A Testbed for investigating personality-based multiagent cooperation

C Castelfranchi [1], F de Rosis[2], R Falcone [1] and S Pizzutilo [2]

[1] Istituto di Psicologia, CNR, Rome

cris, falcone@pscs2.irmkant.rm.cnr.it

[2] Dipartimento di Informatica, Università di Bari

derosis, pizzutil@gauss.uniba.it

Abstract

We describe a framework which is aimed at testing theories about personality-based multiagent cooperation. Personalities affect the agents behaviour in the delegation, help and reaction phases. The testbed enables programming agents in a flexible and domain-independent way, by defining their mental state, abilities and communication language. Agents cooperation in different domains can then be simulated, to test the effect of various personality combinations, also in situations of goal conflict.

1. Introduction

Autonomy is a distinguishing feature of the agent paradigm: designing a 'supporting' agent equates to defining the circumstances in which the agent will activate itself, whether it will follow literally the request received or will interpret it to abduce what it is expected to do in the specific circumstance, whether it will respond critically to this request, and so on. These design features may be seen in terms of 'type of delegation' given to the agent and 'type of help' that the agent is programmed to offer. For an analysis of the different types of delegation/help which may hold in multiagent or human-agent cooperation, see [Cas96].

Any software agent includes an implicit definition of the form of delegation it presumes to receive and of the form of help it is able to offer: when the agent will come into action and for which task, what it knows about other agents' characteristics, when it will remit its commitment and so on. In the majority of frameworks aimed at simulating multiagent systems' behaviour, every agent reasons only 'on itself' (that is, on what it can and wants to do) and applies for other agents' help only in case of need. Delegation originates from awareness of not being able to do by itself, and request of help always concerns a well defined task. Symmetrically, offer of help is bound to the received request. Partial knowledge is contemplated in these systems, but not insincerity: agents are usually 'gullible', in that they are ready to believe to messages received and to retract their previous knowledge accordingly (see for instance [OHa97]).

In our multiagent cooperation theory, the main assumption is that no 'optimal' delegation/help attitude can be defined, and that efficient multiagent systems may stem from a combination of different cooperation attitudes. In a recent paper [Cas97], we called 'personality' this combination of attitudes, coherently with a software anthropomorphization in which the program's knowledge base, control structure and input/output are seen as 'mental state', 'reasoning forms' and 'communication language'. We are convinced that a future society of artificial agents built in a partially uncontrollable way will have to cope with etherogeneous and typically 'human' behaviours such as lie, elusion and similar. To test whether this world might, in some cases, be more efficient than a world in which more rigid and uniform behaviours are combined, we designed a simulation framework that we called GOLEM. GOLEM is aimed at formalising, implementing, and experimenting different kinds and levels of social delegation and adoption, represented as different social attitudes or personality traits. In this paper, we describe its functions and architecture, by showing an example of its functioning.

2. GOLEM's characteristics

In order to respond to the above mentioned goals, we introduced the following features in our testbed:

a. *domain independence:* agents are built separately from the application domain, and their behaviour is described in a domain-independent way;

b. *flexibility in the description of agents behaviour*: decision strategies that agents apply in the different phases of cooperation can be revised easily; new individual personalities can be added if needed, and new combinations of personalities, for groups of interacting agents; new relations between personality and reasoning can be introduced, new inter-agent communication forms, and so on;

c. *flexibility in the representation of mutual knowledge*: an agent may have an incomplete or even incorrect knowledge of other agents, whereas it knows exactly itself,

d. *flexibility in the levels of sincerity in communication*; one of the personality traits that can be introduced in an agent description is its 'propensity to lie or to be reticent': this affects the way that an agent's decision is transformed into a communicative act.

Opposite to these abstraction and flexibility features in the agent definition, we made some initial simplifying assumption about the system functioning:

e. *we limit to two the number of interacting agents;*

f. *we serialize their activity*, with a synchronous 'turn taking' behaviour.

The first prototype of GOLEM is designed, however, so as to relax these assumptions in the future releases.

3. GOLEM's functions

The testbed includes the three main functions that are typical of these systems [Dec96]. The first two of them enable building a 'world' by defining the two agents characteristics and the domain in which they will play: a graphical interface guides the user in this description and an interpreter translates it in an internal form, by checking syntax errors. The third function enables simulating the agents activity, starting from an initial state of the world.

3.1. Domain facility

An application domain is represented as an oriented graph, whose nodes correspond to domain-states and whose arcs correspond to domain-actions bringing from one state to the next. Nodes are objects to which we associate: (i) a state-name `s`, (ii) a symbolic description of the state, (iii) the name of an image representing the state. Arcs are objects to which we associate an action-name `a`.

This description enables computing the values of several properties of a state `s`. For example:

`(Performable (a))` : "`a` can be performed in `s`";

`(Achieve (a s'))` : "`a`, performed in `s`, brings to `s'` ";

`(Currently-Unachievable a s')`: "`s'` cannot be achieved, in the present context, with `a`";

`(Conflicts a s')`: "`a` brings into a state which is in conflict with `s'`".

Two states are defined to be 'in conflict' when their descriptions (with the addition of some 'frame condition') are contradictory. For instance, in a domain which includes small and large blocks, 'big-tower' (a tower of large blocks) is in conflict with 'bell-tower' (the same tower, with a small block on top of it), by virtue of the frame condition: `On (x y) -> not (Clear y)`.

This domain description enables us to easily implement algorithms of path-searching in a graph which are required by plan-evaluation and plan-recognition functions[1].

3.2. Agent development facility

Our rational agents have a *mental state* which includes knowledge about themselfs and about the way that personalities affect reasoning, in general.

The components of the mental state are:

- a set of *private and communicative actions* that the agent can perform;
- a set of *reasoning and commitment rules* which settle the agent behaviour in the various phases of the play;
- a set of *personality* traits;
- a set of *basic belief*s;
- a *domain-goal* (domain-state that the agent desires achieving).

Private and communicative actions are the same for all agents, as well as commitment rules. Agents differ in the personality traits (and consequently, as we will see, in the reasoning rules), in the basic beliefs and in the domain goals.

a. actions

Like in [OHa96], we classify private actions into physical and cognitive ones. P*hysical actions* correspond to domain transformations or control activities:

`(Perform Ai a)`: "Ai performs a domain-action `a`" ,

`(WaitUntil Ai a)`: "Ai periodically controls the domain state, to verify that `a` has been performed". *Cognitive actions* correspond to forms of reasoning. For instance:

- cognitive diagnosis (abductive reasoning about another agent's mental state);
- plan recognition and plan evaluation;
- ATMS-based updating of own mental state.

[1] With the first term, we mean "finding the plans to achieve a given goal, from an initial state"; with the second one "making hypotheses about a goal, given an initial state and other information such as the initial action of the plan, the 'category' of goal or the agent's personality".

Cognitive diagnosis and plan recognition-evaluation processes are influenced by the agent personality.
Given the main purpose of our testbed (to test theories about multiagent cooperation), the object of *communicative actions* are the two agents' abilities and intensions. They can therefore be classified in three categories: REQUEST, INFORM (about intention or about capability) e QUERY (idem).

b. rules

Differently from other testbeds or agent-based languages, we model separately the intention forming process from translation of intentions into a -private or communicative- action: we claim that the first process is personality-dependent, whereas the second is not. Within this distinction, we further classify rules according to the cooperation phase to which they apply, by distinguishing among delegation, help and reaction. We then have six types of rules overall: delegation/help/reaction-reasoning and delegation/help/reaction-commitment.

c. personalities

In [Cas97], a detailed description of the main traits introduced in GOLEM's agents is presented; a personality trait is modeled as a logically coherent combination of reasoning rules. Different traits affect the delegation and help attitudes: an agent's personality is therefore described by a plausible combination of a delegation and an help attitude, and a set of reasoning rules that correspond to the two traits is attributed to the agent's mental state, accordingly.

d. basic beliefs

these are ground atomic formulae which represent the agent's beliefs and goals about the domain, about itself or about other agents.

3.3. Evaluation facility

Our agents play in a domain, by trying to achieve their compatible or conflicting goals. The user can set the conditions of a simulation and follow how it proceeds, through a graphical interface. A particular simulation starts by selecting a 'world' and by setting the initial domain state and the agent which 'moves first'. The two agents introduce themselves by declaring their personalities and abilities; in this introduction, they may give partially incorrect or 'abstract' descriptions, or may even lie about them. For example: an agent might omit the description of its abilities, might say that it is able to 'make towers' without specifying whether they are big or small, or might tell that it is not able to do some action whereas it can do it.
In any phase of interaction, the user can look at the agents' mental state in a graphical window. It can permanently follow the progress of the play by means of a *domain window* (which shows the domain state with actions made at each turn) and a *dialog window* (which shows the communicative acts that the two agents exchange, translated into natural language).

4. Agent programming language

As reasoning by an agent is aimed at deciding 'what to do in the present turn', we do not need representing time in our language. This means that, in GOLEM, agents cannot intend to perform a specific action in a specific time instant (like, for instance, in [Sho93] or in [OHa96]) : they can only decide whether or not to do it *at their turn*. As we anticipated in Section 3.2, an agent Ai is programmed by defining its rules and its basic beliefs .

a. rules:

Rules are logical combinations of belief and goal-atoms, commitments and messages.

Given: a domain-action `a` and a domain-state `s`, we call

- *belief-atom* an atomic formula: (BEL Ai β), with β = positive or negative literal which represents:
  - a domain-property:

    `(T s)` "the present domain-state is `s`"

    or one of the properties defined in 3.1.

    We call *domain-belief* an agent's belief about a domain-property.
  - an agent's ability or intentional attitude:

    `(CanDo Ai a)`: "Ai is able to perform `a`"

    `(IntToDo Ai a)`: "Ai intends to do `a`"

    We call *ability and intentional belief* an agent belief about such an attitude.

- *goal-atoms:* (GOAL Ai γ), with γ = positive or negative literal.

  Our agents may hold goals about the domain-state:

  `(GOAL Ai (T s))`: "Ai desires that `s` is achieved".

  As a consequence of such a goal, they may have the goal that an action `a` is performed:

  `(GOAL Ai (EvDoneFor a s))`. We call these atoms *domain-goals*.

  Agents may have the purpose of influencing other agents' behaviour so as to satisfy their domain-goal; this purpose is represented as a goal about other agents' intentions:

  `(GOAL Ai (IntToDo Aj a))`: "Ai desires that Aj intends to do `a`".

  We call this an *intentional-goal*.

- *commitments* to perform *private or communicative actions*:

  Do (<private-action> | <communicative-action>).

  For example:

  `(DO (Perform Ai a))` : "Ai commits itself to perform `a`"

  `(DO (WaitUntil Ai a))` : "Ai waits until `a` is performed (withouth doing anything)"

  `(DO (Request Ai Aj a))`: "Ai asks to Aj to adopt the intention of performing `a`"

  `(DO (RespondP Ai Aj a))`: "Ai responds positively to Aj's request"

  ...and so on

- *messages:* Done (<private-action> | <communicative-action>).

  Like in [Ric97], our agents communicate by exchanging a communicative action or by looking at the state of the world. In addition to messages of the type:

  `(DONE (Request Aj Ai a))`, or `(DONE (RespondP Aj Ai x))`

  Ai may receive a message from the world, that corresponds to 'seing' that agent Aj performed a domain action: `(DONE (Perform Aj a))`,

  or passed the hand without doing anything:

  `(DONE (WaitUntil Aj a))`.

Rules establish a logical relationship among the described elementary atoms:

- *reasoning rules* state how an intentional belief or goal is achieved:

  <reasoning-rule> ::= [<message>] (AND (<domain-belief> | <domain-goal> | <ability-belief>))* →
  <intentional-belief> | <intentional-goal>
- *commitment rules* state how an intentional belief or goal is translated into a commitment:

  <commitment-rule>::= <intentional-belief> | <intentional-goal> (AND (<domain-belief>))*→
  <commitment>

As we anticipated in the previous Section, personality traits are consistent blocks of reasoning rules which establish when to delegate or adopt a domain-action, which kind of help to provide and how to react to a refusal of help. Other attitude-dependent rules define criteria to assess when a goal conflict exists and when to conclude that the other agent really needs help. In [Cas97], the main personality traits introduced in GOLEM are described; let us here see some examples.

- <u>delegation-personalities:</u>

- a lazy agent Ai always delegates an action `a` if there is another agent Aj which is able to do it:

  ```
  (GOAL Ai (EvDoneFor a g)) AND (BEL Ai (CanDo Aj a)) → (GOAL Ai (IntToDo Aj a)).
  ```

  It acts by itself only when there is no alternative:

  ```
  (GOAL Ai (EvDoneFor a g) AND (BEL Ai (CanDo Ai a)) AND (BEL Ai NOT (CanDo Aj a)) →
     (BEL Ai (IntToDo Ai a))
  ```
- an hanger-on tends to never act by itself; it either delegates:

  ```
  (GOAL Ai (EvDoneFor a g)) AND (BEL Ai (CanDo Aj a)) → (GOAL Ai (IntToDo Aj a))
  ```

  or concludes that its domain-goal g cannot achieved, in the present context:

  ```
  (GOAL Ai (EvDoneFor a g) AND (BEL Ai NOT (CanDo Aj a)) →
     (BEL Ai (CurrentlyUnachievable a g))
  ```

...and so on.

- <u>helping personalities:</u>

- a hyper-cooperative always helps if it can:

  ```
  (BEL Ai (GOAL Aj (IntToDo Ai a))) AND (BEL Ai (CanDo Ai a)) →
     (BEL Ai (IntToDo Ai a))
  ```

- a benevolent first checks that the request does not conflict with its goal:

  `(BEL Ai (GOAL Aj (IntToDo Ai a))) AND (GOAL Ai(T g)) AND`

  `(BEL Ai NOT(Conflict a g)) AND (BEL Ai (CanDo Ai a))` $\rightarrow$ `(BEL Ai (IntToDo Ai a))`

...and so on.

Other helping attitudes further diversify the behaviour of the helping agent:

- *level of engagement* in helping: a literal helper restricts itself to considering the requested action, an overhelper goes beyond this request through plan recognition, to hypothesize a delegating agent's higher order goal and helps accordingly, and so on.
- *control of conflicts* between the requested action and its own goals:
  an action can immediately bring to a state which is in conflict with the helper's domain-goal (we call this a 'surface-conflict'); or, it can be part of a delegating agent's plan which, in the long term, will produce a conflict (we call this a 'deep-conflict'). A deep-conflict-checker will check that no such conflicts are created by the requested action, by making a plan-recognition on the delegating agent's mental state. A surface-conflict-checker will restrict itself to examining the immediate consequences of the requested action.
- *control of the delegating agent's know-how*:
  this, again, can be restricted to examining whether that agent would be able to perform the requested action (in a surface-knowhow-checker) or can go deeper to examine whether alternative plans exist, which bring to the delegating agent's presumed goal and that this agent would be able to perform by itself (in a deep-knowhow-checker).

b. basic beliefs or goals

these are ground beliefs and goal atoms about a particular domain and about own and other agent's abilities.

For example:

let us take Adam and Eve as agents and the blocks world as a domain; let us call "big-tower" a domain state which contains a tower of big blocks, "small-tower" a state with a tower of small blocks and "twin-tower" a state with a big and a small tower. Let us call "make-big-tower" and "make-small-tower" the corresponding building action, from what we call the 'stock' state (a state with all clear blocks on the table). The following are examples of Eve's basic beliefs and goals, in the considered case:

```
(GOAL Eve (T twin-tower)),
(BEL Eve (CanDo Adam make-big-tower))
(BEL Eve (CanDo Adam make-small-tower))
(BEL Eve (CanDo Eve make-small-tower))
```

....and so on.

5. Architecture

Agents of GOLEM do not interact directly. They exchange messages through a *Message Board*, which holds knowledge about the domain and about the agents abilities and personalities that they declared in their self introduction. A *Message Board Handler* responds to agent queries about this knowledge; it also receives "Do (action)" messages at the end of each turn: it changes the domain state according to private actions, forwards communicative actions to the interface and sends back a "Done (action)" message. An *Interface Handler* is responsible for user interaction: it displays graphically the domain state and the agents' characteristics and translates communicative actions into natural language sentences.

Let us just give the blueprint of the basic steps each agent considers during its turn:

```
a. receive 1) "It is your turn" message, and
           2) messages about the current state of the world, and
           3) messages about the action executed at the previous turn, and
           4) possible messages of request;
b.      update mental state accordingly
c.      if there is a request (from the other agent)
        c1.    then   either
                      - (refuse and c2) or
                      - (examine own goals and plans (if any) and on the basis
                      of own personality and possible conflicts decide to help,
                      or not, and how)*
               else
        c2.      if own goals exist
                     then   find a feasible plan and (on the basis of own personality)
                            either delegate (request) and send the message
                            or perform the first action
                     else   pass the hand
* In the case of help: commit itself to perform the requested action; perform the action and
pass the hand.
```

The module that implements the resolution-based inference is in Lisp; other modules (including the Interface and the Message Board Handlers) are implemented in Java.

6. An example

To give an idea of the kind of simulation we can implement in GOLEM, let us extend the example that we introduced in Section 4. The initial domain state is 'stock'; Eve moves first.

turn 1:

Eve's goal is to build a 'twin-tower'; she could start building the small-tower, but is lazy, and therefore decides to request to Adam to do it.

turn 2:

Adam is benevolent, and will help Eve if the requested action does not conflict with his goal. Let us suppose that his goal is a 'cathedral' with a small-tower and a big-building (two overlapping big blocks): so, there is no surface-conflict between Eve's request and his goal. However, Adam is a deep-conflict-checker and he knows that Eve likes 'many-towers-worlds' (maybe because she told it in her self-introduction). In his plan-recognition on Eve's mind, he will then conclude that maybe Eve wants to come to insert a big tower in the world, and this is in conflict with his own goal. He therefore refuses to help.

turn 3:

Eve interprets Adam's refusal as due to his inability to make the requested action, and decides to do the job by herself; she builds the small-tower.

turn 4:

Adam completes his plan by making the big-building.

turn 5:

Eve recognizes the conflict and gives up. Interpreting Adam's refusal of help as due to his inability of helping rather than to a goal conflict was at the origin of her defeat.

Main References

[Cas96] C Castelfranchi and R Falcone (1996): Levels of help, levels of delegation and agent modeling. *AAAI-96 Agent Modeling Workshop*.

[Cas97] C Castelfranchi, F de Rosis and R Falcone: Social Attitudes and Personalities in Co-operation. Submitted to ATAL97.

[Dec96] K S Decker (1996): Distributed artificial intelligence testbeds. In: Foundations of Distributed AI, (G M P O'Hare and N R Jennings Eds), John Wiley & Sons.

[OHa96] G M P O'Hare (1996): Agent Factory: an environment for the fabrication of multiagent systems. In the same book.

[Rao91] A S Rao and M P Georgeff (1991): Modeling rational agents within a BDI-architecture. In: *Principles of Knowledge Representation and Reasoning*.

[Ric97] C Rich and C Sidner (1997): COLLAGEN, when agents collaborate with people. Proceedings of Agents'97, pp. 284-291.

Y Shoham (1993): Agent-oriented programming. *Artificial Intelligence*, 60.