Conflict Management in Multiagent Robotic System: FSM and Fuzzy Logic Approach

Witold Jacak* and Stephan Dreiseitl* and Karin Proell* and Jerzy Rozenblit**

* Dept. of Software Engineering, Polytechnic University of Upper Austria, Hagenberg, Austria
** Dept. of Computer Engineering, University of Arizona, Tucson; USA

1 Introduction

Intelligent agents are a new paradigm for developing complex system applications. The most powerful tools for handling complexity in system development are modularity and abstraction. Agents represent a powerful tool for making systems modular. Industrial application of agent technology were among the first to be developed, in such domain as process control, manufacturing and robotic systems. The agent systems are based on autonomous software-hardware components (technical agents) that cooperate within an environment to perform some task. An agent can be viewed as a self-contained, concurrently executing thread of control that encapsulates some states and communicates with its environment and possibly other agents via some sort of message passing.

The environment may contain other agents whose environments are disjoint with or only partially overlap with the environment of a given agent. In this paper we present the conflict management problem in a multiagent robotic system. The multiagent robotic system consists of the group of autonomous agents on the first (lower) level of its hierarchy, and the management agents on the second (upper) level of the hierarchy [1,2,6,7,8].

A multiagent system has a hierarchical structure. This system is the group of cooperating agents coordinated by two managers, the *contract* and *conflict* managers. On the lower level, the robotic system consists of a group of independently acting robotic agents. Each of the robotic agents is under control of an intelligent software subagent. The intelligence of the software agents comes from the ability to solve subproblems locally and to propose a global solution as a result of communication and/or interactions between different agents. This group of agents constitutes the lower level of the multi-agent robotic system, i.e. the operating level [4,6].

The main goal of a multiagent system is to solve a common task that is split up into several subtasks, which in turn are distributed to the individual robotic agents by the contract manager. Task distribution on the one side and task execution on the other side require two additional agents for cooperation and negotiation.

The contract management agent considers task distribution when a new job enters the system. The contract manager has to direct autonomous agents by specifying individual goals (subtasks) for each of them [11,12,13,14].

The conflict management agent is responsible cooperation and negotiation while each agent performs its assigned subtask. As the individual behavior of all robotic agents involved in task achievement cannot be predicted in advance, the goals of two or more robotic agents can be in direct conflict with one another and a given task might not be completed. In order to resolve a conflict situation, a negotiation process among all conflict parties has to take place. The result of the negotiation should be a solution that results in goal achievement for all involved agents.

Contract manager and conflict manager together form the *management level*, the higher level of the multi-agent system.

In this paper we focus on the second unit of conflict management methods in a robotic system and present solutions for detecting and resolving conflicts between two or more robotic agents [3,4,5,6]. A multi-agent robotic system is presented in Fig. 1.

0-7803-7087-2/01/$10.00 © 2001 IEEE

2 Robotic Agent

A robotic agent consists of many components that are grouped in a layered architecture. This architecture comprises four layers: the action execution layer (hardware component), the behavior-based layer (safety protection component), the planning

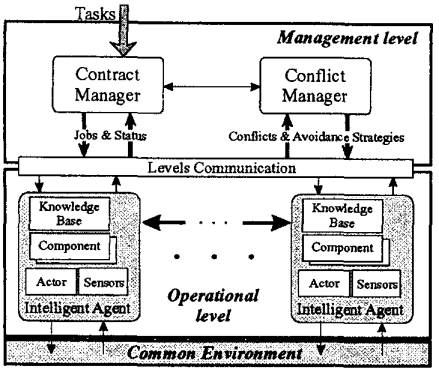

Fig. 1: Components of a multiagent system

layer (action planning component), and the cooperative planning layer (negotiation and communication component) [4,6,9]. In this article we focus our presentation on robotic agents whose hardware components are represented by sensor-equipped robots. Each agent has a knowledge base, which is a set of models and methods needed for decision-making.

The typical knowledge base contains different models and their formal representations: a *world* model, a *social* model, a *mental* and a *self*-model. The world model represents knowledge about the surrounding environments of an agent, the social model represents knowledge about other agent acting in the system, the mental model represents the knowledge about the reactive behavior and risks associated with actions, and the self model contains the knowledge about construction, properties and structure of the agent's hardware component, for example a robot's manipulator.

The reactive part of the agent is implemented by the behavior-based layer, i.e. the safety protection component, which contains a set of situation-action rules. These describe the agent 's reactive skills for implementing fast situation recognition in order to react to time-critical situations. This component operates with the world model of the knowledge base and sensor signals from the hardware component.

The intermediate local planning layer, i.e. the action planning component, implements local goal-directed behavior. This layer operates with mental and plant models of the knowledge base.

The topmost cooperative planning layer, i.e. the negotiation and communication component, enables the agent to plan or to cooperate with other agents in order to resolve conflicts, as well as to achieve multiagent plans. This layer operates with the social model of the knowledge base.

All layers work asynchronously with different models in the agent's knowledge base. The structure of the robotic agent is presented in Fig. 2.

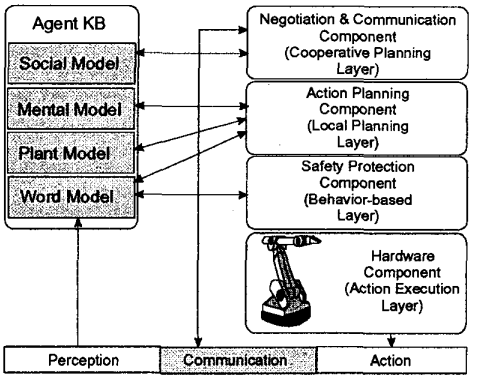

Fig. 2: Structure of a robotic agent

2.1 Social model:

This model contains knowledge about negotiation protocols and joint planning, as well as knowledge about the other agents in the environment.

2.2 Mental model:

For safe realization of the preplanned action, we propose installing a *security zone* that determines the safe distance between an agent and dynamic obstacles. When a dynamic obstacle or other agent penetrates the security zone, a conflict is recognized and the rules of the mental model can be used to determine the new goal-directed behavior. The mental model of the agent contains the knowledge necessary for decision making when a new object is recognized inside the security zone. This model can be presented in form of simple rules as follows.

Case A: *Agent realizes its goal and is not in conflict.*

IF the agent's current position is not in conflict with other agents and the agent realizes its goal THEN the action planning algorithm, as well as the current behavior, should not be changed.

Case B: *Agent realizes its goal and a conflict with other agents are recognized.*
IF the agent's current position is in conflict with one or more agent(s) and the agent realizes its goal THEN based on negotiation with the other agent, the distance to the other agent(s) should be increased and the distance to the goal decreased. The degree of importance of the first or second criterion is depends on the priority of the current action.
Case C: *Agent does not realize its goal.*
IF the agent does not realize its goal THEN the agent should always increase the distance to other agents.
Such a model is easy to implement in the action-planning component.

2.3 World model:
The knowledge represented here is the geometrical model of the robotic agent environment [9,10]. Many different methods can be used for geometrical representation of the agent service space. The model is modified based on sensor data. Depending on the amount of knowledge, different plant and world models can be used in the negotiation and coordination process. If only partial knowledge about the environment is available, the method is called *negotiation* among concurrently acting agents. If full knowledge is available, the method is called *synchronized negotiation.*
2.4 Plant-model: Model of the agent's hardware component:
The plant model contains the knowledge about construction, properties and structure of the hardware component of the agent. Here, knowledge of the kinematical properties of the robotic agent is provided in order to decide about collision avoidance mode and avoidance path. Therefore, the forward and the inverse kinematics models of the robot should be known. These models can be created in different ways using various formal tools, e.g. symbolically computed models, numerical models or neural network-based models [9]. Depending on the amount of knowledge, different models of plant can be used for action planning. If we have only partial knowledge about the surrounding environment we can apply the learning-based neural network model of agent actor (robot). If we have full knowledge about the static environment we can use a finite state machine model of agent actor based on the discretization of the actor's joint space. The complete formal explanation of the FSM model of agent kinematics is presented in [9,10].

3 Goals of Multiagent Robotic System

The behavior of the multiagent system can be described as the set of actions performed by each individual agent. Each agent follows its specific goal. Different dependencies can occur between the goals of agents in a multiagent system. The set of all goals can change over time. Robotic agents should realize different manipulator motions to achieve the common goal of the system. On the management level of an intelligent robotic system, the jobs are decomposed into the several motions and the contract manager decides which robotic agent should execute which motion. The contract manager assigns the final position of motion to each agent, so that they can realize a job from the current task [9].
At each time instance, these final positions constitute the goal set of the system. The principal task of the intelligent robotic agent is the execution of a given path of movement so that the robot action does not result in a collision with dynamical objects, such as the other robotic agents. The problem of finding a safe motion for all active agents is a reachability problem: how to get from the set of start states to the set of goal states. The problem can be stated as:
Find the sequence of global states and sequence of global inputs such that the last state achieves the final position and all the states are feasible.
Not all configurations of agents can be feasible. We say that the configuration (state) is *collision-free* if it does not collide with any static obstacle in the world [9].
To test this condition, we have to have full knowledge about the surrounding static world, i.e., the geometrical model of the agent's environment is completely known. Collision freeness does not determine if there is a collision with a dynamic obstacle, such as another agent. A conflict between two agents occurs if the distance between the agents is less than the security zone. We call a configuration *feasible* if it is collision and conflict free.

4 Local Planning Layer

In order to solve this complex reachability problem, we are going to sequentially apply a graph searching procedure to the state transition graph of global state set, which is the Cartesian product of the state sets of each agent.
Expanding the current state with the state transition function, then the successors of the current state etc. ad infinitum makes explicit the state transition graph that is implicitly defined by the current state and the transition function. As evaluation function we can use the weighted sum of the cost functions.

Using the standard A* procedure we can find the state trajectory (if it exists) from the current state to the goal state consisting only of feasible states.

Let each agent's manipulator has *n-DOF*. With joint space discretisation the number of successors grows exponentially with *n*. Thus, it becomes essential to quickly check for the non-repeatability of the nodes generated, and their feasibility.

If there is a conflict between different agents, the goal state set can be empty. In this case, there is no solution to the reachability problem [6,9].

To avoid this situation, we propose the decomposition of the global search problem into sequential searches for each agent separately. This means that an unachievable simultaneous solution will be replaced by a sequence of partial solutions that can be achieved one after another.

Depending on the knowledge level, different plant and world models can be used to perform the negotiation and synchronization process among agents.

4.1 M-step delay - sequential synchronization of agent group action

We show how the finite state machine model of agent behavior can be applied to solve the conflict avoidance problem by replacing the concurrent planning and realization of agent group actions with sequential one-step communication, planning and action realization.

A round-robin procedure for time synchronization of agent action leads to the concurrent realization of actions with *M*-step delay and to a solution of the conflict avoidance problem by using the sequence of time-synchronized partial solution. Based on the current state of the job queues and messages from agents in the last full activity cycle, the conflict manager prepares the ordered list of agents and establishes security zones, goals of motions, and the priority levels of agent actions. These priorities will the basis for the coordination in case of conflict between agents.

4.2 Action planning and action execution component individual agents

The task of the motion-planning component is to plan the safe configuration of the robot's manipulator based on information coming from the conflict manager, the safety protection component, and the world model. To realize this task, the component calculates the changes of robot configuration to avoid the obstacle in the best possible way. The action planning process of an agent activated by conflict manager, which sends a message including the agent's goal position and priority level, its security zone and its status (free or busy).

The action planner of the agent generates the new movement based on mental model rules in the following steps:

step 1 The action planner requests the current positions of other agents in the common environment,

step 2 based on the positions of other agent, its own status and the message from the safety protection component, the action planner recognizes its own situation and establishes the parameters for the motion search algorithm, such as type of evaluation function and type of feasibility testing function.

Agent is busy and is not in conflict:

If the agent's position is not in conflict with other agents and its status is busy, then the evaluation function is in standard form, i.e. it is the sum of the cost function and a heuristic function. The configuration is feasible if is collision and conflict free.

Agent is busy and is in conflict:

If the agent's position is in conflict with one or more of the other agents and its status is busy, then the evaluation function is the sum of a negative distance function and a situation dependent heuristic function with weight value equal to the priority level. After evaluation the configuration is feasible if is collision free and only is not in direct conflict with other agents.

Agent is free:

If the agent status is free, then the evaluation function is always the negative distance function. The configuration is feasible if is collision free and not in direct conflict with other agents.

step 3 Action planner starts the state-graph searching algorithm A* from its current position with previously established evaluation and feasibility testing functions.

The searching stops if the goal position is reached or if the OPEN set is empty or if the OPEN set has more than a given number of elements. To calculate the successor set, the planner uses the FSM model of the agent's hardware component.

step 4 The temporary path of motion is calculated. Depending on the length and conflict freeness, the new configuration of motion is chosen and realized.

step 5 The new current state (configuration) is analyzed and a message is send to the conflict manager. The message includes the information whether the agent has achieved its goal and/or if a conflict occurs and/or if the agent is in a deadlock position.

By sending the message, the agent ends his activity cycle and waits for a new command from the conflict manager. The flow of action planning is presented in Fig. 3.

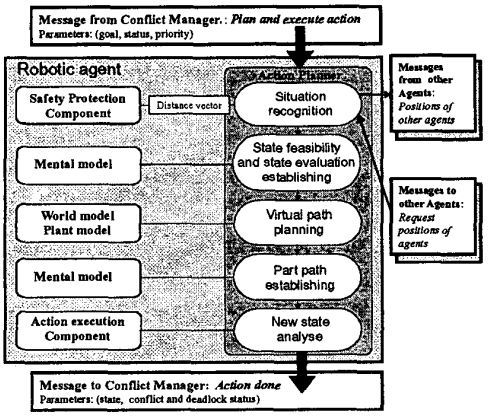

Fig. 3: Action planning in an intelligent agent

5 Conflict Manager

The conflict manager activates each agent sequentially. He sends an activation message to each agent and waits for an answer message. After a full cycle through the list of agents, the conflict manager calculates the priorities that are used to sort the agent list.

The priority calculation is based on the following factors.

General priority of task

This information can be obtained from the agent's task description, where the resulting priority factor is independent of the current state of the agent [9]. A temporary priority factor must also be specified. This factor depends on

- the current distance to finish position *d*,
- the rest duration of the motion *t*,
- the current delay of the motion *m*

To determine a temporary priority we use a fuzzy decision system. As input to the system we use the priority factors presented above.

These input values are not a fuzzy set, but indicate crisp values. Therefore they first have to be fuzzified. The same linguistic values can be assigned for each parameter: *small* (S), *medium* (M) and *large* (L). For each term a trapezoid-shaped membership function is used to cover the whole domain.

We consider fuzzy decision rules with four inputs - (*d,t,m*) - and one output temporary priority. Some examples of fuzzy rules are listed below.

R_1: *if d is S and t is S and m is path preference then priority is L*

R_2: *if d is S and t is S and m is advance then priority is M*

R_3: *if d is S and t is M and m is advance then priority is S*

The antecedents are compared with the input values of the fuzzy decision system so that it can be decided which rules can be used, and with which firing strength (amount of influence). This strength depends on how much the input values and the premises of the rule correspond to one another. The rules can be fired according to the fuzzification interface, each with a particular strength. The conclusion of a particular rule influences the general conclusion of the system. The result of this is a set of "cut off" fuzzy sets. They are combined using the union function and passed to the defuzzification interface to determine one single crisp value that summarizes this output fuzzy set. For this, we use the center-of-gravity method.

Based on these new priorities, the conflict manager establishes the new security zones. The agent list is then sorted with increasing agent priority.

6 Example

Let the agent group contain two agents acting in a common workspace. For the configuration shown in Fig. 4a), there does not exist a global solution, i.e. the agents do not achieve their final positions. We can start with different priorities. If we assume that agent 1 has the higher priority than agent 2, then the conflict manager allows the first agent to realize its motion and after reaching its goal position, the second agent obtains the greater priority and achieves its goal. This sequence of goal achievements is shown in Fig. 4 b), c), and d). Time, energy and length of motion trajectories are presented bellow.

Case	Priority	Agent	Length	Time	Energy
1	optimal	1	1,8 m	2,1, sec	119,7 J
1	optimal	2	2,1 m	1,9 sec	206,8 J
2	Agent1 winner	1	1,9 m	3,1 sec	156,5 J
2	Agent1 winner	2	4,6 m	6,2 sec	351,7 J

Case 1 presented parameters of motion for optimal trajectories, calculated separately for both agent. Case 2 - agent 2 subordinated, shown the parameters of trajectories from start position to the goal position for both agent.

Bibliography

1. FIPA, *Foundation for Intelligent Physical Agents, Agent Management*, FIPA97 Specification Part 1/2, Version 2.0, Geneva, Switzerland, 1998.

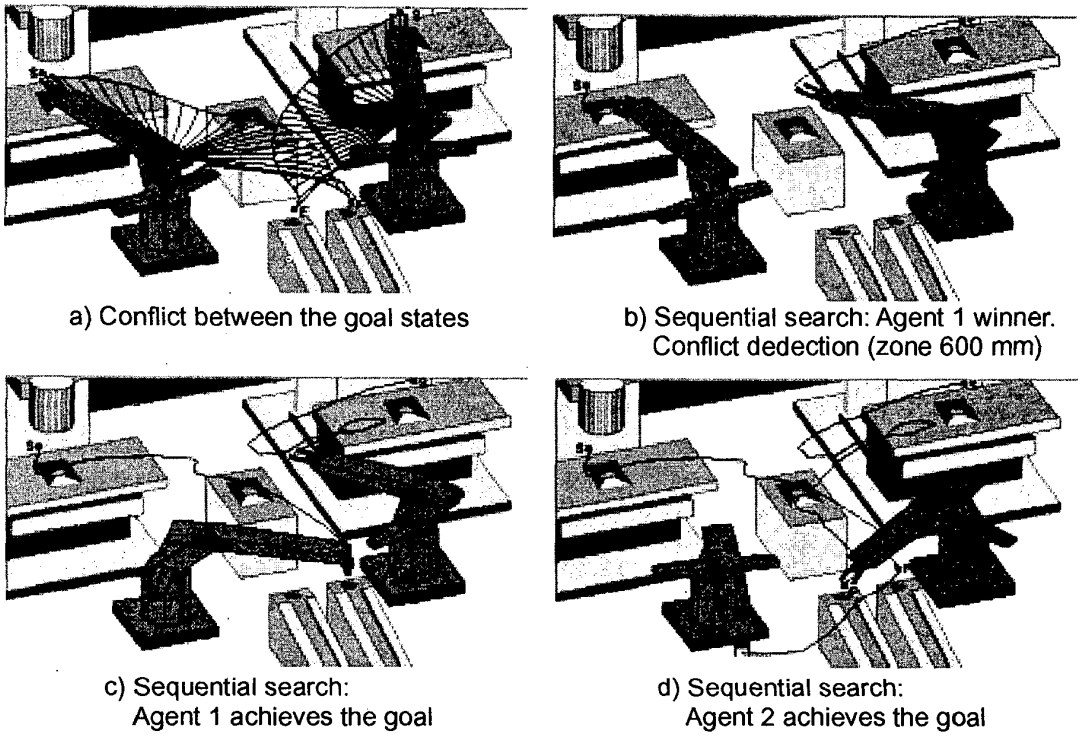

Fig. 4: Conflict management

2. Cohen, P. and Levesque, H., Communicative Actions for Artificial Agents, *Proceedings of the First International Conference on Multiagent Systems*, San Francisco, Cambridge, AAAI Press, 1995.

3. Jacak W., Proell K., Multiagent Approach to intelligent control of robot, *Robotics and Applications RA99, IASTED International Conference*, Santa Barbara, USA, 1999

4. Jacak W., Proell K., Multiagent based intelligent control and internal communication in an autonomous system, *SCI'99 and ISAS'99, World multiconference on systemics, cybernetics and informatics*, 1999.

5. Jacak W., Proell K., Multiagent Approach to Intelligent Control of Robot, *Lecture Notes in Computer Science*, No. 1789, 2000.

6. Jacak W., Proell K. S. Dreiseitl, FSM Theory based Conflict Management in Multiagent System, *EUROCAST'2001*, Las Palmas; Spain, 2001

7. Brenner W., Zarnekow R., Wittig H., *Intelligent Software Agents,* Springer-Verlag, 1998

8. Haddadi A., *Communication and Cooperation in Agent Systems*, Springer Verlag, 1995

9. Jacak W., *Intelligent Robotic Systems*, Kluwer Academic/Plenum Publishers, 1999

10. Latombe J.C. *Robot motion planning*, Kluwer Academic Pub., Boston, Dordrecht, 1991.

11. Sandholm, T. Automated Negotiation. *Communications of the ACM* 42(3), 1999.

12. Sandholm T. Distributed Rational Decision Making, *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence*, Weiss G., ed., MIT Press, 1999.

13. Andersson M. and Sandholm, T. Sequencing of Contract Types for Anytime Task Reallocation. In Agent-Mediated Electronic Trading, Carles Sierra (ed.). *Lecture Notes in Artificial Intelligence* 1571, 1999.

14. Sandholm T., Sikka, S. and Norden, S. Algorithms for Optimizing Leveled Commitment Contracts. *International Joint Conference on Artificial Intelligence (IJCAI),* Stockholm, Sweden 1999.