UBCDynamo02: A Team of Robot Soccer Players

Jefferson D. Montgomery

Laboratory for Computational Intelligence
Computer Science Department
University of British Columbia
jdm@cs.ubc.ca

1 Introduction

The *Dynamo* (Dynamics and Mobile Robots) project provides a testbed for real-time control of multiple mobile robots. The UBCDynamo02 RoboCup team implements recent work in this area aiming to model a system of intelligent, dynamic agents. As a first generation RoboCup team, research has focused on issues which are fundamental for a successful situated robot. These include probabilistic model updating for localisation and object tracking based on Bayesian sensor and actuator models, and an architecture based on an online computational model.

A multi-robot system consists of a number of robots coupled with their environment. Further, a robot can be described as a controller (which is designed to regulate the dynamic relationship between the robot and the environment) embedded in a plant. In general, such systems entail hybrid dynamics consisting of continuous, discrete, and event-driven components. Constraint Nets (CN) provide formally rigorous model of hybrid dynamic systems based on online constraint satisfaction. Appropriate models based on CN allow control synthesis, verification, and a deeper understanding of the behaviour of the system.

A specification of agents situated in real environments must also model noise in sensation and action. UBCDynamo02 uses Kalman filters to track dynamic objects in the environment and Markov models to detect exogenous actions on them. An action accounting procedure is used to infer whether commanded actions are executed, and the world model is updated accordingly. Real-time performance is improved by the use of adaptive synchronisation between the agents and the environment that promotes optimal command times.

This paper provides a brief introduction to the UBCDynamo02 team. Section 2 describes an architecture based on Constraint Nets. Section 3 describes some key features of the robot's internal model and how it is updated.

2 Modeling Robotic Systems

Constraint Nets (CN) are a general abstraction of dataflow networks that provide a semantic model of hybrid, dynamic systems [5]. Using CN, causal systems can be modeled with time and state variables that are either discrete or continuous and with synchronous and asynchronous event structures. As such, CN has proven useful for developing, analysing, and understanding dynamic systems of varying complexity including elevator control [6] and teams of robot soccer players [4, 7].

A robot is a coupling of a controller with a plant and a robotic system is a coupling of a robot with its environment. Using CN, the dynamics of each system component (the environment as well as the robot plant and controller) can be modelled individually and then integrated to derive the behavior of the overall system [4]. The modular nature of CN allows hierarchical modelling and the use of aggregation operators as architectural tools, which makes a CN-based architecture flexible and extensible.

2.1 A CN-Based Robot Controller

The purpose of the robot controller is to regulate the behaviour of the robot in the environment. A CN-based controller is a set of online constraint solvers that act to impose constraints on the robot's behaviour space. Therefore, developing a robot controller reduces to designing constraints on the robots behaviour. In most cases, this is an intuitive way of developing robot behaviour.

The constituent constraint solvers are layered and interactive, where solvers at the same level of the interaction hierarchy are coordinated via various arbitrations. One arbitration technique is to combine these constraints using utility theory [8]. This idea is extended by UBCDynamo02 in using the *Contextually Weighted Utility Constraint*, which is to maximize the utility function

$$U(a) = \sum_{i,j} k_{ij} P_i(a|s = j) P(s = j) \tag{1}$$

where $P_{ij}(a|s = j)$ is the probability that action a will lead to solving the i^{th} constraint in the arbitration hierarchy during situation j, and k_{ij} are constant weights that can be learned using various techniques [8]. The robot's policy is to perform the action that produces the highest utility, U, given its belief in the current situation, $P(s = j)$.

3 Maintaining a Consistent World Model

Stable agents that operate in real time require an accurate notion of the environment's state. Acting correctly involves acquiring sensor information, deliberating, and acting in a timely manner so that when actions are executed, the relevant internal information is still valid in the real world. Incorrect actions may not be possible or may not have the desired effect.

The internal world model for UBCDynamo02 is updated based on sensor and actuator information using Kalman filters [2]. Actuator information is extended beyond dead reckoning by analysing sensors to determine which action commands have been executed, ensuring that model updates are accurate.

3.1 Keeping Track of Commanded Actions

UBCDynamo02 uses a procedure for mapping action commands to execution states (i.e., executed on time, executed late, or not executed) [3]. Such a procedure is useful for model prediction, online robot activity analysis, and providing coarse estimates of the optimal time to command actions. The basis of the procedure is the strict rules that govern how an action is handled by the environment.

Action confirmation is based on the inherent causal relationship between time, action execution state, and the count of executed actions that is included in the `sense_body` message. Effectively, any two messages can be used to determine the number of actions executed between the time the messages were sent. So, if the agent commands an action after receiving a `sense_body` message, the next message will tell the client whether an action was executed or not. However, ambiguities arise if a second action is commanded before a previous action has been confirmed. In such circumstances, all hypotheses are maintained until the ambiguity can be resolved or is proven to be undecidable.

3.2 Synchronising Actuator Commands with Action

In order to send correct actions, agents must synchronise their sensing and acting through the environment. Several synchronisation techniques have been explored based on an internal model of the environment, or on external signals [1]. Because of environment variability, internal synchronisation produces poor results on average. However, it is consistent and not prone to communication errors. External synchronisation uses direct external evidence to determine the environment state of the environment. However, its reliance on these measurements causes performance to degrade as the communication channel becomes less reliable. This classic trade-off makes it difficult to optimally combine internal models with measurements based on external evidence.

UBCDynamo02 utilises a novel synchronisation algorithm that combines probabilistic analysis of the communication channel with coarse event analysis based on the server-client protocol to infer optimal action [3]. The technique is shown to be a generalization of internal and external synchronisation, and combines the benefits of each. Experiments have shown that this synchronisation algorithm enables the agent to execute a higher percentage of correct actions, especially as network communications become less reliable [3].

4 Conclusion

The UBCDynamo02 soccer team was developed using Constraint Nets to model the agents and environment. CN models allow the development of agent controllers by specifying constraints on the desired behaviour. Controllers solve the constraints using an online computational model. Agents adapt their action during play using a new synchronisation algorithm that infers the optimal frequency at which the agents should command action. Acting optimally allows the controller to utilize the most available computation while still executing the actions on time.

We believe that these contributions are critical components of a situated agent. Though the focus has not yet been on team strategy, tactics or deliberation, a strong theoretical foundation such as is demonstrated in UBCDynamo02 will support further research in these areas and contribute to their success.

References

1. Marc Butler, Mikhail Prokopenko, and Thomas Howard. Flexible Synchronisation within RoboCup Environment: a Comparative Analysis. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup 2000: Robot Soccer. World Cup IV*, Lecture Notes in Artificial Intelligence 2019, pages 119–128. Springer-Verlag, Berlin, 2000.
2. R.E. Kalman. A new approach to linear filtering and prediction problems. Trans. ASME, *Journal of Basic Engineering* **82**(D):35–45, 1960.
3. Jefferson D. Montgomery and Alan K. Mackworth. Adaptive Synchronisation for a RoboCup Agent. Submitted to RoboCup 2002 International Symposium.
4. Ying 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.
5. Ying Zhang and A. K. Mackworth. Constraint Nets: A Semantic Model for Hybrid Dynamic Systems. *Journal of Theoretical Computer Science* **138**(1):211–239, 1995, Special Issue on Hybrid Systems.
6. Ying Zhang and A. K. Mackworth. Modelling and Analysis of Hybrid Systems: An Elevator Case Study. In H. Levesque and F. Pirri, editors, *Logical Foundations for Cognitive Agents*, pages 370–396. Springer, Berlin, 1999.
7. Yu Zhang and A. K. Mackworth. A Constraint-based Controller for Soccer-playing Robots. In proceedings, IEEE Conference on Intelligent Robotic Systems '98 Victoria, B.C., Canada, October 13–17, 1998.
8. Yu Zhang. A Constraint-based Approach to Real-time Cooperative Multiagent Systems: A Soccer-playing Robot Team. Department of Computer Science, University of British Columbia, 1998. M.Sc. Thesis.