Using Reactive Deliberation for Real-Time Control of Soccer-Playing Robots

Yu Zhang and Alan K. Mackworth

Department of Computer Science, University of British Columbia, Vancouver B.C. V6T 1Z4, Canada, yzhang@cs.ubc.ca, mack@cs.ubc.ca

Abstract. Soccer meets the requirements of the Situated Agent approach and as a task domain is sufficiently rich to support research integrating many branches of AI. Reactive deliberation is a robot architecture that combines responsiveness to the environment with intelligent decision making. Under Reactive Deliberation, the robot controller is partitioned into a deliberator and an executor; the distinction is primarily based on the different time scales of interaction. A controller for our team entry in the Robocup97 Simulation League, *UBC Dynamo97*, has been developed using the Reactive Deliberation architecture.

1 Introduction

The Good Old Fashioned Artificial Intelligence and Robotics (GOFAIR) [3] research paradigm has shaped the area of intelligent robotics since the time of the robot Shakey. Some of the typical fundamental assumptions made about the world were that there is only one agent, that the environment is static unless the agent changes it, that actions are discrete and are carried out sequentially and that the world the robot inhabits can be accurately and exhaustively modeled by the robot. These assumptions proved to be overly restrictive and ultimately sterile. In the usual dynamic of the scientific dialectic, a new movement has emerged as a synthesis of GOFAIR and "Nouvelle AI": the Situated Agent approach. A situated agent is a real physical system grounded and embedded in a real world, here and now, acting and reacting in real-time. Mackworth [3] proposed that playing soccer be a paradigmatic task domain since it breaks with nearly all of the restrictive assumptions on which GOFAIR is based and meets the standards proposed in the Situated Agent approach. The soccer domain has the following characteristics:

1. Neutral, friendly, and hostile agents
2. Inter-agent cooperation and communication
3. Real-time interaction
4. Dynamic environment
5. Real and partially unpredictable world
6. Objective performance criteria
7. Repeatable experiments

Soccer meets the requirements of the Situated Agent approach and as a task domain is sufficiently rich to support research integrating many branches of AI [4].

The Dynamite testbed has been developed in our laboratory for testing theories in the soccer domain using multiple mobile robots [1]. The testbed consists a fleet of radio controlled vehicles that perceive the world through a shared overhead perceptual system. The Vision Engine produces the absolute position of all the objects on the soccer field. Each vehicle is controlled by a distributed user program running on two transputer nodes. The movement of all vehicles is controlled through radio transmitters attached to a single shared transputer node. A physics-based real-time graphics simulator for the Dynamite world is also available for testing and developing reasoning and control programs [2,5].

Two approaches are taken in our laboratory to robot control in the soccer domain: *Constraint Nets (CN)* and *Reactive Deliberation*. CN is a formal model for robotic systems and behaviours. CN programs are composed of modules with I/O ports. CN provides a theoretical foundation for systems design and analysis. The dynamics and control for each robot, as well as the interaction between robots, can be modeled using CN [3,9].

Reactive deliberation [7] is a robot architecture that combines responsiveness to the environment with intelligent decision making. Several controllers based on reactive deliberation have been implemented to allow Dynamites (soccer robots) to compete in complete two-on-two games of soccer [4,7]. A controller for our team entry in the Robocup97 Simulation League, *UBC Dynamo97*, has also been developed using the Reactive Deliberation architecture.

2 The Reactive Deliberation Architecture

Reactive deliberation is a robot architecture that integrates reactive and goal-directed activity. Even deliberation must be, to some extent, reactive to respond to changes in the environment. Under reactive deliberation, the robot controller is partitioned into a deliberator and an executor; the distinction is primarily based on the different time scales of interaction. Informally, the deliberator decides what to do and how to do it, while the executor interacts with the environment in real-time. These components run asynchronously to allow the executor to interact continuously with the world and the deliberator to perform time consuming computations.

2.1 The Executor

The executor is composed of a collection of action schemas. An *action schema* is a robot program that interacts with the environment in real-time to accomplish specific actions. The deliberator enables a single action schema with a set of run-time parameters that fully defines the activity. Only one action schema is enabled at a time and it interacts with the environment through a tight feedback loop. In the world of real-time control there is no room for time consuming planning

algorithms. Computations in action schemas are restricted to those that can keep pace with the environment, so lengthy computations are performed in the deliberator.

There are three action schemas implemented in the controller for UBC Dynamo97, our team in the Robocup97 Simulation League:

1. *Intercept ball* generates a sequence of actions to move to the ball according to the dynamics of the ball and the robot.
2. *Goto position* generates a sequence of actions to move to a certain position on the field.
3. *Kick ball* decides how much force the player should use to kick the ball.

2.2 The Deliberator

The focus of the deliberator is on an effective mechanism for selecting actions or goals in a timely manner. A central feature of reactive deliberation is that the deliberator is composed of concurrently active modules called *behaviours* that represent the goals of the robot.

A *behaviour* is a robot program that determines an action that may, if executed, bring about a specific goal. Behaviours *propose* actions whereas action schemas *perform* actions. Each behaviour must do the following: 1) select an action schema, 2) compute run-time parameters for the schema (plan the action), and 3) generate a bid describing how appropriate the action is. The most appropriate behaviour, and hence action, is determined in a distributed manner through inter-behaviour bidding.

Each bid is an estimate of the expected utility of the proposed action and is based on the current state of the world as well as the results of planning. Currently, the criteria for generating the bids are hand-coded and tuned so that the most appropriate behaviour is active in each situation. This approach requires the designer of a system to explicitly state the conditions under which certain behaviours are suitable or favorable. The bids are simple algebraic formulas that are easily computed. Each behaviour has a basic bid that is modified through the addition and subtraction of weighted factors that depend on the environment and the results of motion planning. At any point in time, the *ruling behaviour* is the behaviour with the currently highest bid.

The principal advantage of behaviour-based bidding is modularity. Since bids are calibrated to an external measure of utility, behaviours can be added, modified or deleted without changing the bidding criteria of the established system. A new behaviour must, of course, be tuned to be compatible with existing ones. Behaviours are independent, so they can have different representations and approaches to generating actions. There is no central decision maker that evaluates the world and decides the best course of action, so behaviours can be run concurrently on different processors (instead of timesharing a single processor), thus improving the speed of the system.

There are five behaviours implemented in the controller for UBC Dynamo97:

1. *Intercept ball.* The player intercepts the ball. It has a base bid that is modified by environmental factors and planning results. Some of the factors are: the speed and heading of the ball, the distance to the ball, the speed and heading of other players and their distances to the ball.
2. *Defend.* The player goes to its assigned defensive position.
3. *Offend.* The player goes to its assigned offensive position. The Defend and Offend behaviours have the same factors affecting their bids as the first behaviour has. These defensive and offensive positions could be set by learning algorithms, though here they are set by hand.
4. *Kick.* When the player can kick the ball, he should choose a direction in which to kick the ball. He should decide whether he should pass the ball to his teammates or shoot it at the goal. (He shouldn't kick the ball to his opponents but sometimes this does happen because of imperfect information coming from the soccer server.) A sub-bid system is used here to choose the appropriate sub-behaviour among passing, shooting and kicking to an open area. Passing is a inter-robot cooperation problem. Due to the restricted communication mechanism provided by soccer server, we don't use any explicit communication to implement inter-robot cooperation. Using communication, each robot would broadcast its intended action and a bid which estimates the appropriateness of that action. Robots whose actions are in conflict would reevaluate their decision to include the bid information of other robots. A robot with a lower bid than another robot will reevaluate its internal bid for that action, since that action may not be the best one in the current situation.
5. *Avoid.* When the player is obstructed by another player, it plans to find a new path.

3 Conclusions and Future Work

Under Reactive Deliberation, the robot controller is partitioned into a deliberator and an executor; the distinction is primarily based on the different time scales of interaction. Informally, the deliberator decides what to do and how to do it, while the executor interacts with the environment in real-time. These components run asynchronously to allow the executor to interact continuously with the world and the deliberator to perform time-consuming computations.

In Reactive Deliberation, there is no central decision maker that evaluates the world and decides on the best course of action, so behaviours can be run concurrently on different processors, thus improving the speed of the system.

Some learning techniques could be used here to estimate the bid for each behaviour and to set the defensive and offensive positions.

Using constraint-based behaviours, the Constraint Net and Reactive Deliberation approaches could be integrated for a more formal and modular approach.

References

1. R. A. Barman, S. J. Kingdon, J. J. Little, A. K. Mackworth, D. K. Pai, M. Sahota, H. Wilkinson, and Y. Zhang. DYNAMO: Real-time experiments with multiple mobile robots. In *Intelligent Vehicles Symposium*, pages 261–266, Tokyo, July 1993.
2. R. A. Barman, S. J. Kingdon, A. K. Mackworth, D. K. Pai, M. K. Sahota, H. Wilkinson, and Y. Zhang. Dynamite: A testbed for multiple mobile robots. In *Proc. IJCAI Workshop on Dynamically Interacting Robots*, pages 35–45, Chambery, France, August 1993.
3. A. K. Mackworth. On seeing robots. In A. Basu and X. Li, editors, *Computer Vision: Systems, Theory, and Applications*, pages 1–13. World Scientific Press, Singapore, 1993.
4. M. Sahota and A. K. Mackworth. Can situated robots play soccer? In *Proc. Artificial Intelligence 94*, pages 249 – 254, Banff, Alberta, May 1994.
5. Michael K. Sahota. Dynasim user guide. Available at http://www.cs.ubc.ca/nest/lci/soccer, January 1996.
6. Michael K. Sahota. Real-time intelligent behaviour in dynamic environments: Soccer-playing robots. Master's thesis, University of British Columbia, August 1993.
7. Michael K. Sahota. Reactive deliberation: An architecture for real-time intelligent control in dynamic environments. In *AAAI94*, pages 1303–1308, 1994.
8. Michael K. Sahota, Alan K. Mackworth, Rod A. Barman, and Stewart J. Kingdon. Real-time control of soccer-playing robots using off-board vision: the dynamite testbed. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 3690–3663, 1995.
9. Y. Zhang and A. K. Mackworth. Synthesis of hybrid constraint-based controllers. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 552 – 567. Springer Verlag, 1995.