Goal-Based Personalities and Social Behaviors in Believable Agents * †

Paola Rizzo [a] [c] ‡, Manuela M. Veloso [b], Maria Miceli [c], and Amedeo Cesta [c]

[a] Center of Cognitive Science, University of Turin
Via Lagrange 3, I-10123, Turin, Italy
paola@di.unito.it

[b] Department of Computer Science, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
mmv@cs.cmu.edu

[c] IP-CNR, National Research Council of Italy
Viale Marx 15, I-00137 Rome, Italy
{paola, maria, amedeo}@pscs2.irmkant.rm.cnr.it

Abstract

Agents are considered "believable" when viewed by an audience as endowed with behaviors, attitudes, and emotions, typical of different personalities. Our work is aimed at realizing believable agents that perform helping behaviors influenced by their personalities; the latter are represented as different clusters of prioritized goals and preferences over plans for achieving goals. The article describes how such model of personality is implemented in planning with the PRODIGY system and in execution with the RAP system. Both systems are integrated in a plan-based architecture where behaviors characteristic of different "helping personality types" are automatically designed and executed in a virtual world. The article also shows examples of the kinds of plan produced by PRODIGY for different personalities and contexts, and how such plans are executed by RAP when a helping character interacts with a user in a virtual world.

Running head: *Goal-based personalities*

* Corresponding author: Amedeo Cesta, IP-CNR, Viale Marx 15, I-00137 Rome, Italy, phone: +39-6-86090-209, fax: +39-6-824737, e-mail: amedeo@pscs2.irmkant.rm.cnr.it.

† This article is an extended version of a paper originally appeared in the Working Notes of the AAAI Fall Symposium on "Socially Intelligent Agents", AAAI Press Technical Report FS-97-02, American Association for Artificial Intelligence, pp. 109–114.

‡Part of this work was done while the author was visiting the Department of Computer Science at Carnegie Mellon University. The work has benefited from discussions with Leonardo Lesmo, Michael Mateas, Karen Haigh, and the PRODIGY research group members.

Contents

1 Introduction

"There is a notion in the Arts of 'believable character'. It does not mean an honest or reliable character, but one that provides the illusion of life, and thus permits the audience's suspension of disbelief [1]*. The idea of believability has long been studied and explored in literature, theater, film, radio, drama, and other media"* [Bates, 1994]. Believability therefore refers to a character's features producing the illusion of an "animated" being, able to think, feel emotions, and having attitudes toward others, etc., in a way very similar to a human being.

Recently several research projects and groups (the Oz project at CMU [Bates, 1994], the Agents group at MIT [Maes, 1995], the Virtual Theater project at Stanford [Hayes-Roth et al., 1995], etc.) have focused their efforts on the creation of computerized believable agents (BAs). The important novelty of computer-based BAs compared with those created with traditional media concerns their *interactivity*: in fact, while it is only possible to passively witness the actions performed by characters in movies or cartoons, with interactive BAs one can talk, play, fight, and so on. The frontiers of virtual reality make this possibility even more interesting. The applications of BAs concern mainly human-computer interaction, interactive entertainment, education, and some artistic domains.

The realization of computer-based interactive BAs presents problems that are different from those typical of traditional media. In fact, while movies and cartoons are designed and realized off-line, computer-based BAs must be able to interact with a human user in many kinds of situation; therefore, their behavior cannot be a priori defined, but it must be as autonomous and flexible as possible.

From the relevant literature it emerges that among the most important features of a BA are a marked personality and emotions; therefore, most works on BAs are focused on the physical and behavioral features more suited to express emotions, and study how to realize believable characters by varying the mapping between emotions and behaviors, or by varying the "traits" or features underlying personality.

We acknowledge the importance of emotional reactions and typical behaviors for making agents believable. However, we believe that representing personalities in such terms can be very expensive and, more importantly, might lack generality. While attention is focused on the details of BAs' manifest behaviors, their underlying general motivations risk to be neglected. This implies little attempt at defining personalities at a higher level. We try to make such an attempt, and look at general goals and preferences as the building blocks of personality types.

Believable agents are usually implemented by means of purely reactive architectures, where the behaviors are hand-written by the designer. On the contrary, we realize our agents by using a hybrid plan-based

[1] The concept of "suspension of disbelief" comes from the literature on theater; for clarifications, see [Laurel, 1991].

architecture, that includes a generative and a reactive planner. Our model of personality is explicitly represented in that architecture, and plans for the agents are automatically built by the planner from their personalities and then translated into sequences of actions that can be performed by the execution system.

Most of BA reseachers start from the strong claim that the personality of an agent is unique of that agent. Hence the need for a "computer artist" to hand-write the details of each agent's behavior. This work provides a different perspective to the construction of BAs. We aim at automatically producing personality-based behaviors starting from high level declarative goal descriptions. The first approach can be seen as a "bottom-up" development methodology while the one described here is mainly "top-down". Indeed, we would like to contribute to a methodological discussion on BAs design, by emphasizing the relevance of a goal-based analysis in developing agents endowed with personalities. We of course understand the need for uniqueness in BAs and have built a development tool that allows a "computer artist" to realize different characters automatically and subsequently refine them by customizing the declarative behavior descriptions.

The contributions of this article can be summarized as follows:

(a) to support our view of a goal-based approach to personality and present our model;

(b) to instantiate our model of personality by identifying the goals which influence the behavior of help-giving and to describe some "helping personality types";

(c) to explain how our model of personality is represented in the PRODIGY deliberative planner, which is enabled to produce personality-based plans, and in the RAP reactive planner; and how both systems are integrated in the overall architecture, by translating operators and plans from PRODIGY to RAP;

(d) to give an example of the kinds of interaction that can occur between a user and a helping character in a virtual world.

2 A goal-based model of personality

In this section we present our approach to modeling personalities. We first try to support our goal-based view; then we describe our model in terms of personality goals and preferences; finally we instantiate the model in the context of the help-giving behavior, and define a number of "help-giving personality types".

2.1 Why goal-based personalities?

There are several approaches at realizing believable agents with markedly different personalities.

Personality can be represented as a pattern of attribute-value pairs, in which each attribute is assigned a different (usually numerical) value for each agent. Such attributes refer to features like intelligence, amiability, dexterity, etc. [Goldberg, 1997], or resemble psychological traits, such as introversion/extroversion, shyness/assertiveness, and the like [Rousseau and Hayes-Roth, 1996]. Many actions are assigned typical values for the personality attributes. Each character is able to select and perform the actions that are most typical of its personality, by comparing the values of its own personality attributes with those assigned to the same attributes in the actions. For example, a shy character will more likely perform actions characterized by a high "shyness", while a "friendly" agent will select "friendly" actions.

When using the "attribute-based" approach to defining personality, BAs are usually assigned roles and must follow some sort of "script" set by the designer. Differences among their personalities can manifest themselves only in the way the scripts are performed, through the action-selection mechanism.

In order to maintain believability, it is extremely important that agents endowed with different personalities perform the same high-level behaviors in different ways, by choosing the actions that are most typical of them. As it will be explained in Section 2.2, our model of personality takes this issue into account by including personality-based preferences over actions. However, in our view the "attribute-based" approach does not cover some important aspects of an agent's personality. In particular, agents can manifest their personalities not only by performing the same actions in different ways, but also (and above all) *by pursuing different goals*. For instance, an altruistic and a selfish agent may be both extrovert, intelligent, and cheerful, but what makes their personalities different from each other is their typical goals. The altruist wants to be useful to others and tries to make them happy, while the selfish wants to exploit others for its own benefit. By using only an attribute-based approach we would not be able to capture such differences between their personalities. Consequently, it is important (1) to add an explicit representation of typical goals to a model of personality for BAs, and (2) to let the agents autonomously pursue such goals, in addition to performing role- or script-based behaviors. We will address these two issues in sections 3.2.1 and 3.4 respectively.

Given the importance attributed to emotions for BAs, personalities are often constructed also by varying the mapping between goals, emotions and reactions. In other words, each emotion which is triggered by the current state of a goal (e.g., success, failure, threat, etc.) can be associated with a class of reactions typical of each personality: for example, fear can be associated with a class of flight reactions for realizing a fearful personality, or with aggressive behaviors in order to realize an aggressive personality [Bates, 1994, Bates et al., 1992, Elliott, 1992, Reilly, 1996] [2].

[2] As far as social behaviors are concerned, other characteristics have also been analyzed and used for representing personality, such as competence, quirks, relationships, and so on [Reilly, 1996].

In many emotion-based approaches, e.g., [Bates, 1994, Loyall, 1997], goals are in fact recognized to be the "engine" of behaviors and emotions themselves. Emotions and behaviors are signs of underlying desires and goals, i. e. they make such goals manifest. But, of course, without goals and desires, there would be neither any behavior nor any emotion. Positive or negative emotions occur when the agent's goals (are likely to) succeed or fail. However, what we call "emotion-based" approaches are characterized by the strong emphasis they put on emotions in order to better give the "illusion of life". So, if characters do not show emotional reactions "...if they don't care, then neither will we. The emotionless character is lifeless, as a machine" [Bates, 1994].

In our view, though emotional reactions are no doubt very important for making characters believable, they might prove somehow limiting or insufficient for modeling some characters. Consider for instance a "normative" agent, i. e. one eager to comply with rules. How to realize it? On the one hand, it might be difficult or arbitrary to assign it some "typical" emotion. However we would not say that, since it does not show particular emotional reactions to events, it "does not care". A normative agent is an agent that "cares" to conform to social rules and avoid transgressions and immoral or illegal conduct, and such goals will characterize its behavior, however "cool" and emotionless it might appear. On the other hand, if we suppose that some emotional reaction can in fact be associated with the normative agent's behavior, not necessarily (or always) such emotion will allow to distinguish the normative agent from other characters. Suppose the normative agent shows anger when it suffers damage. Such emotional reaction is insufficient per se to distinguish this character from others. What makes the normative agent a distinct personality is that anger will be expressed only when the damage is viewed as a "wrong", that is, it implies some infringed right.

Finally, leaving aside the issue of the role played by emotions in modeling and manifesting distinct characters, we would like to stress what we mean by a "goal-based" model of personality. A goal-based model is not only one where goals play some role, but a model where goals are used as the basic building blocks for defining personalities. Such a model implies a commitment to define personality at a high level, in terms of clusters of stable, high order goals.

We want to explore the feasibility of this approach. In fact, personality is mainly considered by psychologists to be a coherent pattern of kinds of behaviors and interactions with the environment [Page, 1983, Pervin, 1989]. This is best explained by assuming that the agent is pursuing some stable general goals with different priorities, which produce coherent patterns of behaviors across multiple contexts.

By grounding our model on psychological models of human personality, we would like to explore an approach to "believability" that, however radically simplified, takes into account how humans represent

and "model" each other's personality [3].

2.2 Our model

We propose a model of personality based on goals and goal-based preferences. As explained above, personality is mainly concerned with the motivations of behavior; therefore we first want to find, for each possible personality we are interested in, the set of main goals that constitute it and regulate the agent's behavior. We also take into account the preferences that agents with specific personalities can have for different plans for achieving their goals.

In the following, goals are considered in the first place. We start from an AI work [Carbonell, 1980] which has proposed to model human personality by using goal trees, and a psychological work [Ford, 1992] that has provided a taxonomy of basic human goals. Secondly, preferences are analyzed: we propose that each personality has goal-based preferences for the use of some plans over others. Finally, the relevant features of our model of personality are summarized and some of its current limits are outlined. In section 2.3 our model is instantiated into a set of personalities which influence a specific social behavior (help-giving).

2.2.1 Goals

Our model is inspired by [Carbonell, 1980]. As Carbonell notices, a personality could be defined as the set of actions characteristic and non-characteristic of it; but such actions are very many and can vary from one context to another. Therefore it would be impossible to list them exhaustively; furthermore, such a list would ignore the hierarchical relations among behaviors, and would not grasp their underlying motivations. In summary, a representation of personalities based on the set of typical behaviors would be too expensive and would lack generality. Since each action is based on underlying personal motivations, Carbonell proposes to use a more compact representation of personality traits under the form of a goal tree, where each goal would underlie general types of action. A "standard" tree is used for a prototypical person, where each goal is assigned a fixed priority with respect to others; each personality modifies such tree by changing the priorities assigned to one or more goals.

While we share the idea of using the underlying motivations for representing personalities, we do not need to represent the goals for each personality as a "deviation" from the standard goal tree of a prototypical agent. In fact, Carbonell's work is related to the work by [Schank and Abelson, 1977], who have drawn the attention on the need for representing plans, goals, and "themes" for a better

[3] This kind of requirement is also expressed in [Dryer, 1997], where the use of psychologically-grounded models is placed in the context of agent-based help systems for traditional software.

comprehension of human behavior in story understanding. While in story understanding a standard tree is essential for making default inferences when no information about the character's personality is available, in designing BAs, we do not need to mind about the relative importance of goals for a hypothetical standard agent. As a consequence, a different cluster of prioritized goals has been built for each of the personalities that have been chosen.

[Ford, 1992, Ford and Nichols, 1987] have developed a psychologically-based taxonomy, with no hierarchical arrangement, where each category represents classes of goals at an abstract and decontextualized level of analysis. For example, the goal of entertainment, i.e. the goal of experiencing excitement and avoiding boredom, subsumes the goals pursued by behaviors like going to parties, seeking risk, avoiding daily routines, etc. From that taxonomy we have chosen the following goals relevant for our purposes [4]:

Resource provision. Giving approval, support, assistance, advice, or validation to others. Avoiding selfish or uncaring behavior.

Resource acquisition. Obtaining both tangible goods and support, assistance, and advice from others. Avoiding the loss of material possessions or social support.

Social responsibility. Keeping interpersonal commitments, meeting social role obligations, and conforming to social and moral rules. Avoiding social transgressions and unethical or illegal conduct.

Belongingness. Building or maintaining attachments, friendships, intimacy, or a sense of community. Avoiding feelings of social isolation or separateness.

Image. Receiving positive evaluations from others, both in the realm of competence –being evaluated as skillful, intelligent, resourceful, etc.– and in the moral sphere –being evaluated as honest, generous, considerate, etc. Avoiding social disapproval.

Entertainment. Experiencing excitement or heightened arousal. Avoiding boredom or stressful inactivity.

Safety. Being unharmed, physically secure, and free from risk. Avoiding threatening, depriving, or harmful circumstances.

Understanding. Gaining knowledge or making sense out of something. Avoiding misconceptions, erroneous beliefs, or feelings of confusion.

[4] Each description is taken from [Ford, 1992] (Table 4.2, p. 88), except for the goal of Resource Acquisition (which implies both Resource Acquisition and Material Gain as defined in Ford's taxonomy) and the goal of Image (implied in another, wider category in Ford's taxonomy)

It can be empirically found that almost all of the listed goals have different priorities from one individual to another; such goal priorities are a central feature of human personalities [Emmons, 1989, Winell, 1987]. As already explained, from a BAs point of view, we need to design the clusters of prioritized goals for our agents.

2.2.2 Preferences

There are of course many different ways (i.e., actions and plans) for pursuing each goal: personalities manifest themselves not only by pursuing different goals, but also by assigning preferences to such actions and plans. It is important to model preferences because some goals may be common to several agents, but for maintaining believability it is useful that the differences among their behaviors always emerge.

Our agents have a set of actions at their disposal which differ from one another not only for the main goals they achieve, but also for their *side-effects*, i.e. changes in the world that affect goals that are not actively pursued when a given plan is executed. If the affected goals are important in the personality goal cluster, problems or opportunities may arise. For instance, the goal of "resource aquisition", when instantiated in "getting an interesting object from another agent", can be achieved in different ways (each of them is a plan involving the performance of several related actions): by stealing the object from the agent; by bargaining the object; by asking the agent for it. Each plan (when successful) produces the main effect of obtaining the desired object, but it differs from the others in terms of its side-effects: for instance the first plan has the side-effect of violating social rules, thereby thwarting a goal belonging to the "social responsibility" category. Therefore, the "steal" plan would more probably be chosen by a selfish agent rather than an honest one, because the selfish does not care about the goal of respecting social rules. As it will become clearer in Section 3.2, since in our model preferences for each personality concern the relationships between its high priority goals and the side-effects of actions and plans, preferences can be automatically computed on the basis of such relationships by a suitable planning algorithm.

2.2.3 Features and limits of the model

In summary, in our model personality can be basically defined as:

- a cluster of goals with different priorities,
- a set of goal-based preferences over actions and plans for achieving goals.

It is important to note that goals may be pursued both actively and in reaction to suitable environmental contexts. In the latter perspective any occasion may be an opportunity for pursuing personality goals. As Sloman explains, "character and personality include long-term attitudes", which "are expressed in

tendencies to make certain choices *when the opportunity arises*" [Sloman, 1987] (original italics, p. 229). We will see later on how opportunities for manifesting personality-based plans are taken into account in our architecture during both planning and execution.

Of course, many other aspects implied in the treatment of personality are not yet considered here. Among them are the criteria for goal activation and achievement, concerning the processes which regulate the pursuit of goals on the basis of evaluations about their relevance, importance and attainability [Ford, 1992]. Another missing feature concerns the so-called "goal-orientations" [Ford, 1992], such as "maintenance-change", or "approach-avoidance", which refer to general personality-related styles of goal pursuit. Anyway, we believe that the features we have outlined are already a good starting point for representing and realizing complex personalities of BAs.

2.3 Help-Giving and Personalities

It would be hard or even impossible to define goal clusters for generic personalities, suitable to any situation or environment. Therefore, in order to instantiate our goal-based model of personality, we need to choose a situation or behavior in which to test several specific personalities. As a "case study" for realizing our agents, we have chosen the help-giving behavior, for several reasons.

First, help-giving is a very interesting complex social behavior. It is at the basis of exchange and co-operation, and involves the issue of "goal adoption" (that is, how and why does an agent adopt someone else's goals) typical of multi-agent systems and of the works (either computational or not) on dialogue. Consider for instance the so-called "master-slave assumption" underlying human-computer natural language dialogues [Grosz and Sidner, 1990], and the "benevolence assumption" discussed in Distributed AI [Rosenschein, 1985]. Secondly, help-giving implies sophisticated abilities of reasoning about others which are interesting by themselves. For instance, a help-giver could wonder about the reasons for the recipient's need, and decide not to help when the recipient is held responsible for it. Finally, the study of help-giving and of the behavior symmetrical to it (help-seeking) in multi-agent environments has already been pursued in some works of ours [Cesta and Miceli, 1993, Miceli and Cesta, 1993, Rizzo et al., 1995, Rizzo et al., 1997], and in other authors' works [Castelfranchi et al., 1997] addressing their basic motivations and outlining some typical personalities, i.e. kinds of agents with different modalities of help-seeking and help-giving activated by different attitudes.

We suggest the following possible "help-giving personality types", that is personalities which influence the modalities of help-giving, as well as the conditions under which agents will be likely to help someone else:

Altruist. It sincerely cares about others, and is willing to help them, even at its own disadvantage.

Normative. It is willing to help others if it should do it according to some norm (by role, reciprocity, equity), or at least if its helping behavior does not imply breaking any norm.

Selfish. It cares only about itself; therefore it helps others only if it can profit from such behavior (for instance, by offering its help in exchange for some reward).

Spiteful. It enjoys making fun of others, by interfering with their plans, refusing to help, or playing tricks on them.

Suspicious. It is afraid of being exploited by others, so it tries to deeply inquire into the reasons for their requests, and usually refuses to help.

2.3.1 Goals influencing help-giving

Since believability is positively related to strong personalities, we have looked at the goals which can characterize strongly marked behaviors. The listed goals and their relative priorities within a given personality are described in the following, with a more important goal preceded by a "+" and a less important one by a "–"[5]:

- **Altruist**
  - \+ *Resource Provision.* The altruist wants to give others what they need.
  - \+ *Belongingness.* This goal is useful to characterize the considerate and friendly attitude which usually accompanies an altruist's behavior.
  - \+ *Image.* We assume the altruist likes to receive appreciation of its altruistic behavior; so, it welcomes others' positive evaluations, with special reference to the moral domain (being evaluated as generous, considerate of others, etc.)
- **Normative**
  - \+ *Social Responsibility.* The normative agent is very much concerned about social rules, and wants its own and others' behavior to comply with them.
  - \+ *Image.* The normative likes to receive appreciation of its normative behavior; so, it welcomes others' positive evaluations, with special reference to the moral domain. In addition, such a positive appraisal of a "normative image" can be viewed as a means for socially reinforcing norm compliance itself.

[5] At the moment, a goal is given as generically more (or less) important than others, without specifying to what extent.

- **Selfish**
  - \+ *Resource Acquisition.* The selfish wants primarily to acquire and maintain resources.
  - − *Resource Provision.* The selfish is not interested in helping others, unless it is a means for its own "resource acquisition".
  - − *Social Responsibility.* The selfish is ready to break rules in order to pursue the goal of resource acquisition.
  - \+ *Image.* The selfish is interested in receiving others' positive evaluations, as a means for obtaining their adoption of its goals, with a consequent increase of its power (for instance, by presenting an image of "competence", it can be viewed by others as a good partner for exchanges); for the same reason, the selfish might be interested in hiding its selfishness, to avoid negative evaluations in the moral domain (for instance, being viewed as a potential cheater).
- **Spiteful**
  - \+ *Entertainment.* The spiteful likes to play jokes at others; it finds amusing that others get into trouble.
  - − *Social Responsibility.* The spiteful does not care very much about social rules.
  - − *Resource Provision.* Since it takes pleasure in others' troubles, the spiteful is unlikely to help others.
  - − *Image.* The spiteful does not care about its social image and is likely to risk negative evaluations in view of making fun of others and interfering with their plans.
- **Suspicious**
  - \+ *Safety.* The suspicious agent is all the time worried about possible dangers to itself, especially about dangers that could come from others' intentions.
  - \+ *Understanding.* The suspicious agent aims at investigating about such possible dangers, especially when they seem to come from other agents.
  - − *Belongingness.* Being so worried about others' bad faith, the suspicious is not very interested in building friendships or other close relationships.

2.3.2 Examples of personality-driven communicative behaviors

To give a more concrete idea of the behaviors typical of the listed personalities, and explain how the listed goals can influence such behaviors, here are some examples of possible interactions between BAs

endowed with the given personalities, and a human agent (HA), supposedly the agents' sister, that asks them to help her wash the dishes in an every-day life scenario. So, the HA says "There are a lot of dishes to do...", while the various BAs reply to this request in different ways, according to their personalities. The examples described below are based on communicative actions, but many other non-communicative behaviors could be typical of such agents:

Altruist: (going toward the kitchen) "Don't worry! I will do all the dishes by myself". The Altruist is always eager to help, given the high priority assigned to the goal of resource provision.

Normative: (simply standing still) "Today it's your turn!". The Normative wants that the high important goal of social responsibility is respected by HA.

Selfish: (looking at HA's pocket) "I'm too busy, I cannot. Unless you give me some money...". The Selfish may accept to help if it is in exchange for some reward, in order to pursue its very important goal of resource acquisition.

Spiteful: (smiling) "I'll do them before mum arrives..." ...and it doesn't do them. The Spiteful knowingly deceives HA and gives no help, in order to make her get into trouble, and hence take pleasure in this.

Suspicious: (looking intently at HA) "Are you asking me to do the dishes? Do you think that I am at your disposal?" The Suspicious reasons about HA's intentions and requests according to its very important goal of understanding.

3 An architecture for realizing personality-driven behaviors

In this Section we describe our software architecture for realizing different personalities. In this architecture PRODIGY, a generative planner, and RAP, a reactive planner, are integrated and work to produce personality-driven behaviors. The different aspects of the integration are described and finally a trace is presented as an example of social behaviors performed by a BA in interaction with a user.

3.1 A software environment based on planning techniques

Given the relevance of goals in our model of personality, we have considered how to make use of planning for building a computational system to realize different personalities.

Our aim was to obtain agents able to show a good deal of variability in their behaviors, according to both the differences among their personalities and the situations occurring in a changing social

environment; more specifically our purpose was to realize agents that:

1. have general high level goals with different priorities, according to their personalities;
2. choose different ways to pursue (even the same) goals according to (a) their personality preferences over actions and plans and (b) a projection of the situations that may be true in the world at execution time;
3. perform personality-driven behaviors, according to their general goals, personality preferences, and contingencies of the external environment.

An integrated architecture with two planning modules, a generative planner, PRODIGY [Veloso et al., 1995], and a reactive planner [6], RAP [Firby, 1987] fits our purposes. The software architecture relies on PRODIGY's features for the first and the second objective and to RAP's for the third one.

In the current implementation, PRODIGY is used off-line to automatically generate in advance a library of precompiled plans, and RAP is used on-line to adaptively execute such precompiled plans according to the current perception of the external environment.

This Section shortly introduces the two systems and then discusses some issues implied by their integration.

3.1.1 The PRODIGY deliberative planner

A deliberative (or generative) planner is able to produce a sequence of actions that, when executed, should cause the transition from an initial state to a goal state in a given domain.

The domain knowledge is specified to a planner mainly [7] as a set of *operators*. Each operator corresponds to a generalized atomic planning action, described in terms of its effects and the necessary conditions for the application of the operator.

A *planning problem* is defined by: (a) a set of available objects; (b) an *initial state*, that is represented as a set of literals; (c) a *goal statement*, i.e., a logical formula that may contain typed variables, conjunctions, negations, disjunctions, and universal and existential quantifications.

When an operator is used in planning, i.e., it is instantiated, its variables are replaced with specific objects of corresponding types. The precondition expression and the effects of an instantiated operator are then described in terms of *literals*, that is predicates whose variables are instantiated with specific objects. In order to execute an instantiated operator in the internal state of the planner, its preconditions

[6] The term *reactive planning* broadly covers systems able to select and execute actions according to environmental contingencies.

[7] For the sake of simplicity, we will not take the type hierarchy of objects into account.

must be satisfied in the state. Its effects specify the set of literals that must be added and deleted to the state.

A solution to a planning problem is a sequence of instantiated operators that can be applied to the initial state, by transforming it into a state that satisfies the goal.

PRODIGY [Veloso et al., 1995] is a planner of the family described above with a number of distinctive features. The current PRODIGY 4.0 follows a means-ends analysis backward chaining search procedure. It reasons about multiple goals and multiple alternative operators relevant to achieving the goals. PRODIGY 4.0 combines state-space search corresponding to a simulation of plan execution and backward-chaining responsible for goal-directed reasoning.

The strategies used in PRODIGY for directing its choices in decision points are called *control knowledge*. These strategies include the use of control rules (usually domain-dependent) [Minton, 1988], complete problem solving episodes to be used by analogy [Veloso and Carbonell, 1993], and domain-independent heuristics [Stone et al., 1994]. Here we focus on explaining control rules.

A *control rule* is a production (*if-then*) rule that tells the system which choices should be made (or avoided) depending on the current state, unachieved preconditions of the operators, and other meta-level information based on previous choices or subgoaling links. All the control rules used by PRODIGY are distinguished into three groups: *select*, *reject*, and *prefer* rules. If several control rules are applicable in the current decision point, PRODIGY will use all of them. *Select* and *reject* rules are used to prune parts of the search space, while *prefer* rules determine in what order to explore the remaining parts.

3.1.2 The RAP reactive planner

A reactive planner [Lyons and Hendriks, 1992] is a planning system devoted to the execution of plans in a dynamic environment. Usually systems of this family (e.g., [Georgeff and Lansky, 1986, Firby, 1987, McDermott, 1990]) are not able to generate plans by themselves but rather rely on a complex plan specification language to hand-write plans. These languages have several constructs to differentiate the execution according to dynamic environmental conditions.

In RAP [Firby, 1989] the agent's goals are explicitly represented, and for pursuing each of them there are several precompiled plans, instantiated at execution time according to the perceived situation. Such situation-driven execution, among other things, allows the agent to profit from opportunities, to suspend pursuing a goal and give priority to another that has become more urgent, and to change plan in face of execution failures or modified environmental conditions.

The RAP system is composed of a library of Reactive Action Packages (or RAPs)[8], a task agenda, and

[8] RAP refers to the whole system, while RAP(s) refer to Reactive Action Package(s).

a memory. A Reactive Action Package is the basic execution unit. Each task (i.e., achieving a goal) is mapped on a particular RAP, which specifies the set of methods (i.e., unistantiated plans) for achieving it and their conditions of applicability. The methods may consist in primitive actions (i.e. they can be directly executed) or in subtasks to be executed by using other RAPs.

The initial tasks to be performed (which must be set by the designer) are put on the task agenda. The interpreter selects one task at a time from the agenda by considering task selection constraints (e.g. explicit ordering or temporal constraints) and heuristics (e.g. higher priorities or closer deadlines). If the selected task has not yet been achieved, the RAP associated with it is taken from the RAPs library, and an applicable method is chosen for execution. The choice of a method from those listed in the RAP is based on the contents of the memory, which is RAP's internal model of the world. If the method is a primitive action, it is executed; otherwise all of its subtasks are put on the agenda, and the main task is put on the agenda to be checked for success after all of its subtasks are finished.

3.1.3 The integration of PRODIGY and RAP

It is worth noting that in most architectures for building believable agents, such as Hap [Loyall, 1997], BB1 [Hayes-Roth, 1995], Improv [Goldberg, 1997], etc., the reactive aspects of planning have been emphasized. In these systems, the agents' behaviors are usually represented as sequences of actions precompiled by the designer. However, writing such plans by hand has some disadvantages: (a) it is a time-consuming process, especially because it is necessary to predict and avoid negative interactions among actions; (b) it is not possible to build personality-driven behaviors automatically from the definition of the personality of the agent, and in fact personality is usually embedded in the precompiled mapping between situations and actions; (c) finally, it is not possible to build new behaviors automatically on the fly when they might be needed during interactions among agents.

In our work we address issues (a) and (b) and pave the way to addressing (c) by the integration of PRODIGY and RAP in a hybrid planning architecture. Our model of personality is explicitly represented in PRODIGY, and plans for the agents are automatically built by it from their personalities and then translated into sequences of actions (RAPs) that can be executed by RAP.

More precisely, we have designed a software environment, shown in Figure 1, with the following features:

1. The designer defines a synthetic scenario composed by domain objects, planning operators, and agents with distinctive personalities, and then defines a set of problems, typical of such scenario, to be solved by PRODIGY.

2. PRODIGY solves each problem by producing possibly multiple plans according to potentially different

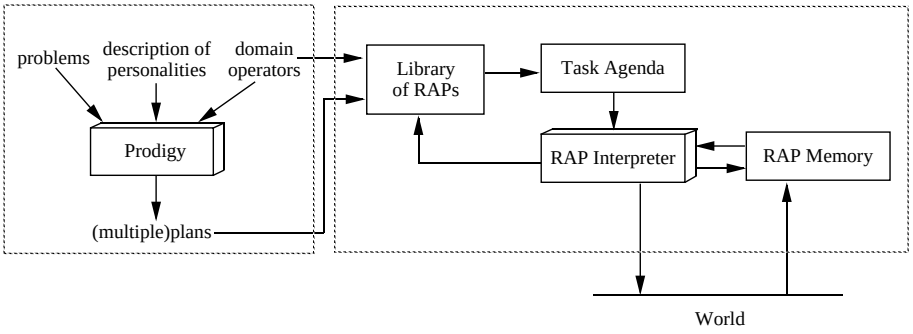

Figure 1: The Complete Software Architecture

contexts; different plans are generated for each agent according to its personality.

3. PRODIGY's operators and plans are translated into RAPs ready for execution; the designer is given the possibility of later inspecting the RAPs library to customize the generated knowledge if necessary.

4. Once the agent's knowledge has been created, the designer sets the RAP's task agenda with some initial goals and calls the RAP interpreter: in this way the synthetic agent starts to be executed, that is it starts to perform behaviors in the artificial world, following RAP plans in the attempt to make the task agenda empty.

5. An external user may interact with the artificial agent through a simple text-based interface; the user's behaviors are interpreted by the agent with a pattern-matching mechanism through the RAP's memory and this can cause the selection of new tasks and an adaptive execution of plans by RAP.

The following Sections illustrate in more detail the basic steps of our implementation: we first describe how plans are generated by PRODIGY (Section 3.2), then how PRODIGY's operators and plans are translated into RAPs (Section 3.3); finally we give details on the RAP execution and present a trace generated by our system in interaction with a user (Section 3.4).

3.2 Producing personality-based plans with PRODIGY

As explained in Section 2.2, our model of personality consists in a cluster of goals with different priorities, and in a set of preferences over actions and plans for achieving goals.

One might represent the set of personality preferences in PRODIGY as a direct mapping between each personality and the operators suitable for it, in either of the following ways:

(a) as in the "attribute-based" approach to personality (see Section 2.1), by "annotating" operators

with respect to personality variables, i.e. by stating in the preconditions of some operators which personalities are allowed to use such operators. For instance, we might think of two operators suitable for getting an object from an agent: `steal-obj-from-agent`, having among its preconditions *(personality <agent> selfish)*, and `bargain-obj-with-agent`, having among its preconditions *(personality <agent> normative)*;

(b) by using a number of control rules of the kind

```
(control-rule REJECT-OP-STEAL-FOR-NORMATIVE-AGT
    (IF (and (current-goal (has <agt> <obj>))
             (true-in-state
                  (personality <agt> normative)))
    (THEN (reject operator steal-obj-from-agent)))
```

that would select, reject, or prefer operators during problem solving according to the agent's personality.

On the basis of preconditions or control rules, PRODIGY would produce plans that differ according to the mapping directly set by the designer between personalities and operators. However, such an approach would present the following drawbacks:

1. It would be hard to make extensions or changes to the planning domain. On the one hand, when adding new operators, each of them should be analyzed with respect to the existing personalities in order to devise the right preconditions or control rules. On the other hand, when introducing new personalities, new preconditions should be added to many existing operators, or new control rules would be needed. Finally, if the designer decides to change the set of preferences over operators for a particular personality, analyzing all the relevant operators would be a very time-consuming process.

2. The reasons for the mapping between operators and personalities would not be explicit. For instance we would neither know why a selfish agent prefers to steal rather than to bargain, nor the reason why the opposite is true for the normative agent.

3. The relationship between preferences, side-effects of actions, and goals typical of each personality (explained in Section 2.2.2) would not be taken into account. For instance, by using preconditions or control rules of the kind above, we would not capture the fact that the operator `steal-obj-from-agent` has a side effect that negatively affects the goal of "social responsibility", which is why a normative agent would reject that operator.

We implement our model of personality in PRODIGY with a flexible approach, that avoids the problems mentioned above, and automatically computes the agents' preferences over operators and plans by considering side-effects of operators on high-level goals typical of different personalities. The implementation is based on the following features: a special kind of predicate (currently without arguments) called "general goals"; "side-effects" in some operators that add or delete general goals; and two control rules used during problem solving for preferring or rejecting operators according to the relationships between their side-effects and the general goals.

3.2.1 General goals

"General goals" or G-GOALS are a special kind of predicate that represent the high-priority goals for each personality. Such predicates are stored in a permanent data structure, that consists in an association table where each personality is mapped into a list of G-GOALS.

For example, let us suppose to have the following two personalities: altruist and selfish [9]. Referring to Section 2.3.1, the table of G-GOALS for such personalities is the following:

Altruist	*resource-provision, belongingness, image*
Selfish	*resource-acquisition, image*

Currently, the G-GOALS table contains only those predicates representing goals that are important (i.e. have a high priority) in a personality, and omits the unimportant ones [10].

It is worth observing that: (a) G-GOALS are not considered part of the goals to be achieved in a given problem. While the goal of a problem is a very specific objective to be attained (e.g., *(cleaned dishes)*), G-GOALS rather represent abstract states of the world (e.g., to respect rules) that are important for an agent with a given personality. So, there will never be a problem whose goal contains for instance *(belongingness)*; (b) G-GOALS are also left out of the description of the initial and goal state of the problems to be solved by PRODIGY for economy of representation.

3.2.2 Side-effects

Planning operators consist in preconditions and effects. The latter represent how the world changes after executing the operator, i.e. which predicates are added or deleted from the state of the problem. In backward chaining planners, operators are selected according to which of their effects produce the desired changes in the world.

[9] From now on, for the sake of brevity, we will consider examples with selfish vs. altruist solely.

[10] In future works we will address possible modifications to the G-GOALS table that make it more flexible.

In our approach to representing personality, we add "side-effects" to some operators, that differ from the main effects because they only refer to those changes in the world that affect some G-GOAL. Positive side-effects add G-GOALS to the state of the problem, while negative side-effects delete them. It is important to notice that additions and deletions do not affect the G-GOALS table itself, which is static.

Side-effects are not taken into account by PRODIGY for deciding which operator to use when solving a problem, because they add or delete from the problem state only G-GOALS that are not part of the goal of the problem.

As an example, let us consider the following two operators [11] that can both be used for getting an object from an agent (their side-effects are shown in italics):

```
(operator STEAL-OBJ-FROM-AGT
    (params <obj> <agt>)
    (preconds (has <agt> <obj>))
    (effects (del (has <agt> <obj>))
             (add (has myself <obj>))
             (del (social-responsibility))
             (del (belongingness))
             (del (resource-provision))))
```

```
(operator BARGAIN-OBJ-WITH-AGT
    (params <obj1> <obj2> <agt>)
    (preconds (and (has <agt> <obj2>)
                   (has myself <obj1>)))
    (effects (del (has <agt> <obj2>))
             (add (has <agt> <obj1>))
             (del (has myself <obj1>))
             (add (has myself <obj2>))
             (add (resource-provision))))
```

The operators `steal-obj-from-agt` and `bargain-obj-with-agt` have identical main effects. When solving a problem whose goal contains for instance *(has myself jewel)*, and the planning algorithm is not biased by any heuristic, either of the two operators can be chosen for solving the problem. However, the operator `steal-obj-from-agt` deletes three G-GOALS: *social-responsibility*, *belongingness*, and *resource-provision*; the operator `bargain-obj-with-agt` adds *resource-provision*.

Let us see how it is possible to bias the planning algorithm in order to choose different operators according to their side-effects, and therefore to produce plans that differ in terms of operators according to differences in personalities.

3.2.3 Control rules

For biasing the planning algorithm we have defined two control rules that can reject or prefer operators during problem solving according to the operators' side-effects on G-GOALS.

The first rule is used for "maintaining" G-GOALS, i.e., for avoiding that the latter are deleted by some operators that have negative side effects. The second rule is used for "opportunistically" achieving G-GOALS i.e., for preferring operators that add G-GOALS to the state of the problem. Both rules are domain independent; in fact, they do not refer to specific operators or states of of the world.

[11] We have omitted the object declarations for reasons of space.

Maintaining G-GOALS. Usually the goal to be achieved by a planner is an "attainment goal", i.e. a logical expression describing a single world state that will be obtained after the plan is executed; on the other hand, "maintenance goals" refer to general constraints on an agent's behavior over time.

G-GOALS are maintained in PRODIGY by using the following control rule:

```
(control-rule REJECT-BAD-OPERATOR
    (IF (and (candidate-operator <op>)
             (bad-effects-p <op>)))
    (THEN (reject operator <op>)))
```

Such rule, through the user-defined meta-predicate *bad-effects-p*, looks for possible negative side-effects in the candidate operator by matching the deleting effects listed in the operator with the predicates listed in the G-GOALS table; it is used for maintaining G-GOALS by rejecting operators that would delete them.[12]

Opportunistically achieving G-GOALS. We have seen in Section 3.2.1 that G-GOALS are not part of the goal to be achieved in a given problem; however, since G-GOALS are important for an agent, when solving a problem the planner will prefer to use operators that achieve them. It is important to notice that G-GOALS can be achieved again and again during problem solving, because they do not represent a single world state achieved after executing the plan (an attainment goal), but rather highly abstract states of the world that do not clearly specify their conditions of fulfillment. Think for instance of a goal like *(rich myself)*: no operator allows to achieve it once and for all, but it can be partially achieved every time a suitable operator is executed [13].

G-GOALS can be opportunistically achieved by using the following control rule, that is symmetrical to the first one:

```
(control-rule PREFER-USEFUL-OPERATOR
    (IF (and (candidate-operator <op>)
             (good-effects-p <op>)))
    (THEN (prefer operator <op> <anything>)))
```

Through the meta-predicate *good-effects-p*, the rule looks for possible positive side-effects in the candidate operators by matching the adding effects listed in the operator with the predicates listed in the G-GOALS table.

3.2.4 A planning trace

Here we give an example of how PRODIGY solves a simple problem in the case of an altruistic personality. We will see that, in addition to rejecting or preferring operators according to the preferences of the

[12] This rule causes Prodigy to fail when working on a problem that cannot be solved without violating a G-GOAL; we are currently working on making the rule more flexible.

[13] Currently we do not specify "how much" a G-GOAL can be achieved by the side-effect of an operator.

altruistic personality (whose G-GOALS are described in the table in Section 3.2.1), PRODIGY is able to produce multiple plans.

The planner runs separately for each agent. Here are the description of the problem (where the object declarations are omitted) and the running trace (that has been edited for reasons of space and clarity):

```
(setf (current-problem)
      (create-problem
        (name have-fun)
        (state
         (and (has other-agt toy)
              (has myself book)
              (willing-to-play other-agt)))
        (goal (have-fun myself))))
```

Each line in the trace corresponds to a node in the search tree. The line begins with the depth of the node in the search tree, followed by a label for the node, and a short description. Nodes within parentheses represent subgoals; nodes within braces are instantiated operators, and nodes within braces in capital letters (for instance node *n2*) represent applied operator nodes in the search tree.

```
USER(14): (run)

[. . . .]

2 n2 (done)
4 n4 <*finish*>
5 n5 (have-fun myself)

---> general goal (BELONGINGNESS) can also be reached
     operator #<OP: PLAY-WITH-AGT> is preferred

Firing pref rule PREFER-USEFUL-OPERATOR for (#<OP: PLAY-WITH-AGT>) over
#<OP: PLAY-WITH-OBJ>

7 n7 <play-with-agt other-agt>
7 n8 <PLAY-WITH-AGT OTHER-AGT>
Achieved top-level goals.

Solution #1: <play-with-agt other-agt>
```

[Here we ask for more solutions, and the problem solving process starts again]

```
7 n10 <play-with-obj toy>
8 n11 (has myself toy)

---> general goal (RESOURCE-PROVISION) threatened!
     operator #<OP: STEAL-OBJ-FROM-AGT> rejected

Firing reject operator REJECT-BAD-OPERATOR STEAL-OBJ-FROM-AGT

---> general goal (RESOURCE-PROVISION) can also be reached
     operator #<OP: BARGAIN-OBJ-WITH-AGT> is preferred
```

```
Firing pref rule PREFER-USEFUL-OPERATOR for (#<OP: BARGAIN-OBJ-WITH-AGT>)
over #<OP: GET>

10 n13 <bargain-obj-with-agt book toy other-agt>
11 n14 <BARGAIN-OBJ-WITH-AGT BOOK TOY OTHER-AGT>
11  n15 <PLAY-WITH-OBJ TOY>
Achieved top-level goals.

Solution #2:
<bargain-obj-with-agt book toy other-agt>
<play-with-obj toy>
```

[No more solutions are available.]

In sum, PRODIGY has produced two alternative solutions that differ in terms of preferences over operators according to the G-GOALS of the altruistic personality. The first solution has been produced by preferring the operator **play-with-agt**, because it allows to solve the problem in a single step, and at the same time to opportunistically achieve the G-GOAL of *(belongingness)*. The second solution has been produced by rejecting the operator **steal-obj-from-agt** in order to maintain the G-GOAL of *(resource-provision)*, and by preferring the operator **bargain-obj-with-agt** that opportunistically achieves the same G-GOAL. According to the way these two solutions are produced, they can be ranked in terms of preferences as the first one being preferred over the second.

The two solutions also differ with regard to the initial states to which they can be applied: by taking into account the preconditions of operators (that cannot be shown here for reasons of space) it is possible to see that solution number 1 needs for its application a subset of the initial state corresponding to **(willing-to-play other-agt)**, while solution number 2 needs the subset **(and (has myself book) (has other-agt toy))**.

Producing multiple solutions that can be mapped on different subsets of the initial state according to the "footprint principle" introduced in [Veloso, 1994] is used for having a good variability in plans that solve the same problem; later on we will see that such multiple plans will be translated in multiple methods within Reactive Action Packages, so that the RAP system is able to use different methods according to the different contexts that can occur in the virtual world.

3.3 Translation of PRODIGY operators and plans into RAPs

Our architecture is composed of a deliberative planner that off-line builds plans to be executed by the reactive planner. Since PRODIGY's operators and plans and Reactive Action Packages do not have the same syntax, it is necessary to automatically translate the former into the latter.

From PRODIGY operators to RAPs and memory rules. PRODIGY operators are translated into RAPs consisting in directly executable primitive actions; the effects of PRODIGY operators are translated into "memory rules", which change the contents of the RAP memory after the successful execution of a primitive RAP. Currently, in such translations we deal with operators containing neither conditional effects nor quantifications; later work will be devoted to such issues.

The translation from operators into RAPs is straightforward: an operator is defined through its name, parameters, preconditions and effects [14]; the name and parameters are translated into the RAP index, while the preconditions are mapped into the RAPs preconditions. Since the RAP's effects are not represented inside the RAP itself, the operator's effects are mapped into a so-called memory rule in the RAP system; such rule represents how the RAP affects the world after its successful execution.

From PRODIGY plans to RAPs. PRODIGY's (possibly multiple) plans are translated into complex RAPs; these are composed of partially ordered primitive RAPs previously translated from the planning operators. This kind of translation is described by referring to the problem named "have-fun" solved by PRODIGY in Section 3.2.4. PRODIGY had produced 2 solutions, the first one being preferred over the second. The Reactive Action Package shown here is composed by both plans: each of them maps into a different method. The first method has a higher preference, expressed as a lower integer, than the second method, thereby preserving the relative preferences between PRODIGY plans.

```
(define-RAP (index (have-fun))
            (succeed (entertained myself))
            (method 0
              (context (willing-to-play ?agt))
              (task-net
                   (1 (play-with-agt ?agt))))
            (method 1
              (context (and (has ?agt ?toy) (has myself ?obj)))
              (task-net
                   (1 (bargain-obj-with-agt ?obj ?toy ?agt) (for 2))
                   (2 (play-with-obj ?toy)))))
```

Notice that the ordering constraints of the solutions (taken from a contingency table not shown here) are represented also in the methods through the use of *for* clauses, that specify which task depends on the successful execution of another. The context for each method refers to a subset of the initial state of the problem that must be true for using that particular method. Finally, notice that RAPs represent a generalization of PRODIGY plans, by transforming literals into predicates with variables.

[14] Once again, we omit the type declarations.

3.4 Executing personality-based behaviors with RAP

In this section we will see how the RAP system uses the RAPs (translated from PRODIGY operators and plans) to pursue different goals in different ways according to the agents' personalities and to the changing situations of the environment.

Going back to the RAP architecture, as agents designers, we put in the RAP task agenda the tasks that we want our agents to perform. Some of these tasks will be related to the agents' personalities through their relation with the G-GOALS, while some other tasks will be "neutral". For instance a task like "satisfy-hunger" will be common to all agents, while a task like "help-beggar" will be very relevant to the altruistic personality because it is an instantiation of the G-GOAL *(resource-provision)*. In order to define task agendas that are peculiar for each agent's personality, it is possible to assign priorities to tasks, similarly to what is done in other BAs architectures (e.g., Hap [Loyall, 1997]). In fact, tasks on the agenda can be assigned a numerical value; the higher the value the more important the task. Agents with specific personalities can have different values attached to the same tasks, and this will produce differences in their behaviors.

As for how tasks are pursued, it has been shown in Section 3.3 that RAPs can represent multiple methods for achieving the same task; each method can be assigned a preference through a numerical value. The lower the value, the higher the preference over a particular method. In our implementation, preferences are assigned automatically when translating multiple plans produced by PRODIGY into multiple RAPs methods.

The variability among agents' behaviors due to their personalities is therefore produced in two ways: first, by the interpreter's activation of different tasks depending on their priorities (e.g. the "entertainment" task can be pursued by any agent at any time, but it will be pursued more often by the Spiteful agent); second, by selecting for each task-related RAP the methods for which there are personality preferences.

These concepts are going to be illustrated in an example of how RAP executes the behaviors of an agent that can interact with a human player in a text based virtual environment.

3.4.1 An execution trace

The architecture described in the paper has been used to realize several "help-giving characters"; from them, we have chosen to show an altruistic agent, Tony, interacting in alternate turns with a human player in an everyday scenario, through a very simple text-based interface [15]. Tony's agenda has been set

[15] Currently we do not process natural language.

by the designer, and contains the following tasks: "have-fun" (with priority 1), "satisfy-hunger" (with priority 3), and "do-dishes" (with priority 2). The trace has been edited for purposes of clarity: here and there the internal state of the RAP interpreter is shown with some explanations.

We suppose that the human player is an 11 year-old girl who lives with her younger brother, Tony, who is 8 year-old. The player has finished her homework, and now she would like to play a little, so she goes to the living room, looking for her favorite building blocks.

[the next instruction starts the RAP interpreter together with the user interface]

```
(go!)

*******PLAYER> (observe environment)
```

[Here the RAP interpreter starts executing the task "have-fun" and instantiates the method "watch-TV" for pursuing it]

```
==> Switching to family 1 (HAVE-FUN) :: #{Goal 1}
==> Considering goal (HAVE-FUN) :: #{Goal 1} :NEW
... Instantiating METHOD - METHOD-116
... Adding goal to agenda: T1 WATCH-TV :: #{Goal 5}

[...]

==> Considering goal (WATCH-TV CHANNEL) :: #{Goal 5} :NEW
... Instantiating METHOD - PRIMITIVE-117
--- Enabling skill: (WATCH-TV CHANNEL) - #{Goal 5}

*******Tony> (WATCH-TV CHANNEL)

*******PLAYER> (get building-blocks)

*******Tony> (WATCH-TV CHANNEL)

*******PLAYER> (play-with-obj building-blocks)
```

[The player's behavior is interpreted by the RAP memory, and consequently an alternative method is selected for pursuing the task "have-fun". Such method corresponds to a plan previously produced by PRODIGY *with a higher preference for the altruistic personality.]*

```
==> Considering goal (HAVE-FUN) :: #{Goal 1} :ACTIVE
... Instantiating METHOD - METHOD-114
... Adding goal to agenda: 1 PLAY-WITH-AGT :: #{Goal 6}

==> Considering goal (PLAY-WITH-AGT PLAYER) :: #{Goal 6} :NEW
... Instantiating METHOD - PRIMITIVE-120
--- Enabling skill: (PLAY-WITH-AGT PLAYER) - #{Goal 6}

*******Tony> (PLAY-WITH-AGT PLAYER)
```

[Now Tony is hungry, and the task "satisfy-hunger" takes priority over the task "have-fun" which is suspended.]

```
==> Switching to family 2 (SATISFY-HUNGER) :: #{Goal 2}
==> Considering goal (SATISFY-HUNGER) :: #{Goal 2} :NEW
... Instantiating METHOD - METHOD-111
... Adding goal to agenda: 1 GOTO-OBJ :: #{Goal 6}
... Adding goal to agenda: 2 GET :: #{Goal 5}
... Adding goal to agenda: 3 COOK :: #{Goal 7}
... Adding goal to agenda: 4 SHARE-AND-EAT :: #{Goal 8}
```

[Notice that our altruistic agent chooses the prosocial method of sharing its food with the player.]

```
*******Tony> (GOTO-OBJ POPCORN)
             (GET POPCORN)
             (COOK POPCORN)
             (SHARE-AND-EAT POPCORN PLAYER)
[...]

==> Goal achieved: (SATISFY-HUNGER) :: #{Goal 2} at time 16
... with result: :SUCCEED OKAY
... Removing goal: #{Goal 2}
==> Top level goal (SATISFY-HUNGER) :: #{Goal 2} finished with result
(:SUCCEED OKAY)

*******PLAYER> (eat popcorn)
```

[The task "have-fun" is resumed]

```
==> Switching to family 1 (HAVE-FUN) :: #{Goal 1}
==> Considering goal (HAVE-FUN) :: #{Goal 1} :ACTIVE
... Instantiating METHOD - METHOD-114
... Adding goal to agenda: 1 PLAY-WITH-AGT :: #{Goal 8}

[...]

==> Considering goal (PLAY-WITH-AGT PLAYER) :: #{Goal 8} :NEW
... Instantiating METHOD - PRIMITIVE-120
--- Enabling skill: (PLAY-WITH-AGT PLAYER) - #{Goal 8}

*******Tony> (PLAY-WITH-AGT PLAYER)
```

[Now the player sees that the dishes are dirty after having eaten the pop-corn, and she asks Tony to do the dishes.]

```
*******PLAYER> (request Tony do-dishes)
```

[The player's request is accomplished by activating a new task for washing dishes.]

```
==> Switching to family 4 (DO-DISHES) :: #{Goal 4}
==> Considering goal (DO-DISHES) :: #{Goal 4} :NEW
... Instantiating METHOD - METHOD-113
... Adding goal to agenda: 1 CLEAN-DISHES :: #{Goal 8}
... Adding goal to agenda: 2 STOW-DISHES :: #{Goal 7}

[...]

*******Tony> (CLEAN-DISHES)
             (STOW-DISHES)
```

```
[...]

==> Goal achieved: (DO-DISHES) :: #{Goal 4} at time 21
... with result: :SUCCEED OKAY
... Removing goal: #{Goal 4}
==> Top level goal (DO-DISHES) :: #{Goal 4} finished with result
(:SUCCEED OKAY)

*******PLAYER> (play-with-obj building-blocks)
```

[After washing dishes, the task "have-fun" is resumed again.]

```
==> Switching to family 1 (HAVE-FUN) :: #{Goal 1}
==> Considering goal (HAVE-FUN) :: #{Goal 1} :ACTIVE
... Instantiating METHOD - METHOD-114
... Adding goal to agenda: 1 PLAY-WITH-AGT :: #{Goal 7}

==> Considering goal (PLAY-WITH-AGT PLAYER) :: #{Goal 7} :NEW
... Instantiating METHOD - PRIMITIVE-120
--- Enabling skill: (PLAY-WITH-AGT PLAYER) - #{Goal 7}

*******Tony> (PLAY-WITH-AGT PLAYER)
```

In summary, this trace shows three main features in the agent's behavior:

1. spontaneous social interaction: Tony stops watching TV and starts playing with the user when the latter shows, by using the building blocks, that she is willing to play.

2. autonomous goal activation: this can be observed both at the beginning of the simulation, when Tony starts watching TV, and later on, when it cooks and eats the pop-corn for satisfying its hunger. In the second episode it is interesting to notice that Tony uses an "altruistic" method that consists in sharing food with the user.

3. response to an explicit request: when the player asks it, Tony suspends playing and washes the dishes, and then starts playing again with the user.

All the behaviors performed by the agent had been automatically produced in advance by PRODIGY and then translated into RAPs. It is interesting to notice that (1) the preference over RAP methods like `play-with-agt` and `share-and-eat` has been derived by the preferences over operators that had been taken into account by PRODIGY when solving the problems "have-fun" and "satisfy-hunger" respectively; (2) RAP's ability to switch from one task to another when the opportunities arise allows to show personality-driven prioritizations of different tasks.

4 Discussion

In this work we have tried to model and realize believable agents that perform social behaviors influenced by their own personalities. As a case study we have chosen help-giving, which we view as an important social behavior, at the basis of both cooperation and exchange. A couple of general features characterize our approach:

- a model of personality in terms of an agent's typical motivations, or clusters of general and stable goals, and its preferences over plans for achieving them;
- the automatic realization of different personalities and personality-driven behaviors by means of an architecture including a deliberative and a reactive planner.

An agent's personality can be identified on the grounds of its coherent patterns of behaviors and interactions with the environment. Such typical behaviors can in turn be explained in terms of a number of underlying stable goals with different priorities. However, different personalities may share some general goals; in such cases, differences can emerge from how those goals are pursued, that is, from the different plans each personality can prefer for achieving the same goal. In our view, this way to model personalities allows quite a wide range of different personalities to be developed, as well as to distinguish among personalities that may happen to share some features in given situations.

Our hybrid architecture integrates deliberative and reactive planning. The PRODIGY deliberative planner is able to generate a plan library according to the agent's personality. Each personality is implemented by means of relationships among G-GOALS (that is, general high level goals typical of that personality), anticipated side-effects of actions, and preferences over plans. The RAP reactive planner is able to produce actual behaviors according to such personality goals and preferences, as well as to specific contingencies (opportunities, occurred failures, etc.) of the given situation and environment. The main advantages of such an integration lie in the flexibility of representation of the different personalities, and in an automated generation of personality-driven behaviors. The latter, in fact, are produced as a consequence of the model of personality which is explicitly represented in PRODIGY and RAP rather than being compiled ad hoc by the designer.

As far as planning techniques are concerned, other work exists that is somehow similar to ours. RAP has been coupled with PRODIGY also in [Blythe and Reilly, 1993], and with another generative planner (AP) in [Bonasso et al., 1995]. Both works differ from ours mainly because of their rather "conventional" view of the integration of generative and reactive planning. In fact, the following differences emerge: (a) RAPs are not automatically created by translating the planner's operators and plans, but are written and mapped

into the planning operators by hand; (b) the generative planner interacts with RAP on-line rather than off-line, by passing its plans to RAP for immediate execution, and by monitoring the results. Wilkins and Myers [Wilkins and Myers, 1995] have addressed the problem of making a generative planner (SIPE-2) and a reactive planner (PRS-CL) speak a common language. Their solution differs from ours because they have created an interlingua (the ACT formalism) used for handwriting plans that can subsequently be understood and used by both systems. On the contrary, in our work there is a unidirectional translation from PRODIGY to RAP. Only the planning operators must be written by the designer, while plans are produced by PRODIGY and automatically transformed into RAPs.

In this paper, we have used a software architecture for the automatic generation of behaviors grounded on an explicit goal-based model of personality for BAs, and we have explained that such "top-down" approach has several advantages with regard to the "bottom-up" manual compilation of the agents' behaviors.

However, we believe that these methods of producing behaviors for BAs can be considered as the two extremes of a continuum, rather than opposed approaches, and they can be integrated in several ways. From this point of view, the plan-based architecture described in this paper should be considered as a development tool that enables the agent creator to use both methods. On the one hand, the designer can start from a high level, declarative representation of each agent's personality, and let the generative planner automatically produce several individual libraries of typical behaviors. On the other hand, the designer can (1) hand-write further behaviors from scratch, using the RAP description language; (2) customize the automatically produced behaviors in several ways (e.g., by arbitrarily modifying their triggering conditions, or by deciding which are the conditions for success or failure of the action); and (3) define how "primitive" actions can be differently realized at low level by the graphical and/or text-based user interface.

As for the social interactions of our different characters, for the time being they are restricted to dyadic interactions between a given character and the user. The trace we have presented in Section 3.4.1 offers an example of how social interaction can be biased by the personality of the artificial agent, in terms of both the priorities it gives to its possible tasks, and the preferences it shows for specific methods for realizing them. For instance, an altruistic character, being very likely to take into account the other's needs, when hungry and intending to cook something to eat, prefers to share its pop-corn with the user. In the same circumstance, a selfish character would behave quite differently: it would cook the pop-corn just for itself, and, whenever possible, it would even try not to comply with an explicit request by the user, unless it gets some particular reward from its cooperative behavior.

Further developments of our work concern the use of our architecture for realizing believable agents that interact with each other; they should perform a variety of personality-driven behaviors in reaction to others' personalities. For instance, it would be interesting to observe the kinds of interaction that might occur between a selfish and an altruistic agent, or between a normative and a suspicious agent. To this aim, we are currently working on a multi-agent version of RAP. In addition, we aim at realizing a tighter integration of the deliberative and the reactive planner, in which the former would inspect the task agenda, and build new plans on the fly during interactions among agents. This would allow to take into account the interactions between tasks occurring at execution time, and to model changes in the personalities of the BAs according to the feedback coming from social interaction. More importantly, an on-line integration of the two planners will put the overall goal-based model to the test, and will help understand how good such model is in producing robust and widely different behaviors.

Acknowlegments

Paola Rizzo is currently supported by a scholarship from CNR Committee 12 on Information Technology. Prof. Veloso's research is sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement number F30602-97-2-0250. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory (AFRL) or the U.S. Government. Maria Miceli and Amedeo Cesta have been partially supported by CNR Committee 12 on Information Technology (Projects SARI and SCI*SIA), and by IP-CNR — Division of Artificial Intelligence, Cognitive Modeling and Interaction.

References

[Bates, 1994] Bates, J. (1994). The Role of Emotion in Believable Agents. *Communications of the ACM*, 37(7):122–125.

[Bates et al., 1992] Bates, J., Loyall, A. B., and Reilly, W. S. (1992). Integrating Reactivity, Goals, and Emotion in a Broad Agent. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society.*

[Blythe and Reilly, 1993] Blythe, J. and Reilly, W. (1993). Integrating Reactive and Deliberative Planning for Agents. Technical Report CMU-CS-93-155, Carnegie Mellon University, School of Computer Science.

[Bonasso et al., 1995] Bonasso, R., Kortenkamp, D., Miller, D., and Slack, M. (1995). Experiences with an Architecture for Intelligent, Reactive Agents. In *Proceedings of ATAL-95*.

[Carbonell, 1980] Carbonell, J. (1980). Towards a Process Model of Human Personality Traits. *Artificial Intelligence*, 15:49–74.

[Castelfranchi et al., 1997] Castelfranchi, C., de Rosis, F., and Falcone, R. (1997). Social Attitudes and Personalities in Agents. In *Working Notes of the AAAI Fall Symposium on "Socially Intelligent Agents"*. AAAI Technical Report FS-97-02.

[Cesta and Miceli, 1993] Cesta, A. and Miceli, M. (1993). In Search of Help: Strategic Social Knowledge and Plans. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, Hidden Valley (PA).

[Dryer, 1997] Dryer, D. (1997). Ghosts in the Machine: Personalities for Socially Adroit Software Agents. In *Working Notes of the AAAI Fall Symposium on "Socially Intelligent Agents"*. AAAI Technical Report FS-97-02.

[Elliott, 1992] Elliott, C. (1992). *The Affective Reasoner: A Process Model of Emotions in a Multi-Agent System*. PhD thesis, Northwestern University. Technical Report no. 32.

[Emmons, 1989] Emmons, R. A. (1989). The Personal Striving Approach to Personality. In Pervin, L. A., editor, *Goal Concepts in Personality and Social Psychology*. Lawrence Erlbaum Associates, Hillsdale (NJ).

[Firby, 1987] Firby, R. J. (1987). An Investigation into Reactive Planning in Complex Domains. In *Proceedings of the 6th National Conference on Artificial Intelligence*, pages 202–206, Seattle (WA).

[Firby, 1989] Firby, R. J. (1989). *Adaptive Execution in Complex Dynamic Domains*. PhD thesis, Yale University. Technical Report YALEU/CSD/RR #672.

[Ford, 1992] Ford, M. E. (1992). *Motivating Humans. Goals, Emotions, and Personal Agency Beliefs*. Sage, Newbury Park (CA).

[Ford and Nichols, 1987] Ford, M. E. and Nichols, C. W. (1987). A Taxonomy of Human Goals and Some Possible Application. In *Humans as Self-Constructing Living Systems: Putting the Framework to Work*. Lawrence Erlbaum Associates, Hillsdale (NJ).

[Georgeff and Lansky, 1986] Georgeff, M. and Lansky, A. L. (1986). Procedural Knowledge. *Proceedings of IEEE*, 74(10):1383–1398.

[Goldberg, 1997] Goldberg, A. (1997). Improv: A System for Real-Time Animation of Behavior-Based Interactive Synthetic Actors. In Trappl, R. and Petta, P., editors, *Creating Personalities for Synthetic Actors*. Springer, Berlin.

[Grosz and Sidner, 1990] Grosz, B. J. and Sidner, C. L. (1990). Plans for Discourse. In Cohen, P. R., Morgan, J., and Pollack, M. E., editors, *Intentions in Communication*. MIT Press, Cambridge (Mass.).

[Hayes-Roth, 1995] Hayes-Roth, B. (1995). An Architecture for Adaptive Intelligent Systems. *Artificial Intelligence*, 72:329–365.

[Hayes-Roth et al., 1995] Hayes-Roth, B., Brownston, L., and vanGent, R. (1995). Multi-Agent Collaboration in Directed Improvisation. In *First International Conference on Multi-Agent Systems*, Cambridge (Mass.). MIT Press.

[Laurel, 1991] Laurel, B. (1991). *Computers as Theatre*. Addison-Wesley, Reading (Mass.).

[Loyall, 1997] Loyall, B. (1997). *Believable Agents: Building Interactive Personalities*. PhD thesis, School of Computer Science, Carnegie Mellon University. Technical Report CMU-CS-97-123.

[Lyons and Hendriks, 1992] Lyons, D. and Hendriks, A. (1992). Reactive Planning. In *Encyclopedia of Artificial Intelligence (Second Edition)*, pages 1171–1181. John Wiley & Sons, Inc.

[Maes, 1995] Maes, P. (1995). Artificial Life meets Entertainment: Lifelike Autonomous Agents. *Communications of the ACM*, 38(11):108–114.

[McDermott, 1990] McDermott, D. (1990). Planning Reactive Behavior: A Progress Report. In Sycara, K., editor, *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, San Mateo (CA). Morgan Kaufmann.

[Miceli and Cesta, 1993] Miceli, M. and Cesta, A. (1993). Strategic Social Planning: Looking for Willingness in Multi-Agent Domains. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, Boulder (CO).

[Minton, 1988] Minton, S. (1988). *Learning Search Control Knowledge: An Explanation-based Approach.* Kluwer Academic Publishers.

[Page, 1983] Page, M. M., editor (1983). *Personality. Current Theory and Research.* University of Nebraska Press, Lincoln.

[Pervin, 1989] Pervin, L. A. (1989). *Goal Concepts in Personality and Social Psychology.* Lawrence Erlbaum Associates, Hillsdale (NJ).

[Reilly, 1996] Reilly, W. S. (1996). *Believable Social and Emotional Agents.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Technical Report CMU-CS-96-138.

[Rizzo et al., 1995] Rizzo, P., Miceli, M., and Cesta, A. (1995). On helping behavior in cooperative environments. In *Proceedings of the International Workshop on the Design of Cooperative Systems (COOP'95)*, pages 96–108, Le Chesnay Cedex (France). INRIA.

[Rizzo et al., 1997] Rizzo, P., Veloso, M., Miceli, M., and Cesta, A. (1997). Personality-Driven Social Behaviors in Believable Agents. In *Working Notes of the AAAI Fall Symposium on "Socially Intelligent Agents"*. AAAI Technical Report FS-97-02.

[Rosenschein, 1985] Rosenschein, J. S. (1985). *Rational Interaction Among Intelligent Agents.* PhD thesis, Computer Science Department, Stanford University, Stanford (CA).

[Rousseau and Hayes-Roth, 1996] Rousseau, D. and Hayes-Roth, B. (1996). Personality in Synthetic Agents. Technical report, Knowledge Systems Laboratory, Department of Computer Science, Stanford University, Stanford (CA).

[Schank and Abelson, 1977] Schank, R. C. and Abelson, R. P. (1977). *Scripts, Plans, Goals and Understanding. An Inquiry into Human Knowledge Structures.* Lawrence Erlbaum Associates, Hillsdale (NJ).

[Sloman, 1987] Sloman, A. (1987). Motives, Mechanism, and Emotions. *Cognition and Emotion*, 1(3):217–233.

[Stone et al., 1994] Stone, P., Veloso, M. M., and Blythe, J. (1994). The Need for Different Domain-Independent Heuristics. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94).*

[Veloso, 1994] Veloso, M. M. (1994). *Planning and Learning by Analogical Reasoning.* LNAI 886. Springer-Verlag.

[Veloso and Carbonell, 1993] Veloso, M. M. and Carbonell, J. (1993). Derivational Analogy in PRODIGY: Automating Case Acquisition, Storage, and Utilization. *Machine Learning*, 10:249–278.

[Veloso et al., 1995] Veloso, M. M., Carbonell, J., Perez, A., Borrajo, D., Fink, E., and Blythe, J. (1995). Integrating Planning and Learning: The PRODIGY Architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:81–120.

[Wilkins and Myers, 1995] Wilkins, D. E. and Myers, K. L. (1995). A Common Knowledge Representation for Plan Generation and Reactive Execution. *Journal of Logic and Computation.*

[Winell, 1987] Winell, M. (1987). Personal Goals: The Key to Self-Direction in Adulthood. In *Humans as Self-Constructing Living Systems: Putting the Framework to Work.* Lawrence Erlbaum Associates, Hillsdale (NJ).