Dynamic Configuration of a Team of Robots

Robert Lundh and **Lars Karlsson** and **Alessandro Saffiotti**[1]

Abstract. We study teams of autonomous robotic agents in which agents can help each other by offering information-producing resources and functionalities. Depending on the current situation and tasks, the team may need to change its functional configuration, that is, which agents provide which functionalities to whom. We propose to use knowledge-based techniques to automatically synthesize new team configurations in response to changes in the situation or tasks. This note summarizes our approach, and reports our preliminary steps in this direction.

1 Introduction

Consider a society of autonomous robotic systems embedded in a common environment. By an *autonomous robotic system* we mean here any computer-controlled system able to sense the environment, take decisions about actions to perform in the environment, and perform those actions. These include mobile robots, like the one pictures in Fig. 1, as well as simpler devices like domestic appliances or monitoring apparatuses. We do not assume that the systems in the society are homogeneous: they may have different sensing, acting, and reasoning capacities.

From an abstract point of view, this society can be seen as one distributed robotic system. The system usually includes a number of *functionalities* organized in some way, for instance, in a generic two-layer hybrid architecture like the one shown in Fig. 2 (left). In these architectures, the top layer implements higher cognitive processes for world modeling (M) and for planning and deliberation (D). The bottom layer implements sensori-motor processes for sensing and perception (P) and for motion control (C), which are connected to a set of sensors (S) and actuators (A).

In practice, the above functionalities can be distributed across different physical units in the society (robots, devices, etc). Each unit includes several functionalities in each one of the {P, M, D, C, S, A} classes, which it can use to perform the tasks assigned to it. In addition, each unit may use functionalities from other units in order to compensate for some one that it is lacking, or to improve its own. Consider for example the following scenario involving a pair of outdoor robots, A and B, equipped with pan-tilt stereo cameras. Robot A needs to perform the action to cross a gate in a metalic fence, as shown in Fig. 1. To do so, it must have a P functionality to measure the relative position and orientation of the gate, since this information is needed by the controller. Robot A can use its stereo camera to observe the edges of the gate during the crossing, but the measure obtained when these edges are near is not very reliable. Robot B, however, could observe the entire scene from a distance and compute a better estimate of the relative position and orientation between robot A and the gate. Robot B can therefore offer this functionality to A so that A can perform its task more reliably — see Fig. 2 (right).

In general, the same task can be performed by using different functionalities in different robots and connected in different ways. For example, the previous gate-crossing task can be achieved by either (1) connecting the camera functionality in A to the gate-crossing functionality in A, or (2) connecting the camera functionality in B to the gate-crossing functionality in A. We informally call *configuration* any way to allocate and connect the functionalities of a distributed multi-robot system. Note that we are interested in functional software configurations, as opposed to the hardware configurations usually considered in the field of reconfigurable robotics (e.g., [7, 12]). Clearly, which configuration should be preferred depends on the task, situation and resources. This suggests that the system should be able to switch to a new configuration whenever these conditions change.

The focus of our work is to study configurations of a society of robotic agents. Our objective is threefold:

1. To define the concept of functional *configuration* of a robot society: which robot is providing which functionalities to which one, and how.
2. To study how to *dynamically change* the configuration of a robot society in response to changes in the environment, in the tasks, or in the available resources.
3. To use knowledge-based techniques (e.g., planning and monitoring) to automatically *detect* when a configuration is not adequate any more, and *synthesize* a new one for the current situation.

Figure 1. An outdoor robot about to cross a gate in a fence.

[1] Center for Applied Autonomous Sensor Systems, University of Örebro, Sweden. {robert.lundh, lars.karlsson, alessandro.saffiotti}@aass.oru.se

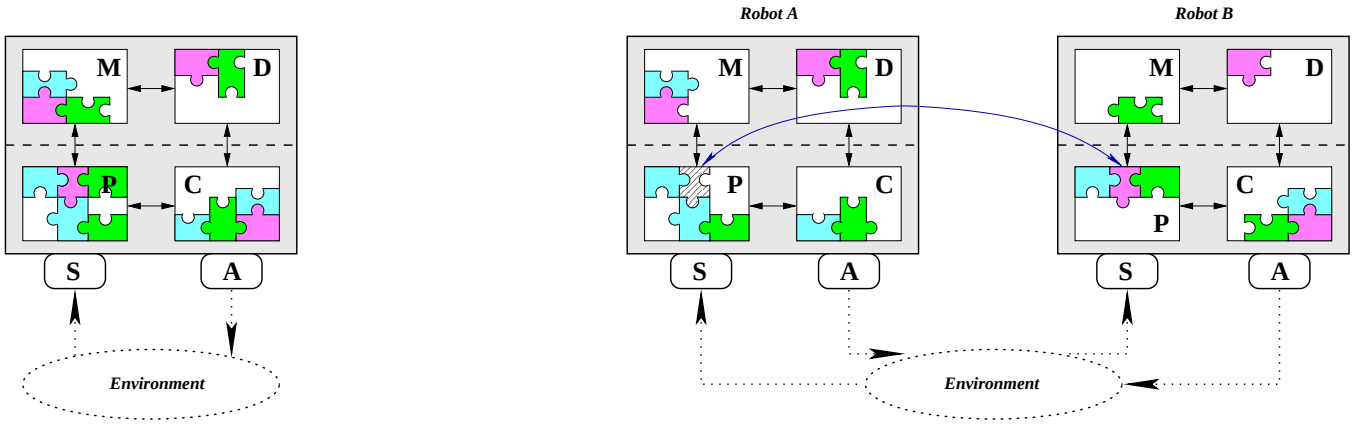

Figure 2. Left: abstract view of a team as a distributed robotic system. Right: a simple team configuration consisting of two-robots: Robot B is providing a missing perceptual functionality to Robot A.

At this initial stage of our work, we focus on the first objective: to define a concept of configuration which is adequate from the point of view of the other two objectives above. The rest of this paper outlines our first steps in this direction.

2 Related work

There are several relevant areas from which one might take inspiration to address the above objectives. In the area of multi-robot systems, much work has been done on the problem of multi-robot task allocation, that is, how to allocate a number of tasks to a number of robots taking into account that different robots may be differently adequate for different tasks (see, e.g., [9] for an overview and analysis). Some examples are the ALLIANCE architecture [13] and Local Eligibility approach [25] based on local utility estimates, and the M+ [2] and MURDOCH [8] approaches. Closely related to task allocation are the issues of robotic team configuration and of dynamic role assignment [23, 21, 11]. Chaimowicz *et al* [3] consider roles as the part of an individual agent in a cooperative task. They define a role as a control mode in a hybrid automaton, and a role assignment is a transition in that automaton. The approach that we propose in this paper departs from the above works since we focus on the distribution and — in particular — the interconnection of atomic (action and perception) functionalities. These are combined to form behaviors, which achieve tasks.

The problem of distributing the performance of a task across a number of agents according to their respective capabilities has been widely addressed in the Distributed AI (DAI) and in the Multi-Agent Systems (MAS) communities. Early work in DAI considered distributed problem solving settings with a precedence order among sub-tasks [6]. Later work has included the notion of coalitions between sub-groups of more closely interacting agents [17]. The notions of team-work [14], capability management [22] and norms [1] have also been used in the MAS community to account for the different forms of interactions between the sub-tasks performed by the agents in a team. These works, however, typically assume software agents, and are not concerned with issues of physical action, mobility, and perception, which play a central role in our work.

Another area of interest is program supervision, where program modules are combined, tuned and evaluated in order to solve specific computational tasks such as image processing, often using planning techniques [10, 4, 18]. Our work adds several dimensions to program supervision since we deal with multiple physical agents with both sensing and acting capabilities.

In a paper more similar in spirit to this one, Simmons et al [20] consider a task involving a heterogeneous team of robots — a crane, a robot with a manipulator, and a robot with stereo cameras — solving a construction task where a beam is placed on top of a stanchion. This task requires tight cooperation between the robots involved. Cooperation between the robots is hand-coded, although the authors declare their intention to use planning techniques to set up the cooperation. For specifying tasks, they use TDL (task description language) [19], an imperative language which is a superset of C++. This language does not appear to be adequate for automatic reasoning about configurations by, e.g., a planner.

3 Framework

The first goal in our research program is to develop a definition of configuration that is adequate for the three objectives presented in the introduction. In general, a configuration of a team of robots may include interconnected functionalities of two types: information providing functionalities, that is, functionalities that change the internal state by providing or processing information; and action executions, that is, functionalities that change the state of the environment. (Some functionalities can have both aspects.) In the work presented here we focus on the information providing functionalities, since these are a less studied aspect in the planning literature. The extension of our framework to include action functionalities is left as a second step.

To define our notion of configurations, a clarification of the three concepts of functionality, resource and channel is in order.

3.1 Functionality

A *functionality* is an operator that uses information provided by other functionalities to produce additional information. Each instance of a functionality is located in a specific robot (or other agent). The functionality consists of:

- a specification of inputs, to be provided by other functionalities. For each input, it contains information about domain (e.g. video images) as well as timing information (e.g. every 100 ms).

- a specification of outputs, to be provided for other functionalities. They also contain domain and timing information.
- a specification of relations between inputs to outputs.
- a set of causal preconditions, that is conditions in the environment that have to hold in order for the functionality to be operational.
- a set of causal postconditions, that is conditions in the environment which the functionality is expected to achieve.
- possibly also a specification of costs in terms of e.g. computation and energy.

3.2 Resource

A *resource* is a special case of a functionality. There are two different types of resources: sensing resources and action resources. As mentioned previously, only sensing resources will be considered in this paper. A *sensing resource* has no input from other functionalities, and is typically a sensor that gives information about the current state of the surrounding environment (e.g., a camera) or perhaps information about the internal state of the robot.

3.3 Channel

A *channel* transfers data from one functionality to another. A channel can be in terms of either inter-robot or intra-robot communication, and be on different mediums (radio, network, internal connections). A channel may have requirements of band width, speed and reliability.

3.4 Configuration

A *configuration* is the set of functionalities and the set of channels that connects functionalities to each other. Each channels connects the output of one functionality to the input of another functionality.

In the context of a specific world state, a configuration is *admissible* if the following conditions are satisfied:

- each input of each functionality is connected via an adequate channel to an output of another functionality with a compatible specification (information admissibility).
- all preconditions of all functionalities hold in the current world state (causal admissibility).
- the combined requirements of the channels can be satisfied.

3.5 Examples

In order to illustrate the above concepts, we consider a concrete example inspired by the scenario described in the introduction. In order to more easily test the example on real robots (see next section), we consider an indoor office building. A robot is assigned the task of moving from one room to another one by crossing a door between the two rooms. The "cross-door" action requires information about position and orientation of the door with respect to the robot performing the action. The resources available are two indoor robots (including the one crossing the door) each one equipped with a camera and a compass. The door to cross is equipped with a wide-angle camera.

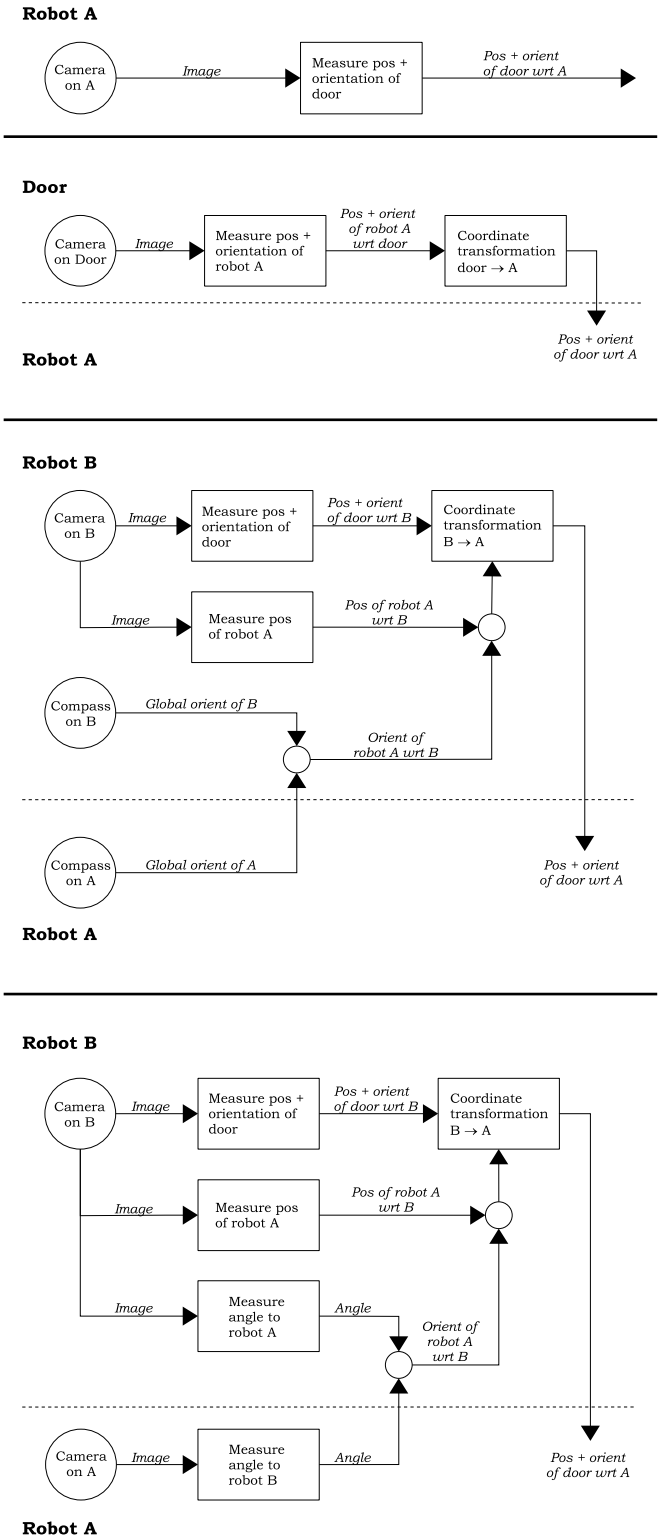

Figure 3. Four different configurations that provide the position and orientation of a given door with respect to robot A. See explanation in the text.

Two functionality operators from this scenario are shown below:

```
(Op Measure_Door (?Y)
    Inputs:    Image (?X)
    Outputs:   Door position & orientation
               of ?Y relative to ?X
    Preconds:  Door ?Y fully visible
               in image from ?X
    Postconds: -
    Transform: measuring door procedure
    )

(Op Camera (?X)
    Inputs:    -
    Outputs:   Image (?X)
    Preconds:  CameraOn
    Postconds: -
    Transform: image retrieval procedure
    )
```

The input and output of a functionality represent the data flow associated with the functionality. In the `Measure_Door` example we have an image taken by camera `?X` as input and from that we are able to compute the position and orientation of the door `?Y` relative to `?X` as output. The second example is an operator for a camera. Output from `Camera` is an image taken by camera `?X`. Since `Camera` is a sensing resource no input is specified. There are also certain conditions that need to be satisfied in order for the functionality to operate, and conditions that will be satisfied if the functionality is executed. This causal flow is represented as preconditions and post-conditions in the operator. For instance the precondition for `Measure_Door` is that the door `?Y` is fully visible in the input image and the precondition for `Camera` is that the camera is switched on. The body of the operator describes the computations performed on the input in order to generate the specified output provided that the preconditions are satisfied. Notice that the output of `Camera` matches the input of `Measure_Door`. Intuitively, this means that a channel between these two functionalities can legally be created.

Fig. 3 illustrates four different (admissible) configurations that provide the information required by the action "cross-door", which include the functionalities above.

The first configuration involves only the robot performing the action. The robot is equipped with a panoramic camera that makes it possible to view the door even during the passage. The camera produces information to a functionality that measures the position and orientation of the door relative to the robot.

The second configuration in Fig. 3 shows the other extreme, when all information is provided by the door that the robot is crossing and the robot is not contributing with any information. The door is equipped with a camera and functionalities that can measure the position and orientation of the robot relative to the door. This information is transformed into position and orientation of door with respect to the robot before it is delivered to robot A.

The third and fourth configurations in Fig. 3 consist of two robots (A and B), each with its own set of resources and functionalities.

In the third configuration, robot A (the robot performing the "cross-door" action) only contributes with one resource, a compass. Robot B's resources are a compass and a camera. The camera provides information to two functionalities: one that measures the distance and orientation to the door, and another one that measures the distance to robot A. All these measurements are computed relative to robot B. In order to compute the position and orientation of the door relative to robot A, we need to use a coordinate transformation. This in turn requires that we know the relative position and orientation of robot A relative to B. The relative position is obtained from the camera information. The relative orientation can be obtained by comparing the absolute orientations of the two robots, measured by their two on-board compasses.

The fourth configuration in Fig. 3 is similar to the third one, except that the orientation of robot A relative to B is obtained in another way, i.e., no compasses are used. Both robots are equipped with cameras and have a functionality that can measure the bearing to an object. When the robots are looking to each other, each robot can measure the bearing to the other one. By comparing these two measurements, we obtain the orientation of robot A relative to robot B.

4 A Simple Experiment

In order to test whether sharing of functionalities in different configurations would actually allow us to solve simple coordination examples, we have conducted a series of experiments using a pair of real robots equipped with different sensors. These experiments were also aimed at assessing the mechanisms for the switching between configurations. In these first experiments, the configuration generation and configuration switches were hand-coded. We intend to eventually make both aspects automatic.

We present here a simple experiment based on the third and fourth configurations in Fig. 3. The platform used were two Magellan Pro robots from iRobot, shown in Fig. 4. Each robot runs an instance of the layered hybrid architecture Thinking Cap [16].

Both robots are equipped with compasses and fixed color cameras. They have additional sensors (e.g., sonars, laser, and an electronic nose) not used in our experiments. Since the cameras are fixed, they can only measure distances to objects further away than 2 meters. The environment consists of two rooms (R1 and R2) with a door connecting them. The door and the robots have been marked with uniform colors in order to simplify the vision task (see Fig. 4).

The following scenario describes how the two configurations were used, and demonstrates the importance of being able to reconfigure dynamically. Robot A and robot B are in room R1. Robot A wants to go from room R1 to room R2. Since the camera can only measure distances to objects further away than 2 meters, robot A is not be able to perform the action on its own. Robot B is equipped with the same sensors as robot A, but since robot B is not crossing the door it is able to observe both the door and robot A from a distance during the whole procedure. We therefore configure our team according to the third configuration in Fig. 3, and execute the task. Robot A continuously receives information about the position and orientation during the execution of "cross-door".

When robot A enters room R1 it signals that the task is accomplished. This signal is received by robot B and the current configuration is played out. Next, robot B is assigned the task of going from room R1 to room R2. The same configuration as before is used to solve this task, but with the roles exchanged — i.e., robot A is now guiding robot B. This time, however, during the execution of the "cross-door" behavior a compass fails due to a disturbance in the magnetic field. This makes the current configuration not admissible, and a reconfiguration is necessary to proceed. The fourth configuration in Fig. 3 is still admissible even with no compass, and we therefore use this one to carry out the remaining part of the task. Fig. 5 shows the trajectories performed by the robots in a sample run of this experiment. In the picture, robot A is standing still at the observing position and robot B has just accomplished its task.

Figure 4. Robot B is guiding robot A through the door.

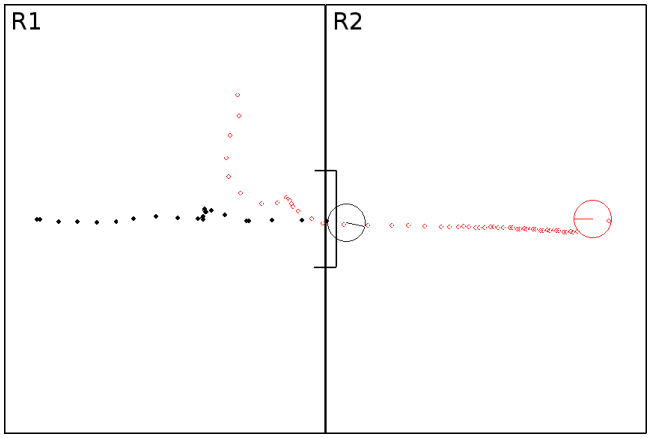

Figure 5. Robot A and B have both reached room R2. Circles show robot A's trajectory and dots show robot B's trajectory.

5 Conclusions

This paper has shown the first steps toward our goal to automatically synthesize a team configuration using knowledge-based techniques. Differently from most current works on team formation, our atomic unit of decomposition is not a task or a role, but a single functionality that a robot can make available to another one. The preliminary experiments indicate that we can achieve flexible forms of cooperations in this way. Moreover, we are able to describe information- and action-producing functionalities as abstract operators similar to the ones which are customary in planning systems. This suggests the possibility to build a system that uses planning techniques [24, 5] to automatically create team configurations given the current tasks, resources, and situation. We would like this system to be able to determine the most adequate configuration in terms of a set of criteria, like efficiency, reliability, or cost of resources and communication. Our research efforts are in this direction.

The distributed nature of our configuration is expected to be an important aspect in configuration planning. For instance, the cost and unreliability of inter-robot communications should be taken into account when evaluating alternative configurations. Moreover, configuration planning may have to be done in a distributed manner, or if centralized it must be preceded and followed by information exchanges between the robots.

Automatic re-configuration of a robotic team will be important as the task, environment and maybe also the composition of the team change dynamically. An important issue to consider here is how to decide when it is time to change configuration. A reconfiguration may be needed if the available functionalities change (e.g., due to malfunctioning), if the external conditions change, if the current task is completed, or if a new task is given. From an experimental perspective, we intend to apply automatic reconfiguration online on robots in increasingly complex and dynamic environments (e.g. 4-legged robotic soccer [15]).

ACKNOWLEDGEMENTS

This work was supported by the Swedish KK Foundation, the Swedish National Computer Graduate School in Computer Science (CUGS), and the Swedish Research Council.

REFERENCES

[1] G. Boella, 'Norms and cooperation: Two sides of social rationality', in *Agent Autonomy*, eds., H. Hexmoor, C. Castelfranchi, and R. Falcone, Kluwer, Boston/Dordrecht/London, (2003).
[2] S. Botelho and R. Alami, 'M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement', in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1234–1239, (1999).
[3] L. Chaimowicz, M. Campos, and V. Kumar, 'Dynamic role assignment for cooperative robots', in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 293–298, (2002).
[4] S. A. Chien and H. B. Mortensen, 'Automating image processing for scientific data analysis of a large image database.', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**(8), 854–859, (1996).
[5] M. E. desJardins, E. H. Durfee, C. L. Ortiz, Jr, and M. J. Wolverton, 'A survey of research in distributed continual planning', *AI Magazine*, **20**(4), 13–22, (1999).
[6] E.H. Durfee, V.R. Lesser, and D.D. Corkill, 'Coherent cooperation among communicating problem solvers', in *Readings in Distributed AI*, eds., A.H. Bond and L. Gasser, 268–284, Morgan Kaufmann, San Mateo, CA, (1988).
[7] T. Fukuda and S. Nakagawa, 'Approach to the dynamically reconfigurable robot systems', *Intelligent Robtics Systems*, **1**, 55–72, (1988).
[8] Brian P. Gerkey and Maja J Matarić, 'Sold!: Auction methods for multi-robot coordination', *IEEE Transactions on Robotics and Automation*, **18**(5), 758–768, (October 2002).
[9] Brian P. Gerkey and Maja J Matarić, 'Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures', in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, (May 2003).
[10] L. Gong and A.C. Kulikowski, 'Composition of image analysis processes through objectcentered hierarchical planning', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(10), (1995).
[11] J. S. Jennings and C. Kirkwood-Watts, 'Distributed mobile robotics by the method of dynamic teams', in *Proc. of the Intl. Symp. on Distributed Autonomous Robotic Systems (DARS)*, Karlsruhe, Germany, (1998).
[12] F. Mondada, M. Bonani, S. Magnenat, A. Guignard, and D. Floreano, 'Physical connections and cooperation in swarm robotics', in *Proc. of the 8th Int. Conf. on Intelligent Autonomous Systems (IAS8)*, pp. 53–60. IOS Press, (2004).
[13] L. Parker, 'ALLIANCE: An architecture for fault tolerant multi-robot cooperation', *IEEE Trans. on Robotics and Automation*, **14**(2), (1998).
[14] D.V. Pynadath and M. Tambe, 'Automated teamwork among heterogeneous software agents and humans', *Journal of Autonomous Agents and Multi-Agent Systems*, **7**, 71–100, (2003).
[15] A. Saffiotti, A. Björklund, S. Johansson, and Z. Wasik, 'Team Sweden', in *RoboCup 2001*, eds., A. Birk, S. Coradeschi, and S. Tadokoro, Springer-Verlag, Germany, (2002).

[16] A. Saffiotti, K. Konolige, and E. H. Ruspini, 'A multivalued-logic approach to integrating planning and control', *Artificial Intelligence*, **76**(1-2), 481–526, (1995).
[17] O. Shehory and S. Kraus, 'Methods for task allocation via agent coalition formation', *Artificial Intelligence*, **101**, 165–200, (1998).
[18] C. Shekhar, S. Moisan, R. Vincent, P. Burlina, and R. Chellappa, 'Knowledge-based control of vision systems', *Image and Vision Computing*, **17**, 667–683, (1998).
[19] R. Simmons and D. Apfelbaum, 'A task description language for robot control', in *Proceedings of the Conference on Intelligent Robotics and Systems*, Vancouver Canada, (October 1998).
[20] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith, 'First results in the coordination of heterogeneous robots for large-scale assembly', in *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Honolulu Hawaii, (December 2000).
[21] P. Stone and M. Veloso, 'Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork', *Artificial Intelligence*, **110**(2), 241–273, (1999).
[22] I. J. Timm and P-O Woelk, 'Ontology-based capability management for distributed problem solving in the manufacturing domain', in *Multiagent System Technologies – Proceedings of the First German Conference, (MATES 2003)*, eds., M. Schillo and et al., pp. 168–179. Springer Verlag, (September 2003).
[23] D. Vail and M. Veloso, 'Multi-robot dynamic role assignment and coordination through shared potential fields', in *Multi-Robot Systems*, eds., A. Schultz, L. Parker, and F. Schneider, Kluwer, (2003).
[24] D. S. Weld, 'Recent advances in AI planning', *AI Magazine*, **20**(2), 93–122, (1999).
[25] B. B. Werger and M. J Matarić, 'Broadcast of local eligibility for multi-target observation', in *Distributed Autonomous Robotic Systems*, eds., L. E. Parker, G. Bekey, and J. Barhen, pp. 347–356. Springer Verlag, (2000).