Linköping Electronic Articles in
Computer and Information Science
Vol. 2(1997): nr 1

A Decision-Mechanism for Reactive and Cooperating Soccer-Playing Agents

Silvia Coradeschi
Lars Karlsson
Department of Computer and Information Science
Linköping University
Linköping, Sweden

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/1997/001/

Published on January 15, 1997 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

Typeset by the authors using TeX
Formatted using étendu *style*

Recommended citation:
<Authors>. <Title>. Linköping electronic articles
in computer and information science, Vol. 2(1997): nr 1.
http://www.ep.liu.se/ea/cis/1997/001/. *January 15, 1997.*

This URL will also contain a link to the authors' home pages.

Abstract

In this paper we present some preliminary results regarding a system for developing autonomous agents that combine reactivity to an uncertain and rapidly changing environment and commitment to prespecified tactics involving cooperation. A central requirement is that the behavior specification should be done by people who are not computer and AI specialists. The original application was air-combat simulation. Here we consider the simulated soccer domain in perspective of the RoboCup competition. The behavior of an agent is specified in terms of prioritized rules organized into a decision tree. Coordinated behaviors are encoded in the decision trees of the individual agents. What link individual behavior descriptions together are explicit communication and common means to describe the situation the agents are in.

1 Introduction

An important issue in the development of multi-agent systems is how to create a decision mechanism for an agent that combine the abilities to react to a dynamically changing environment and to coordinate with other agents.

Linköping University and Saab Military Aircraft AB, Sweden, have undertaken a cooperation project with the aim to investigate the design and implementation of automated agents for the air combat domain (Coradeschi, Karlsson & Törne 1996). In this project we are developing means for specifying the behavior for automated pilots that attempts to combine on one side reactivity to the uncertain and dynamic battlefield environment and on the other side commitment to prespecified tactics involving coordination. A third requirement is that the behavior specification is not to be done by experts in computer science and AI, but mainly by experts in air combat and military air technology, and therefore, the system should be relatively easy to use and conceptually simple. The types of behaviors to be specified are those following the types of tactics used by human pilots in real situations. The specification is done in terms of hierarchically structured and prioritized decision rules.

Our long term goal is to create a general decision-mechanism for multiple automated agents in complex real-time environments that can be used in applications that have similar characteristics to the one that we are now examining. Soccer is an area with some strong similarities to air combat: there are two opposing forces/teams, a high degree of coordination within the teams and real-time decision making in a rapidly changing situation.[1] Therefore it seemed like a natural step to consider how our decision-mechanism can be applied to autonomous agents for synthetic soccer robot competition. In particular, it seemed to be a very suitable domain for us to study how a coordinated behavior can be specified using the type of rule-based system presented here.

Also in the TacAir-Soar project (Tambe et. al. 1995) the development of intelligent agents for the air combat domain is now coupled with the development of intelligent agents for the soccer domain with special emphasis in the tracking of agents team (Tambe 1996). Also the analysis of the means for coordination of behavior of computer generated forces in TacAir-Soar (Laird, Jones & Nielsen 1994) offers interesting suggestions for coordination of soccer players.

In both the soccer domain and the air combat domain coordination between agents is essential, but it cannot be achieved by mainly communication. In fact, giving the real-time constrains in these domains, the time for communication is limited and the presence of hostile agents makes the communication risky and unreliable (jamming). Several techniques have been developed for communication-

[1] Of course, there are differences as well. Soccer has far more restrictive rules regarding how you can treat your opponents.

poor coordination including physics oriented approach to coordination (Shehory & Kraus 1996), focal points (Fenster, Kraus & Rosenschein 1995) and observation-based coordination (Huber & Durfee 1995). In general these techniques have as an aim coordination of isolated agents performing rather simple tasks as respectively filling holes of various sizes in a planar surface, trying to choose the same object out of a set of objects and moving in the same location or in an opposite location respect another agent in a discrete time, two dimensional, simulated grid world.

In our case the agents are organized in teams and perform complex and in general not repetitive tasks. Several agents should then coordinate performing sequences of actions that would lead the team to success. To achieve this coordinated behavior, the agents rely on common tactics similarly to humans trained to coordinate in specific domains.

In our approach the experts of the air combat domain are intended to specify the behavior of each agent directly, without the aid of a system expert. This is one of the central requirements on the system, as it lets the domain experts directly give the directives that the agents should follow, test the behavior and change the directives in case the agents do not behave as expected. In the case of the soccer domain we have no intentions to employ a professional football coach to specify the behavior of the players. However, also in this case it is advantageous to have a easy way to specify and incrementally improve the behavior of the players.

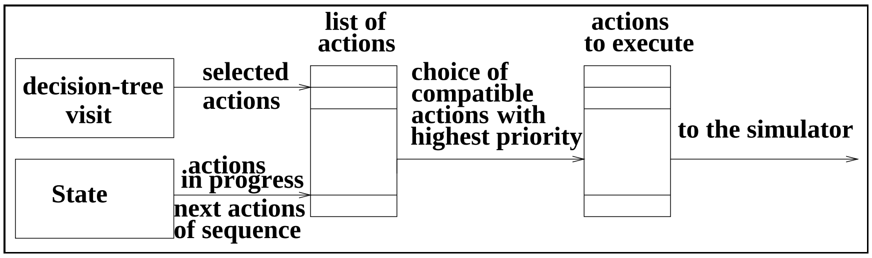

Figure 1: Phases of the decision mechanism

2 The decision-mechanism

In our approach, each agent is equipped with a state describing the information that he uses for making decisions, and a decision-tree. The decision-tree consists of a hierarchy of decisions with one decision for each node. At every step of the simulation the state is updated with information that is received from the simulator and interpreted. Then the decision-tree is visited and as a result of the visit a list of actions is created (Fig. 1). The actions in the list are the actions that the agent will consider to perform. Each action is associated with a

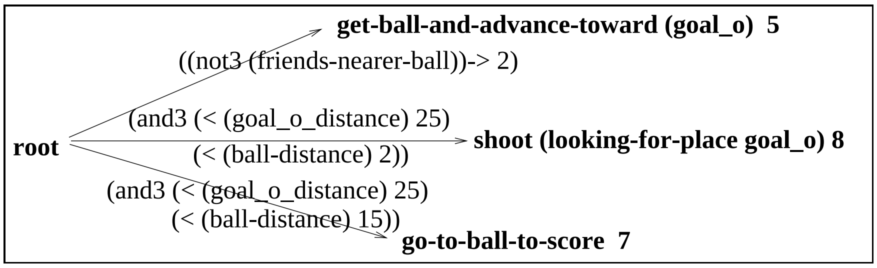

Figure 2: Simple example of decision tree

priority value that changes dynamically. To this list the actions and the activities still in progress are added. The agent then selects the actions to actually execute and then send them to the simulator. Some of the actions in the list are mutually exclusive as they use the same resource and in this case the agent selects those he will actually perform depending on their priority values. Some actions can be done concurrently and in this case the agent performs them in parallel. The agent can also decide to perform sequences of actions. Anyhow, the agent is not committed to complete the sequence of actions as the sequence can be interrupted if something else becomes more important.

Let us consider a small example of a decision tree (Fig. 2) in which no coordination between players is considered. The agent, if there are no team mates nearer to the ball *((not3 (friends-nearer-ball))*, tries to get the ball and advances toward the opponent goal, **get-ball-and-advance-toward(goal_o)**. When he is near the opponent goal and he can kick the ball, *(and3 (< (goal_o_distance) 25) (< (ball-distance) 2))*, he tries to score, **shoot(goal_o)**. If he is near the goal but not enough near to the ball to kick he goes toward the ball **go-to-ball-to-score**. In this example the conditions are not mutually exclusive, the correct action is chosen depending on the priority. For example the **shoot(goal_o)** action has priority 8, while the **go-to-ball-to-score** action has priority 7 so if the player is near the ball he will perform the shooting action.

The conditions on the branches can refer to both information obtained from perception and information stored in the agent's internal state. An example of the latter is when a defender follows an opponent due to a commitment to mark this opponent. This commitment is registered in the defender's state. Making the commitment to mark a particular opponent constitute an internal action that update the state of the agent.

The external actions presented in the decision tree will be interpreted in terms of the low-level actions available in the soccer simulator. For instance, **go-to-ball-to-score** will make the agent turn and move towards the ball. Also the information received from the simulator is interpreted. For instance, at each simulation cycle the value of *friends-nearer-ball* is derived from the incoming message. This permits the user to work on a level of abstraction that is more meaningful and convenient than working with the low-level data and actions of the simulator. He/she can focus more on what to do than the details of how to do it. Actually, in the original air-combat application, a certain level of abstraction is built into the simulator. In the soccer application, we had to provide this abstraction ourselves.

2.1 State

Each of the agents has a state where the information is maintained that is necessary for deciding what actions to perform. This information is of four types. First, the state contains the characteristics of the agent specified before the starting of the simulation, for example team and position (midfielder, left-attacker, etc.) of the player.

Secondly there is the information the agent receives from the simulator. This information is partly information directly received from the simulator (for example distance and direction of other players, flags, goals and lines) and partly information that is derived during the processing of the messages received. Examples of this last kind of information are which opponents or team mates are near the ball (*friends-nearer-ball* in the previous section) and the approximate position and direction of the agent. The information is stored with a time stamp to make it possible to recognize if the information is up-to-date.

Thirdly, there is the information about the present status of the agent for example the angle of view and the quality chosen for the visual information and the actions presently performed.

Finally, one part of the state represents the "memory" of the agent. In the memory are recorded, among other things, important past events, for example the number of goals done and received, decisions previously taken, for example the decision to mark an opponent, and messages received via communication channels.

2.2 Decision-tree structure

In a decision-tree a node is entered if the entry condition associated with it is satisfied. In the leaves there are actions with a basic priority value. In addition modifier conditions can be associated with a node. Modifier conditions have the form *(condition $\rightarrow$ number)*. If the condition is true the number is added to the priority of the actions associated with the branches. If the priority of the action fall to 0 or below the action is not performed.

2.3 Conditions

The conditions are defined as follows:

- **Primitive Conditions**

- Atomic Conditions

  Atomic conditions are true if the information is present in the state. For example, the atomic condition *friends-nearer-ball* is true if there is information that there are team mates nearer to the ball.

- Relational Conditions

  Relational conditions consist of relational operators such as $<$, $>$, $\leq$, $\geq$, $=$, $\neq$, applied to numbers and numerical functions. An example of a relational condition is *(< (distance-player x) 30)* where *(distance-player x)* is a function that returns the distance of the agent to another player x.

- **Composite Conditions**

  Composite conditions are formed by applying the operators *and3, or3, not3, for-all* and *there-is* to other composite or primitive conditions. *And3, or3, not3* are operators in a three-valued logic: true, false and unknown. In fact the agent can be in the situation in which he does not know the value of a part of a condition, for example he does not know his relative position with respect to the ball, but he can still decide the true value of the condition, for example using the rule that *and3* of false and unknown is false.

  If the condition associated with a node has the form "*(for-all x set condition)*" the condition is tested for each agent x in the set. The rest of the branch is visited once for each of the agents for which the condition is satisfied. All the actions selected are then taken into account as candidate actions to perform. "*(there-is x set condition)*" finds the first agent in the set that satisfies the condition and continues to evaluate the tree with x as this agent.

2.4 Actions

There are three kinds of actions: primitive actions, concurrent actions and sequential actions. Primitive actions are for example **pass(x)**, i.e., pass the ball to the agent x. They can also be internal actions, i.e., actions that update the state of the agent. For example **set-state(mark, x)** records in the state the information that the agent is committed to mark the player x. The actions sent to the simulator can be instantaneous, for example **pass(x)**, or can have a duration. An example of action with duration is the action **go-behind-ball** in Fig. 3. The part of the tree in this example is accessed when the game is suspended and the player should put the ball back in game. The player first goes behind the ball and then passes the ball to a fellow player. The action of going behind the ball requires a certain amount of time and it is performed in different phases. The agent keeps track of the actions that he is currently performing in the

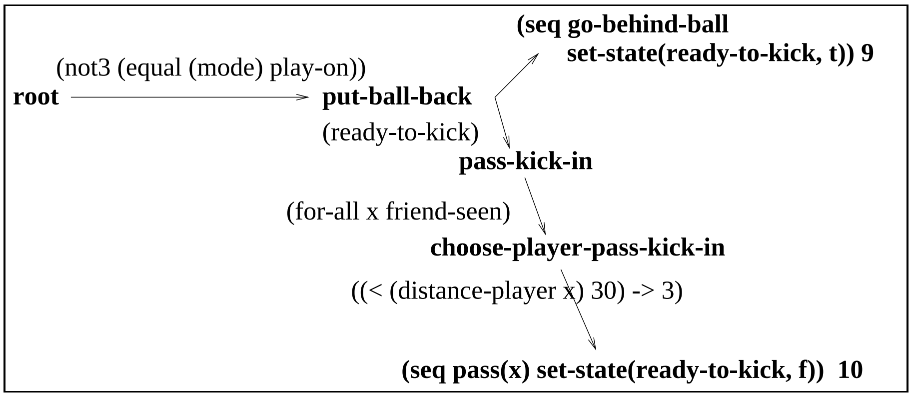

Figure 3: part of the decision-tree used to put the ball back in game

state and in which phase the action is. At each step of the simulation the agent decides if continuing performing the action or interrupting it in order to perform a more important action (that is one with a higher priority value).

Sequential actions are composed by actions that should be performed one after another. When a sequential action is started the first action of the sequence is started and it is automatically recorded in the state which are the following actions in the sequence that should be performed. When the first action of the sequence is completed the second action of the sequence becomes a candidate action to be started and it competes with the actions currently performed and the other actions selected at the present cycle of the simulation. If the second action is started the same thing is then done with the third action and so on until the sequence is completed. If an action that is in turn is not started the sequence is aborted. An example of sequential action is **(seq go-behind-ball set-state(ready-to-kick, t))** in Fig. 3. When the action of going behind the ball is completed the agent record in the state that he is ready to kick the ball.

When a sequential action is started the priority of the following actions in the sequence is increased with a value decided by the user and depending on how important it is in his opinion to complete the sequence.

Concurrent actions consist of a collection of actions that should be started at the same time. There is a concurrency of actions inherent to the system as actions in different branches can be selected and, if they are compatible, started at the same time. With explicit concurrent actions, however, we are sure that either all the actions are performed or none of them are. The actions can be a composition of both concurrent and sequential actions.

2.5 Compatibility of actions

The test to check if two actions are compatible is based on the physical resources they use. Two primitive actions are compatible if they do not use the same resources. For example the action of passing is compatible with the action of saying a message as they use different resources.

A concurrent action is compatible with another action if all the actions that form the concurrent action are compatible with the other actions. A sequential action is compatible with another action if all the actions that are present in the sequence are compatible with the action.

Compatibility of actions is more significant in the air combat domain than in the soccer domain, as the pilots can in general do more actions in parallel than the soccer players. In the soccer domain the actions that are compatible with each other are the actions of sending a message and the actions that involve movement. The internal actions are in general compatible with all the other actions and between themselves.

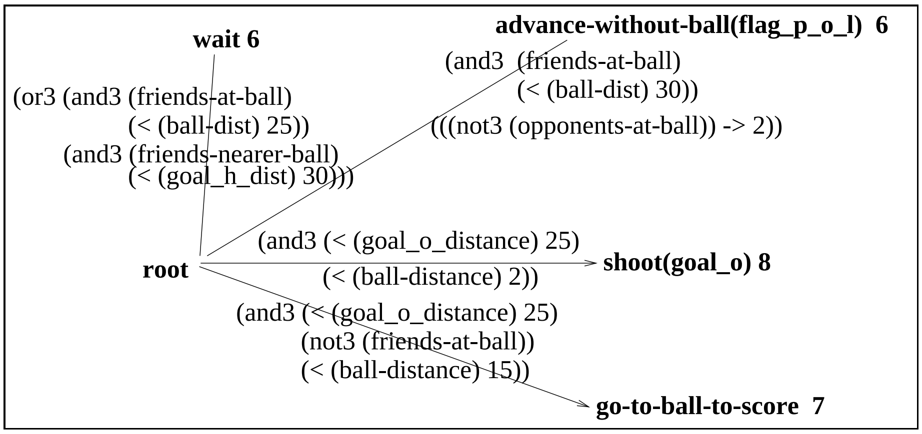

Figure 4: part of the decision-tree of a left attacker

3 Coordination among agents

In both the soccer and the air combat domain team work is essential, but communication between team mates due to real time constrains and possibility of interception by the opponents should be reduced to short and simple messages. Coordination is therefore obtained by common tactics and strategies learned and practiced during training. Each agent knows the tactics of the team and his role in the tactics and tries to act according to this knowledge. The agent should initiate tactics depending on the situation and recognize the tactics that the other team members have initiated in order to do his part in it.

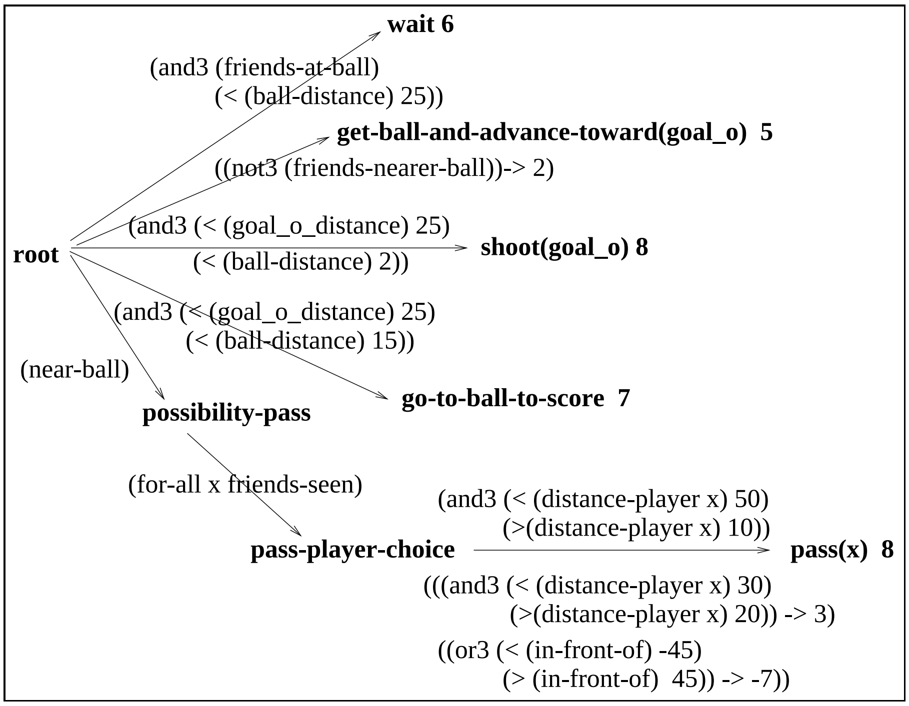

Figure 5: part of the decision-tree of a midfielder

The agent has three ways of recognizing the tactics that the team is following: observation of the general situation and knowledge of the tactics appropriate for the situation, observation of indicative actions of the team mates, and explicit communication. In general there can be a combination of these three aspects.

In our approach the knowledge required for applying the tactics is codified in the decision-tree and the observations of the situation and of the actions of other agents influence the decisions about which tactics to adopt. Explicit communication is performed through communication actions present in the decision tree. The message is stored in the state of the agent and will influence his future decisions. Tactical decisions made by the agent are also registered in the state. The agent can change tactics as the situation changes. Further, the use of priorities lets him take advantage of opportunities or react to threats that are not considered in the tactics selected.

Let us consider a simple example of coordinated behavior between a left attacker and a midfielder and how we codify it in the decision-tree. Part of the decision-tree of the left attacker is shown in Fig. 4 and part of the decision-tree of the midfielder is shown in Fig. 5.

If no team mate has the ball, the midfielder tries to get the ball and to advance toward the opponent goal **go-to-ball-and-advance-toward(goal_o)**. If a team mate has the ball he tries to keep a distance from the ball that would allow him to support the team mate action. If he has the ball, *(near-ball)*, then he decides to which fellow player to pass the ball among those that he can see (*for-all x friends-seen*) and that are at the right distance:

(and3 (< (distance-player x) 50)
(> (distance-player x) 10)).

The preference to which player to pass is done in base of the distance of the player (if his distance is between 20 and 30 the priority of the action to pass to him is increased by 3), and in base of the distance to the opponent goal (if a player is further than him from the goal the priority of passing to him is decreased by 7).

The left attacker, if a team mate has the ball, expects that the player that has the ball will advance and pass the ball to him. Therefore he will tend to take position at the left corner of the opponent penalty area, **advance-without-ball(flag_p_o_l)**, anyhow keeping a position not further from the ball than 30. It is possible that the player that has the ball will do something else, but the attacker should position himself in the best place for continuing the action depending on the schema that the player that has the ball is most likely to follow. When he is near the opponent goal he will try to get the ball and to score (**go-to-ball-to-score** and **shoot(goal_o)**).

Notice that with the approach presented here, the different elements of a coordinated behavior of a group of agents have to be specified in terms of the behavior of each agent individually. However, it would be straight-forward to provide a decision-tree editor that permits the user to visualize, edit and test the parts of the decision tree that constitute some specific coordinated maneuver for several agents at the same time. This could simplify the work with coordinated behaviors considerably, without additional complexity to the representation being used.

Good guidelines for the user when he describes a cooperative behavior are to structure the tree in similar ways for different agents and to use the same names for state parameters for all agents. In addition, good support from the message processing part is necessary; the user should be able to work with meaningful conditions describing the situation on the field. For instance, a defender playing a zone defense should be able to obtain information regarding how many opponents there are in his fellow defender's zone in order to judge whether this other defender will need his support or not.

4 Summary

We have described a decision mechanism for autonomous agents which attempts to provide a relatively simple and easy-to-learn way to specify the behavior of autonomous agents. It is based on a decision-tree where actions are assigned dynamically changing priorities and decision are based both on sensory input and the internal state of the agent. We have discussed how this decision tree can be used to obtain a reactive and coordinated behavior for a simulated soccer team.

Coordinated behaviors are encoded in the decision trees of the individual agents. What link these individual behavior descriptions together are explicit communication and common means to describe the situation the agents are in, both in terms of their observations and their internal states.

5 Acknowledgment

We would like to thank Jacek Malec for the helpful comments. This work is partly supported by the Centre for Industrial Information Technology, Linköping University (CENIIT), and the Swedish National Board for Industrial and Technical Developement (NUTEK).

6 References

Coradeschi, S., Karlsson, L. and Törne, A. 1996 Intelligent Agents for Aircraft Combat Simulation, in Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL. Available on WWW: `http://www.ida.liu.se/ silco`.

Fenster, M., Kraus, S. and Rosenschein, J. S. 1995 Coordination without Communication: Experimental Validation of Focal Point and Techniques, in Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA.

Huber, M. J. and Durfee, E. H. 1995 Deciding When to Commit To Action During Observation-based Coordination, in Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA.

Laird, J. E., Jones, R. M. and Nielsen, P. E. 1994 Coordinated Behavior of Computer Generated Forces in TacAir-Soar, in Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL.

Shehory, O. and Kraus, S. 1996 Cooperative goal-satisfaction without communication in large-scale agent-systems, in Proceedings of ECAI96, Budapest, Hungary.

Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S. and Schwamb, K. 1995 Intelligent Agents for Interactive Simulation Environments in *AI Magazine* 16:1.

Tambe, M. 1996 Tracking Dynamic Team Activity, in Proceedings of AAAI96, Portland, Oregon.