On the Role of BDI Modelling for Integrated Control and Coordinated Behaviour in Autonomous Agents

Ferguson, Innes

Publisher’s version / Version de l'éditeur:

Journal of Applied Artificial Intelligence, 9, 4, 1995

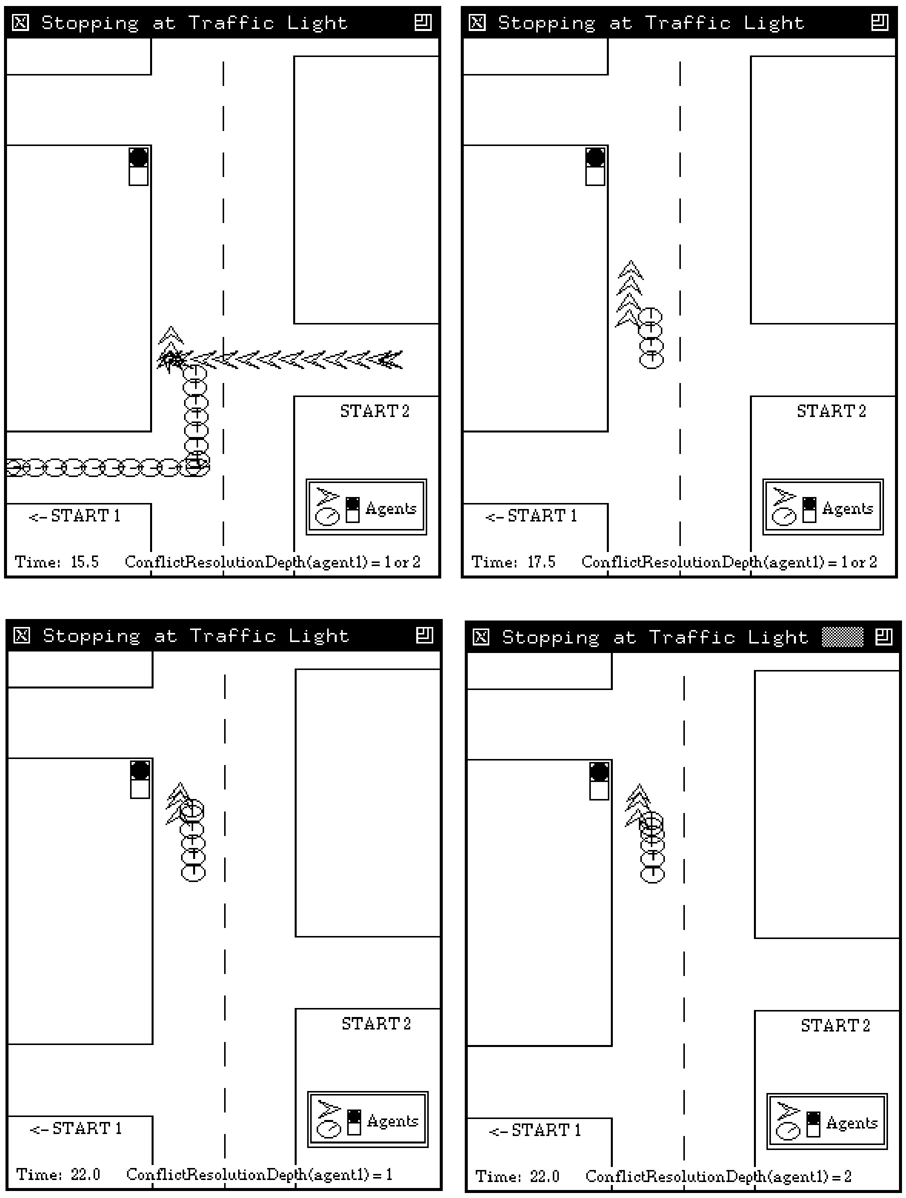

Stopping at Traffic Light
START 2
<- START 1
Agents
Time: 15.5 ConflictResolutionDepth(agent1) = 1 or 2
Stopping at Traffic Light
START 2
<- START 1
Agents
Time: 17.5 ConflictResolutionDepth(agent1) = 1 or 2
Stopping at Traffic Light
START 2
<- START 1
Agents
Time: 22.0 ConflictResolutionDepth(agent1) = 1
Stopping at Traffic Light
START 2
<- START 1
Agents
Time: 22.0 ConflictResolutionDepth(agent1) = 2

Figure 5. Altering the value of an agent's **ConflictResolutionDepth** parameter can affect the timeliness and effectiveness of any predictions it might make.

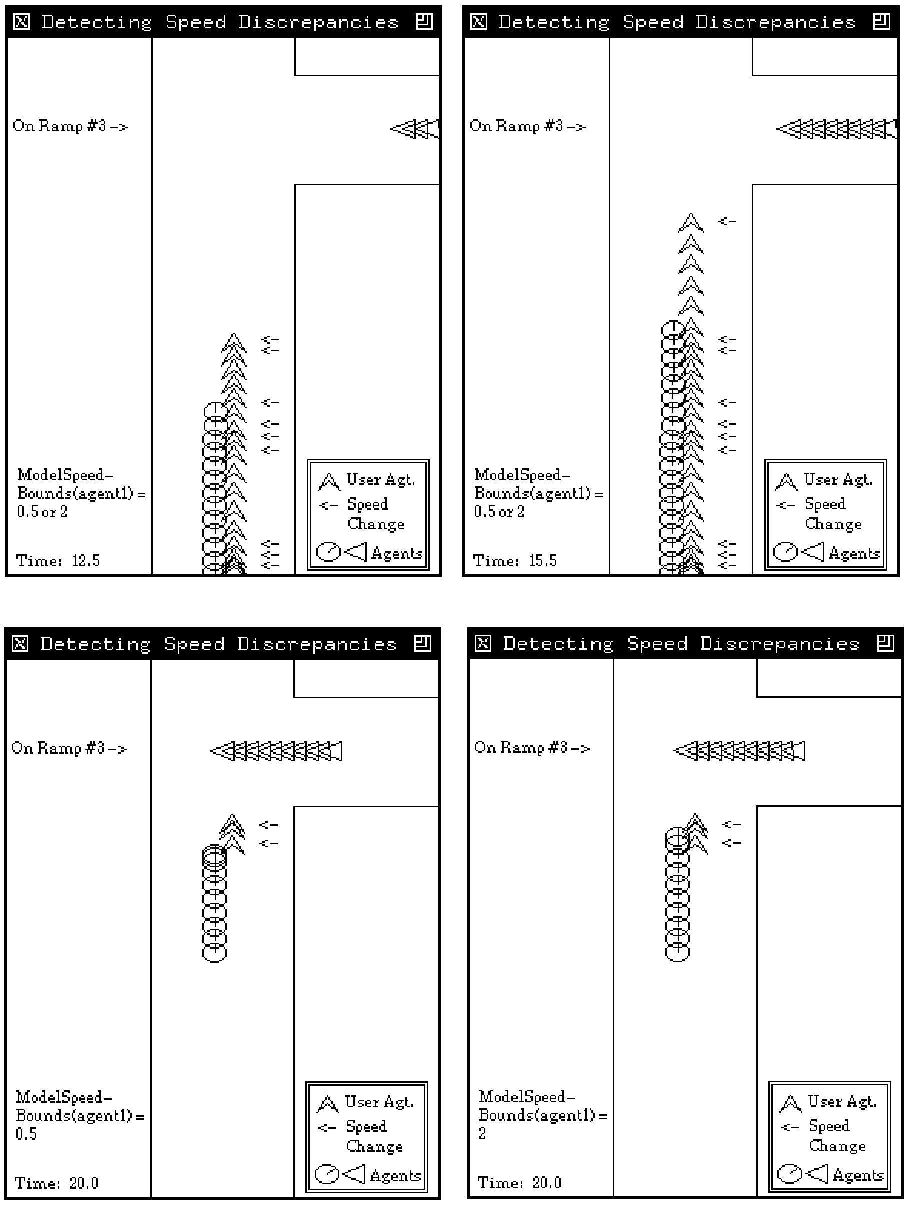
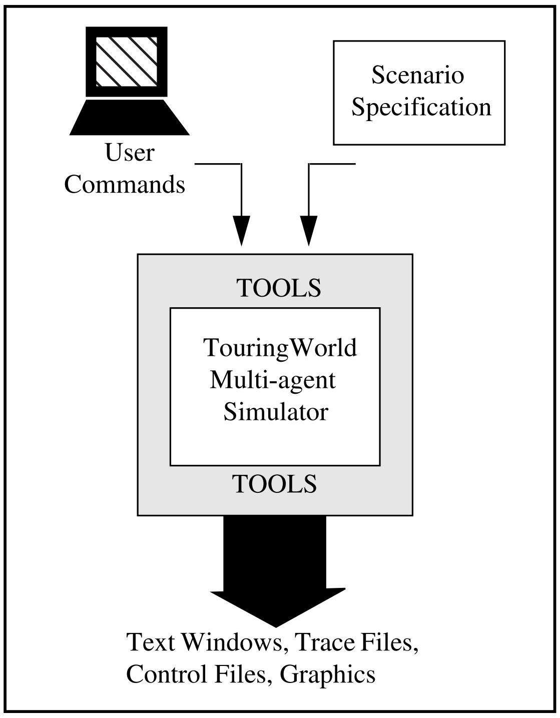
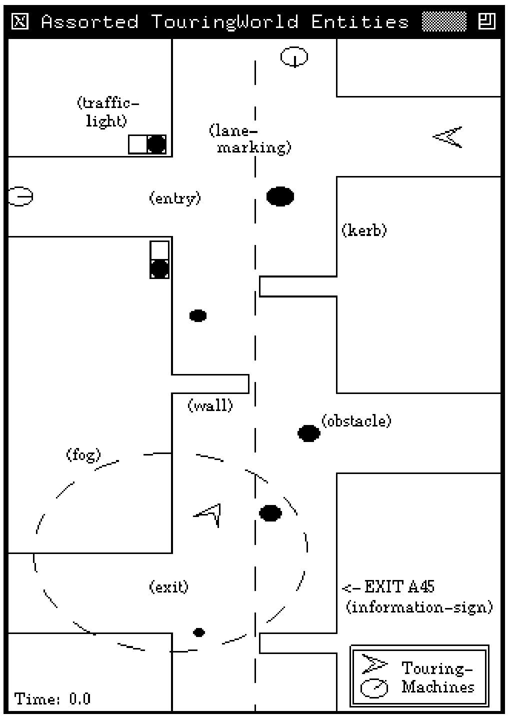

Detecting Speed Discrepancies
On Ramp #3 ->
ModelSpeed-Bounds(agent1) = 0.5 or 2
Time: 12.5
User Agt.
<- Speed Change
Agents
Detecting Speed Discrepancies
On Ramp #3 ->
ModelSpeed-Bounds(agent1) = 0.5 or 2
Time: 15.5
User Agt.
<- Speed Change
Agents
Detecting Speed Discrepancies
On Ramp #3 ->
ModelSpeed-Bounds(agent1) = 0.5
Time: 20.0
User Agt.
<- Speed Change
Agents
Detecting Speed Discrepancies
On Ramp #3 ->
ModelSpeed-Bounds(agent1) = 2
Time: 20.0
User Agt.
<- Speed Change
Agents

Figure 4. Varying the value of an agent's **ModelSpeedBounds** parameter can affect the agent's level of sensitivity to environmental change.


User Commands
Scenario Specification
TOOLS
TouringWorld Multi-agent Simulator
TOOLS
Text Windows, Trace Files, Control Files, Graphics



(a) **(b)**

Figure 3. (a) Top-level view of the TouringWorld multi-agent testbed. **(b)** A snapshot of a particular TouringWorld scenario showing various types of (labelled) objects and agents.

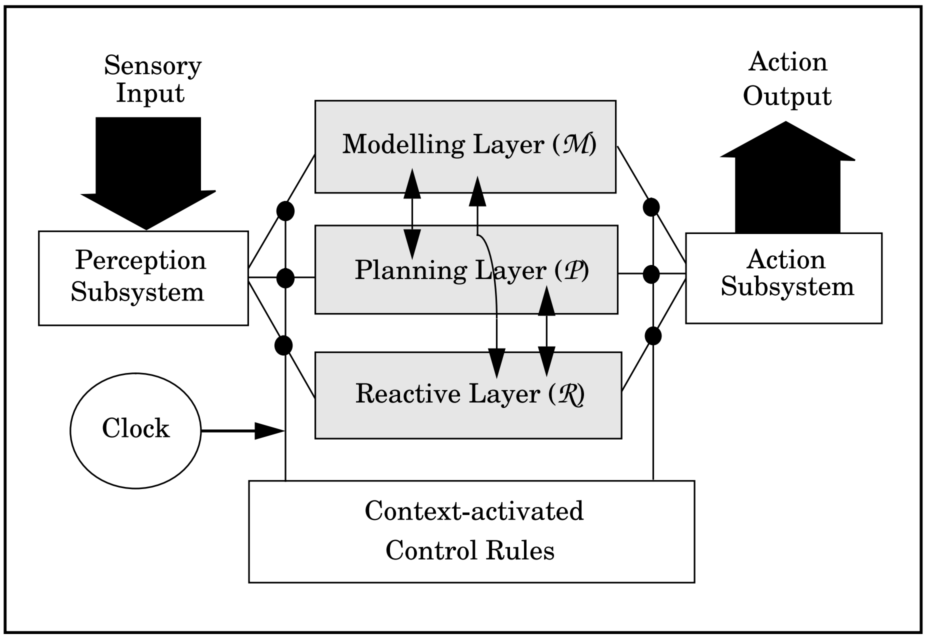


Figure 1. A TouringMachine's mediating control framework.

```
censor-rule-1:
    if   entity(obstacle-6) ∈ Perception-Buffer
    then
         remove-sensory-record(layer-𝓡, entity(obstacle-6))

suppressor-rule-3:
    if   action-command(layer-𝓡-rule-6*,
                      change-orientation(_))† ∈ Action-Buffer
         and
         current-intention(start-overtake)
    then
         remove-action-command(layer-𝓡, change-orientation(_))
         and
         remove-action-command(layer-𝓜, _)
```

* `layer-𝓡-rule-6` is the reactive (layer 𝓡) rule which is invoked in order to avoid crossing a path lane marking.

† "_" simply denotes a don't-care or anonymous variable.

Figure 2. Two example control rules: **censor-rule-1** and **suppressor-rule-3**.

Figure Captions:

Figure 1. A TouringMachine's mediating control framework.

Figure 2. Two example control rules: **censor-rule-1** and **suppressor-rule-3**.

Figure 3. (a) Top-level view of the TouringWorld multi-agent testbed. **(b)** A snapshot of a particular TouringWorld scenario showing various types of (labelled) objects and agents.

Figure 4. Varying the value of an agent's **ModelSpeedBounds** parameter can affect the agent's level of sensitivity to environmental change.

Figure 5. Altering the value of an agent's **ConflictResolutionDepth** parameter can affect the timeliness and effectiveness of any predictions it might make.

Shoham, Y. 1993. Agent-oriented programming. *Artificial Intelligence*, 60(1):51—92.

Vere, S. and Bickmore, T. 1990. A basic agent. *Computational Intelligence*, 6(1):41—60.

Miller, M.S. and Drexler, K.E. 1988. Markets and computation: agoric open systems. In B.A. Huberman, editor, *The Ecology of Computation*, pages 133—176. Elsevier Science Publishers B.V. (North-Holland), The Netherlands.

Minsky, M.L. 1986. *The Society of Mind.* Simon and Schuster: NY, NY.

Müller, J.P. and Pischel, M. 1994. Integrating agent interaction into a planner-reactor architecture. In *Proceedings 13th International Workshop on Distributed Artificial Intelligence*, Lake Quinalt, WA, July, 1994.

Pollack, M.E. and Ringuette, M. 1990. Introducing the Tileworld: Experimentally evaluating agent architectures. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 183—189.

Poole, D.L., Goebel, R.G., and Aleliunas, R. 1986. Theorist: A logical reasoning system for defaults and diagnosis. Research Report CS-86-06, University of Waterloo, Waterloo, ON, February.

Poole, D. 1988. Representing knowledge for logic-based diagnosis. In *Proceedings International Conference on Fifth Generation Computer Systems*, pages 1282—1289.

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

Shoham, Y. and McDermott, D. 1990. 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.

Finin, T., McKay, D., and Friztson, R. 1992. An overview of KQML: A knowledge query and manipulation language. Available through the Stanford University Computer Science Department, Palo Alto, CA.

Galliers, J.R. 1990. The positive role of conflict in cooperative multi-agent systems. In Jean-Pierre Müller and Yves Demazeau, editors, *Decentralized AI*. Elsevier Science Publishers B.V. (North-Holland), The Netherlands.

Georgeff, M.P. 1990. Planning. In J. Allen, J. Hendler, and A. Tate, editors, *Readings in Planning*, pages 5—25. Morgan Kaufmann: San Mateo, CA.

Hanks, S. and Firby, J. 1990. Issues and architectures for planning and execution. In *Proceedings DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 59—70. Morgan Kaufmann: San Mateo, CA.

Kinny, D.N. and Georgeff, M. 1991. Commitment and effectiveness of situated agents. Technical Note 17, Australian Artificial Intelligence Institute, Carleton 3053, Australia, April.

Kirsh, D. 1991. Today the earwig, tomorrow man? *Artificial Intelligence*, 47:161—184.

Langley, P. 1988. Machine learning as an experimental science. *Machine Learning*, 3:5—8.

Maes, P. 1990. Situated agents can have goals. *Robotics and Autonomous Systems*, 6(1&2):49—70.

Maes, P. 1994. Modeling Adaptive Autonomous Agents. *Artificial Life*, 1:135—162.

Pulling together or pulling apart. *AI Magazine*, 12(1):16—41.

Covrigaru, A.A. and Lindsay, R.K. 1991. Deterministic autonomous agents. AI Magazine, 12(3):110—117.

Davis, E. 1990. *Representations of Commonsense Knowledge*. Morgan Kaufmann: San Mateo, CA.

Davis, R. and Hamscher, W 1988. Model-based reasoning: Troubleshooting. In Howard E. Shrobe and AAAI, editors, *Exploring Artificial Intelligence*, pages 297—346. Morgan Kaufmann: San Mateo, CA.

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

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

Ferguson, I.A. 1994. Autonomous agent control: a case for integrating models and behaviors. In *Working Notes AAAI Fall Symposium on Control of the Physical World by Intelligent Agents*, New Orleans, LA, pages 46—54. NRC 38326.

Ferguson, I.A. 1995. Integrating models and behaviors in autonomous agents: some lessons learned on action control. In *Working Notes AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, Palo Alto, CA, March. NRC 38340.

References

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

Baclace, P.E. 1992. Competitive agents for information filtering. *Communications of the ACM*, 35(12):50.

Bond, A.H. and Gasser, L., editors 1988. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann: Palo Alto, CA.

Bratman, M.E., Israel, D.J., and Pollack, M.E. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349—355.

Brooks, R.A. 1991. Intelligence without representation. *Artificial Intelligence*, 47:139—159.

Carberry, S. 1990. *Plan Recognition in Natural Language Dialogue*, MIT Press: Cambridge, MA.

Chandrasekaran, B. 1994. Understanding control at the knowledge level. In *Working Notes AAAI Fall Symposium on Control of the Physical World by Intelligent Agents*, New Orleans, LA, pages 19—26.

Cohen, P.R., Greenberg, M.L., Hart, D.M., and Howe, A.E. 1989. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):32—48.

Cohen, P.R. 1991. A survey of the Eighth National Conference on Artificial Intelligence:

internal behavior APIs, libraries of code implementing a variety of different sensory and effectory apparatus (e.g. for interpreting and executing commands on the UNIX file system or for HTML page processing), persistent storage management, and (currently under consideration) CORBA[*] compliance.

Another aspect of the current work is to identify and incorporate new capabilities in order to extend the behavioral repertoire of agents; capabilities being considered at present include, among others, reinforcement learning, user modelling, and episodic memory management. Relatedly, a new domain to which TouringMachines —actually, CALVIN agents —are currently being applied involves adaptive information retrieval and filtering on the World Wide Web. In particular, this application comprises a system of information processing agents which can retrieve HMTL documents (e.g. pages from the various Usenet newsgroups) and filter their contents according to a number of weighted document features such as subject, author, chosen keywords, (syntactic) structural organization, and shared ratings from colleagues who have previously read the document(s). The agents, which are specialized by the particular document feature(s) they search on, have limited resources which they must continually compete for: agents which retrieve documents that elicit positive relevance feedback from the user are rewarded; those returning documents that elicit negative feedback are penalized. The agents, thus, comprise an agoric open system (Miller and Drexler, 1988; Baclace, 1992) which adapts through agents learning how to prioritize information that best reflects the user's personal interests.

†. The Knowledge Query and Manipulation Language (KQML) is a protocol intended to support interoperability among intelligent agents in distributed applications (Finin et al., 1992).

*. The Object Management Group's Common Object Request Broker Architecture (CORBA).

cannot underestimate the importance of deploying — from the earliest stages of design — concrete measures for carrying out extensive experimentation. In this respect, the TouringWorld Testbed domain has proved a viable and useful system for evaluating different agent control designs.

The integration of a number of traditionally expensive deliberative reasoning mechanisms (for example, causal 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 powerful 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 and behaviors, internal parameters and constraints) of the agent itself. Relatedly, the TouringMachine architecture is currently being recast as CALVIN[*] — a platform independent, Perl-scripted, open implementation of the TouringMachine architecture which provides application developers with a powerful set of programming tools, including libraries of intra- and inter-agent protocols (e.g. KQML[†]),

*. The Communicating Agents Living Vicariously In Networks (CALVIN) architecture is an agent framework developed at the National Research Council's Knowledge Systems Laboratory (see World Wide Web page http://ai.iit.nrc.ca/software.html for more details).

and productive framework in which to carry out such design activities was thus established. Examples of reviewable TouringMachine design decisions — established with empirical feedback gained through use of the TouringWorld Testbed — include the particular set of reactive rules initially made available to the agent, the contents of its various domain-specific libraries (plan schemas, BDI model templates, conflict-resolution methods, and space-time projection functions), the initial set of heuristics used to program the agent's focus of attention mechanisms, and the precise set of censor and suppressor control rules which are used to mediate the actions of the agent's three control layers.

In conclusion, the evaluation of TouringMachines appeared to support the claim that it is both desirable and feasible to combine non-deliberative and suitably designed and integrated deliberative control functions in a single, hybrid, autonomous agent architecture. As was demonstrated, the resulting architecture, when suitably configured, was capable of effective, robust, and flexible behaviors in a reasonably wide range of complex single- and multi-agent task scenarios. As described above, the behavioral repertoire of TouringMachines is wide and varied, including behaviors which are reactive, goal-oriented, reflective, and also predictive. The evaluation, furthermore, suggested that establishing an appropriate balance between reasoning and acting — that is, between appropriate degrees of deliberative and non-deliberative control — would appear to depend on characteristics of the task environment in which the particular TouringMachine is operating. More generally, and in line with the experiences of both Maes (Maes, 1990) and Pollack (Pollack and Ringuette, 1990), there is evidence to suggest that environmental factors invariably play an important role in determining which agent configuration or parametrization is the most appropriate for any given situational context. Finally, one deliberative functions be: *(i)* latency-bounded in order to provide guaranteed system responsiveness (this in turn demands fairly strict control over internal computational resource use); *(ii)* that they operate incrementally (in other words, that they be capable of suspending operation and state after regular — and arbitrarily small — periods of processing time); and *(iii)* that they serve merely as *resources* for action rather than as strict *recipes* for overall agent control. This last requirement would also become the main motivating force behind the decision to employ a context-sensitive mediatory control policy for establishing control layer priorities. Other design decisions worth mentioning here include the incorporation of functions for reasoning about — or modelling — other agents' actions and mental states and for identifying and flexibly resolving conflicts within and between agents (this is necessary because the TouringWorld domain is populated by multiple intentional agents with limited computational and informational resources); and mechanisms for constantly sensing and monitoring the external world (which are needed since the TouringWorld domain is both dynamic and unpredictable).

Identification and isolation of the second type of design decisions, reviewable decisions, are those which, as their name suggests, can be “reviewed after they are implemented” (Cohen 1991, page 31). The purpose of differentiating fixed and reviewable design decisions was to enable the basic (fixed) design to be implemented and run as early as possible, and to provide an empirical environment in which to develop iteratively this basic agent model, to test hypotheses about how the model should behave, and then to review, subsequently, particular design decisions in the light of observed performance. Also, by providing — in addition to the TouringMachine agent architecture — a highly parametrized and controllable testbed environment like the TouringWorld, a very effective

6 Discussion and Future Work

Apart from matters arising directly from the evaluation process described above, a number of experiential and implementational issues which bear on the applicability and appropriateness of the TouringMachine architecture also merit addressing at this point. As mentioned above, the first stage in designing the TouringMachine architecture involved an analysis of the intended TouringMachine task environment: that is, a characterization of those aspects of the intended environment which would most significantly constrain the TouringMachine agent design. As will now be argued, the main purpose of this analysis was to differentiate between, and therefore establish, what Cohen (Cohen, 1991) terms the system's *fixed* and *reviewable* design decisions.

Fixed design decisions are those which “will not be reviewed anytime soon” (Cohen, 1991, page 31). In the TouringMachine architecture, these design decisions were established upon close examination of the intended TouringWorld domain. For instance, the decision to divide control among multiple, independent concurrent layers was influenced by the fact that TouringMachines would have to deal flexibly and robustly with any number of simultaneous events, each occurring at a potentially different level of space-time granularity. Such differences in event granularity, together with the need for carrying out both long-term, deadline-constrained tasks, as well as short-term reactions to unexpected events, ultimately played a part in the decision to combine both deliberative and non-deliberative control functions into a single hybrid architecture. In turn, the need to ensure that any such (non-real-time) deliberative functions be “suitable” for use in a real-time domain such as the TouringWorld — in other words, that they be efficient and effective on the one hand but flexible and robust on the other — suggested that such conflict — the one between itself and `agent2` — until one clock cycle later; that is, at time $T = 18.0$ instead of at $T = 17.5$. Due to the proximity of the two agents, the relatively high speed of `agent1`, and the inevitable delay associated with any change in intention or momentum, this 0.5 second delay proves to be sufficiently large to make agent1 realize too late that `agent2` is going to stop; an inevitable rear-end collision therefore occurs at $T = 22.0$ (Figure 5, lower left-hand frame).[*] Configured with **ConflictResolutionDepth** = 2 (Figure 5, lower right-hand frame), `agent1` ends up having enough time — an extra 0.5 seconds — to adopt and realize the appropriate intention `stop-behind-agent`, thereby avoiding the collision that would otherwise have occurred.

Having the flexibility to reason about the interactions between other world entities (for example, between `agent2` and the traffic light) and to take into account the likely future intentions of these entities (for example, `stop-at-light`) can enable TouringMachines like `agent1` to make timely and effective predictions about the changes that are taking place or that are likely to take place in the world. In general, however, knowing how deeply agents should model one another is not so clear: since the number of layer $\mathcal{M}$ resources required to model world entities is proportional to both the *number* of entities modelled and the (counterfactual reasoning) *depth* to which they are modelled, agents will ultimately have to strike a balance between breadth of coverage (more entities modelled, little detail) and depth of coverage (less entities, more detail). This issue is investigated in more detail elsewhere (Ferguson, 1992).

*. In fact, this collision need not be “inevitable”: in this scenario both `agent1` and `agent2` have been configured with fairly insensitive (not very robust) layer $\mathcal{R}$ reactions, primarily to emphasize the different behaviours that *could* result from different parametrizations of agents' modelling capabilities.

conflicts.

In the scenario of Figure 5, two TouringMachine agents can be seen following independent routes to one destination or another. The interesting agent to focus on here — the one whose configuration is to be varied — is `agent1` (the round one). The upper left-hand frame of Figure 5 simply shows the state of the world at time *T* = 15.5 seconds. Throughout the scenario, each agent continually updates and projects the models they hold of each other, checking to see if any conflicts might be "lurking" in the future. At *T* = 17.5 (upper right-hand frame of Figure 5), `agent1` detects one such conflict: an `obey-regulations`[*] conflict which will occur at *T* = 22.0 between `agent2` (chevron-shaped) and the traffic light (currently red). Now, assuming `agent1` is just far enough away from the traffic light so that it does not, within its parametrized conflict detection horizon, see any conflict between itself and the traffic light, then, if `agent1` is configured with **ConflictResolutionDepth** = 1, it will predict the impending conflict between `agent2` and the traffic light, as well as the likely event of `agent2` altering its intention to `stop-at-light` so that it will come to a halt at or around *T* = 22.0. If, on the other hand, `agent1` is configured with **ConflictResolutionDepth** = 2, not only will it predict the same conflict between `agent2` and the traffic light and the resolution to be realized by this entity, but it will also, upon hypothesizing about the world state *after* this conflict resolution is realized, predict another impending conflict, this second one involving itself and the soon to be stationary `agent2`.

The observable effects of this parameter difference are quite remarkable. When `agent1` is configured with **ConflictResolutionDepth** = 1, it will not detect this second

*. All agents possess the homeostatic goal `obey-regulations` which, in this particular example, will trigger a goal conflict if the agent in question (`agent2`) is expected to run through the red traffic light.

`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.

5.3 Counterfactual Reasoning: why Modelling other Agents' Intentions can be Useful

In constructing and projecting models of other world entities, a TouringMachine must constrain its modelling activities along a number of dimensions. Implemented as user-definable parameters, these layer $\mathcal{M}$ constraints can be used to define such things as the tolerable deviations between the agent's actual and desired headings, the length of time into the future over which the agent's conflict detection predictions will apply, the rate at which the agent updates its models, and the total number of per-clock-cycle resources available for constructing models. One other layer $\mathcal{M}$ parameter which is of particular interest here is **ConflictResolutionDepth** — the parameter which fixes the number of levels of counterfactual reasoning the agent should undertake when projecting entities' models to discover possible future goal conflicts. In general, when constructing model projections at counterfactual reasoning level *N*, an agent will take into account any conflicts *plus any actions resulting from the anticipated resolutions to these conflicts* which it had previously detected at level *N-1*. Values of **ConflictResolutionDepth** which are greater than 1, then, give agents the flexibility to take into account — up to some fixed number of nested levels of modelling — any agent's responses to any other agent's responses to any predicted

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 4, 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 fast enough to `agent2`'s actions at T = 20.0 (Figure 4, 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 4, 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 `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

*. The process is expensive in the sense that since the agent has only enough computational resources to focus on a subset of entities in the world at any given time, misplaced sensitivity can result in the agent making poor use of its limited resources and potentially missing what might otherwise have been critical events.

5.2 Monitoring the Environment: Sensitivity versus Efficiency

TouringMachines continuously monitor their surroundings for activity or change. In monitoring the state of another agent, and in particular, in determining whether the model it maintains of an agent'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 agent's 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 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 4 (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 with other architectures extremely difficult if not altogether impossible.

Due to the relatively large number of parameters which the TouringWorld testbed provides for specifying different agent configurations, performance evaluation criteria (for example, task completion time, resource utilization), and agent task and environmental characteristics, the present evaluation will necessarily be partial, the main focus being placed on studying selected *qualitative* aspects of TouringMachine behavioral ecology — namely, some of the effects on agent behavior which, in a given task environment, can occur through varying individual agent configuration parameters; and the effects on agent behavior which, for a given agent configuration, can occur through varying certain aspects of the agent's environment. Like with the Tileworld experiments described by Pollack and Ringuette (Pollack and Ringuette, 1990, page 187), a number of TouringWorld “knobs” (for example, world clock timeslice size, total per-timeslice resources available to each agent, agent size, agent speed and acceleration/deceleration rate limits, agent sensing algorithm, initial attention focussing heuristics, reactive rule thresholds, plan schema and model template library entries) have been set to provide “baseline” environments which are dynamic, somewhat unpredictable, and moderately paced. In such environments, a competent (suitably configured) agent *should* be able to complete all of its goals, more or less according to schedule; however, under certain environmental conditions and/or agent parametrizations — a number of which will be analyzed below — this will not always be the case. In order to simplify the analysis of agents' behaviors in *multi-agent* settings, TouringMachine configurations — both mental and physical — should be presumed identical unless otherwise stated.

trivial challenges to TouringMachine agents.

It is not the aim of the present evaluation to show that the TouringMachine architecture is in any sense “optimal”. As argued elsewhere (Ferguson, 1992), optimal rational behavior will in general be impossible if the agent is resource-bounded, has several goals, and is to operate in a real-time multi-agent environment in which events are able to take place at several levels of space-time granularity. As such, one should more realistically expect a TouringMachine to behave satisficingly, but at times — for example, when under extreme real-time pressure — to fail to satisfy every one of its outstanding goals. What is really of interest here is understanding how the different configurations of agents and the different environmental characteristics to which such configurations are subjected affect, positively or negatively, the ability of agents to satisfy their goals.

It is also not the aim of the present evaluation to show that TouringMachines are “better” than other integrated agent architectures at performing their various tasks. Rarely is it the case that the actual and/or intended task domains of different agent architectures are described in sufficient detail so as to permit direct comparisons of agent performance. The lack, at present, of any common benchmark tasks or of any universally agreed upon criteria for assessing agent performance — previous evaluations have relied either on a single performance criterion; for example, the total point score earned for filling holes in specific single-agent Tileworld environments (Pollack and Ringuette, 1990; Kinny and Georgeff, 1991), or on a small number of performance criteria which can only be interpreted with respect to the particular architecture being measured; for example, the total number of behaviors communicated between agents in selected MICE environments (Durfee and Montgomery, 1990) — combine to make detailed *quantitative* comparisons design) decisions with the use of predictive models of a system's behaviors and of the environmental factors that affect these system behaviors. Like IRMA agents in the Tileworld domain (Pollack and Ringuette, 1990), TouringMachine agents can be viewed as having been developed via an incremental version of MAD, in which the (causal) model of TouringMachine behavior is developed incrementally, at the same time as the agent design. In other words, the agent design (or some part of its design) is implemented as early as possible, in order to provide empirical data (or feedback) which flesh out the model, which then become the basis for subsequent redesign (Cohen, 1991). The implications of adopting such a design method, as well as the roles played in this method by the environmental and behavioral analyses referred to above, are discussed in detail elsewhere (Ferguson, 1992).

The present evaluation of TouringMachines is realized through a series of interesting task scenarios involving one or more agents and/or zero or more obstacles or traffic lights. The scenarios have been selected with the aim of evaluating some of the different capabilities and behaviors which TouringMachines will require if they are to complete their tasks in a competent and effective manner — for example, reacting to unexpected events, effecting of goal-directed actions, reflective and predictive goal monitoring, spatio-temporal reasoning, plan repair, coping with limited computational and informational resources, as well as dealing with real-time environmental change. The scenarios can be considered interesting because they succinctly exercise agents' abilities to carry out time-constrained tasks in complex — partially-structured, dynamic, real-time, multi-agent — environments. Although the chosen scenarios are simplified to deal only with mentally and structurally homogeneous agents possessing noiseless sensors, perfect actuators, and approximately similar non-shared relocation tasks these still present a number of nonenvironment (Cohen et al., 1989), the Tileworld (Pollack and Ringuette, 1990), and MICE (Durfee and Montgomery, 1990).

The power of the TouringWorld testbed domain, and of artificial domains in general, arises from the insights it can provide toward the improved understanding of agent — in this case, TouringMachine — behavioral ecology: in other words, the understanding of the functional relationships that exist between the designs of agents (their internal structures and processes), their behaviors (the tasks they solve and the ways in which they solve these tasks), and the environments in which they are ultimately intended to operate (Cohen et al., 1989).

The characterization of TouringMachines as a study of agent behavioral ecology exemplifies a research methodology which emphasizes complete, autonomous agents and complex, dynamic task environments. Within this methodological context, the focus of the present evaluation has been centered on two particular research tasks. Cohen *et al.* (Cohen et al., 1989) refer to these as *environmental analysis*, in other words, understanding what characteristics of the environment most significantly constrain agent design; and the *design task*, in other words, understanding which agent design or configuration produces the desired behaviors under the expected range of environmental conditions.

These two tasks, in fact, are the first two stages of a more complete research methodology which Cohen (Cohen, 1991) refers to as the *MAD* methodology, for *modelling*, *analysis*, and *design.*[*] This methodology aims to justify system design (and re-

*. The remaining design activities — *predicting* how the system (agent) will behave in particular situations, *explaining* why the agent behaves as it does, and *generalising* agent designs to different classes of systems, environments, and behaviours — are beyond the scope of this work. See Cohen (Cohen, 1991, pages 29—32) for details.

(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. 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 *extended prediction problem* (Shoham and McDermott, 1990). Yet other experiments have been carried out which suggest that predicting future world states through causal modelling of agents' mental states, 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, two particular experiments that illustrate the role of causal modelling of agent behavior will be described in some detail. Before this, however, a few comments on the adopted experimental methodology are worth giving.

5.1 Some Methodological Issues

One useful approach toward understanding the reasons for the behaviors exhibited by the TouringMachine agent design — and, more specifically, for identifying the conditions under which one configuration of the architecture performs better than another — is to vary the environment in which it operates. The simplest approach to this issue, Langley (Langley, 1988) argues, involves designing a set of benchmark problems, of which some, for the purposes of scientific comparison (that is, for the purposes of enabling independent variation of different task environment attributes), should involve artificial domains. The TouringWorld environment is one such domain; other examples include the Phoenix

Ferguson, 1995).

5 Experimenting 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 functional capabilities within agents might combine to produce different behaviors. 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 (see Figure 3.a) 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, a declarative specification language for defining the agents to be observed, and several text and graphics windows for displaying output (see Figure 3.b). 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 behavior.

A number of experiments have been carried out on TouringMachines which illustrate, in particular, that the balance between goal-orientedness (effectiveness) and reactivity internal clock. Theorist's reasoning strategy then tries to accumulate consistent sets of facts and instances of hypotheses, or defaults, as explanations for which the observations are logical consequences. The facts and defaults reside in the *Theorist KBase*, a knowledge base containing a domain model of the TouringWorld expressed in terms of the various "faults" that can be used to explain entities' "errant" behaviors. In the present context, faults can be viewed as the causes for — or the intentions behind — why certain events — or certain observed actions of some entity — might have occurred in the world. Details on agents' domain models can be found elsewhere (Ferguson, 1992).

5.2 Generating Expectations and Closing the Loop

Once all BDI 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 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 (for example, the presence of other entities) or motivational forces (for example, an agent's own internal goals) that may constrain the possible actions that the agent can take (Ferguson, 1992;

which imply the observations. More formally, G is said to be *explainable* if there is some subset D of Δ such that

$$F \cup D \models G \text{ and}$$

$$F \cup D \text{ is consistent.}$$

D is said to be a *theory that explains G*. D should then be seen as a “scientific theory” (Poole et al., 1986, page 4).

Theorist has been described as both a theory and an implementation for default and *abductive* reasoning (Poole, 1988). One of the several ways in which Theorist can been used, then, is for performing *abductive diagnosis*; namely, finding a set of causes (for example, diseases) which can imply the observed effects (for example, patients' symptoms). Now, by taking the system or artifact that is being diagnosed as the entity that our TouringMachine agent is modelling, and by re-interpreting “symptoms” as the entity's observed actions, then the causes behind this entity's actions can be regarded as the entity's intentions.* Note, then, that in the context of TouringMachines, the process of finding the intentions which are the cause of some other entity's actions is effectively one of performing plan inference or recognition (Carberry, 1990).

Theorist is invoked once for every one of the agent's entity models that displays a model-entity (or expectation-observation) discrepancy. In particular, Theorist is called by supplying it with the name of the agent that is doing the modelling, the name of entity that is being modelled, the agent's observations of that entity (that is, all relevant details of the entity's current configuration as modelled by the agent), and the current value of the agent's

*. It should be noted that, for the purpose of simplifying the implementation of TouringMachines, agents' beliefs and desires are assumed to be common and so can be ignored during the theory formation process.

model.

Reasoning from a model of an entity essentially involves looking for the "interaction of observation and prediction" (Davis and Hamscher, 1988); 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 explanations about any entity (object/agent) behaviors which have recently been observed. Explanations (models) are then used in detecting discrepancies between these entities' current behaviors and those which had been anticipated from previous encounters. If any such behavioral discrepancies are detected, the agent will then strive to infer, via intention ascription, plausible explanations for their occurrence.

5.1 Generating Explanations with Theorist

Theorist (Poole et al., 1986) is a logic programming system for constructing scientific theories — that is, for constructing explanations of *observations* in terms of various *facts* and *hypotheses*. Theorist is a system for both representation and reasoning. A Theorist knowledge base consists of a collection of first order clausal form logic formulae which can be classified as: *(i)* a closed set of consistent formulae or facts, F, which are known to be true in the world; *(ii)* the possible hypotheses, Δ, which can be accepted as part of an explanation; and *(iii)* the set of observations, G, which have to be explained. Given these, the Theorist reasoning strategy attempts to accumulate consistent sets of facts and instances of hypotheses as explanations for which the observations are logical consequences. An explanation or *theory* is then a subset of the possible hypotheses which are consistent and would then enable it to make changes to its own plans 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 structures used by an agent to model an entity's behavior are 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. Plan 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 (Poole et al., 1986) — more on this shortly.

These Belief-Desire-Intention (BDI) models used by an agent are, in fact, 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 hybrid control approach that integrates a number of deliberative and non-deliberative action control mechanisms.

4 Modelling Agent Behavior

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 (Georgeff, 1990) 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, desires, 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 (Kirsh, 1991).

The potential gain from incorporating knowledge level 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 — thus enhancing the agent's ability to coordinate its actions with other agents (Chandrasekaran, 1994). This will not be able to be changed dynamically and there will be no way to reason about alternative plans for carrying them out. Maes (Maes, 1990) also argues that without explicit goals it is not clear how agents will be able to learn or improve their performance.

Complex agents will need complex goal or desire systems — in particular, they will need to handle a number of goals, some of which will vary in time, and many of which will have different priorities that will vary according to the agent's situational needs. The implications of this, Kirsh (Kirsh, 1991) argues, is that as agents' desire systems increase in size, there will be a need for some form of desire management, such as deliberation, weighing competing benefits and costs, and so on.

There are undoubtedly a number of real-world domains which will be suitable for strictly non-deliberative agent control architectures. It is less likely whether there exist any realistic or non-trivial domains which are equally suited to purely deliberative agents. What is most likely, however, is that the majority of real-world domains will require that intelligent autonomous agents be capable of a wide *range* of behaviors, including some basic non-deliberative ones such as perception-driven reaction, but also including more complex deliberative ones such as flexible task planning, strategic decision-making, complex goal handling, or predictive reasoning about the beliefs and intentions of other agents.

A central goal of the research presented here was to demonstrate that it is both *desirable* and *feasible* to combine non-deliberative and suitably designed deliberative control functions to obtain effective, robust, and flexible behavior from autonomous, task-achieving agents operating in complex environments. The arguments put forward so far have attempted both to outline some of the broader functional and behavioral requirements for intelligent agency in complex task domains like the TouringWorld, and also to justify a local minima, or generally undesirable outcomes. It follows, then, that if the agent's task requires knowledge about the environment which is not immediately available through perception and which can, therefore, only be obtained through some form of inference or recall, then it cannot truly be considered situationally determined. Kirsh (Kirsh, 1991) considers several such tasks, a number of which are pertinent to the TouringWorld domain: activities involving other agents (as these often require making *predictions* of their behavior and reasoning about their plans and goals (Davis 1990, page 395)); activities which require responding to events and actions beyond the agent's current sensory limits (such as taking precautions now for the future or when tracking sequences of behaviors that take place over extended periods of time); as well as activities which require some amount of reasoning or problem solving (such as calculating a shortest route for navigation). The common defining feature of these tasks is that, besides requiring reliable and robust *local* control to be carried out, they also possess a non-local or *global* structure which will need to be addressed by the agent. For instance, to carry out a navigation task successfully in the TouringWorld an agent will need to coordinate various locally constrained (re-)actions such as slowing down to avoid an obstacle or slower moving agent with other more globally constrained actions such as arriving at a target destination within some pre-specified deadline.

While non-deliberative control techniques ensure fast responses to changing events in the environment, they do not enable the agent's action choices to be influenced by deliberative reasoning. In most non-deliberative architectures, the agent's goals are represented implicitly — in effect, embedded in the agent's own structure or behavioral rule set. When goals are not represented explicitly, Hanks and Firby (Hanks and Firby, 1990) argue, they

Similarly, while the inclusion of at least some degree of non-deliberative control in TouringMachines would seem essential — particularly since the agents will need to be closely coupled to their environment, robust to unexpected events, and able to react quickly to unforeseen events and operate with guaranteed levels of responsiveness — it is questionable whether non-deliberative control techniques *alone* will be sufficient for providing TouringMachines with the complete behavioral repertoire necessary for successful operation in the TouringWorld environment. This argument deserves closer consideration.

3.1 Limitations of Pure Non-deliberative Control

The strength of purely non-deliberative architectures lies in their ability to identify and exploit *local* patterns of activity in their current surroundings in order to generate more or less hardwired action responses (using no memory or predictive reasoning, and only minimal state information) for a given set of environmental stimuli. Successful operation using this method of control presupposes: *(i)* that the complete set of environmental stimuli required for unambiguously determining subsequent action sequences is always present and readily identifiable — in other words, that the agent's activity can be strictly *situationally determined*; *(ii)* that the agent has no *global* task constraints — for example, explicit temporal deadlines — which need to be reasoned about at run-time; and *(iii)* that the agent's goal or desire system is capable of being represented *implicitly* in the agent's structure according to a fixed, pre-compiled ranking scheme.

Situationally determined behavior will succeed when there is sufficient local constraint in the agent's environment to determine actions that have no irreversibly detrimental long-term effects. Only then, as Kirsh (Kirsh, 1991) argues, will the agent be able to avoid representing alternative courses of actions to determine which ones lead to dead ends, loops, agents' activities. In this respect, each TouringMachine must have the capacity to *objectify* particular aspects of the world — that is, to construct and deploy internal models of itself and of other agents — to see where it fits in the coordinated process and what the outcomes of its own actions might be (Bond and Gasser, 1988, page 25).

Although much of the above functionality could be described as deliberative (for example, reasoning about the temporal extent of actions, conflict resolution, reflexive modelling), it is unclear whether a strictly deliberative control approach based on traditional planning techniques would be adequate for successful operation in the TouringWorld domain. Most classical planners make a number of important simplifying assumptions about their domains which cannot be made about the TouringWorld: namely, that the environments remain static while their (often arbitrarily long) plans are generated and executed, that all changes in the world are caused by the planner's actions alone, and that their environments are such that they can be represented correctly and in complete detail. Given that the TouringWorld is dynamic and multi-agent and given that TouringMachines also have inherently limited physical and computational means for acquiring information about their surroundings, it seems clear that a strictly traditional planning approach to controlling TouringMachines would be unsuitable. Also, while it is true that planning systems capable of execution monitoring and interleaved planning and execution represent a significant advance on the earlier traditional planners, their usefulness in a highly dynamic and real-time domain like the TouringWorld is questionable, particularly given the reservations expressed by Georgeff (Georgeff, 1990) and Bratman *et al.* (Bratman et al., 1988) concerning their computational efficiency and inability to cope with situationally-varying time constraints.

- A TouringMachine should be **robust** to unexpected events. Successful operation in a real-time dynamic environment will require that TouringMachines be able to identify and handle — in a timely manner — a host of unexpected events at execution time. For many events (such as the sudden appearance of a path-blocking obstacle) an agent will have little or no time to consider either what the full extent of its predicament might be or what benefits consideration of a number of different evasive maneuvers might bring. In order to cope with such events, TouringMachines will need to operate with guaranteed responsiveness (for example, by using latency-bounded computational and execution techniques) as well as being fairly closely coupled to their environments at all times. Since the time and location of such events will be unpredictable, TouringMachines will need to monitor their surroundings continually throughout the course of their goals.
- A TouringMachine should be **flexible** in the way it carries out its tasks. Due to the dynamic and unpredictable nature of the TouringWorld environment, and the fact that its multiple inhabitants must operate in real time with limited world knowledge, TouringMachines will inevitably be faced with various belief and/or goal conflict situations arising from unforeseen interactions with other agents. Agents operating cooperatively in complex domains must have an understanding of the nature of cooperation. This, Galliers (Galliers, 1990) argues, involves understanding the nature and role of multi-agent conflict. To behave flexibly and to adjust appropriately to changing and unpredicted circumstance, TouringMachines should be designed to recognize and resolve unexpected conflicts rather than to avoid them. Also, for the purposes of control and coordination, TouringMachines must be able to reason about their own and other

would enable TouringMachines to carry out tasks and act on their environments autonomously and in accordance with a set of domain-specific evaluation criteria; namely, effectiveness, robustness, and flexibility. These criteria suggested a broad range of behavioral and functional capacities that each TouringMachine might need to possess:

- A TouringMachine should be capable of **autonomous** operation. Operational autonomy requires that the agent have its own goals and be able to select among these as and when required. In addition, as Covrigaru and Lindsay (Covrigaru and Lindsay, 1991) argue, the agent should, among other things, be capable of interacting with its environment, be able to move (preferably fluidly) around its environment, have selective attention (this is also desirable since TouringMachines have limited computational resources), have a varied behavioral repertoire, and have differential responsiveness to a variety of environmental conditions.
- A TouringMachine should carry out its goals in an **effective** manner. Effective goal achievement requires that the agent be capable of carrying out its multiple tasks in an efficient and timely manner. Since among its various tasks, a TouringMachine must navigate along some route within a pre-specified time limit, the agent should be able to reason predictively about the temporal extent of its own actions. Also, because TouringMachines will operate in a partially-structured multi-agent world, they should, in order to complete their tasks, be able to *coordinate* their activities with other agents that they might encounter: that is, they should be capable of cooperation.[*]

[*]. Following Bond and Gasser (Bond and Gasser, 1988, page 19), cooperation in the TouringWorld is viewed simply as a special case of coordination among *non-antagonistic* agents. While TouringMachines are not actually benevolent (they are selfish with respect to their own goals and have the ability to drop or adopt different intentions according to their own preferences and situational needs) they are also not antagonistic since they do not intentionally try to deceive or thwart the efforts of other TouringMachines.

ancing two “reasonable" approaches to acting in the world: the first, deliberation, involves making as many decisions as possible as far ahead of time as possible; the second approach, reaction, is to delay making decisions as long as possible, acting only at the last possible moment. At a glance, the first approach seems perfectly reasonable since, clearly, an agent which can think ahead will be able to consider more options and thus, with forethought, be more informed when deciding which action to take. On the other hand, since information about the future can be notoriously unreliable and, in many real-world situations, difficult or even impossible to obtain given the agents' changing time constraints, it would also seem reasonable that acting at the last moment should be preferred. In fact, except perhaps for a small number of special-case task domains, it would seem much more reasonable to assume that neither approach — deliberation or reaction — should be carried out to the full exclusion of the other.

TouringMachines are autonomous, mobile agents which are capable of rationally carrying out a number of routine tasks in a complex multi-agent traffic navigation domain — the *TouringWorld*. These tasks are prioritized in advance by the agent's designer and, as mentioned above, include goals like avoiding collisions with other mobile agents and fixed obstacles, obeying a commonly accepted set of traffic regulations, and relocating from some initial location to some target destination within certain time bounds and/or spatial constraints. Besides being limited in terms of its internal computational resources, each TouringMachine will start out with limited knowledge of its world and with limited means for monitoring and acquiring information from its surroundings; in addition, each agent will be will be restricted in its capacity to communicate with other agents.

The initial goal of this research was to produce an integrated control architecture which

Mediation remains active at all times and is largely “transparent” to the layers: each layer acts as if it alone were controlling the agent, remaining largely unaware of any “interference” — either by other layers or by the rules of the control framework — with its own inputs and outputs. The overall control framework thus embodies a real-time opportunistic 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. In this respect, the TouringMachine architecture offers many of the desirable characteristics of so-called hybrid architectures (Hanks and Firby, 1990; Müller and Pischel, 1994). Before further describing TouringMachines’ behavioral modelling capabilities, a few comments justifying the choice of a hybrid control architecture seem appropriate.

3 Hybrid Architectures: a Rationale

An autonomous agent operating in a complex environment is constantly faced with the problem of deciding what action to take next. As Hanks and Firby (Hanks and Firby, 1990) point out, formulating this problem precisely can be very difficult since it necessitates consideration of a number of informational categories which are often difficult to ascertain — for example, the benefits and costs to the agent of executing particular actions sequences; or which have been demonstrated from previous research to be problematic to represent — for example, models of agents' beliefs and desires about a world which is complex and unpredictable.

The control problem in an agent is the problem of deciding how to manage these various sources of information in such a way that the agent will act in a competent and effective manner. This problem, Hanks and Firby (Hanks and Firby, 1990) suggest, amounts to balconditional parts are conjunctions of statements that test for the presence of particular sensory objects recently stored in the agent's Perception Subsystem (see Figure 1). Censor rules' action parts consist of operations to prevent particular sensory objects from being fed as input to *selected* control layers. In Figure 2, for example, the censor rule **`censor-rule-1`** is used to prevent layer $\mathcal{R}$ from perceiving (and therefore, from reacting to) a particular obstacle which, for instance, layer $\mathcal{M}$ might have been better designed to deal with. In the case of suppressor control rules, conditional parts are conjunctions of statements which, besides testing for the presence of particular outgoing action commands in the agent's Action Subsystem, can also test the truth values of various items of the agent's current internal state — in particular, its current beliefs, desires, and intentions (more on these in the next section). Suppressor rules' action parts consist of operations to prevent particular action commands from being fed through to the agent's effectors. In Figure 2, for example, the suppressor control rule **`suppressor-rule-3`** is used to prevent layer $\mathcal{R}$ from reacting to (steering away from) a lane marking object whenever the agent's current intention is to overtake some other agent that is in front of it. Any number of censor control rules can fire (and remove selected control layer input) when these are applied at the beginning of a synchronization timeslice. Suppressor control rules, on the other hand, are assumed to have been crafted by the agent's programmer in such a way that *(i)* at most one will fire in any given situational context (an agent's situational context is taken to be the combination of its perceptual input set and its current internal state); and *(ii)* at most one action command will remain in the Action Subsystem after the suppressor control rule's action part has been executed. By crafting suppressor control rules in this way, a TouringMachine's effectors can be guaranteed to receive no more than one action command to execute during any given timeslice.

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 framework so that the agent, as a single whole, may behave appropriately in each different world situation.

Implemented as a combination of inter-layer message-passing and context-activated, domain-specific control rules (see Figure 1), the control framework's mediation enables each layer to examine data from other layers, inject new data into them, or even remove data from the layers. (The term *data* here covers sensed input to and action output from layers, the contents of inter-layer messages, as well as certain rules or plans residing within layers.) This has the effect of altering, when required, the normal flow of data in the affected layer(s). So, in the road driving domain for example, the reactive rule in layer $\mathcal{R}$ to prevent an agent from straying over lane markings can, with the appropriate control rule present, be overridden should the agent embark on a plan to overtake the agent in front of it (more on this shortly).

Inputs to and outputs from layers are generated in a synchronous fashion, with the context-activated control rules being applied to these inputs and outputs at each synchronization point. The rules, thus, act as filters between the agent's sensors and its internal layers (*suppressors*), and between its layers and its action effectors (*censors*) — in a manner very similar to Minsky's suppressor- and censor-agents (Minsky, 1986). Both types of rules are of the *if-then* condition-action type. In the case of censor rules, the

2 TouringMachines

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 (Ferguson, 1992, pages 26—32), the TouringMachine architecture has been designed through integrating a number of reactive and *suitably designed* deliberative control functions.

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 (see Figure 1) 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 behavioral 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 considered interesting because it presents agents with a series of challenging tasks including having to cope with multi-agent interactions, unpredictability, uncertainty, resource-constrained tasks, and real-time environmental change. While it is true that a number of simplifications have been in order to analyze more easily the behavior of TouringMachines (these are detailed elsewhere (Ferguson, 1992, pages132—134)), it is fair to say that the TouringWorld is a reasonably faithful approximation to certain classes of real-world, multi-agent domains; for example, automated factory/office floors or road traffic environments.

This article is concerned with the design and implementation of an integrated agent control architecture, the *TouringMachine* architecture (Ferguson, 1994; Ferguson, 1995), suitable for controlling and coordinating the actions of autonomous rational agents embedded in a partially-structured, dynamic, multi-agent world. In addition to providing an overview of the proposed agent control architecture (see section 2), this article presents, in section 3, a detailed rationale justifying the use of a *hybrid* architecture — that is, one which integrates both deliberative and non-deliberative task behaviors — for controlling agents in complex real-world domains such as the TouringWorld. In section 4, a description is given of TouringMachines' Belief-Desire-Intention modelling capabilities. Section 5 provides some results obtained upon experimenting with several such intentional agents, in a number of different environmental settings; this section also discusses several important methodological issues concerning the experimental approach followed in this project. The article concludes with a discussion of a number of critical issues surrounding the practical design and implementation of autonomous agents such as TouringMachines.

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 (Agre and Chapman, 1987; Schoppers, 1987; Brooks, 1991; Maes, 1994) nor purely deliberative (Bratman et al., 1988; Durfee and Montgomery, 1990; Shoham, 1993; Vere and Bickmore, 1990) 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.

In the example application domain used to evaluate TouringMachines, one or more autonomous route-planning agents (vehicles) are considered, each with the task of relocating from some starting location to some goal location within certain time bounds and/or spatial constraints. Each agent starts out with some topological knowledge of the world (e.g. locations of paths, path junctions, and certain landmarks associated with these junctions), but has no prior knowledge of the whereabouts of other agents or of any obstacles. An agent can communicate its intentions to turn or overtake by signalling — much like a driver does in a car — and can only consume up to some fixed number of computational resources per unit of world time. This domain, the *TouringWorld*, can be

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 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

Abstract

This paper describes an architecture for controlling and coordinating autonomous agents, building on previous work addressing reactive and deliberative control methods. The proposed multi-layered hybrid architecture allows a rationally bounded, goal-directed agent to reason predictively about potential conflicts by constructing knowledge level models which explain other agents' observed behaviors and hypothesize their beliefs, desires, and intentions; at the same time it enables the agent to operate autonomously, to react promptly to changes in its real-time environment, and to coordinate its actions effectively with other agents. A principal aim of this research is to understand the role different functional capabilities play in constraining an agent's behavior 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.

Appears in Applied Artificial Intelligence Journal - Special Issue on Multi-agent Systems, 9(4), 1995.

On the Role of BDI Modelling for Integrated Control and Coordinated Behavior in Autonomous Agents

Innes A. Ferguson[*]

Knowledge Systems Laboratory

Institute for Information Technology

National Research Council

Ottawa ON

Canada K1A 0R6

Abbreviated title:

BDI Modelling for Coordinated Behavior

Mailing address for proofs:

Innes A. Ferguson

IIT/KSL, Building M-50

National Research Council

Ottawa, ON

Canada K1A 0R6

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