DALI Logical Agents into Play

Stefania Costantini, **Annalisa D'Andrea**, **Giovanni De Gasperis**,
Niva Florio, and **Arianna Tocchio**

Dip. di Ingegneria e Scienze dell'Informazione e Matematica, Università di L'Aquila, Italy

1 Introduction

This paper stems from an experience made by Annalisa D'Andrea (with her colleague Daniela Pavone) as a student of the 'Artificial Intelligence and Intelligent Agents' graduate course at the University of L'Aquila (held by Prof. Stefania Costantini and Dr. Arianna Tocchio). The experience has been further developed by Annalisa as a Ph.D. student, with her collegue Niva Florio and with the help of Dr. Giovanni De Gasperis.

The application that we illustrate here is developed in DALI, an agent-oriented logical language that extends Prolog with reactive and proactive features (see, e.g. [1,2], or [3] for a full list of references on DALI). We show how DALI logical agents and their communication capabilities can be exploited to put into play one's favorite movie or book, where in particular we discuss "Il berretto a sonagli" ("Cap and Bells") by Luigi Pirandello. This by implementing agents that "impersonate" the main characters. In particular, the agents that Daniela and Niva have implemented in DALI do not simply propose chunks of Pirandello's text: rather, each agent is intended to reproduce the main features of the corresponding character's "personality". A particular contribution is given by the DALI communication architecture that provides a meta-level for specifying which messages can be sent or received, from which agents and in which cases. Then, a DALI agent can choose whether to take a message received by some other agent into consideration, or instead discard it. This according to conditions that will be defined depending upon the agent's goals, role, objective and subjective evaluation (the latter according to the agent's "personality"). Symmetrically, the agent is able to decide which agents to "talk" to, and which to exclude from consideration.

These DALI features allowed us to model a basic element of Pirandello's plot: along with the interactions, the characters grow more and more suspicious against each other, and their willingness to communicate decreases until finally arriving to a total blackout. This is rendered by modeling trust management in the communication meta-layer, thus subjecting message exchange to the agent's level of trust in the other agents.

With this example we envisage for future agent-based Artificial Intelligence an extended Turing test, where (in perspective) a human should find it not so easy to understand whether (s)he is talking to a real character or to its agent counterpart. In fact, several students of the Artificial Intelligence and Intelligent Agents graduate course have chosen to model their favorite movie or book in DALI as the subject of their final project, with surprisingly interesting results. In the future we will try to propose "The impossible interviews" ("Le interviste impossibili"), drawing inspiration from a programme dating back to 1975 by RAI2 (one of the three radio networks owned by the Italian government) where a contemporary scholar pretended to interview the "ghost"

of some famous person lived in the past. Agents might play the role of the interviewed but possibly also of the interviewer.

The paper is structured as follows. In Section 2 we briefly introduce the DALI language. In Section 3 we summarize some aspects of "Cap and Bells" and how we represent them in DALI. Finally, in Section 4 we conclude.

2 The DALI Language in a Nutshell

DALI [1,2,4] is an Active Logic Programming language designed for executable specification of logical agents. The DALI interpreter is freely available [5]. A DALI agent is a logic program that contains a particular kind of rules, reactive rules, aimed at interacting with an external environment. The environment is perceived in the form of external events, that can be exogenous events, observations, or messages by other agents. In response, a DALI agent can perform actions, send messages, adopt goals, etc. The reactive and proactive behavior of the DALI agent is triggered by several kinds of events: external events, internal, present and past events. It is important to notice that all the events and actions are time-stamped, so as to record when they occurred. The new syntactic items, i.e., predicates related to events and proactivity, are indicated with special postfixes (which are coped with by a pre-processor) so as to be immediately recognized while looking at a program.

2.1 DALI Basic Features

External Events. The external events are syntactically indicated by the postfix *E*. When an event comes into the agent from its "external world", the agent can perceive it and decide to react. Reaction is defined by a reactive rule which has in its head that external event. The special token $:>$, used instead of $:-$, indicates that reactive rules performs forward reasoning. Operationally, incoming events are added to a list called EV and "consumed" according to the arrival order, unless different priorities are specified. Priorities are listed in a separate initialization file including various kinds of directives. By means of these directives the user, in the spirit of [6], can "tune" the agent's behavior under several respects. The advantage of introducing a separate initialization file is that for affecting control there is no need to modify (or even to understand) the agent's main program.

Actions. Actions are the agent's way of affecting her environment, possibly in reaction to an external or internal event. In DALI, actions (indicated with postfix *A*) may have or not preconditions: in the former case, the actions are defined by actions rules, in the latter case they are just action atoms. An action rule is just a plain rule, but in order to emphasize that it is related to an action, we have introduced the new token $:<$, thus adopting the syntax $action :< preconditions$. Similarly to events, actions are recorded as past actions.

Internal Events. The internal events define a kind of "individuality" of a DALI agent, making her proactive independently of the environment, of the user and of the other agents, and allowing her to take initiatives, to adopt goals and intentions, to execute

plans and to manipulate and revise her knowledge. An internal event is syntactically indicated by the postfix *I*, and its description is composed of two rules. The first one contains the conditions (knowledge, past events, procedures, etc.) that must be true so that the reaction (in the second rule) may happen. Internal events are automatically attempted with a default frequency customizable by means of directives in the initialization file. The user directives can tune several parameters: at which frequency the agent must attempt the internal events; how many times an agent must react to the internal event (forever, once, twice,...) and when (forever, when some specific triggering conditions occur, ...); how long the event must be attempted (until some time, until some terminating conditions, forever). Internal events are treated by the interpreter analogously to external events. A particular kind of internal event is the *goal*, postfix *G*, that stops being attempted as soon as it succeeds.

Present Events. When an agent perceives an event from the "external world", it doesn't necessarily react to it immediately: (s)he can first reason about the event, before (or instead of) triggering a reaction. At this stage, the event is called *present event* and is indicated by the suffix *N*.

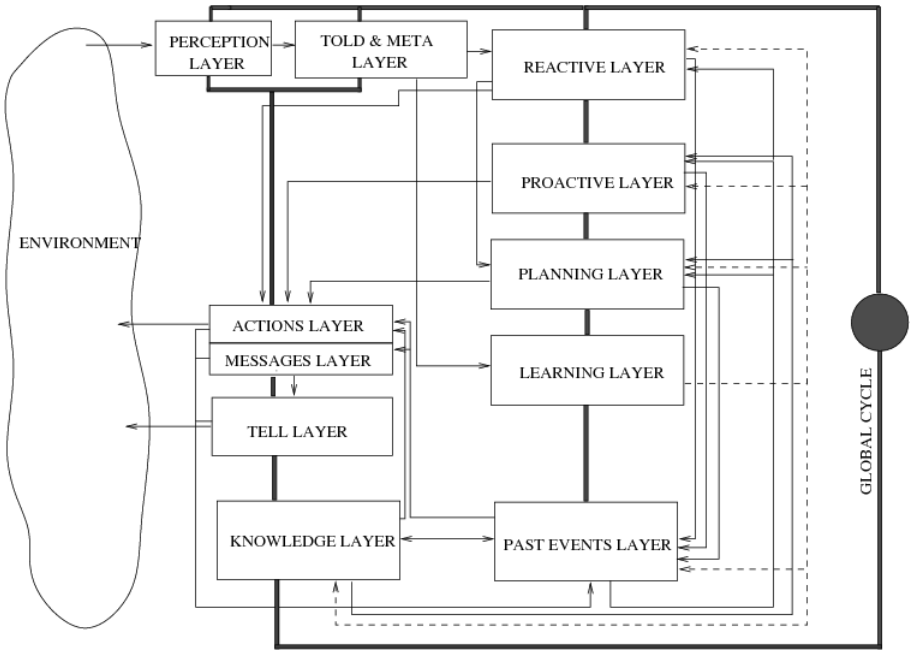

Fig. 1. Architecture of a DALI Agent

Past events Past events represent the agent's "memory", that makes her capable to perform its future activities while having experience of previous events, and of its own previous conclusions. Each past event has a time-stamp T indicating when the recorded

event has happened (i.e., when it has been reacted to). Memory of course is not unlimited, neither conceptually nor practically: it is possible to set, for each event, for how long it has to be kept in memory, or until which expiring condition. In the implementation, past events are kept for a certain default amount of time, that can be modified by the user through a suitable directive in the initialization file. Implicitly, if a second version of the same past event arrives, with a more recent time-stamp, the older event is "overridden", unless a directive indicates to keep a number of versions. Old versions of past events are placed in a special structure called PNV, as they may have a role in allowing the agent to reason about the past.

The overall structure of a DALI agent, built out of formerly described elements, is depicted in Figure 1, where the communication components are discussed below.

2.2 DALI Communication Architecture

The DALI communication architecture [7], outlined in Figure 2, consists of four components. The first component (TOLD layer) implements the DALI/FIPA communication protocol and a filter on incoming communication, i.e. a set of rules that decide whether or not to accept reception of a message. The second component is a meta-reasoning user-customizable module that tries to understand message contents, possibly based on ontologies and/or on forms of commonsense reasoning. The third component is the 'core' DALI interpreter. The fourth component implements a filter for the out-going messages. The DALI/FIPA protocol consists of the main FIPA primitives, plus few new primitives which are peculiar of DALI.

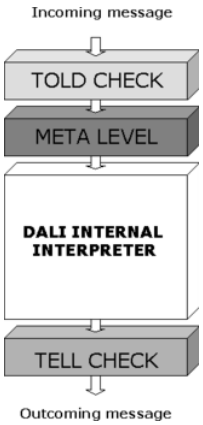

Fig. 2. DALI Communication architecture

When a message is received, it is examined by the TOLD check layer, which is adaptable to the context and modifiable by the user. This filter checks sender and content of the message, and verifies if the conditions for reception are verified. If the conditions are false, this security level eliminates the supposedly wrong message. A similar check is performed to out-going messages via the TELL layer.

DALI communication filters and meta-reasoning on messages are specified by means of meta-level rules defining the distinguished predicates $tell$, $told$ and $meta$ (the latter not discussed here). Some standard rules of general use are pre-defined, to which the user can add new others, specific for the application at hand.

Whenever a message is received, coming from a $Sender$ agent, with content part $Content$, the DALI interpreter automatically looks for a matching $told$ rule. If such a rule is found, the interpreter attempts to prove $told(Sender, Content)$. If this goal succeeds, then the message is accepted, and $Content$ is added to the set of the external events incoming into the receiver agent. Otherwise, the message is discarded. Semantically, this can be understood as implicit reflection up to the filter layer, followed by a reflection down to whatever activity the agent was doing, with or without accepting the message. Symmetrically, messages that an agent sends are subjected to a check via $tell$ rules. For every message that is being sent, the interpreter automatically checks whether an applicable $tell$ rule exists. If so, the message is actually sent only if the goal $tell(Receiver, Content)$ succeeds. Via $tell$ rules one can manage, e.g., mailing lists. In general, via the TELL/TOLD filters useless message exchange can be considerably reduced.

$tell$, $told$ and $meta$ rules are contained in a separate file with respect to the main agent program. Thus, they can be changed without affecting or even knowing the DALI code. The FIPA/DALI communication protocol is implemented by means a piece of DALI code including suitable $tell/told$ rules. This code is in turn specified in a separate file that each DALI agent imports as a library, so that the communication protocol can be seen an "input parameter "of the agent.

The overall DALI communication architecture is aimed at improving *elaboration tolerance* where, [8]:

> "A formalism is elaboration tolerant to the extent that it is convenient to modify a set of facts expressed in the formalism to take into account new phenomena or changed circumstances. ...

Communication in DALI is in fact devised in order to be elaboration-tolerant with respect to both the protocol, and the filter.

3 Description of the Application and Demo

As the representation of knowledge and trust plays in general an important role in the interactions among agents, we are going to show in practice how trust can affect the exchange of information among agents. To this purpose, we have defined as a DALI application a small reduction of the Pirandello's comedy "Il berretto a sonagli" ("Cap and Bells"). Typical for Pirandello is to show how illusion mixes with reality and how people see things in very different way: words are unreliable and everything can be seen at the same time as true and false. In "Il berretto a sonagli" he emphasizes that telling everyone the truth, the sole and cruel truth, regardless of manners or respect and in spite of social habits, will soon produce isolation and a reputation of being mad.

The three main characters of this play have been implemented by three agents of a DALI system: we have not merely reproduced dialogues and lines written by Pirandello,

but, based on the dialogues of this play, we have tried to realize agents that mimic the personality of the characters and that reason spontaneously as a spectator would expect.

The title of this play refers to the hat worn by court jesters, that represents the hat of shame to be worn in public. In this comedy, Pirandello pushes the characters into a dilemma with no apparent solution. It is set in Sicily in the early twentieth century and tells the story of Ciampa, a middle-aged bank clerk who lives with his beautiful young wife, Nina, in a small town where sexual infidelity and dishonor are matters of life and death. His boss's wife, Mrs. Beatrice Fiorica, a jealous and dissatisfied woman, becomes convinced (though without proof) that her husband is sleeping with Nina. She is shocked by the discovery of the betrayal and, thinking only of revenge, issues an official "denunciation" of the alleged affair. Although a police investigation turns up no improprieties (also because the police officer Spanò does not want to stand against Beatrice's husband, Cavalier Fiorica), Ciampa's name is still sullied, on the theory that where there is smoke there still must be fire. Ciampa, who paradoxically has always liked to think of himself as the sole voice of reason and sanity in a crazy context, for the sake of respectability does not abandon his wife, but he seriously consider killing her in order to regain his reputation. Luckily for him however, due to Spanò's deception Beatrice is declared legally insane, and is hospitalized for three months in a mental hospital for acting like a fool.

In the practical demonstration of the application, the first act of the comedy is represented. The involved agents communicate through messages. Each agent adopts a communication strategy implying that messages can be exchanged only with trusted agents (in particular, with those for which the level of trust is more than a minimum). At the beginning, each agent trusts the others. Along with the interactions, the level of trust of each agent towards the others will change, and in our particular case it will decrease. In the end, it will become so low as to prevent any further exchange of messages. Trust management for this example is implemented in DALI in a very simple way by means of the following rules:

$$told(Sender, Content) :- trust(Sender, Trust), Trust > k.$$
$$tell(Receiver, Content) :- trust(Receiver, Trust), Trust > k.$$

These rules specify that a certain minimum degree of trust is required in order to enable message exchange. In fact, the first rule accepts reception of incoming messages only if the degree of trust in sender agent is greater than threshold k. The second rule does the same for out-going messages, that are not allowed to be sent to a receiver trusted k or less. Thanks to this mechanism of interaction based on trust, one can leave the agent's "core" program untouched, and still determine an evolution on the behavior of communicating agents.

For the internal reasoning of each agent we widely used internal events. We do not report the full DALI code here, but we just notice that by means of internal events each agent/character "meditates" about her/his circumstances: Beatrice about how to seek revenge, Ciampa about how to restore his good reputation, Spanò about how to avoid to acquire enemies. Based on their internal conclusions, they will then decide to do something. In particular, their internal reasoning process affects the level of trust in other agents. Also, for instance, a character can at a certain point decide that another

one is an enemy. This conclusion directly cuts-off communication (even independently of the present level of trust) whenever the following $told$ rule belongs to the TOLD layer:

$$told(Sender, Content) :- not(enemy(Sender)).$$

The demonstration related to this paper shows examples of communication between agents representing Pirandello's characters. It should be possible to notice that the interaction is entirely based on the reconstruction in the DALI system of the Pirandello's character personalities, and that these characteristics determine the generation of events and the corresponding agent answers.

4 Conclusions

The present work can be seen as part of a long-termed experimentation, for which we thank all students of the 'Artificial Intelligence and Intelligent Agents' course: in fact, via the projects that they have to prepare for their exam, they learn how to program agents and at the same time they help us understand how to improve the DALI framework. The specification of the project (cf. the Course web site http://www.di.univaq.it/stefcost/AI2.htm) requires students to choose a movie or a novel of interest for them, and implement as DALI agents the main characters and their interactions. In the opinion of all involved parties, this is an example of how learning can sometimes be fun!

References

1. Costantini, S., Tocchio, A.: A logic programming language for multi-agent systems. In: Logics in Artificial Intelligence, Proc. of the 8th Europ. Conf.,JELIA 2002. LNAI 2424, Springer-Verlag, Berlin (2002)
2. Costantini, S., Tocchio, A.: The DALI logic programming agent-oriented language. In: Logics in Artificial Intelligence, Proc. of the 9th European Conference, Jelia 2004. LNAI 3229, Springer-Verlag, Berlin (2004)
3. Costantini, S.: The DALI agent-oriented logic programming language: References (2012) at URL http://www.di.univaq.it/stefcost/info.htm.
4. Costantini, S., Tocchio, A., Verticchio, A.: Communication and trust in the dali logic programming agent-oriented language. Intelligenza Artificiale, J. of the Italian Association **2**(1) (2005) in English.
5. Costantini, S., D'Alessandro, S., Lanti, D., Tocchio, A.: DALI web site, download of the interpreter (2010) http://www.di.univaq.it/stefcost/Sito-Web-DALI/WEB-DALI/index.php, With the contribution of many undergraduate and graduate students of Computer Science, L'Aquila. For beta-test versions of the interpreter (latest advancements) please ask the authors.
6. Kowalski, R.A.: Algorithm = logic + control. Commun. ACM **22**(7) (1979) 424–436
7. Costantini, S., Tocchio, A., Verticchio, A.: Communication and trust in the DALI logic programming agent-oriented language. Intelligenza Artificiale, J. of the Italian Association of Artificial Intelligence **2**(1) (2005) (in English).
8. McCarthy, J.: Elaboration tolerance. In: Proc. of Common Sense'98. (1998) Available at http://www-formal.stanford.edu/jmc/ elaboration.html.