtenir les intentions de l'agent. Nous montrons aussi que l'ensemble des désirs peut être divisé en trois catégories :

1. Les intentions de l'agent.
2. Les désirs rejetés.
3. Les désirs en suspens.

## 2 Les définitions de base

Dans cette section, nous présentons les notions de base d'un système de traitement des désirs à savoir : un langage logique, un désir, une action, un arbre de réalisation d'un désir et enfin la contrariété entre les actions.

### 2.1 Le langage logique

Soit $\mathcal{L}$ un langage propositionnel. $\vdash$ dénote l'inférence classique et $\equiv$ dénote l'équivalence logique.

**Définition 1** *Un* littéral *est soit une proposition atomique p de $\mathcal{L}$ soit la négation d'une proposition atomique $\neg p$.*

**Définition 2 (Règle)** *Une* règle *est une expression de la forme $\varphi_1 \wedge \ldots \wedge \varphi_{n-1} \rightarrow \varphi_n$ avec $\varphi_1$, . . ., $\varphi_n$ sont des littéraux.*

Chaque agent est supposé doté de trois bases $<\mathcal{D}, \mathcal{P}, \Sigma>$ telles que :

- $\mathcal{D}$ contient des formules de $\mathcal{L}$. Les éléments de $\mathcal{D}$ représentent les désirs de l'agent. Par exemple, l'agent $ag$ peut avoir comme désirs : *finir un article*, *faire un voyage en Afrique*, etc...
  Notons que l'ensemble $\mathcal{D}$ peut être inconsistant. Cela veut dire qu'un agent peut avoir des désirs contradictoires.
- $\mathcal{P}$ est considérée comme une base des plans. Elle contient des *règles* telles présentées dans la définition 2. Une telle formule veut dire que pour réaliser $\varphi_n$, il faudra réaliser $\varphi_1$, . . ., $\varphi_{n-1}$.
- $\Sigma$ contient des formules de $\mathcal{L}$. Ces dernières représentent les connaissances de l'agent et les contraintes d'intégrité.

### 2.2 La notion de désir

Un désir est soit un élément $h$ de $\mathcal{D}$, soit un désir induit de $h$, appelé aussi *sous-désir* de $h$. Notons que les sous-désirs ne sont pas stockés dans la base $\mathcal{D}$. Formellement :

**Définition 3 (Désir)** *Un* désir *est :*
- *une formule $h \in \mathcal{D}$.*
- *une formule $h$ telle que $\exists \varphi_1 \ldots \wedge \varphi_n \rightarrow h \in \mathcal{P}$*
- *une formule $h$ telle que $\exists \varphi_1 \wedge h \ldots \wedge \varphi_n \rightarrow h' \in \mathcal{P}$. Dans ce cas, $h'$ est appellée* sous-désir *de h. La fonction Subdesire(h) retourne l'ensemble $\{\varphi_1, \ldots \varphi_n\}$ des sous-désirs d'un désir $h'$.*

### 2.3 La notion d'action

Un désir peut avoir un plan pour le réaliser. Nous regroupons la notion de désir et son plan de réalisation dans la notion d'*action*.

**Définition 4 (Action)** *Une* action *est une paire $d = < h, H >$ où :*
- *$h$ est un désir.*
- *Si $\exists \varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow h \in \mathcal{P}$ alors $H = \{\varphi_1, \ldots, \varphi_n\}$ sinon $H = \emptyset$.*

*$h$, le désir à réaliser, est dénoté par $h = Desire(d)$. $H$, le plan à suivre pour la réalisation du désir $h$, est dénoté par $H = Plan(d)$.*
*On dénote par $\aleph$ l'ensemble de toutes les actions construites sur $<\mathcal{D}, \mathcal{P}, \Sigma>$.*

**Remarque 1** –
- *Lorsque l'ensemble $H$ ou le plan est vide, cela veut dire que soit le désir n'a pas encore de plan pour le réaliser, soit c'est un* désir atomique *et donc il n 'a pas besoin de plan.*
- *Il peut y avoir plusieurs plans pour réaliser un même désir. Dans ce cas, il y aura autant d'actions pour le désir qu'il a de plans.*

L'exemple suivant illustre les définitions précédentes :

**Exemple 1** *Considérons un agent qui a les deux désirs suivants :*

1. *Faire un voyage en Afrique centrale. $(voy)$*
2. *Finir un article avant de partir en voyage. $(fa)$*

*Dans ce cas la base des désirs de l'agent est $\mathcal{D} = \{voy, fa\}$. En plus de cette base, nous supposons que l'agent possède les deux bases $\mathcal{P}$ et $\Sigma$ suivantes :*

$$\mathcal{P} = \begin{cases} b \wedge vac & \rightarrow & voy \\ w & \rightarrow & fa \\ ag & \rightarrow & b \\ a & \rightarrow & b \\ hop & \rightarrow & vac \\ med & \rightarrow & vac \end{cases}$$

$$\Sigma = \left\{ \begin{array}{lcl} w & \rightarrow & \neg ag \\ w & \rightarrow & \neg med \end{array} \right.$$

*avec :*

| | | |
|---|---|---|
| $b$ | = | *"avoir les billets"* |
| $vac$ | = | *"être vacciné"* |
| $voy$ | = | *"faire le voyage"* |
| $w$ | = | *"aller au travail"* |
| $fa$ | = | *"finir l'article"* |
| $ag$ | = | *"passer à l'agence"* |
| $a$ | = | *"avoir un ami"* |
| $hop$ | = | *"aller à l'hôpital"* |
| $med$ | = | *"aller chez le médecin"* |

*L'agent possède deux désirs. Cependant, nous avons aussi les désirs induits ou les sous-désirs suivants :* $w$, $b$, $vac$, $ag$, $a$, $hop$ *et* $med$.
*Considérons maintenant les actions suivantes :*
– $a_1$ = <*voy*, {*b, vac*}>
– $a_2$ = <*fa*, {*w*}>

*Cela veut dire que pour voyager, l'agent doit i) avoir les billets et ii) se faire vacciner. Et pour finir son article, l'agent doit aller à son travail.*

*En outre on a aussi l'action suivante :*
– $a_3$ = <*w*, $\emptyset$ >

*Cette dernière action signifie que :*

1. *Soit le désir d'aller au travail est un désir atomique : l'action d'aller au travail est atomique, on y va et c'est tout.*
2. *Soit le plan n'a pas encore été donné dans* $\Sigma$*, peut être est-il nécessaire de prendre sa voiture ou le bus.*

La réalisation d'une action, cad l'exécution de son plan, nécessite dans certaines situations la décomposition de ce plan. Chaque élément du plan initial donne lieu à une nouvelle action avec un nouveau plan. On appelle ce genre d'action des *sous-actions*. Formellement :

**Définition 5 (Sous-action)** *Soit* $a_1$ *et* $a_2$ *deux actions de* $\aleph$. $a_1$ *est une* sous-action *de* $a_2$ *ssi* $Desire(a_1) \in Plan(a_2)$. *Autrement dit,* $Desire(a_1)$ *est un sous-désir de* $Desire(a_2)$.

**Exemple 2** *Dans l'exemple précédent,* $a_1$ = <*voy*, {*b, vac*}> *a comme sous-actions :*
– $a_{11a}$ = <*b*, {*ag*}>
– $a_{11b}$ = <*b*, {*a*}>
– $a_{12a}$ = <*vac*, {*med*}>
– $a_{12b}$ = <*vac*, {*hop*}>

*Pour chercher les billets, l'agent a deux possibilités : soit qu'il se présente lui-même à l'agence soit demander à un ami à lui d'aller les chercher. De même pour sa vaccination. L'agent peut soit aller chez un médecin soit aller dans un hôpital.*

Une action qui n'est pas une sous-action d'aucune autre action est appelée *une racine*. Formellement :

**Définition 6 (Racine)** *Soit* $a$ *une action de* $\aleph$. *Elle est dite* une racine *ssi* $\not\exists\ a' \in \aleph$ *tel que* $Desire(a) \in Plan(a')$.

**Remarque 2** *Notons que les actions correspondantes aux désirs de* $\mathcal{D}$ *ne sont pas forcément des actions racines. Considérons l'exemple suivant :*

**Exemple 3** *Soient* $\mathcal{D} = \{b, c\}$ *et* $\mathcal{P} = \{a \rightarrow b, b \rightarrow c\}$ *et* $\Sigma = \emptyset$. *L'action* $< b, \{a\} >$ *n'est pas une racine car c'est une sous-action de* $< c, \{b\} >$.

Une action peut avoir des conséquences.

**Définition 7 (Conséquences d'une action)** *Soit* $a \in \aleph$. $Consequence(a) = \{\phi_i$ *telle que* $Plan(a) \cup \mathcal{P} \cup \Sigma \vdash \phi_i\}$.

### 2.4 La contrariété entre actions

Deux actions $a_1$ et $a_2$ peuvent entrer en conflit entre elles pour une des raisons suivantes :
– un conflit désir-désir ie $Desire(a_1) \cup Desire(a_2) \vdash \bot$
– un conflit plan-plan ie $Plan(a_1) \cup Plan(a_2) \vdash \bot$.
– un conflit conséquence-conséquence ie $Consequence(a_1) \cup Consequence(a_2) \vdash \bot$.
– un conflit plan-conséquence ie $Plan(a_1) \cup Consequence(a_2) \vdash \bot$ ou bien $Plan(a_2) \cup Consequence(a_1) \vdash \bot$.

**Exemple 4** *Considérons un agent doté des bases suivantes :* $\mathcal{D}$ = {*a, ps, e*}, $\Sigma$ = $\{ps \rightarrow \neg ws\}$ *et*

$$\mathcal{P} = \left\{ \begin{array}{lcl} ws \wedge wd & \rightarrow & a \\ v & \rightarrow & ps \\ \neg v & \rightarrow & e \end{array} \right. \quad \textit{avec :}$$

*ws* = *"travailler samedi"*
*wd* = *"travailler dimanche"*
*a* = *"l'article est réalisé"*
*v* = *"prendre la voiture"*
*ps* = *"aller à la plage samedi"*
*e* = *"avoir fait des économies"*

$\aleph$ *contient les actions suivantes :*
– $a_1$ = <*a*, $\{ws, wd\}$>.
– $a_2$ = <*ps*, $\{v\}$>.
– $a_3$ = <*e*, $\{\neg v\}$>.
*Il existe bien un conflit plan-plan entre* $a_2$ *et* $a_3$. *Sans même regarder les conséquences des deux actions, le conflit est apparu. Les deux actions ont ici deux plans contradictoires.*
*L'action* $a_2$ *a pour conséquence* $\neg ws$ *(*$Consequence(a_2) = \{ps, \neg ws\}$*). Elle est donc en conflit du type plan-conséquence avec l'action* $a_1$. $a_1$ *et* $a_2$ *sont aussi en conflit du type conséquence-conséquence. Contrairement au conflit entre* $a_2$ *et* $a_3$, *les plans de* $a_1$ *et* $a_2$ *semblaient compatibles, c'est dans l'implication des plans que surgit le conflit.*

Ces quatre formes de conflits entre les actions sont regroupées dans une seule notion d'*attaque* définie comme suit :

**Définition 8 (Attaque)** *Soit* $a_1$ *et* $a_2$ *deux actions de* $\aleph$. $a_1$ attaque $a_2$ *ssi* $\{Desire(a_1), Desire(a_2)\} \cup Plan(a_1) \cup Plan(a_2) \cup \mathcal{P} \cup \Sigma \vdash \bot$.

**Exemple 5** *Dans l'exemple 1,* $a_{11a}$ = <*b*, $\{ag\}$> *attaque* $a_2$ = <*fa*, $\{w\}$>. *En effet, Plan(*$a_{11a}$*)* $\cup \Sigma \vdash \{\neg w\}$ *et Plan(*$a_2$*)* = $\{w\}$, *ainsi* $a_{11a}$ *et* $a_2$ *sont incompatibles.*

**Propiété 1**
– *La relation attaque n'est pas* reflexive.
– *La relation attaque est* symétrique.
– *La relation attaque n'est pas* transitive.

**Exemple 6** *Soit un agent doté de la base des plans suivante :*

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

*Avec la base de désirs D=*$\{h_1, h_2, h_3\}$ *et* $\Sigma = \emptyset$.
*Considérons les trois actions suivantes :*
– $a_1$ = <$h_1$, $\{a\}$>
– $a_2$ = <$h_2$, $\{b\}$>
– $a_3$ = <$h_3$, $\{\neg a, \neg b\}$>

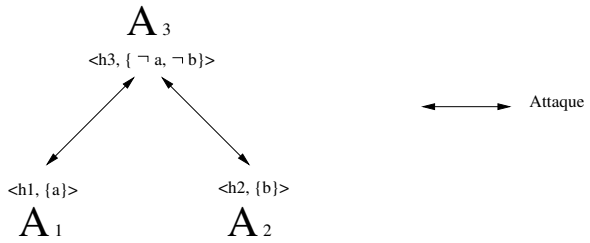


FIG. 1 – Les conflits

La figure 1 synthétise les différents conflits existant entre les 3 actions. $a_1$ attaque $a_3$ et $a_3$ attaque à son tour $a_2$ et pourtant $a_1$ n'attaque pas $a_2$.

**Remarque 3** *Une action peut s'attaquer elle-même. Considérons l'exemple suivant :*

**Exemple 7** *Soit* $\mathcal{D} = \{\neg a'\}$, $\mathcal{P} = \{a \wedge b \wedge c \rightarrow \neg a'\}$ *et* $\Sigma = \{a \rightarrow a'\}$. *L'action* $< \neg a', \{a, b, c\} >$ *est en conflit avec elle-même.*

### 2.5 La notion d'arbre de réalisation

Dans l'exemple du voyage en Afrique, l'action $a_2$ n'attaque pas l'action $a_1$, par contre elle attaque bien une de ses sous-actions comme $a_{11a}$. Dans ce cas, on ne peut pas dire que les deux désirs $Desire(a_1)$ et $Desire(a_2)$ sont réalisables au même temps.
Pour vérifier si un désir donné est réalisable, l'action correspondante et toutes ses sous-actions doivent être prises en compte. D'ou la necessité d'introduire la notion d'arbre de réalisation d'un désir. Il s'agit d'un *arbre ET*. Les noeuds de l'arbre sont des actions, les arcs représentent la relation de parenté (sous-action) et la racine de l'arbre est une action pour le désir que l'on veut tester.
Il s'agit d'un noeud ET car toutes les sous-actions doivent être réalisées. Lorsque pour un même désir, il y a plusieurs plans pour le réaliser, on prend un seul plan (ie on choisit une action possible). Formellement

**Définition 9 (Arbre de réalisation)** *Etant donné un triplet ¡*$\mathcal{D}, \mathcal{P}, \Sigma$*¿, un* arbre de réalisation $g$ *d'un désir* $h$ *est un arbre fini tel que :*
– $< h, H >$ *est la racine de l'arbre.*
– *Un noeud* $< h', \{\varphi_1, \ldots, \varphi_n\} >$ *possède exactement* $n$ *fils*[1] $< \varphi_1, H'_1 >$, ..., $< \varphi_n, H'_n >$.

[1] Si un désir possède plusieurs plans pour le réaliser, uniquement un seul est considéré dans un arbre.

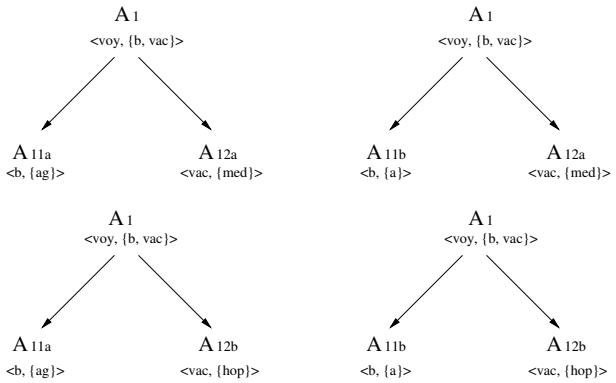


FIG. 2 – Des arbres de réalisation

– *Les actions des feuilles de l'arbre sont des actions atomiques.*

*La fonction* $Racine(g) = h$ *retourne le désir de la racine de l'arbre. La fonction* $Noeuds(g)$ *retourne l'ensemble de toutes les actions de l'arbre* $g$.
$\mathcal{G}(\aleph)$ *dénote l'ensemble de tous les arbres de réalisation que l'on peut construire à partir de l'ensemble* $\aleph$.

**Remarque 4** *Un désir peut avoir plusieurs arbres de réalisation. Ce cas se présente lorsqu'un de ses sous-désirs possède plusieurs plans.*

**Exemple 8** *Toujours sur le même exemple 1, le désir "voy" dont l'action est* $a_1$ = <*voy,* $\{b,$ *vac*$\}$> *possède quatre arbres de réalisation :*

Considérons maintenant l'exemple suivant :

**Exemple 9** *Soit un agent doté des bases suivantes :* $\mathcal{D} = \{c\}$, $\Sigma = \emptyset$ *et* $\mathcal{P} = \{a \wedge b \rightarrow c, e \rightarrow a, d \wedge g \rightarrow b, \neg e \rightarrow g, x \rightarrow d\}$. *L'arbre de réalisation du désir c contient les actions suivantes :* $< c, \{a, b\} >$, $< a, \{e\} >$, $< b, \{d, g\} >$, $< g, \{\neg e\} >$, $< d, \{x\} >$, $< e, \emptyset >$, $< \neg e, \emptyset >$ *et* $< x, \emptyset >$. *Il est clair que* $< a, \{e\} >$ *attaque* $< g, \{\neg e\} >$.

Cet exemple montre bien qu'un arbre de réalisation d'un désir peut contenir des actions qui s'attaquent mutuellement. Un arbre de réalisation peut aussi contenir une action qui s'attaque elle même. De tels arbres de réalisation sont dit inconsistants. Formellement :

**Définition 10 (Consistance)** *Soit* $g \in \mathcal{G}(\aleph)$. $g$ *est dit* inconsistant *ssi* $\exists a_1, a_2 \in Noeud(g)$ *tels que* $a_1$ *attaque* $a_2$. *Sinon* $g$ *est dit* consistant.

Il est clair que les désirs dont tous les arbres de réalisation sont inconsistants sont des *désirs impossibles à réaliser*.

# 3 Un modèle formel de traitement des désirs

A partir des définitions précédentes, on peut introduire maintenant ce qu'est un système de traitement des désirs d'un agent.

**Définition 11 (Système de traitement des désirs)** *Etant donné un triplet ¡*$\mathcal{D}, \mathcal{P}, \Sigma$*¿, un système de traitement des désirs (STD) est un couple* $< \aleph, Attaque >$ *avec* $\aleph$ *un ensemble d'actions et* $Attaque$ *une relation binaire de conflit entre les actions (*$Attaque \subseteq \aleph \times \aleph$*).*

Notre but est de trouver parmi les désirs de l'agent (les éléments de $\mathcal{D}$), ceux qui deviennent ses intentions. A partir de l'ensemble $\mathcal{D}$, nous distinguons trois catégories de désirs :

1. $\mathcal{I}$ est l'ensemble des ***intentions*** de l'agent.
2. $\mathcal{D}_{nr}$ = $\{h \in \mathcal{D}$ tq $\forall\ g \in \mathcal{G}(\aleph)$ et $Racine(g)$ = $h$, alors $g$ est inconsistant$\}$ regroupe les *désirs non réalisables* de l'agent. Il s'agit des désirs dont tous les arbres de réalisation sont inconsistants.
3. $\mathcal{D}_s = \mathcal{D} \setminus (\mathcal{I} \cup \mathcal{D}_{nr})$ est l'ensemble des *désirs en suspens*.

Dans ce qui suit, nous montrons comment calculer les intentions de l'agent ainsi que ses désirs en suspens. Pour cela, nous introduisons une nouvelle notion qui est celle de *sans-conflits*.

**Définition 12 (Sans-conflits)** *Soit* $< \aleph, Attaque >$ *un STD et* $S \subseteq \mathcal{G}(\aleph)$. $S$ *est dit* sans conflits *ssi :*
– $\forall g_i \in S$, $g_i$ *est consistant.*
– $\nexists\ g_1$ *et* $g_2$ *dans* $S$ *tels que :* $\exists a_1 \in Noeud(g_1)$ *et* $\exists a_2 \in Noeud(g_2)$ *telles que* $a_1$ *attaque* $a_2$.

**Exemple 10** *Dans la figure 3, l'ensemble* $S_1$ *est sans-conflits, alors que* $S_2$ *ne l'est pas car* $D_{11a}$ *attaque* $D_2$.

Pour calculer les intentions d'un agent, nous commençons d'abord par calculer les sous-ensembles maximaux (pour l'inclusion) sans-conflits de $\mathcal{G}(\aleph)$. Ces sous-ensembles sont appelés des *extensions*. Formellement :

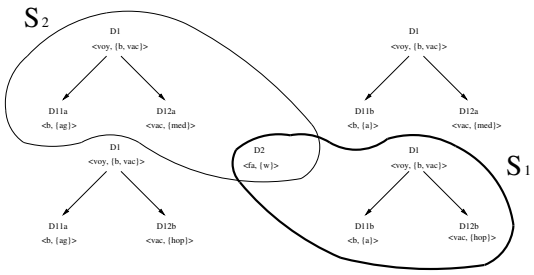


FIG. 3 – Les extensions

**Définition 13 (Extension)** *Soit* $< \aleph, Attaque >$ *un STD et* $S$ *un sous-ensemble de* $\mathcal{G}(\aleph)$. *S est une* extension *ssi :*

1. $S$ *est sans-conflits.*
2. $S$ *est un ensemble maximal (pour l'inclusion) qui vérifie 1).*

*Notons qu'un STD peut avoir plusieurs extensions. Soit* $S_1$, ..., $S_n$ *les extensions trouvées. La fonction* $Desirs(S_i) = \{h \in \mathcal{D}$ *tel que* $\exists\ g \in S$ *et* $Racine(g) = h\}$. *En d'autres termes, cette fonction retourne l'ensemble des désirs qui appartiennent à* $\mathcal{D}$ *et qui sont des racines des éléments de* $S_i$ .

**Propiété 2** *Soit* $< \aleph, Attaque >$ *un STD et* $S_1$, ..., $S_n$ *les différentes extensions du système. Les ensembles* $Desirs(S_1)$, ..., $Desirs(S_n)$ *ne sont pas tous maximaux pour l'inclusion.*

**Exemple 11** *Dans l'exemple 1, nous pouvons construire 5 arbres de réalisation. Les quatre correspondant au désir* $voy$ ($g_1$, $g_2$, $g_3$, $g_4$) *et un autre correspondant au désir* $fa$ *soit* $g_5$. *Nous avons deux extensions :*
– $S_1 = \{g_1, g_2, g_3, g_4\}$ *avec* $Desirs(S_1) = \{voy\}$
– $S_2 = \{g_4, g_5\}$ *avec* $Desirs(S_2) = \{voy, fa\}$
*Nous avons bien* $Desirs(S_1) \subseteq Desirs(S_2)$.

Le but est de réaliser le maximum de désirs de l'agent. Dans l'exemple précédent, l'agent peut réaliser ses deux désirs dans l'extension $S_2$. Donc parmi les extensions trouvées, nous ne gardons que celles qui réalisent le maximum de désirs.
Soit $Desirs(S_1)$, ..., $Desirs(S_i)$ les ensembles maximaux pour l'inclusion parmi $Desirs(S_1)$, ..., $Desirs(S_n)$.

**Définition 14** *Soit* $< \aleph, Attaque >$ *un STD.* $\mathcal{I} = \bigcap \{Desirs(S_1), ..., Desirs(S_i)\}$.

# 4 Conclusion

Dans cet article nous avons défini un modèle formel de calcul des intentions d'un agent. Nous avons montré comment et quand est ce les désirs d'un agent peuvent être en conflit entre eux. Nous avons formalisé cette notion de conflits et puis nous avons montré comment les résoudre. La résolution de ces conflits nous permet de trouver l'ensemble des intentions de l'agent.
Une première voie d'élargissement de ce travail serait d'introduire des préférences entre les désirs pour affiner la classification des désirs en laissant moins de désirs en suspens.
Nous pouvons aussi enrichir le modèle d'une autre forme de conflit. Outre un conflit de plans, deux actions peuvent être en conflit pour un problème de raison : le pourquoi du désir. Deux actions peuvent ainsi être en conflit mais l'une d'entre elles peut avoir une "bonne" raison de se réaliser.

# Références

[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. Istrael, 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, logic programming and $n$-person games. *Artificial Intelligence*, 77 :321–357, 1995.

[8] J. Land, L. van der Torre, and E. Weydert. Two kinds of conflicts between de-

sires (and how to resolve them). In *Proceedings of AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents (GTDT2001) AAAI press, Stanford.*, 2001.

[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.