Towards a BDI Framework for Rational Behaviour Programming in Autonomous Robots

Fabrizio Messina, Giuseppe Pappalardo, Corrado Santoro
University of Catania – Dept. of Mathematics and Computer Science
Viale Andrea Doria, 6 — 95125 - Catania, ITALY
EMail: {messina, pappalardo, santoro}@dmi.unict.it

I. Background and Motivations

The design of *autonomous robots* poses a series of challenges which range from low-level control aspects to higher-level behavioural concerns. While low-level issues (e.g. motion control, arm control, servo driving) are dealt with using traditional control systems techniques (e.g. PID controllers), the design of an *"intelligent behaviour"* requires to take into account proper AI techniques to let the robot to reach its goal.

With the advent of more sophisticated robots (e.g. humanoids), also aimed at accomplishing assistance tasks at home (i.e. *home robots*), the aspect of intelligence is particularly stressed: indeed, these robots are expected to perform tasks whose complexity is still increasing, while living in a physical and human environment which is often highly unpredictable and not fully observable. In order to face such issues, the software designed to control the behaviour of these kind of robots has to always consider the possible occurrence of unexpected situation and then be able to adopt countermeasures, in order to—sooner or later—achieve the goals. In other words, the software should exhibit human-like characteristics such as *rationality* and *deliberation abilities*.

In such a context, classical programming models range from AI techniques, such as logic-/knowledge-based systems, to state machine-based abstraction, often employed in the field of autonomous software agents. However, in all of these models, the robot's behaviour is expressed by a predictable and prefixed sequence of actions which, even if it can feature branches, it does not allow a clear emergence of the deliberation aspect.

One of the most widely known rational models, in the field of software intelligence agents, that presents a certain form of deliberation is the *belief-desire-intention (BDI)* [5]. Basically, it is a model which tries to mimic human thinking: given an objective to achieve, starting from the current *beliefs* (which in turn are given by proper sensors), the set of the possible actions to perform is determined (*desires* or *goals*) and, on this basis, an *intention* is selected to be executed.

One of the well-known implementations of the BDI model is the abstract language AgentSpeak(L) [4] which has also been implemented in a Java-based framework, called Jason [1]. However, while the AgentSpeak(L)/Jason proposal is sound, it presents only a limited adherence to the BDI-model since it lacks one of its basic aspect: the *deliberation*; indeed, the concept of goal in AgentSpeak(L) is quite similar to that of a "procedure" (sequence of statements) and the ability of selecting the proper intention to executed is not present at all.

$$
\begin{array}{rcll}
P & ::= & g & \\
g & ::= & sg \mid cg & \\
sg & ::= & (f, p, act) & \\
f & ::= & functor \rightarrow \{true, false\} & \\
p & ::= & functor \rightarrow \mathcal{N} & \\
act & ::= & functor \rightarrow status & \\
status & ::= & ACHIEVED & \\
 & & \mid T\_FAIL & \\
 & & \mid P\_FAIL & \\
cg & ::= & (rel, g\_set) & \\
rel & ::= & ALL & \\
 & & \mid ALL\_SEQ & \\
 & & \mid AT\_LEAST(k) & \\
 & & \mid SEQ\_UNTIL & (k \geq 1) \\
g\_set & ::= & \{g_1, \ldots, g_n\} & (n > 1)
\end{array}
$$

Fig. 1. Basic GOLEM Syntax

II. The GOLEM System

In this context, with the aim of providing a way to easily program the behaviour of rational autonomous robots, the authors designed an abstract framework, called GOLEM [2], [3], whose central aspect is the concept of *goal* together with the *opportunity* to achieve it in a certain time instant. A GOLEM program is an unordered set of *goals*, which may also have dependencies to one another; executing/achieving the goals of the set implies to achieve the overall objective for which the robot has been designed.

In order to ensure the deliberation ability, the order of execution of GOLEM goals is not fixed at design stage but decided at run-time on the basis of an *aware choice* made by the robot itself. A scheduler is provided in a GOLEM system, which governs the execution of goals and their selection policy; the latter is provided, in a GOLEM program, by the designer itself, who has the responsibility of implementing not only the code of the goals but also the algorithm to perform their run-time selection.

Figure 1 reports the basic elements of GOLEM and its abstract syntax. The main entity of a GOLEM program is the *goal*, which may be *simple* or *composite*.

A goal is *simple*, sg, if it requires no specific further decision on the actions to be undertaken. It is represented with the

tuple (f, p, act), where: f is the *feasibility function*, a functor returning a boolean value stating whether the goal could be feasible or not[1]; p is the *opportunity evaluator*, a functor returning a numerical value stating the *importance*[2] of the goal w.r.t. other goals; and act is the *goal action*, a computation made of a sequence of statements aimed at achieving the goal and that do not require a particular "intelligent choice". A goal action terminates with a return value indicating a *success* or a *failure*; in the latter case, the failure may be *permanent*—i.e. there are no condition which can lead to a success—or *temporarily*—i.e. the impeding condition could disappear in the near future; in this last case, the GOLEM system takes care of re-scheduling (at the next opportunity) the goal in order to make further trials.

A composite goal, cg, is instead represented as a pair (rel, g_set) comprising a *set of sub-goals* g_set and a *relationship condition* rel; each sub-goal may be, in turn, both simple and composite while the relationship can be chosen among one of the following:

- **ALL**. The goal succeeds when *all* of its sub-goals are achieved but the order of achievement does not matter.
- **ALL_SEQ**. The goal succeeds when *all* of its sub-goals are achieved but the achievement must be performed in strict sequence.
- **AT_LEAST(k)**. The goal succeeds when *at least* k of its sub-goals (with k specified) are achieved (no constraints on the order of achievement).
- **SEQ_UNTIL**. The goal succeeds when (as soon as) *any sub-goal* is achieved; the set g_set is *ordered* and sub-goal achievement is tried in strict sequence.

Algorithm 1 Sketch of GOLEM Machine Execution Loop

```
1: procedure GAM(P:goal)
2:     while ¬ACHIEVED(P)∧¬PERM_FAILED(P) do
3:         feasibles ← {g ∈ P : IS_FEASIBLE(g)}
4:         candidates ← {g ∈ feasibles :
5:                         max OPPORTUNITY(g)}
6:         selected ←ONE_OF(candidates)
7:         EXEC(selected)
8:     end while
9: end procedure
```

A developer has to design the behaviour of its robot by identifying the relevant goals, with the proper relationships, and thus implementing the program code for the feasibility functions, opportunity functions and goal actions. The execution of the program is then governed by a GOLEM machine whose behaviour is described in Algorithm 1. As the algorithm suggests, the choice of the goal to execute is performed by analysing goal relationships and the return values of both feasibility and opportunity functions; the execution sequence is thus highly dynamic and chosen by the robot itself on the basis of current the state of the environment (which could make some goal infeasible) and the decision to make a certain goal, in a given time instant, more important than another one.

[1] A goal is infeasible if there are no condition (in the environment or in the robot) to achieve it with success.

[2] i.e. *priority*

III. TOWARDS GOLEM-BDI

The GOLEM framework described so far has been implemented by the authors in the form of software libraries for the programming languages Erlang and C. The C implementation is designed to work in microcontrollers in bare-metal, thus without the support of an operating system. Such implementations have been used to program some autonomous robots developed in author's laboratory.

Tests performed have proved the effectiveness of the approach highlighting, in particular, the ability to (*i*) autonomously choose the strategy to follow and (*ii*) face failures thus adopting countermeasures. However, despite the cited advantages, our tests showed that GOLEM lacks some features which, instead, could help a lot the design of a rational behaviour, i.e. (*i*) an adequate representation of the environment and robot's state, (*ii*) the storage of past experiences in order to reuse them in future, or (*iii*) a form of *forward reasoning* which could help the robot to infer additional information useful to perform a more aware goal selection.

Such aspects are proper of AI system, while GOLEM has been initially designed as a dynamic scheduler of goals, with the aim of having a light system in both the model and the implementation[3]. But the basic behaviour of a GOLEM system, which lets the main actor to choose itself the strategy to adopt, is quite similar that of a BDI system.

Given these premises, a natural evolution of GOLEM is to make it a *real* BDI system, by properly extending it, but without forgetting the initial requirement of having an "as light as possible" environment.

Making GOLEM BDI-compliant requires to clearly map GOLEM concepts to the basic entities of the BDI model (beliefs, desires and intentions), but ensuring to keep the deliberation feature which, in GOLEM, is paramount.

A first thing to be noticed is that the concept of BDI *desire*, which can be expressed as "the things which an agent/robot would like to do", is quite similar to a GOLEM *goal*.

The concept of *intention* is instead something like "given a certain desire, an intention is what the agent/robot intend to do in order to meet that desire". According to the BDI model, intentions are selected from desires on the basis of the beliefs, i.e. on the basis of reasoning aspects and the knowledge the agent/robot has on itself and the environment. By making a comparison with a GOLEM system, an intention is indeed a *candidate goal* for execution and thus one element of the set computed at line 4 of Algorithm 1. It should be also noticed that, according to the BDI model, intentions represent a *possible set of options* to fulfill a certain desire; one of such options is then selected to be executed. This behaviour is similar to that of goal selection in line 6 of Algorithm 1, but is also well represented by a composite goal with **AT_LEAST(1)** or **SEQ_UNTIL** relationships. Indeed, the former relationship models the case in which several sub-goals are *alternative options* to achieve a certain main goal: one option (intention) may be selected to make the goal (desire) successful, but, if it fails, other alternatives are still possible. The latter relationship

[3] One objective were to have a small-sized system running on microcontrollers.

$$
\begin{array}{rcl}
P & ::= & (mg, mp) \\
mg & ::= & g \\
mp & ::= & \{lf_1, \ldots, lf_n\} \\
g & ::= & sg \mid cg \\
sg & ::= & (f, p, act) \\
f & ::= & bel_1 \text{ “,” } \ldots \text{ “,” } bel_n. \\
p & ::= & functor \rightarrow \mathcal{N} \\
act & ::= & a_1 \text{ “,” } \ldots \text{ “,” } a_n \\
a & ::= & \mathsf{A}(t_1, \ldots, t_n) \mid \text{“+” } bel \mid \text{“-” } bel \mid st \\
st & ::= & \mathtt{ACHIEVED}(sg) \mid \mathtt{T\_FAIL}(sg) \\
 & & \mid \mathtt{P\_FAIL}(sg) \\
cg & ::= & (rel, g\_set) \\
rel & ::= & \mathtt{ALL} \mid \mathtt{ALL\_SEQ} \mid \mathtt{AT\_LEAST}(k) \\
 & & \mid \mathtt{SEQ\_UNTIL} \\
g\_set & ::= & \{g_1, \ldots, g_n\} \\
lf & ::= & bel \text{ “:-” } bel_1 \text{ “,” } \ldots \text{ “,” } bel_n. \\
bel & ::= & \mathsf{B}(t_1, \ldots, t_n)
\end{array}
$$

Fig. 2. Basic Syntax of GOLEM-BDI

models a set of alternative options as well, but adds an *order of precedence/preference*: first try sub-goal 1, but if it fails, try sub-goal 2, and so on.

The last concept of the BDI model is the *belief*, which is used to model the knowledge of the agent/robot. A similar concept is however not present in GOLEM but is highly demanded in order to make the framework BDI-compliant. Beliefs not only must be properly represented, but also be manipulated by GOLEM goals in order to (*i*) perform checks on certain knowledge; (*ii*) add or remove knowledge on the basis of the evolution of the program; (*iii*) infer new knowledge by using reasoning.

While there are many alternatives to model beliefs, the most widely accepted approach implies the use of logic programming, and this is what we used to in *GOLEM-BDI*, the BDI-compliant extension of GOLEM.

GOLEM-BDI includes the goal model of GOLEM adding the expression and manipulation of beliefs. As it is sketched in Figure 2, which reports the basic syntax of GOLEM-BDI, a program P is composed by a *main goal* (mg) and a set of *logic predicates* (mg). The latter is a set of Prolog-style first-order logic predicates (lf) each one representing an implication and thus used to perform reasoning on the knowledge. Each predicate is expressed using one or more *beliefs* (bel) which are represented as atomic formulae with one or more variables/parameters.

Beliefs, which at run-time are supposed to be stored in a knowledge base, are manipulated as follows:

- Through pieces of code which poll the environment or robot state by using proper *sensors*[4];
- As a result of a reasoning process, according to a Prolog-style predicate (lf);
- By using an explicit *assert* or *retract* command in the sequence of statements placed in the goal's action part.

[4]Such a piece of code is intended to be *outside* the context of a GOLEM program.

As for goal structure, both simple and composite goals are expressed as in GOLEM provided the following modifications.

First of all the feasibility f function uses a first-order logic predicate on the beliefs present in knowledge base. A goal is thus feasible if the beliefs expressed in the predicate, which may also include constraint on parameters, are present in the knowledge base.

Secondly, the action act, which in GOLEM is not specified, here is expressed as a list of commands which can be:

- executing an *atomic* action, that is an explicit command to e.g. drive a robot's actuator (robot motion, arm movement, etc.);
- asserting or retracting a certain belief, in order to update the knowledge according to the evolution of the robot state;
- specifying the outcome of the goal execution by using one of the special beliefs in st, i.e. ACHIEVED, T_FAIL or P_FAIL.

Obviously, the GOLEM-BDI system described here is an abstract framework. To make it concrete, an implementation is needed in a proper programming language. In this sense, while a logic-based paradigm seems the most appropriate, such a choice is not mandatory and any other approach can be chosen, given that the semantics of the program constructs and execution is respected. This implementation activity will be performed in future work.

IV. Conclusions and Workshop Proposal

This aim of this paper is the description of GOLEM-BDI, a *belief-desire-intention* abstract framework for autonomous systems. The framework derives from GOLEM, which is goal-based framework specifically designed by the authors to program autonomous robots. GOLEM is extended in order to include the basic concepts of the BDI model, and, more precisely, the *belief* abstraction, which is then properly interfaced to all of the GOLEM mechanisms for goal selection and execution. While GOLEM has been successfully implemented and tested on some robots, GOLEM-BDI is currently under developed and it will be subject of future work.

References

[1] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*. Wiley, 2007.

[2] F. Messina, G. Pappalardo, and C. Santoro, “A Goal-centric Framework for Behaviour Programming in Autonomous Robotic Systems,” in *10^{th} IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications - MESA2014*. IEEE, 2014.

[3] ——, “Designing Autonomous Robots Using GOLEM,” in *XV Workshop “Dagli Oggetti agli Agenti” - WOA2014*. CEUR-WS, 2014.

[4] A. Rao, “AgentSpeak (L): BDI agents speak out in a logical computable language,” *Lecture Notes in Computer Science*, vol. 1038, pp. 42–55, 1996.

[5] A. Rao and M. Georgeff, “BDI agents: From theory to practice,” in *Proceedings of ICMAS*. San Francisco, CA, 1995, pp. 312–319.