A framework for modeling the relationships between the rational and behavioral reactions of assisting conversational agents

François Bouchet[1] and Jean-Paul Sansonnet[1]

LIMSI-CNRS, BP 133, F-91403 Orsay Cedex, France

Abstract. In order to facilitate the access and the usage to the general public of the rapidly expanding applications and services, particularly on the Internet, new assisting tools are needed with two main requirements: naturalness and acceptability. Conversational Agents are a promising approach for the support of the Function of Assistance, especially when they focus on the Natural Language modality. In this context, the assisting agents cannot rely only on rational reasoning over the structure and the functioning of the assisted application in order to resolve the user's questions. Agents must also express behavioral reactions that involve social relationships, character traits and affects. Once studied in separate communities, the relationships between rational and behavioral reactions are now considered a key issue. Some models have been proposed where the relationships are preset and often rigid while we think that more flexible tools would be handy to make experimental studies of this problem. In the first part of this paper, we propose a flexible framework for modeling the relationships between the rational and behavioral reactions of an assisting agent. Then this framework is used to support a first case-study, based on cognitive biases.

1 Introduction

1.1 Assisting Conversational Agents

With the increase of computer applications and services together with their functionalities and the number of general public users, the need for an efficient assistance to novice users, long time addressed [1], is now becoming crucial. However the renewal of the scientific interest in the Function of Assistance is above all prompted by the new issues engendered by this population of novice users. In the paper, the term 'Function of Assistance' is employed in a generic way to refer to the activity of providing help to a human user involved in some task; it ranges from butler agents [2] to conventional help systems but excludes for example the domains of learning and training. As opposed to computer experts who rely on reference manuals or to corporate employees who are often trained before they are compelled to use daily a small amount of desktop applications, it has been shown that novice computer users, when in need for assistance, tend to prefer asking help from a "friend behind their shoulder", as defined in [3], rather than from the traditional help system available on their computer. Indeed, the acceptability factor is now considered a major usage barrier to the expanding of assisting tools. Although the salience of the task at hand can partially explain the so-called "paradox of motivation"

defined by [4] where people are not motivated to learn how an application works but rather to achieve their task, the observation of cognitive phenomena like the "Persona effect" defined by [5] stating that the personification of an agent increases its acceptability or the positive impact of natural language interaction for assistance purposes exhibited by [6] proves it can also be related to a need for a more intuitive interaction. From there, Embodied Conversational Agents (ECA) [7] have been envisioned to provide more natural assistance for novice users, thus defining a subclass of ECA dedicated to the Function of Assistance that we shall refer to as Assisting Conversational Agents (ACA) [2] .

In the same way as many studies have been focusing on the improvement of the physical believability of ECA, for instance through expressive emotions [8, 9], we believe that to go across the "uncanny valley" exhibiting a decorrelation between the realism of the personification and the acceptability [10] would require agents not only physically but also cognitively believable, *i.e.* able to exhibit complex behaviors similar to human beings' ones (an increased believability improving as well the perceived human-likeness [11]). To go towards this direction, we propose to provide ACA with: 1) personality parameters similar to the ones used in psychological studies to characterize human beings, 2) cognitive constraints deeply integrated into the agent underlying architecture to emulate restrictions human beings would have in similar situations.

Researchers in Multi-Agents Systems (MAS) have explored, since the beginning, the concept of "cognitive agents", using cognitive theories to model agents' reasoning capacities. For example, by adding a layer over existing agent creation tools, like CoJACK [12, 13] for JACK which takes into account parameters simulating some physiological human constraints like the duration taken for cognition, working memory limitations (*e.g.* "losing a belief" if the activation is low or "forgetting the next step" of a procedure), fuzzy retrieval of beliefs, limited focus of attention or the use of moderators to alter cognition. Attempts to add emotions to classical BDI architectures [14] have also been undertaken, for instance to take into account fear, anxiety or selfconfidence by adding parameters like fundamental desires, capabilities and resources [15]. The idea of adding degrees in multivalued logic for beliefs, desires and intentions has also been explored in [16], with the case of the Łukasiewicz logic. It has been shown as well that the order in which heuristics are applied can significantly impact the agent's perceived personality: if we consider classes of rules (like Beliefs, Desires, Intentions or Obligations), it can even be a way to characterize the agent's personality, with traits like stable, selfish or social [17].

1.2 Towards rational and behavioral agents experimentation

A typical ACA architecture (Fig. 1) is composed of two main parts:
1) The user interface ($\mathscr{I}$): is in charge of the conversational interaction between the help system and the user. Conversational agents are inherently multimodal [18] but in the case of the assisting agents, the Natural Language modality plays a prominent role in the process of assistance because novice users prefer to express their problems in Natural Language when in presence of an ACA [3]. Hence, as a simplification, we will not consider the multimodal dimension in this paper. For example, while aware of their

relevance to the acceptability factor [19], we will let aside the issues related to the graphical and gestural personification of the agent (its appearance and animation as a virtual character on screen). Instead, we focus on the processing of the Natural Language help requests put by the users. Consequently the two main devices of the interface are:
— The Natural Language Processing chain (NLP-chain): it translates user's textual utterances into a formal form, the Formal Request Language (FRL), destined to the agent;
— The Natural Language Expressing chain (NLE-chain): it expresses the agent's formal answers to the user, using mainly Natural Language. The NLE-chain uses also the FRL formalism which has been designed to support both users' questions and agent's answers.

2) The assisting agent ($\mathscr{A}$): handles the help requests of the users once they have been translated in a formal form. The assistant is composed of two sub parts:
— The Model of assistance ($\mathscr{M}$): it is a formal model of the application dedicated to the support of the Function of Assistance. Considering several applications to be assisted, all the models share the same ontology, but a specific model must be instantiated for each application.
— The Rational engine ($\mathscr{E}_r$): it processes the formal requests while using a set of symbolic reasoning heuristics that are applied to the model $\mathscr{M}$; then it builds a formal answer that is sent back to the interface which in turn, synthesizes the answer. The heuristics of the rational engine should be as generic as possible, meaning that they should not depend on a particular application. In this context, we have developed a pro-

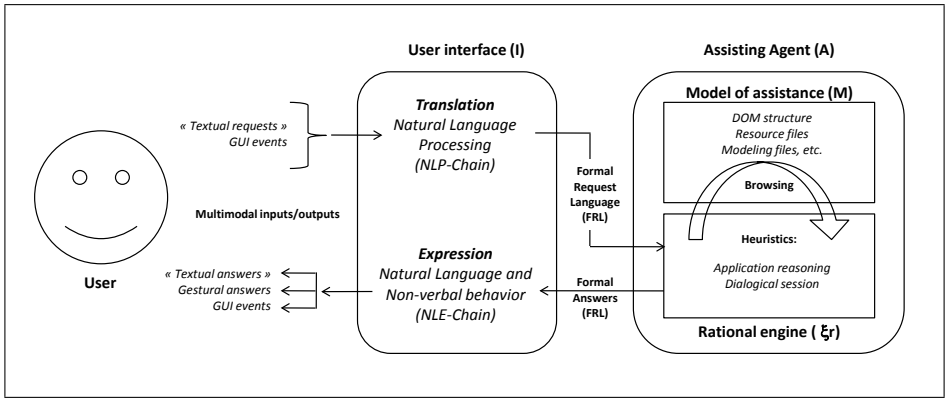

Fig. 1. A typical ACA architecture

gramming framework called DIVA [20], which the objectives to be both:
1) Web-based, that is dedicated to the development of ACAs for applications and services on the Internet. The assisting agent is personified by a virtual animated character, fully integrated in the Web pages. The agent can interact in Natural Language with the users (they type their requests in a 'chatbox' and the agent replies in a 'balloon'). The agent can also access the Document Object Model (DOM) content [21] of the assisted

page: it can consult but also modify the DOM structure. Actually, DIVA stands for "DOM-Integrated Virtual Agents" which emphasizes its specific capability of browsing and reasoning over the DOM structure of the Web pages.
2) open-source for research purposes, to facilitate the development and deployment of controlled experiments where ordinary users can interact with small applications assisted by ACAs.

The DIVA framework has proven to be cost-effective and has enabled us to carry out several experimental applications (*cf.* examples on the DIVA website [20]). However, first experimentations with novice users [11] have revealed some limitations of assisting agents based on a strict rational engine. They have pointed out at least three obvious phenomena that hamper the human-likeness and dialogical naturalness of the agents:
1) Repetition of the agent's cooperative reactions: the agent is always responsive whatever happens during the session, whatever the character it endorses and whatever the user asks. For example, the agent can neither hide information it possesses from the user nor refrain from executing an operation it can do when commanded by the user.
2) Repetition of the answer's schemes: the agent provides the same information several times without any linguistic variability.
3) Repetition of the rational reactions: if the user puts several times the same request, each time the agent reacts to the requests independently from the session that is as if the request hadn't been put before.

These observations emphasize the importance of more human-like behaviors as far as the general public is concerned. This is the reason why we propose in this paper a modified ACA architecture where the assisting agent is provided with both a personality model integrated into the model of assistance $\mathscr{M}$ and a correlated Behavioral Engine ($\mathscr{E}_b$) that works in conjunction with the Rational engine ($\mathscr{E}_r$). In this study, we will rely on the state of the art research [22] [23] for the definition of the personality model and we will focus the discussion on the relationships between the two engines: $\mathscr{E}_r$ and $\mathscr{E}_b$.

Several authors have discussed the relationships between strict rational reasoning (often associated with Artificial Intelligence) and more psychological-oriented behavioral reasoning, (often associated with works in Affective Computing). First, Damasio has claimed the prominence of affects upon rationality [24] and the OCC model [25] has provided a first computationally tractable framework [26] but with a preset model, henceforth criticized and modified by further authors [27]. Indeed, the nature of the relationships between rational reasoning and behavioral reasoning in conversational agents is an open issue that requires many experiments before it could be settled. Consequently, we need to develop specific software tools to support such experimental research [28]. In this context, the main objective of the paper is to propose a flexible software architecture that can support the implementation and the experimentation (*e.g.* through DIVA experiments) of various scenarios of relationships between Rational and Behavioral agents (further abbreviated as *R&B* agents).

The outline of the paper is as follows: Section 2 is dedicated to the description of a supporting architecture for *R&B* agents with the Formal Request Language (FRL), the representation of the model of assistance ($\mathscr{M}$) and the Heuristic Description Language (HDL). In section 3 we put this architecture to the test by implementing an example of *R&B* scenario which is based on Cognitive Biases.

2 Agent architecture

2.1 Formal Request Language

As showed in Fig. 1, the Formal Request Language (FRL) supports the input/output channel between the Interface and the Agent. As we focus on the agent's architecture, we just give the notations used in the following sections. The FRL is the result of our previous work on the design of a NLP-chain for ACAs that is based on a specific corpus of 11 000 help utterances collected both from thesaurus source and experimental collecting with novice users. This work enabled an analysis and a classification of the linguistic domain related to the Function of Assistance [29], thus enabling the definition of a Formal Request Language. In the following we describe only FRL basic requests leaving aside complex requests like reported speech, conditional commands, past/future, *etc.*

A basic FRL request is of the form $F(X)$ where:
– F is inspired from searlian speech acts and is called *performative* in the following,
– X is the content. There is only one argument in basic requests.
There are four kinds of contents: $X \in \{R, A, P, V\}$
– R **reference** is a referential expression in the model $\mathscr{M}$;
– A **action** is the description of an operation executable in the environment;
– P **proposition** is a logical proposition above the model $\mathscr{M}$;
– V **value** is a typical value $\in V$ as defined in the model $\mathscr{M}$;

ask R|A|P	L asks for an information or checks if the proposition currently stands or if the action is known by the int.	Knowledge
tell P	L states an information or that the proposition currently stands	
reply V	L gives a value as an answer of request (ask,..) from the interlocutor	
know R|A|P	L states that he knows something about the content (when P, means he thinks it is true)	
unknown R|A|P	L states that he knows nothing about the content (or if P is true or false)	
mistrust P|V	L states that he thinks that the value is probably erroneous or the proposition is probably false	
why P|V	L asks why the proposition is currently standing or why the value has been replied	R reference A action P proposition V value
possible A	L asks if it is possible (availability, rights...) for him or the agent to execute the action	
how A	L asks how to do the action/procedure	
effect A	L asks what will be the consequences of performing the action A	
execute A|R	L commands the interlocutor to execute the action or to activate the main function of the referenced object	Action
repeat -	L commands the interlocutor to execute again the last executed action	
undo -	L commands asks the interlocutor to undo the last executed action	
suggest A|P	L encourages/suggests/allows the interlocutor to execute the action / to adopt the proposition as a goal	
object A|P	L discourages/objects/forbids the interlocutor to execute the action / to adopt the proposition as a goal	
intent A|P	L states that he has the intention to execute the action in the near future / he has just adopted the proposition as a goal	
judge P	L expresses a subjective opinion by stating the proposition	Feelings
feel P	L expresses a subjective feeling by stating the proposition	
like R|A|P|V	L expresses a subjective preference/liking for the content (sub case of judge)	
dislike R|A|P|V	L expresses a subjective dis-preference/disliking for the content (sub case of judge)	
bravo -	L congratulates the interlocutor about the topic	
criticize -	L criticizes the interlocutor about the topic (e.g. contains abuse)	
agree -	L replies yes to a yes/no question or agree with the topic	Dialogue
disagree -	L replies no to a yes/no question or disagree with the topic	
greet	L greets the interlocutor at the beginning of the session	
bye	L asks for the session to end	

'L' stands for the locutor (the user or the agent in replies)
'-' stands for the current focus of the dialogue session (IT)

Fig. 2. List of FRL performatives

Request subjects: The request's emitter (called the locutor) is always the verbal subject of the performative: $F(X) \equiv F_{\text{CURRENTLOCUTOR}}(X)$. Depending on the dialogue turn, it can be the user $\mathscr{U}$ or the agent $\mathscr{A}$. During a dialogue turn, several FRL requests can be sent in a sequence separated by a semicolon ';'.

Performatives: The list (completely given in Fig. 2) can be divided into four main categories (that can overlap):
– Knowledge: concerns mainly information about environment objects and actions
– Action: concerns mainly the management of actions
– Feeling: concerns mainly expressing subjective opinions and feelings
– Dialogue: concerns mainly the session handling and yes/no answers
When the user types a textual utterance in the agent's GUI 'chatbox', the NLP-chain maps it onto a FRL request (or a sequence of requests) and some semantic operations are performed like indexical processing (filling of 'I', 'you', ...).
USER INPUT: "I like you" $\Longrightarrow \text{LIKE}_{\text{u}}[\text{agent}]$
AGENT OUTPUT: $\text{FEEL}_{\text{a}}[\text{M}_{\text{h}}^{>}] \Longrightarrow$ "I am happy!" (notations are detailed in section 3).

2.2 Model of Assistance

The model of assistance ($\mathscr{M}$) of the agent is a symbolic structure that supports, in a single framework, the symbolic representation of all the information accessible to the agent. This section is only an overview of some main notations related to the modeling issue. We will present successively:
– The syntax and the skeleton of the ontology (main represented entities) of the model;
– The dynamics of the model, *i.e.* how the symbolic representation evolves over time;
– The basic functions of Model Query Language (MQL) which is the API of the model;
– The query *objects*. They are MQL queries that can be manipulated by any engine.
Several formalisms have been proposed for Knowledge Representation (KR): various levels of Logics, Semantic Networks, Description Logics, *etc.* Recently, tree based forms have been widely used in the Internet application in order to enhance the basic DOM (Document Object Model) form (RDF, OWL...). Because we focus on Web-based assisting agents that have to deal with DOM and XML-based representations, we have chosen a tree form for the support of the agent's model. However this model could easily be transposed into other KR formalisms, like a base of Prolog facts for example. The model is a tree where:
– Non terminal nodes are labeled by concepts. A concept is a Symbol or an integer (used to index list of sub concepts);
– Terminal nodes are conventional values: Symbols, Numbers, Boolean, Strings. We use a bracketed form to represent the tree (*e.g.* in Mathematica):

```
Model = Rootconcept[
    Concept1[
        Concept11[...],
        Concept12[...],
    ...]
    Concept2[
```

```
        Concept21[...],
        Concept22[...]
    ...],
...]
```

In this structure, each node is composed of a head (the concept's symbol) and a body (the subtree contained between the brackets), also called the value of the concept. Hence, concepts values can be conventional terminal values or more complex subtrees. This formal tree structure allows the organization of the concepts as a static ontology that defines the actual model of assistance. The first level of the ontology is defined by the root concept $\mathscr{M}$ and five ordered sub domains, considered relevant for an Assisting Conversational Agent. Hence the model is a 5-tuple $\mathscr{M} =< \mathscr{A}, \mathscr{U}, \mathscr{R}, \mathscr{S}, \mathscr{T} >$ where:

1. *The agent ($\mathscr{A}$)* contains the available information about the agent as a conversational character: like its name, age, gender, but also behavioral traits, moods *etc.*
2. *The user ($\mathscr{U}$)* contains the available information available about the user: like its name, its preferences *etc.*
3. *The request ($\mathscr{R}$)* contains the information about the current user's request: its textual and formal representation, but also the information involved in its current state of resolution;
4. *The session ($\mathscr{S}$)* contains the information about the previous requests since the beginning of the dialogical session;
5. *The topic ($\mathscr{T}$)* contains the available information specific to the assisted application.

Note that the term 'available' used above, emphasizes the fact that the model does not contain exhaustive information about the related entities, but only the information that was provided by the application's designers plus the information that the agent was able to gather during the dialogical session. Consequently, the agent has to reason with incomplete knowledge on the world (we cannot rely on the CWA assumption [30]).

Dynamics of the model: Basically, the model $\mathscr{M}$ is an evolving tree structure (as in Evolving algebra [31]). Given a new session, the model $\mathscr{M}_0$ starts at t_0:
$\mathscr{M}_0 =< \mathscr{A}_0, \varnothing, \varnothing, \varnothing, \mathscr{T}_0 >$, where:
– $\mathscr{A}_0$ is the generic submodel of the agent: it is defined by the designers of the help system and does not depend on the assisted applications.
– $\mathscr{T}_0$ is the specific submodel of the assisted application: for each application, the programmer must create a model that is filled with the information (relevant to the Function of Assistance) about the static structure and the dynamic functioning of the application.

Then, the tree structure of the model evolves according to two kinds of events:
1) Agent updates of information in $\mathscr{A}$, $\mathscr{U}$, $\mathscr{R}$, $\mathscr{S}$, according to interactions with users;
2) Application updates of the variables in $\mathscr{T}$ changed by the application runtime.
Hence, the structure of the model is shared between the application and the agent which are two separate processes. The question of their synchronization is a separate issue; see [32] for more information. The synthesis of the model of assistance $\mathscr{M}_0$ at t_0 and its updating during the session, at $t_1, .., t_n$ defining the tree sequence $\mathscr{M}_1, .., \mathscr{M}_n$ is a specific problem (see [33]) which is out of the scope of this paper. Therefore, in the following we will refer to the model as $\mathscr{M}$, whereas it should be $\mathscr{M}_n$, the current tree after n events.

Table 1. Main access functions of MQL where `path` stands for a tree path expression $\mathscr{M}.s_1.s_2,..,s_n$ (s_i being node labeling symbols), `n` is the node referred to by `path` and `expr` is a terminal value or a subtree

Name	Role	Returned value
`GET[path]`	returns the subtree from n	`OK[expr]` | `FAIL[report]`
`SET[path, expr]`	replaces `n` by `expr`	`OK[expr]` | `FAIL[report]`
`MAP[path, func]`	replaces `n` by func(`n`)	`OK[func(n)]` | `FAIL[report]`
`ADD[path, expr]`	appends `expr` to `n`	`OK[expr]` | `FAIL[report]`
`DEL[path,` s_i`]`	deletes subtree of head symbol in n	`OK[expr]` | `FAIL[report]`
`VOID[]`	does nothing	always `OK[]`

Table 2. The four types of agent's mental states according to their dynamicity and to their arity

	Unary	Binary
Static	Trait Ψ_T	Role Ψ_R
Dynamic	Mood Ψ_m	Affect Ψ_a

Model Query Language (MQL): The model is accessed by the agent or by the application, while using the Model Query Language (MQL). The main access functions are described in Table 1. Additional functions like `APPEND`, `FIRST`, `REST`, *etc.* make it easier to deal with lists (enumerated nodes). Also, paths can be simplified when there is no ambiguity: $\mathscr{M}.\mathscr{A}$.name $\rightarrow \mathscr{A}$.name (is the name of the agent in the model). Moreover, because the tree is traversed in depth-first order, 'name' stands for the agent's name rather than the user's name.

Agent query objects: Within the agent, MQL queries are produced by the rational and by the behavioral heuristics. Actually, they are not sent directly by the heuristic scheduler to the model for direct execution. Instead, a given query Q_i is reified in a so-called *query object* that wraps the actual MQL instruction into a symbolic structure expressed in the same formalism as $\mathscr{M}$, providing extra attributes (execution status, result status, *etc.*). The heuristics are then enabled access in read/write mode to the queries produced by other heuristics, and can perform symbolic reasoning about them. This feature is at the core of the implementation of the Cognitive Biases in section 3.

2.3 A simple mind model for the agents

In order to implement the *R&B* architecture a mental model of the agent is required. The *R&B* architecture can support various models provided they can be implemented in the formalism of the model. Here we define a specific mind model that is both simple enough to support the examples presented in section 3, and covering most significant notions discussed in the literature about mental states modeling [34]. Hence, some simplifications have been done here, *e.g.* we consider that traits ans roles are static (whereas

some works in the domain of organizations do not consider authority as a static relationship). We distinguish four types of mental states according to their dynamicity and their arity, as summarized in Table 2. Each of them is associated to a decimal value in $[-1,1]$, where $[0,1]$ denotes the intensity of the concept, $[-1,0]$ is the intensity of the antonym of the concept and 0 the "neutral" position (neither the concept nor its antonym stand).

– *Traits* (Ψ_T) correspond to typical personality attributes that can be considered as stable during the agent's lifetime, implemented using the "Five Factors Model" personality traits commonly used in psychology [35]:

- *Openness*: the appreciation for adventure, imagination and curiosity.
- *Conscientiousness*: the tendency to self-discipline to aim at achieving goals.
- *Extraversion*: the energy, strength of positive emotions and tendency to seek company of others.
- *Agreeableness*: the propensity to be compassionate and cooperative.
- *Neuroticism*: the tendency to easily feel negative emotions: anger, vulnerability, *etc.*

– *Moods* (Ψ_m) are agent's factors varying with time thanks to heuristics and biases, according to previous state of the agent and to the current state of the world. We define:

- *Energy*: the agent's physical strength.
- *Happiness*: the agent's physical contentment regarding its current situation.
- *Confidence*: the agent's cognitive strength.
- *Satisfaction*: the agent's cognitive contentment regarding its current situation.

Since physical properties consider the agent as an entity embodied into the world (like physical attributes of videogames characters), they appear less relevant in the case of an ACA and won't be considered in this article.

– *Roles* (Ψ_R) represent a static relationship between the agent and another entity of the world (typically a user it is assisting). We define two main categories of roles:

- *Authority*: the right the agent feels to be directive and reciprocally to not accept directive behaviors from another one. This role is often antisymmetric such as: `Authority(X,Y) = -Authority(Y,X)` where '-' denotes the antonym relation.
- *Familiarity*: the right the agent feels to use informal behaviors towards another one. This role is often symmetric such as: `Familiarity(X,Y) = Familiarity(Y,X)`

– *Affects* (Ψ_a) here denote dynamic relationships between the agent and another entity (typically the user). We distinguish at least three kinds of affects:

- *Dominance*: the agent feels powerful relatively to another one. It is often antisymmetric such as: `Dominance(X,Y) = -Dominance(Y,X)`
- *Cooperation*: the agent tends to be nice, caring and helpful with another one. It is not necessarily symmetric.
- *Trust*: the agent feels it can rely on another one. It is not necessarily symmetric.

Implementation of the agent's mind in the model $\mathscr{M}$: The proposed mind model is denoted in short R_{af}-T_{ocean}-M_{hesc}-A_{dct} where we just use the capitals of the nodes of the following tree which represents our simple mind model:

```
M.A.mind[
    role[                 // STATIC, SOCIAL RELATIONSHIPS WITH THE USER
        authority[-1..1],
        familiarity[-1..1]
        ],
    trait[                // STATIC, RELATIVE TO ANYTHING
        openness[-1..1],
        conscientiousness[-1..1],
        extraversion[-1..1],
        Agreeableness [-1..1],
        neuroticism[-1..1],
        ],
    mood[                 // DYNAMIC, RELATIVE TO SELF
        happiness[-1..1]
        energy[-1..1],
        satisfaction[-1..1],
        confidence[-1..1],
        ],
    affect[               // DYNAMIC, RELATIVE TO THE USER
        dominance[-1..1],
        cooperation[-1..1],
        trust[-1..1]
        ]
    ]
```

An attribute can be accessed by its full path like '$\mathscr{M}.\mathscr{A}$.mind.mood.happiness.value' or in the R_{af}-T_{ocean}-M_{hesc}-A_{dct} shortened notation as M_h, for the example of the happiness.

Mind intervals: In the formal model described above, values of the attributes v are decimal numbers $\in [-1,1]$, but it is often more convenient to consider a five positions symbolic scale based on a partition of the domain $[-1,1]$ into five contiguous intervals:

$v \in [-1,-0.8]$	<	strongly antonymic
$v \in [-0.8,-0.2]$	-	antonymic
$v \in [-0.2,0.2]$	=	neutral
$v \in [0.2,0.8]$	+	positive
$v \in [0.8,1]$	>	strongly positive

To keep the model simple, we have chosen discrete intervals but Fuzzy Logic could provide a more appropriate segmentation. Values are noted, for example, $M_h^+ \wedge A_c^<$ to mean the agent is happy and completely antagonistic; several intervals can be unioned by juxtaposing the signs after the attribute: *e.g.* $M_h^{+>}$ means 'happy or strongly happy'.

Mind Events Moreover, we are also interested in the *transitions* of a dynamic mind attributes (moods and affects) from an interval to another. Each time a transition is crossed downwards (resp. upwards) an event '$\backslash$' (resp. '/') is generated, which is associated with the updating query. For example, $Q_i^+ \wedge \backslash M_h^-$ means that the query Q_i was successfully carried out and that the happiness of the agent has been lowered from $M_h^{=+>}$ to M_h^-. Because events are associated with their causing queries, they are deleted when the query object is deleted. Here, it prevents an agent from looping on saying M_h^- ("I am sad") again and again, but only once on $\backslash M_h^-$ ("I [suddenly] feel sad").

2.4 Heuristic Description Language

A heuristic defines a rational or a behavioral reaction to a class of formal requests expressed in FRL. In a way; one can say that heuristics hard-code the reaction of the agent. The class of FRL expressions handled by a heuristic is defined by a pattern matching expression (FRL-pattern) over the FRL language. The general form of a heuristic is given by an expression of the form:

$$\text{H : identifier[FRL-pattern]} :\rightarrow \{\text{GuardedScript}_1, \ldots, \text{GuardedScript}_n\}$$

Where:

GuardedScript	$\equiv$	$\{\text{Guard}_1 \rightarrow \text{Script}_1, \ldots, \text{Guard}_n \rightarrow \text{Script}_n\}$
Guard_i	$\equiv$	logical expression $\mid \emptyset$ $\quad (\emptyset = \text{True})$
Script_i	$\equiv$	instruction $\mid \{\text{Instruction}_1, \ldots, \text{Instruction}_n\}$
Instruction_i	$\equiv$	basic operation $\mid$ MQL query object call $\mid$ GuardedScript
MQL query object call	$\equiv$	Q[Query identifier, MQL expression]

Since instructions can recursively be guarded scripts, it is possible to define embedded conditional structures allowing to script decision trees, often used in heuristics engine.

Heuristic Scheduling: When a FRL request is sent to the agent, the Heuristic Scheduler (HS) triggers the rational and behavioral heuristics that match the request and it ensures the coroutining of their execution. Coroutining is carried out thanks to guarded scripts: when HS enters a guarded script construct, the current heuristic is suspended and it is reactivated as soon as one of the guards is satisfied. When the script_i associated with the guard_i is executed the couple $\text{guard}_i \rightarrow \text{script}_i$ is deleted, so it is only executed once (actually, there is no need to handle iteration at the HS level; iteration constructs can be used in basic instructions but they are not coroutined). When several guards are simultaneously actives two strategies are available to the experimenters: a) deterministic: only the script of the first defined one is executed and all guarded scripts are deleted; b) stochastic: a script is randomly chosen and executed, and all guarded scripts are deleted. On the contrary, if no guard is triggered the agent reasoning process stops; two mechanisms can avoid such livelocks: a) always true guards with default actions within the heuristics; or b) default heuristics that handle what is indeed an agent failure.

Example 1: a simple rational reaction Assume that the user puts the question "What is your age?" resulting in the FRL request ASK_u[agent.age]. A possible rational heuristic that can handle questions about the agent's attributes is:

```
1:   H_r : ask-agent-attribute[ASK_u[agent.x_]]:→ {
2:            → Q[i, GET[x_]],
3:        Q_i^+ → Q[j, SET[R.reply, TELL_a[agent.x_, Q_i.value]]],
4:        Q_i^- → Q[j, SET[R.reply, TELL_a[Q_i.failure]]]
5:   }
```

Explanations:

1: x_ is a pattern variable that matches any symbol like age, gender, name, ...
(if needed, x_:list restricts the pattern to a list of authorized symbols).

2: empty guard prompts the script to be executed immediately with x_ being 'age'
Then age will be a shortcut for $\mathscr{M}$ full path M.A.age.value (subtree value[v] $\Rightarrow$ v)
i is a unique query identifier that is used later to refer to the query object as Q_i

3: Q_i^+ is a shortcut for Q_i.status == done $\wedge$ Q_i.head == OK
TELL$_a$[agent.x_, Q_i.value] is a FRL answer to the user
4: Q_i^- is a shortcut for Q_i.status == done $\wedge$ Q_i.head == FAIL
Q_i.failure reports the cause of the failure (*e.g.* NOTFOUND[path, concept])

Example 2: a simple Behavioral reaction Assume that the user types "I dislike you!" resulting in the FRL request DISLIKE$_u$[agent]. A possible behavioral heuristic is:

```
1:   H_b : dislike-agent[DISLIKE_u[agent]]:→ {
2:                                  → { Q[i, MAP[energy, λ x.x∗0.9]],
3:                                      Q[j, MAP[confidence, λ x.x∗0.9]],
4:                                      Q[k, MAP[cooperation, λ x.x∗0.9]],
5:                                  }
6:      Q_i^+ ∧ Q_i.value < −0.5 →  ADD[R.reply, TELL_a[energy, "tired"]]
7:      Q_j^+ ∧ Q_j.value < −0.5 →  ADD[R.reply, TELL_a[confidence, "depressed"]]
8:   }
```

Explanations:
2: The agent executes a sequence of three queries to modify its mind state
energy stands for the path M.A.mind.mood.energy.value in the agent's mind
$\lambda x.x*0.9$ is a λ-expression decrementing its argument x by 10%.
6: if its energy is very low, the agent appends "I feel tired" to the request reply list.
7: Same with the confidence; coroutining makes it possible to have both at once:
"I feel tired and depressed".

3 A case study: the Cognitive Biases

3.1 Principle and functioning of the Cognitive Biases

In order to study the relationships between rational reasoning and behavioral reasoning, it appeared helpful to envision a model where the problem is stretched to its extreme, without jumping to conclusions about its actual psychological soundness. We recall here that we are focused on developing software tools that can support various hypotheses about the *R&B* relationships. Consequently, we have proposed the so-called Cognitive Biases (CB) scenario which postulates that the rational engine and the behavioral engine 1) are constructed independently and 2) work independently.
– The Rational Engine $\mathscr{E}_r$ is a set of rational heuristics H$_{r_i}$ that apply on the user's requests: in order to resolve the current request, $\mathscr{E}_r$ produces MQL queries Q_i on the model $\mathscr{M}$ and reacts to the queries' reports.
– The Behavioral Engine $\mathscr{E}_b$ is a set of behavioral heuristics H$_{b_i}$ (called Cognitive Biases or simply Biases) that apply on the MQL queries Q_i produced by $\mathscr{E}_r$. Given a query Q_i, $\mathscr{E}_b$ can 1) alter the query and/or 2) alter the query report and/or react to the query Q_i by updating $\mathscr{M}\mathscr{A}$.A.mind.(moodlaffect) This process is called query filtering.

In the CB scenario, $\mathscr{E}_r$ has no knowledge about the existence and the effects of $\mathscr{E}_b$ on the queries. Moreover, $\mathscr{E}_r$ has read/write access to all parts of $\mathscr{M}$ except to $\mathscr{M}\mathscr{A}$.mind (for example, it is not possible to use M_h^+ in a $\mathscr{E}_r$ guard). Similarly, $\mathscr{E}_b$ has read/write access to $\mathscr{M}\mathscr{A}$.mind.(moodlaffect) and has only read access to the other parts of $\mathscr{M}$.

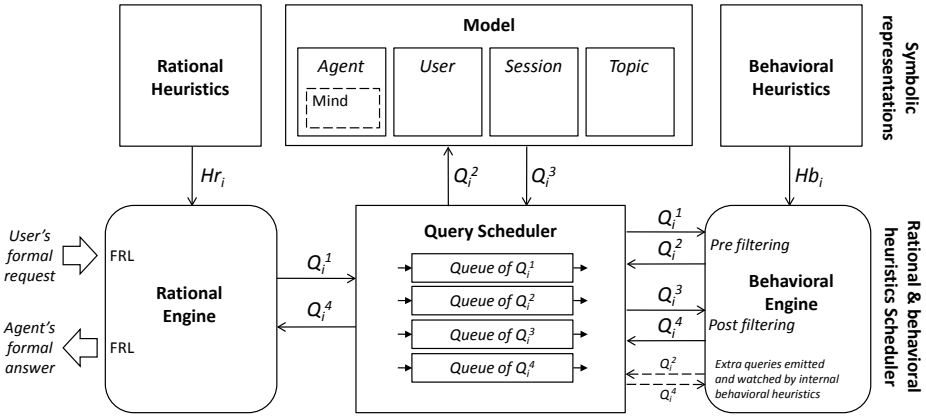

Fig. 3. General *R&B* Architecture with the specific query scheduler for the Cognitive Biases.

Consequently, $\mathscr{E}_b$ has no knowledge about the current rational process of request resolution which is stored in the rational heuristics. Moreover, $\mathscr{E}_b$ is only activated when the scheduler calls it to filter queries.

In order to support this functioning, the attribute '.status' of a query Q_i registers its four successive processing states:

– Q_i.status = 1 the query has been produced by $\mathscr{E}_r$
– Q_i.status = 2 the query is pre filtered by $\mathscr{E}_b$
– Q_i.status = 3 the filtered query is executed over the model and its result attributes are filled
– Q_i.status = 4 the reported query is post filtered, again by $\mathscr{E}_b$, and is now available for $\mathscr{E}_r$ guard expressions and scripts processing.

In the following, we will refer when necessary to $Q_i \wedge Q_i.status = 1$ as Q_i^1 and so on. The general *R&B* architecture and its specific query scheduler for the Cognitive Biases is described in Fig. 3.

3.2 Implementing behaviors through biases

A behavior H_b describes a reaction of an agent according to its current state of mind (*i.e.* the values in the leaves of $\mathscr{M}\mathscr{A}$.mind).

Example 3: If the agent has a high level of satisfaction and is not neurotic $M_s^{>} \wedge T_n^{<-=}$, it would tend to be in denial when facing negativity into user's utterances.
Assume the user tells that s/he dislikes the agent: "I hate you" $\Rightarrow$ DISLIKE$_u$[agent].
Then some rational heuristic will add the 'agent' to the 'dislikes' list of the user:
H_r: dislike-agent[DISLIKE$_u$[agent]] :$\rightarrow$ {$\ldots \rightarrow Q^1$[i, `ADD[`$\mathscr{U}$.dislikes, agent`]`], $\ldots$}
Then a possible bias upon the produced query in state 1 is:

```
1:   H_b : see-life-in-pink[Q^1[i_, ADD[(A |U).dislikes, y_]]]:→ {
2:      M_s^> ∧ T_n^<-= → Q^2[i, VOID[]]
3:   ...}
```

Explanations:
1: The agent refuses to consider a query adding a user's or an agent's dislike into $\mathscr{M}$.
2: So it replaces it by a `VOID[]` query that is produced with state 2.
Note that a positive result `OK[]` will be returned to H_r which is waiting on a condition of the form Q_i^{+4} and the rational process will go on without H_r being aware that a cognitive bias occurred.

Example 4: A neurotic unhappy agent $T_n^- \wedge M_h^-$ has a tendency to perceive extra negativity in everything the user is saying.
Assume the user tells that s/he likes an entity of the application: "I like the color of the title" $\Rightarrow$ LIKE_u[gui.title.color]. Then some rational heuristic will possibly add the entity to the user's 'likes' list (as in example 3) but also set the attribute 'wfu' (meaning worth-for-user) of this entity to the value 'high':
H_r: like-entity[LIKE_u[x_]] :$\rightarrow$ {... $\rightarrow$ Q^1[i, `SET[x_.wfu, 'high']`], ...}
However the agent can react negatively to that kind of query, and a possible bias expressing this is:

```
1:   H_b : see-life-in-black[Q^1[i_, SET[x_.wfu, 'high']]]:→ {
2:     M_s^> ∧ T_n^<-= → {
3:                       Q^2[i, SET[x_.wfu, 'high'],
4:                       ADD[R.reply, REQUEST_a[JUSTIFY_u[M R ]]]
5:                       Q^2[j, MAP[confidence, λ x.x*0.8]
6:                       ...},
7:   ...}
```

Explanations:
3: The query itself is not altered. It just goes from state 1 to 2.
4: A sour reply is added to the request $\mathscr{R}$.reply to ask the user to justify that opinion.
5: Moreover, because it is depressing to perceive high spirits from others when one is in a sad mood itself, the agent can decrease its happiness by 20%.

Now, it is possible that we had defined another bias filtering/reacting to queries in state 3 that are associated with a mind event that was possibly triggered by a mind update (like $\backslash M_c^-$):

```
1:   H_b : on-entering-unconfidence[Q^3[i_, MAP[confidence, f_]]]:→ {
2:     Q_i^+4 ∧ \M_c^- → ADD[R.reply, TELL_a[\M_c^-]]
3:   ...}
```

This, in turn can prompt an extra "I feel suddenly depressed" reply on the event that the agent enters downwards into the M_c^- interval, and so on. Note that extra queries produced by biases are created in state 2 so that they can also be biased (between states 3 and 4) by other biases like the one above, when they are sent back. Again, unaware of these biases, H_r will further add to $\mathscr{R}$.reply an `OK[]` report. Finally the expressing module will have to pragmatically sort and turn the list of produced replies into something like "Ok, but can you justify that!" or even "I take it, but give any reason why! ... I feel suddenly so depressed".

4 Conclusion

In the context of new assistance tools for the general public we have stated that the Function of Assistance must be supported both by rational reasoning over the task at hand but also by behavioral reasoning about the dialogical session between the users and the assisting agent. However, it is an open issue to figure out the nature and the relationships between the rational and the behavioral heuristics that support the reactions of the agents. This has prompted the design of a dedicated architecture for modeling both the rational and the behavioral heuristics in a single framework: the *R&B* agents). The *R&B* framework allows the implementation of flexible relationships between the rational and behavioral engines in order to experiment various scenarios. As a first step to the evaluation of the framework, we have implemented a simple but really trying case study, based on the Cognitive Biases scenario, where the two engines are independent but work in conjunction by sharing internal model queries. In further work, we intend to use the Web-based DIVA toolkit with the *R&B* framework in order to implement research experiments on various scenarios of agents with ordinary users on the Internet.

References

1. Cohill, A.M., Williges, R.C.: Retrieval of help information for novice users of interactive computer systems. Human Factors **27**(3) (1985) 335–343
2. Maes, P.: Agents that reduce work and information overload. Commun. ACM **37**(7) (1994) 30–40
3. Capobianco, A., Carbonell, N.: Contextual online help: elicitation of human experts' strategies. In: Proceedings of HCI'01, New Orleans (August 2001) 824–828
4. Carroll, J.M., Rosson, M.B.: Paradox of the active user. In: Interfacing thought: cognitive aspects of human-computer interaction. MIT Press (1987) 80–111
5. Lester, J.C., Converse, S.A., Kahler, S.E., Barlow, S.T., Stone, B.A., Bhogal, R.S.: The persona effect: affective impact of animated pedagogical agents. In: Proc. of the SIGCHI conference on Human factors in computing systems, Atlanta, Georgia, USA, ACM (March 1997) 359–366
6. Carbonell, N.: Towards the design of usable multimodal interaction languages. Universal Access in the Information Society **2**(2) (June 2003) 143–159
7. Cassell, J., Sullivan, J., Prevost, S., Churchill, E., eds.: Embodied Conversational Agents. MIT Press (April 2000)
8. Bates, J.: The role of emotion in believable agents. Commun. ACM **37**(7) (July 1994) 122–125
9. Martin, J.C., d'Alessandro, C., Jacquemin, C., Katz, B., Max, A., Pointal, L., Rilliard, A.: 3D audiovisual rendering and Real-Time interactive control of expressivity in a talking head. In: Proc. of IVA'2007. (2007) 29–36
10. Mori, M.: Bukimi no tani [The uncanny valley]. Energy **7**(4) (1970) 33–35
11. Xuetao, M., Bouchet, F., Sansonnet, J.P.: Impact of agent's answers variability on its believability and human-likeness and consequent chatbot improvements. In Michaelson, G., Aylett, R., eds.: Proc. of AISB 2009, Edinburgh, Scotland, SSAISB (April 2009) 31–36
12. Norling, E., Ritter, F.E.: Towards supporting psychologically plausible variability in Agent-Based human modelling. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems. (2004)
13. Evertsz, R., Ritter, F.E., Busetta, P., Pedrotti, M.: Realistic behaviour variation in a BDI-based cognitive architecture. In: Proc. of SimTecT'08, Melbourne, Australia (2008)

14. Rao, A.S., Georgeff, M.P.: Modelling rational agents within a BDI architecture. In Fikes, R., Sandewall, E., eds.: Proceedings of Knowledge Representation and Reasoning, San Mateo, CA, USA, Morgan Kaufmann (1991) 473–484
15. Pereira, D., Oliveira, E., Moreira, N.: Formal modelling of emotions in BDI agents. In Sadri, F., Satoh, K., eds.: Proceedings of CLIMA-VIII. Volume 5056 of LNAI., Porto, Portugal, Springer-Verlag (2008) 62–81
16. Casali, A., Godo, L., Sierra, C.: Graded BDI models for agent architectures. In: Proc. of CLIMA-V. Volume 3487 of Lecture Notes in Computer Science., Lisbon, Portugal (2004) 126–143
17. Dastani, M., van der Torre, L.: A classification of cognitive agents. In: Proceedings of Cogsci02. (2002) 256—261
18. Kopp, S., Wachsmuth, I.: Synthesizing multimodal utterances for conversational agents. Computer Animation and Virtual Worlds **15**(1) (2004) 39–52
19. Robbe-Reiter, S., Carbonell, N., Dauchy, P.: Expression constraints in multimodal human-computer interaction. In: Proceedings of the 5th international conference on Intelligent user interfaces, New Orleans, Louisiana, United States, ACM (2000) 225–228
20. Sansonnet, J.P.: DIVA - DOM integrated virtual agent. http:// www.limsi.fr/ jps/ online/diva/ divahome/index.html
21. Apparao, V., Byrne, S., Champion, M., Isaacs, S., Hors, A.L., Nicol, G., Robie, J., Sharpe, P., Smith, B., Sorensen, J., Sutor, R., Whitmer, R., Wilson, C.: DOM: document object model level 1 specification. W3C recommendation, W3C (October 1998)
22. Picard, R.W.: Affective computing. The MIT Press (2000)
23. Picard, R.W.: Affective computing: challenges. International Journal of Human-Computer Studies **59**(1-2) (2003) 55–64
24. Damasio, A., Sutherland, S.: Descartes'error: Emotion, reason, and the human brain. Papermac London (1996)
25. Ortony, D.A., Clore, G.L., Collins, A.: The Cognitive Structure of Emotions. Cambridge university press edn. (1988)
26. Bartneck, C.: Integrating the occ model of emotions in embodied characters. In: Proc. of the Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges, Melbourne (2002)
27. Gebhard, P.: Alma: a layered model of affect. In: Proc. of the fourth international joint conference on AAMAS, ACM New York, USA (2005) 29–36
28. Sabouret, N., Sansonnet, J.P.: Learning Collective Behavior from Local Interaction. LNAI-LNCS **2296** (2002) 273–282
29. Bouchet, F., Sansonnet, J.P.: Tree-kernel and feature vector methods for formal semantic requests classification. In Perner, P., ed.: Machine Learning and Data Mining in Pattern Recognition, Leipzig, Germany, IBaI Publishing (July 2009) 126–140
30. Reiter, R.: An experimentation with novice users. H. Gallaire and J. Minker, editors, Logic and Data Bases (1978) 119–40
31. Gurevich, Y.: Evolving algebras 1993: Lipari guide. Specification and validation methods (1995) 9–36
32. Sansonnet, J.P.: Composants dialogiques generiques. Perspectives et methodes pour une approche integree. Revue RSTI l'objet **102**(3) (2004) 189–202
33. Leray, D., Sansonnet, J.P.: Ordinary user oriented model construction for assisting conversational agents. In: CHAA'06 at IEEE-WIC-ACM Conf. on Intelligent Agent Technology. (2006)
34. Sabater, J., Sierra, C.: Review on Computational Trust and Reputation Models. Artificial Intelligence Review **24**(1) (2003) 33–60
35. Goldberg, L.R.: Language and individual differences: The search for universal in personality lexicons. Review of personality and social psychology **2** (1981) 141–165