High Level Programming Tools for Robotic Interaction Protocols: a Logic Programming Approach

Mariano Tucat and Alejandro J. García

Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering, Universidad Nacional del Sur,
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
{mt, ajg}@cs.uns.edu.ar

Summary. In this paper we consider the problem of programming autonomous mobile robots in a Multi-Robot System. We focus on the development of heterogeneous multi-robot systems, in which each robot is autonomous and cooperates in order to achieve a shared task. We propose to design Multi-Robot Systems in a way in which each robot decides the best action for itself but also considers other robots behavior. The way of designing the robot behavior tries to maximize the benefit of using communication. We describe how to implement an interaction protocol that allow robots to decide which is the best action to execute. In order to facilitate the programming of the robots with the ability to communicate with each other, we use a set of primitives that allow them to interact easily. The interaction among the robots is carried out through FIPA ACL.

Key words: Multi-Robot Systems, Robot Programming, Interaction Protocols, Logic Programming

1 Introduction

In this paper we consider the problem of programming autonomous mobile robots in a Multi-Robot System (MRS). The goal of this paper is to use a Logic Programming framework to define interaction protocols for real robots. We focus our work on programming a group of autonomous robots that will cooperate in order to accomplish a specific task. For instance, the cleaning task problem represents an environment with this kind of specifications, that is, the problem of cleaning a room full of things by a group of robots. Another similar cooperative environment corresponds to the robotic soccer domain, in which a team of autonomous robots cooperate in order to perform a determined goal, *i.e.*, win the game.

In the literature, there exist different alternatives for designing and implementing the behavior of a MRS that range from centralized to distributed approaches. One alternative is to use centralized control. There, a central planner determines the different task or targets for all the robots (see [3, 9, 10]). Another possibility is to use a master-slave configuration (*i.e.*, [11]) in which one robot (the master) decides the best plan for each robot. This plan will try to maximize their efficiency and to avoid wasting resources. In this alternative, slaves robots are simple reactive agents that follow the orders of a master.

In this paper, instead, we focus on a distributed solution. We will consider the development of a heterogeneous MRS (*i.e.*, robots with different hardware or programmed with different languages or technics). We assume that robots are autonomous and cooperate in order to achieve a shared task. Thus, each robot should decide which action is better for the system, considering both the information perceived from the environment and other robots decisions. Hence, we propose to design a MRS in a way in which each robot decides the best action for itself but also considers other robots behavior. In our approach, the robots will interact with each other in order to exchange their intended plans and take into account the decisions made by other robots. We will focus on the communication they may use to coordinate with partners in order to avoid wasting resources.

In order to facilitate the programming of the robots with the ability to communicate with each other, we will use a set of primitives that allow them to interact easily. This set of primitives (see [6] for details) was designed and implemented as an extension of Logic Programming, since this language is widely adopted for the development of intelligent agents. These primitives provide a reliable way for programming communicative agents without dealing with low-level details such as the actual location of an agent, an IP address or a machine name. They also allow the use of standard Agent Communication Languages, such as FIPA ACL [5] or KQML [7], and provides tools for developing standard Agent Conversation Protocols [2, 8].

In this work we consider a physical robot agent as a composition of two basic elements: (i) a robot with fundamental sensing, navigation and locomotion capabilities, (ii) a software agent with fundamental cognitive capabilities, such as problem solving and interaction. In our approach, the interaction among the robots will be carried out through FIPA ACL.

This paper is organized as follows. Section 2 describes an application domain that will be used for examples in this paper and proposes a way of improving cooperation in a MRS using explicit communication among robots. Section 3 shows how to implement interaction protocols for cooperative robots in a high level programming language. Finally, in Section 4 conclusions and related work are included.

2 Motivation

As mentioned above, in this paper we consider the development of heterogeneous MRSs, in which each robot is autonomous and cooperates in order to achieve a shared task. For instance, consider a simple situation of the cleaning task problem where several robots (with partial vision) have to transport scattered boxes to a particular place, minimizing the motion of each robot. Therefore, each robot will try to find the nearest box and pick it. Figure 1(a) shows a possible scenario of this domain where there are three robots (R1, R2 and R3) and three boxes (B1, B2 and B3). Note that the partial vision of each robot is depicted as a dashed area. In this situation the best solution is that R1 picks B1, R2 picks B2 and R3 picks B3. However, without communication, R1 will be the only robot that see B2 and it will pick this box whereas R2 and R3 will keep looking for boxes.

We will show next that using communication robots may arrive to better solutions. For instance, one alternative is that each robot broadcast its perception (in our domain, its location and the boxes it can see). Then, at the moment of deciding which action to execute, all robots will have the same information about the environment. Thus, each robot may try to figure out which decision will take each other robot in order to obtain the better action it may execute. However, determining the possible decision taken by each robot requires a great amount of processing. Also note that there is no guarantee that the predicted decisions will be correct, since robots can be heterogeneous.

In this work we propose another alternative that tries to maximize the benefit of using communication. In our approach, each robot will decide which is the best action to execute next by using its own perception. Then, the robot will interact with its partners and it may reconsider its decision, taking into account the decisions made by the other robots. This interaction allows the agents to exchange relevant knowledge also reducing the complexity of the decision process of each robot. Also note that the information about other robots decisions will be accurate because it is a cooperative system and decisions will be informed by direct communication.

Following our example of the cleaning task problem, whenever a robot finds a box, it will interact with its partners in order to determine which one should pick the box. Thus, this interaction not only informs the others robots about the existence of the box, but also helps them decide which robot may pick the box. Another possible domain in which this kind of interaction may have place is in the robotic soccer. Consider for example that, instead of a box, B1 in Figure 1(b) is the ball, and that the robots are teammates playing a soccer game (again with partial vision). Here, R1 may interact with R2 in order to inform it the ball location and to determine which one will try to catch the ball.

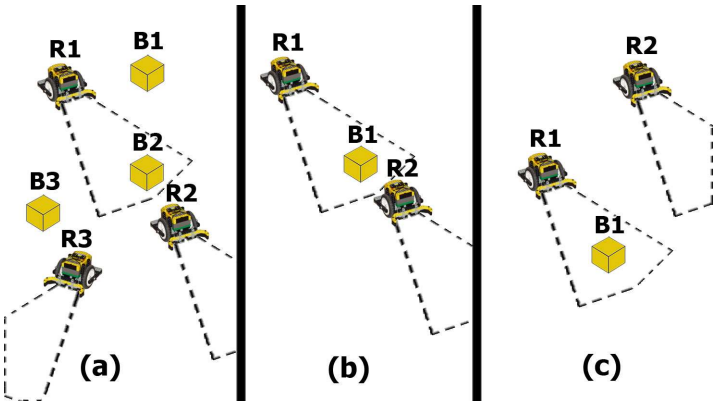

Fig. 1. Cleaning Task Problem: some possible scenarios

3 Implementing Interaction Protocols

In this section we will describe how to implement an interaction protocol for the cleaning task problem that allow robots to decide which is the best action to execute next. In the selected interaction protocol (see Figure 2), a robot R1 (the initiator) proposes to its partners (the participants) that it will pick a determined box, providing both the box location and the distance between it and the box. Thus, a robot receiving the proposal may agree with the initiator (R1) allowing R1 to pick the box, or it may issue a counter-proposal, indicating that it is nearer to the box, and thus, that it should pick the box. In the latter case, the initiator of the interaction (R1) should decide which robot will pick the box and it will inform its partners accordingly.

Consider for example the situations in Figures 1(b) and 1(c). In both situations, following the proposed protocol, since R1 sees B1, it will interact with R2, informing the box location and that it will pick B1 unless R2 has a better proposal. Observe that in the situation of Figure 1(c), R2 will agree with R1 since R1 is nearer, whereas in Figure 1(b) R2 will make a counter proposal. In the latter case, R1 will decide that R2 should pick the box.

In the situation of Figure 1(a) there are three robots and again R1 is the only one that sees a box. Here, R1 will send its proposal of picking B2 to R2 and R3. Then, both robots will send a counter-proposal to R1 and since R2 is the nearest robot to B2, then R1 will decide that R2 should pick it sending the corresponding messages to R2 and R3.

In [6] we have proposed a set of primitives that facilitates the implementation of communicative robots. Figure 3 shows a subset of these primitives that allow both sending messages and retrieving the received ones. The primitive *send* allows the exchange of meaningful information facilitating the use of standard Agent Communication Languages. The received messages are automatically queued to be processed whenever the robot decides. The primitives

Fig. 2. A simple interaction protocol for the cleaning task problem

for retrieving the received messages allow the robots to wait for specific messages (*receive*) and also to associate (*bind*) the arrival of specific messages with the execution of a particular process.

Primitive:	Brief description:
send	Sends a message to other agents
receive/3	Waits for a message from another agent
receive/4	Waits for a message for a given period of time
bind	Binds the arrival of specific messages to the call of a predicate
unbind	Unbinds the association already made

Fig. 3. A brief description of the selected primitives taken from [6]

The code in Figure 4 shows how a robot (in the previous examples R1) may implement the initiating part of the protocol using the primitives mentioned above. Executing the predicate $init\_protocol(Robots, B, D, Result)$, a robot will send to it partners ($Robots$) the proposal that it will pick a box in a given location (B), also informing its distance (D), and the variable $Result$ will return which robot may pick the box. The predicate *init_protocol* will send the appropriate messages and then, it will collect all the answers ($collect\_answers/3$). The predicate $decide/4$ determines which robot should pick the box, based on which of them is nearer to the box. That is, if all the participants of the protocol agree, the robot that should pick the box will be the initiator (the list of $Refused\_Robots$ will be empty). Whenever any robot

sends a counter proposal, this predicate compares all the proposals including the one made by the initiator of the interaction. Finally, the refusal answers are sent to those robots that made a counter proposal and were not selected to pick the box. In the case that the robot selected to pick the box is different from the initiator, the corresponding agree message is sent.

```
init_protocol(Robots, box(X,Y), distance(D), Result):- Box=box(X,Y),
  send(Robots, [sender(r1), communicative_act(propose), conv_id(ID),
       ontology(cleaning_task), content(pick(r1,Box,distance(D)))]),
  collect_answers(Robots, ID, Partner_Answers),
  decide(Partner_Answers, Decision, Selected_Robot, Refused_Robots),
  inform_others(Decision, Selected_Robot, Refused_Robots, ID),
  Result = pick(Selected_Robot, box(X,Y)).

inform_others(Decision, self, Refused_Robots, ID) :-
  send(Refused_Robots, [sender(r1), communicative_act(refuse),
        ontology(cleaning_task), content(Decision), conv_id(ID)]).
inform_others(Decision, Selected_Robot, Refused_Robots, ID) :-
  send(Selected_Robot, [sender(r1), communicative_act(agree),
           ontology(cleaning_task), content(Decision), conv_id(ID)]),
  send(Refused_Robots, [sender(r1), communicative_act(refuse),
           ontology(cleaning_task), content(Decision), conv_id(ID)]).
```

Fig. 4. Sample code for a robot initiating the IP shown in Fig. 2

Figure 5 shows one way in which a robot (for example R2) may implement the participant part of the protocol. The robot associates the arrival of proposals to the execution of the predicate $consider\_action(R, B, D, ID)$, thus, whenever the robot receives a message with a proposal, the mentioned predicate will be automatically called. In this predicate, the robot decides whenever it may agree with the proposal or send a counter proposal. Therefore, the robot determines the distance between itself and the box, and in the case that it is nearer than the initiator of the interaction, it will send a counter proposal. After that, it waits for the answer of its counter proposal and depending on the decision made by the initiator, it will pick the box or it will search for a new box ($new\_action/2$).

As shown in Figures 4 and 5, we use standard FIPA ACL in the communication among robots. There exist proposals in the literature in which the messages the robots exchange are defined ad-hoc. Although ad-hoc approaches facilitates the programming of the robots, these approaches forbid the interaction among heterogeneous robots.

In order to allow the interaction among heterogeneous robots we decided to use FIPA ACL and also to define the interaction protocols following FIPA Standards. Standard agent communication languages (ACLs), like FIPA ACL, allow agents to effectively communicate and exchange knowledge with other

```
:- bind(Robot,[sender(Robot),communicative_act(propose),conv_id(ID),
           ontology(cleaning_task), content(pick(R,B,distance(D)))],
        consider_action(R,B,D,ID)).

consider_action(Robot, Box, Its_Distance, ID) :-
  calculate_distance(Box,My_Distance), My_Distance < Its_Distance,
  send(Robot,[sender(r2), receiver(Robot), ontology(cleaning_task),
              communicative_act(counter-proposal), conv_id(ID),
              content(pick(r2, Box, distance(My_Distance)))]),
  receive(Robot,[sender(Robot),receiver(r2),ontology(cleaning_task),
         communicative_act(Answer), conv_id(ID), content(Content)]),
  new_action(Answer, Box).

consider_action(Robot, Box, Its_Distance, ID) :-
  send(Robot,[sender(r2), receiver(Robot), communicative_act(agree),
              ontology(cleaning_task), conv_id(ID),
              content(pick(Robot,Box,distance(Its_Distance)))]).

new_action(agree, Box) :- do_action(pick(Box)).
new_action(refuse, _ ) :- do_action(search_for_new_box).
```

Fig. 5. Sample code for a robot participating in the IP shown in Fig. 2

agents despite differences in hardware platforms, operating systems, architectures, programming languages and representation and reasoning systems.

4 Related Work and Conclusions

In this paper we have considered the problem of programming autonomous mobile robots in a Multi-Robot System. We focus on the development of heterogeneous multi-robot systems, in which each robot is autonomous and cooperates in order to achieve a shared task. We have proposed to design Multi-Robot Systems in a way in which each robot decides the best action for itself but also considers other robots behavior. The way of designing the robot behavior tried to maximize the benefit of using communication. We have described how to implement an interaction protocol that allows robots to decide which is the best action to execute. In order to facilitate the programming of the robots with the ability to communicate with each other, we used a set of primitives that allow them to interact easily. The interaction among the robots was carried out through FIPA ACL.

There are numerous contributions concerning decentralized cooperation among multiple mobile robots. In [1], a decentralized approach for the conflict-free motion of multiple mobile robots operating in a common 2D workspace is introduced. The authors propose a framework for cooperation in which the robots interact in order to coordinate their motions and to resolve local con-

flicts. In contrast with our approach, in order to accomplish the cooperation, they define a minimum set of four messages, that is, they define ad-hoc messages for their domain. As mentioned above, this facilitates the programming of the robots but also precludes the interaction among heterogeneous robots.

Another approach is presented by [4], corresponding to a distributed architectural framework that enables multiple physical robot agents to coordinate high-level tasks in a collaborative manner. Similar to us, they use interaction to solve interdependency problems, such as conflict of interest and collision avoidance issues. They also use message passing based on FIPA-ACL to the robot interaction. In contrast with our approach, they use JADE, a Java Agent Development Framework, in order to implement the communication among agents. Our framework is based on Logic Programming (LP), and therefore sophisticated Knowledge Representation and reasoning formalisms developed for LP can be easily used.

References

1. K. Azarm and G. Schmidt. Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. Robotics and Automation. Proceedings., 1997 IEEE Int. Conference on, Vol.4, Iss., 20-25 Apr, 1997.
2. R. Scott Cost, Yannis Labrou, and Timothy W. Finin. Coordinating agents using agent communication languages conversations. In *Coordination of Internet Agents: Models, Technologies, and Applications*, pages 183–196. 2001.
3. Michael Erdmann and Tomas Lozano-Perez. On multiple moving objects. IEEE Int. Conference on Robotics and Automation, San Francisco, April 7-10, 1986.
4. J. Eze, H. Ghenniwa, and W. Shen. Distributed control architecture for collaborative physical robot agents. Systems, Man and Cybernetics, 2003. IEEE International Conference on, Vol.3, Iss., 5-8 Oct., 2003.
5. FIPA. Foundation for intelligent physical agents. http://www.fipa.org.
6. Alejandro J. García, Mariano Tucat, and Guillermo R. Simari. Interaction Primitives for Implementing Multi-agent Systems. In *VII Argentine Symposium on Artificial Intelligence*, Rosario, Argentina, August 2005.
7. KQML. Knowledge Query and Manipulation Language. Oficial Web Page: http://www.cs.umbc.edu/kse/kqml.
8. Francisco J. Martín, Enric Plaza, and Juan A. Rodríguez-Aguilar. Conversation protocols: Modeling and implementing conversations in agent-based systems. In Frank Dignum and Mark Greaves, editors, *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 249–263. Springer, 2000.
9. P. Svestka and M. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. Robotics and Automation. Proceedings., 1995 IEEE International Conference on, Vol.2, Iss., 21-27 May, 1995.
10. C.W. Warren. Multiple robot path coordination using artificial potential fields. Robotics and Automation. Proceedings., 1990 IEEE International Conference on, Vol., Iss., 13-18 May, 1990.
11. S. Yuta and S. Premvuti. Coordinating autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots. Intelligent Robots and Systems, Proceedings of the 1992 lEEE/RSJ International Conference on, Vol.3, Iss., 7-10 Jul, 1992.