On the Role of Device Modelling for Autonomous Agent Control

Ferguson, Innes

Appears in Proc. AAAI Workshop on Representing and Reasoning with Device Function, Seattle, WA, July, 1994

On the Role of Device Modelling for Autonomous Agent Control*

Innes A. Ferguson

Knowledge Systems Laboratory
Institute for Information Technology
National Research Council
Ottawa ON, Canada K1A 0R6

Abstract

It is becoming widely accepted that neither purely reactive nor purely deliberative control techniques are capable of producing the range of behaviors required of intelligent computational agents in dynamic, unpredictable, multi-agent worlds. This paper presents a new architecture for controlling autonomous agents, building on previous work addressing reactive and deliberative control methods. The proposed multi-layered architecture allows a resource-bounded, goal-directed agent to reason predictively about potential conflicts by constructing causal theories or device models which explain other agents' observed behaviors and hypothesize their intentions and function; at the same time it enables the agent to operate autonomously and to react promptly to changes in its real-time environment.

A principal aim of this research is to understand the role different behavioral capabilities play in constraining an agent's function under varying environmental conditions. To this end, an experimental testbed has been constructed comprising a simulated multi-agent world in which a variety of agent configurations and behaviors have been investigated. A number of experimental findings are reported.

1 Introduction

The computer-controlled operating environments at such facilities as automated factories, nuclear power plants, telecommunications installations, and information processing centers are continually becoming more complex. As this complexity grows, it will be increasingly difficult to control such environments with centralized management and scheduling policies that are both robust in the face of unexpected events and flexible at dealing with operational and environmental changes that might occur over time. One solution to this problem which has growing appeal is to distribute, along such dimensions as space and function, the control of such operations to a number of intelligent, task-achieving robotic or computational agents.

Most of today's computational agents are limited to performing a relatively small range of well-defined, pre-programmed, or human-assisted tasks. Operating in real world domains means having to deal with unexpected events at several levels of granularity — both in time and space, most likely in the presence of other independent agents. In such domains agents will typically perform a number of complex simultaneous tasks requiring some degree of attention to be paid to environmental change, temporal constraints, computational resource bounds, and the impact agents' shorter term actions might have on their own or other agents' longer term goals. Also, because agents are likely to have incomplete knowledge about the world and will compete for limited and shared resources, it is inevitable that, over time, some of their goals will conflict. Any attempt

* This research was conducted while the author was a doctoral candidate at the Computer Laboratory, University of Cambridge, Cambridge, UK.

to construct a complex, large-scale system in which all envisaged conflicts are foreseen and catered for in advance is likely to be too expensive, too complex, or perhaps even impossible to undertake given the effort and uncertainty that would be involved in accounting for all of one's possible future equipment, design, management, and operational changes.

Now, while intelligent agents must undoubtedly remain reactive in order to survive, some amount of strategic or predictive decision-making will also be required if agents are to handle complex goals while keeping their long-term options open. On the other hand, agents cannot be expected to model their surroundings in every detail as there will simply be too many events to consider, a large number of which will be of little or no relevance anyway. Not surprisingly, it is becoming widely accepted that neither purely reactive [Bro86, AC87, Sch87] nor purely deliberative [DM90, Sho90, VB90] control techniques are capable of producing the range of robust, flexible behaviors desired of future intelligent agents. What is required, in effect, is an architecture that can cope with uncertainty, react to unforeseen incidents, and recover dynamically from poor decisions. All of this, of course, on top of accomplishing whatever tasks it was originally assigned to do.

This paper is concerned with the design and implementation of a novel integrated agent control architecture, the *TouringMachine* architecture [Fer91, Fer92a, Fer92b, Fer92c], suitable for controlling and coordinating the actions of autonomous rational agents embedded in a partially-structured, dynamic, multi-agent world. Upon carrying out an analysis of the intended TouringMachine task domain — that is, upon characterizing those aspects of the intended real-time road navigation domain that would most significantly constrain the TouringMachine agent design — and after due consideration of the requirements for producing autonomous, effective, robust, and flexible behaviors in such a domain, the TouringMachine architecture has been designed through integrating a number of reactive and *suitably designed* deliberative control functions.

2 TouringMachines

Implemented as a number of concurrently-operating, latency-bounded, task-achieving control layers, the resulting TouringMachine architecture is able to produce a number of reactive, goal-directed, reflective, and predictive behaviors — as and when dictated by the agent's internal state and environmental context. In particular, TouringMachines comprise three such independently motivated layers: a *reactive* layer $\mathcal{R}$ for providing the agent with fast, reactive capabilities for coping with events its higher layers have not previously planned for or modelled (a typical event, for example, would be the sudden appearance of some hitherto unseen agent or obstacle); a *planning* layer $\mathcal{P}$ for generating, executing, and dynamically repairing hierarchical partial plans (which are used by the agent, for example, when constructing navigational routes to some target destination); and a reflective-predictive or *modelling* layer $\mathcal{M}$ for constructing functional device models of world entities, including the agent itself, which can be used as a platform for explaining observed behaviors and making predictions about possible future behaviors (more on this below).

Each control layer is designed to model the agent's world at a different level of abstraction and each is endowed with different task-oriented capabilities. Also, because each layer directly connects perception to action and can independently decide if it should or should not act in a given world state, frequently one layer's proposed actions will conflict with those of another; in other words, each layer is an approximate machine and thus its abstracted world model is necessarily incomplete. As a result, layers are mediated by an enveloping control frame-

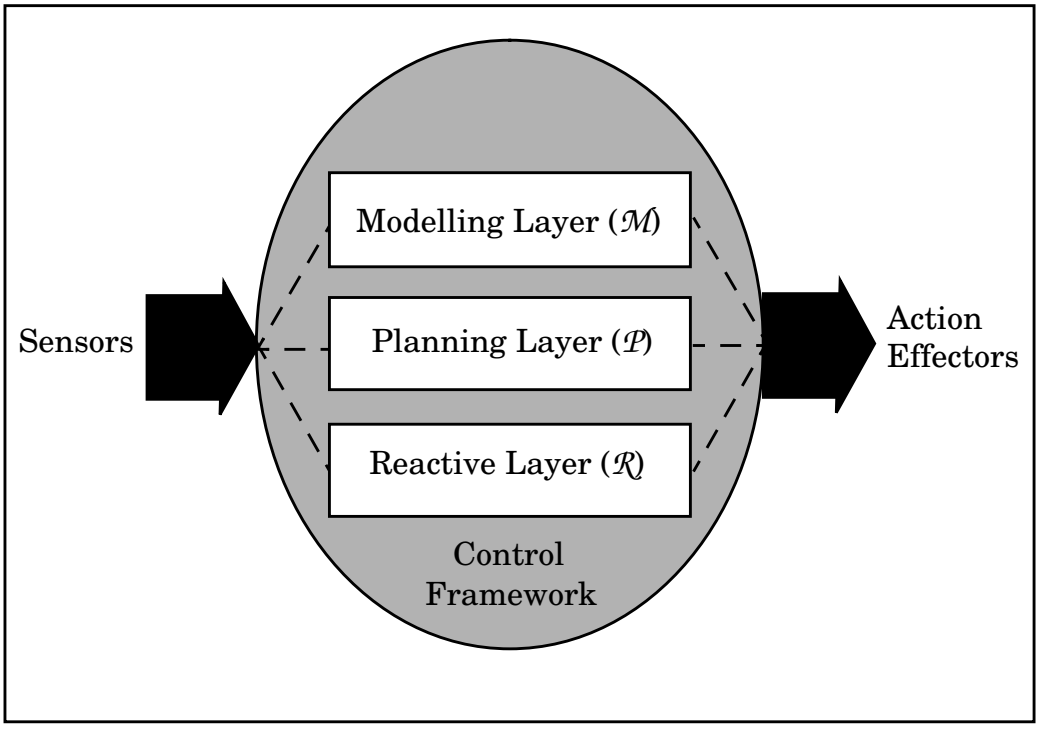

Figure 1: The TouringMachine Architecture.

work (see Figure 1) so that the agent, as a single whole, may behave appropriately in each different world situation. As described in more detail elsewhere [Fer92c], a TouringMachine's control framework is implemented as a combination of inter-layer message-passing and context-activated control rules which are applied repeatedly, in a synchronous fashion, to the inputs and outputs of each of the agent's control layers. The overall control framework thus embodies a scheduling regime which, while striving to service the agent's high-level tasks (e.g. planning, causal modelling, counterfactual reasoning) is sensitive also to its low-level, high-priority behaviors such as avoiding collisions with other agents or obstacles.

3 Modelling Agent Function

Like most real-world domains, a TouringMachine's world is populated by multiple autonomous entities and so will often involve dynamic processes which are beyond the control of any one particular agent. For a planner — and, more generally, for an agent — to be useful in such domains, a number of special skills are likely to be required. Among these are the ability to monitor the execution of one's own actions, the ability to reason about actions that are outside one's own sphere of control, the ability to deal with actions which might (negatively) "interfere" with one another or with one's own goals, and the ability to form contingency plans to overcome such interference. Georgeff [Geo90] argues further that one will require an agent to be capable of coordinating plans of action and of reasoning about the mental state — the beliefs, goals, and intentions — of other entities in the world; where knowledge of other entities' *motivations* is limited or where communication among entities is in some way restricted, an agent will often have to be able to infer such mental state from its observations of entity behavior. Kirsh, in addition, argues that for survival in real-world, human style environments, agents will require the ability to frame and test hypotheses about the future and about other agents' behaviors

[Kir91].

The potential gain from incorporating causal device or mental modelling capabilities in an autonomous agent is that by making successful predictions about entities' activities the agent should be able to detect potential goal conflicts earlier on. This would then enable it to make changes to its own goals or intentions in a more effective manner than if it were to wait for these conflicts to materialize. Goal conflicts can occur within the agent itself (for example, the agent's projected time of arrival at its destination exceeds its original deadline or the agent's layer $\mathcal{R}$ effects an action which alters the agent's trajectory) or in relation to another agent (for example, the agent's trajectory intersects that of another agent). Associated with the different goal conflicts that are known to the agent are a set of conflict-resolution strategies which, once adopted, typically result in the agent taking some action or adopting some new intention.

The device models used by TouringMachines are structured as time indexed 4-tuples of the form $\langle C, B, D, I \rangle$, where C is the entity's *Configuration*, namely *(x,y)*-location, speed, acceleration, orientation, and signalled communications; B is the set of *Beliefs* ascribed to the entity; D is its ascribed list of prioritized goals or *Desires*; and I is its ascribed plan or *Intention* structure. Intention ascription or recognition has been realized in TouringMachines as a process of *scientific theory formation* which employs an abductive reasoning methodology similar to that of the Theorist default/diagnostic reasoning system [PGA86]. In fact, the device models are actually filled-in templates which the agent obtains from an internal model library. While all templates have the same basic 4-way structure, they can be made to differ in such aspects as the depth of information that can be represented or reasoned about (for example, a particular template's B component might dictate that modelled beliefs are to be treated as defeasible), initial default values provided, and computational resource cost. The last of these will subsequently be taken into account each time the agent makes an inference from the chosen model.

Reasoning from a model of an entity essentially involves looking for the "interaction of observation and prediction" [DH88]; that is, for any discrepancies between the agent's *actual* behavior and that *predicted* by its model or, in the case of a self-model, between the agent's actual behavior and that *desired* by the agent. Model-based reasoning in TouringMachines specifically comprises two phases: *explanation* and *prediction*. During the explanation phase, the agent attempts to generate plausible or inferred functional explanations about any entity (object/agent) behaviors which have recently been observed. Explanations — a set of consistent hypotheses about the entity's intended state which logically imply the observations made of the entity —are then used in detecting discrepancies between these entities' current behaviors and those which had been anticipated from previous encounters. If any such discrepancies are detected, the agent will then strive to infer, via intention ascription, plausible explanations for their occurrence.

Once all model discrepancies have been identified and their causes inferred, predictions are formed by temporally projecting those parameters that make up the modelled entity's configuration vector C in the context of the current world situation and the entity's functional model (principally, its ascribed intention). The space-time projections (in effect, knowledge-level simulations*)* thus created are used by the agent to detect any potential interference or goal conflicts among the modelled entities' anticipated/desired actions. Should any conflicts — intra- or inter-agent — be identified, the agent will then have to determine how such conflicts might best be resolved, and also which entities will be responsible for carrying out these resolutions. Determining such resolutions, particularly where multiple goal conflicts are involved, will require consideration of a number of issues, including the priorities of

the different goals affected, the space-time urgency of each conflict, rights-of-way protocols in operation, as well as any environmental and physical situational constraints (e.g. the presence of other entities) or motivational forces (e.g. an agent's own internal goals) that may constrain the possible actions that the agent can take. In the TouringMachine architecture such knowledge is encapsulated in a library of domain-dependent conflict resolution triples, each of which relates a specific conflict state description with the particular agent goal under threat as well as the recommended recovery procedure or resolution that should be adopted [Fer92c].

The device models employed here are in fact functional models since their purpose is to give the agent a global view of what other observed entities are doing [HBJ91]. In particular, an agent uses models to abstract, organize, and index the relevant aspects of the behavioral knowledge it acquires about other entities according to: *(i)* the (hypothesized) intentions which the agent considers to be "responsible" for the entities' observed behaviors; and *(ii)* the current context within which these entities are operating. So, unlike observed behaviors which can be interpreted independently of the context within which they were produced, the definition of device model employed here emphasizes the context-dependent nature of agent function by taking into account the agent's intended state along with the background (beliefs, desires) and environmental (physical and situational) conditions under which the state can be achieved.

4 Experiments with TouringMachines

The research presented here adopts a fairly pragmatic approach toward understanding how complex environments might constrain the design of agents, and, conversely, how different task constraints and behavioral capabilities within agents might combine to produce different agent functions. In order to evaluate TouringMachines, a highly instrumented, parametrized, multi-agent simulation testbed has been implemented in conjunction with the TouringMachine control architecture. The testbed provides the user with a 2-dimensional world — the TouringWorld — which is occupied by, among other things, multiple TouringMachines, obstacles, walls, paths, and assorted information signs. World dynamics are realized by a discrete event simulator which incorporates a plausible world updater for enforcing "realistic" notions of time and motion, and which creates the illusion of concurrent world activity through appropriate action scheduling. Other processes handled by the simulator include a facility for tracing agent and environmental parameters, a statistics gathering package for agent performance analysis, a mechanism enabling the testbed user to control the motion of a chosen agent, and several text and graphics windows for displaying output. By enabling the user to specify, visualize, measure, and analyze any number of user-customized agents in a variety of single- and multi-agent settings, the testbed provides a powerful platform for the empirical study of autonomous agent function.

A number of experiments have been carried out on TouringMachines which illustrate, in particular, that the balance between goal-orientedness (effectiveness) and reactivity (robustness) in agents can be affected by a number of factors including, among other things, the level of detail involved in the predictions agents make about each other, the degree of sensitivity they demonstrate toward unexpected events, and the proportion of total agent resources that are made available for constructing plans or building mental models of other agents' functions. Other experiments point toward a trade off between the reliability and the efficiency of the predictions an agent can make about the future (this turns out to be an instance of the well-known *extended prediction problem* [SM90]). Yet other experiments

have been carried out which suggest that predicting future world states through causal modelling of agents' mental states or function, can, in certain situations, prove useful for promoting effective coordination between agents with conflicting goals. To illustrate some of the diverse opportunities for analysis which are afforded by the TouringMachine testbed, one particular experiment is now described in some detail.

4.1 Monitoring the environment: sensitivity versus efficiency

In monitoring the state of another world entity, and in particular, in determining whether the information it maintains of an entity's current physical configuration (its location, speed, orientation, etc.) is as it should be — that is, satisfies the expectations which were computed when it last projected the entity's functional model in space-time — a TouringMachine makes use of various tolerance bounds to decide whether any discrepancies in fact exist. As with any discrepancies detected in the agent's self model, identification of a discrepancy in the model of another entity typically requires further investigation to determine its cause. Often this reasoning process results in having to re-explain the entity's current behavior by ascribing it a new functional role —and in particular, a new intention. For example, a discrepancy between the modelled entity's current and expected speeds might be indicative of the entity's change of intention from, say, `drive-along-path` to `stop-at-junction`.

In Figure 2 (upper two frames) we can see, at two different time points $T = 12.5$ seconds and $T = 15.5$ seconds, several agents in pursuit of their respective goals: `agent1` (round), `agent2` (chevron-shaped), and `agent3` (triangular, top-most). Furthermore, we can see the effect on `agent1`'s behavior — that is, on its ability to carry out its pre-defined homeostatic goal `avoid-collisions` — of modifying the value of **ModelSpeedBounds**, an internal agent parameter which, when modelling another entity, is used to constrain the "allowable" deviations between this entity's currently observed speed and the speed it was predicted to have had when the entity was last observed. In this scenario, `agent1` has to contend with the numerous and unexpected speed changes effected by `agent2`, a testbed user-driven agent. With fairly tights bounds (for example **ModelSpeedBounds** = +/-0.5 ms^{-1}), `agent1` detects any speed discrepancies in `agent2` which are greater than or equal to 0.5 ms^{-1}. Among such discrepancies detected by `agent1` are those which result from `agent2`'s deceleration just prior to its coming to a halt at a junction at time $T = 20.0$ (Figure 2, lower left-hand frame). As a result, and compared to the situation when `agent1` is configured with **ModelSpeedBounds** = +/-2.0 ms^{-1}, and therefore, in this particular scenario, unable to detect or respond to `agent2`'s actions at $T = 20.0$ (Figure 2, lower right-hand frame), the configuration with tighter speed bounds is more robust, more able to detect "important" events (for example, the agent in front coming to a halt) and also more able to carry out timely and effective intention changes (for example, from `drive-along-path` to `stop-behind-agent`).

This in itself, of course, does not suggest that agents should always be configured with tight speed bounds. Sensitivity or robustness to environmental change can come at a price in terms of increased resource consumption: each time an agent detects a model discrepancy it is forced by design to try to explain the discrepancy through a (relatively expensive) process of abductive intention ascription. Often, however, small changes in the physical configuration of a modelled entity need not be the result of the entity having changed intentions. In the scenario of Figure 2, for example, `agent2`'s speed changes are due entirely to actions effected by the testbed user. Ignorant of this, however, `agent1` configured with **ModelSpeedBounds** = +/-0.5 ms^{-1} will continually attempt to re-explain

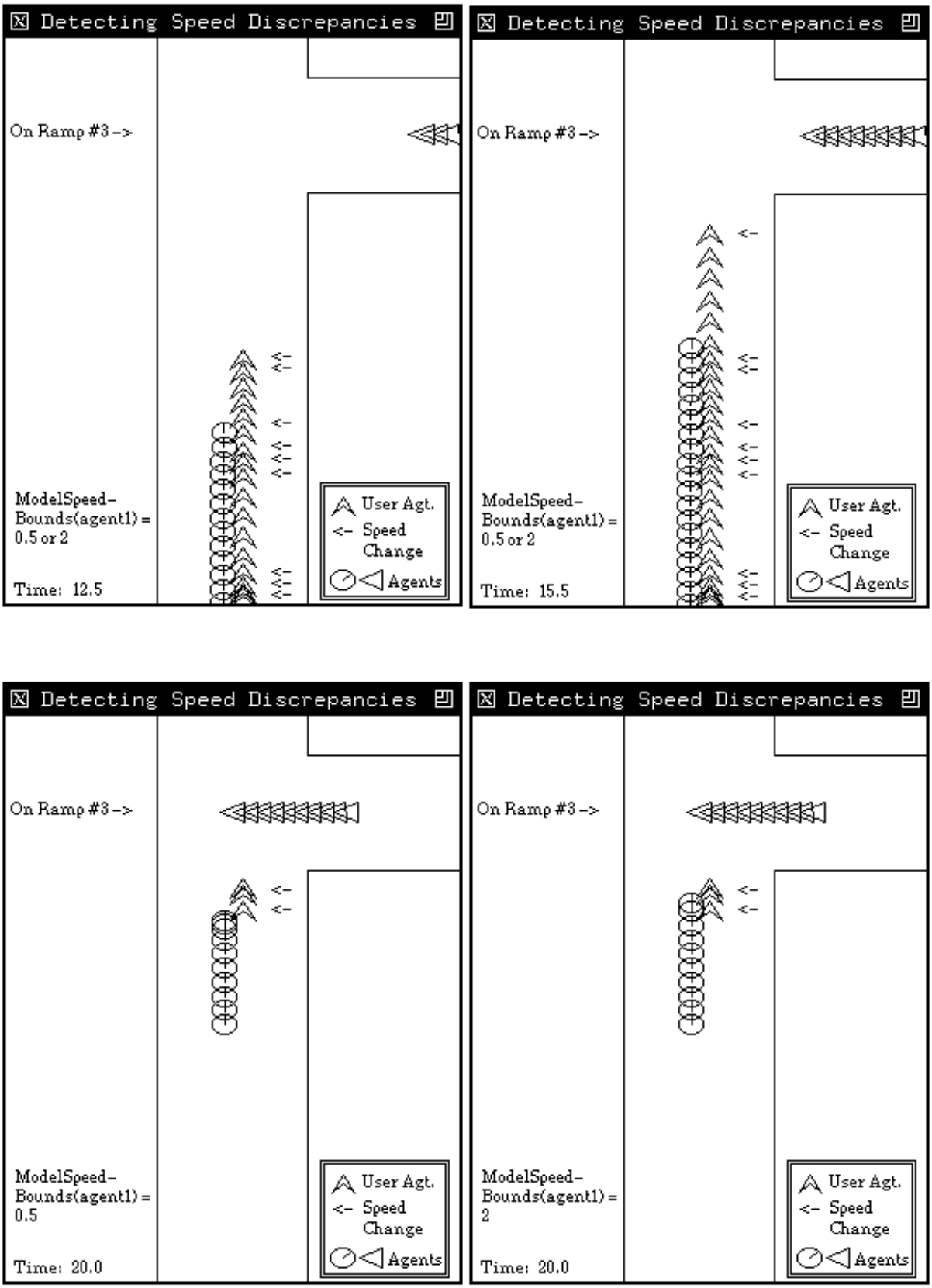

Figure 2: Varying the value of an agent's **ModelSpeedBounds** parameter can affect the agent's level of sensitivity to environmental change.

agent2's changing behavior — despite the fact that this reasoning process will always, except in the case when agent2 stops at the junction, return the same explanation of drive-along-path. Also, although not elaborated on in this paper, it is also important to note that a TouringMachine may only monitor the state of its *own* layer $\mathcal{M}$ goals when there are exactly zero discrepancies to attend to in its current set of modelled (external) agents. A less environmentally sensitive agent, therefore, might well end up with more opportunities to monitor its own progress and so, potentially, achieve its goals more effectively.

5 Conclusions

Through the above and a number of other coordination experiments addressing such issues as the production of emergent behavioral patterns, the TouringMachine architecture has been shown to be feasible and that, when suitably parametrized, can endow rational autonomous agents with appropriate levels of effective, robust, and flexible control for successfully carrying out multiple goals while simultaneously dealing with a number of dynamic multi-agent events.

Among other things, the TouringMachine architecture can be seen as a practical framework for modelling device function within the context of real-time multi-agent environments. In particular, the functional modelling approach adopted helps to abstract and organize reasoning about observed behavior by enabling agents to focus only on those aspects of modelling which are pertinent for constructing approximate — but nevertheless useful — explanations and predictions of activity in the environment.

The integration of a number of traditionally expensive deliberative reasoning mechanisms (for example, functional modelling and hierarchical planning) with reactive or behavior-based mechanisms is a challenge which has been addressed in the TouringMachine architecture. Additional challenges such as enabling effective agent operation under real-time constraints and with bounded computational resources have also been addressed. The result is a novel architectural design which can successfully produce a range of useful behaviors required of sophisticated autonomous agents embedded in complex environments.

The research presented here is ongoing; current work on the TouringMachine agent architecture includes an effort to generalize further the TouringWorld testbed, in particular, by separating the definition of the agent's domain of operation (description of the environment, initial goals to accomplish, criteria for successful completion of goals) from the specified configuration (capabilities, internal parameters and constraints) of the agent itself. Another aspect of the current work is to identify and incorporate new capabilities in order to extend the functional repertoire of agents; capabilities being considered at present include, among others, inductive learning, user modelling, and episodic memory management. Relatedly, a new domain to which TouringMachines are currently being applied involves adaptive information retrieval and filtering on the World Wide Web.

References

[AC87] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 268—272, 1987.

[Bro86] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14—23, 1986.

[DH88] Randall Davis and Walter Hamscher. Model-based reasoning: Troubleshooting. In Howard E. Shrobe and AAAI, editors, *Exploring Artificial Intelligence*, pages

297—346. Morgan Kaufmann: San Mateo, CA, 1988.

[DM90] Edmund H. Durfee and Thomas A. Montgomery. A hierarchical protocol for coordinating multiagent behaviors. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 86—93, 1990.

[Fer91] Innes A. Ferguson. Toward an Architecture for Adaptive, Rational, Mobile Agents (extended abstract). In M.P. Georgeff and A.S. Rao (eds.) *Proc. of the IJCAI-91 Workshop on Theoretical and Practical Design of Rational Agents*, Sydney, Australia, August, 1991.

[Fer92a] Innes A. Ferguson. TouringMachines: Autonomous Agents with Attitudes. *IEEE Computer*, 25(5), 51—55, 1992.

[Fer92b] Innes A. Ferguson. Toward an Architecture for Adaptive, Rational, Mobile Agents. In E. Werner and Y. Demazeau (eds.) *Decentralized AI 3*, Elsevier Science (North Holland): Amsterdam, 1992.

[Fer92c] Innes A. Ferguson. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. Ph.D. thesis, Computer Laboratory, University of Cambridge, Cambridge UK, 1992.

[Geo90] Michael P. Georgeff. Planning. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 5—25. Morgan Kaufmann: San Mateo, CA, 1990.

[HBJ91] Ameen Abu-Hanna, Richard Benjamins, and Wouter Jansweijer. Integrating Functional Models in Model Based Diagnosis. In *Proceedings AAAI-91 Workshop on Model-Based Reasoning*, Anaheim CA, July, 1991.

[Kir91] David Kirsh. Today the earwig, tomorrow man? *Artificial Intelligence*, 47:161—184, 1991.

[PGA86] David L. Poole, Randy G. Goebel, and Romas Aleliunas. Theorist: A logical reasoning system for defaults and diagnosis. Research Report CS-86-06, University of Waterloo, Waterloo, Ont., February 1986.

[Sch87] M.J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 1039—1046, 1987.

[Sho90] Yoav Shoham. AGENT0: A simple agent language and its interpreter. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 704—709, 1991.

[SM90] Yoav Shoham and Drew McDermott. Problems in formal temporal reasoning. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 581—587. Morgan Kaufmann: San Mateo, CA, 1990.

[VB90] Steven Vere and Timothy Bickmore. A basic agent. *Computational Intelligence*, 6(1):41—60, 1990.