Integrated Control and Coordinated Behaviour: A Case for Agent Models

Ferguson, I.A.

Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages (ATAL 94), 1995

6(1):41—60, 1990.

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

6. 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.

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

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

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

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

11. 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.

12. 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.

13. 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.

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

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

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

17. 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.

18. Paul R. Cohen. A survey of the Eighth National Conference on Artificial Intelligence: Pulling together or pulling apart. AI Magazine, 12(1):16—41, 1991.

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

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 [7].

5 Conclusions

Through the above and a number of other single- and multi-agent 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 configured, 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.

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 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 behavioral 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

1. Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14—23, 1986.
2. 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.
3. Yoav Shoham. AGENT0: A simple agent language and its interpreter. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 704—709, 1991.
4. Steven Vere and Timothy Bickmore. A basic agent. *Computational Intelligence*,

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 Fig.4 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 Fig. 4), `agent1` detects one such conflict: an `obey-regulations`[4] 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 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 (Fig. 4, lower left-hand frame).[5] Configured with **ConflictResolutionDepth** = 2 (Fig. 4, 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

4. 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.

5. 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.

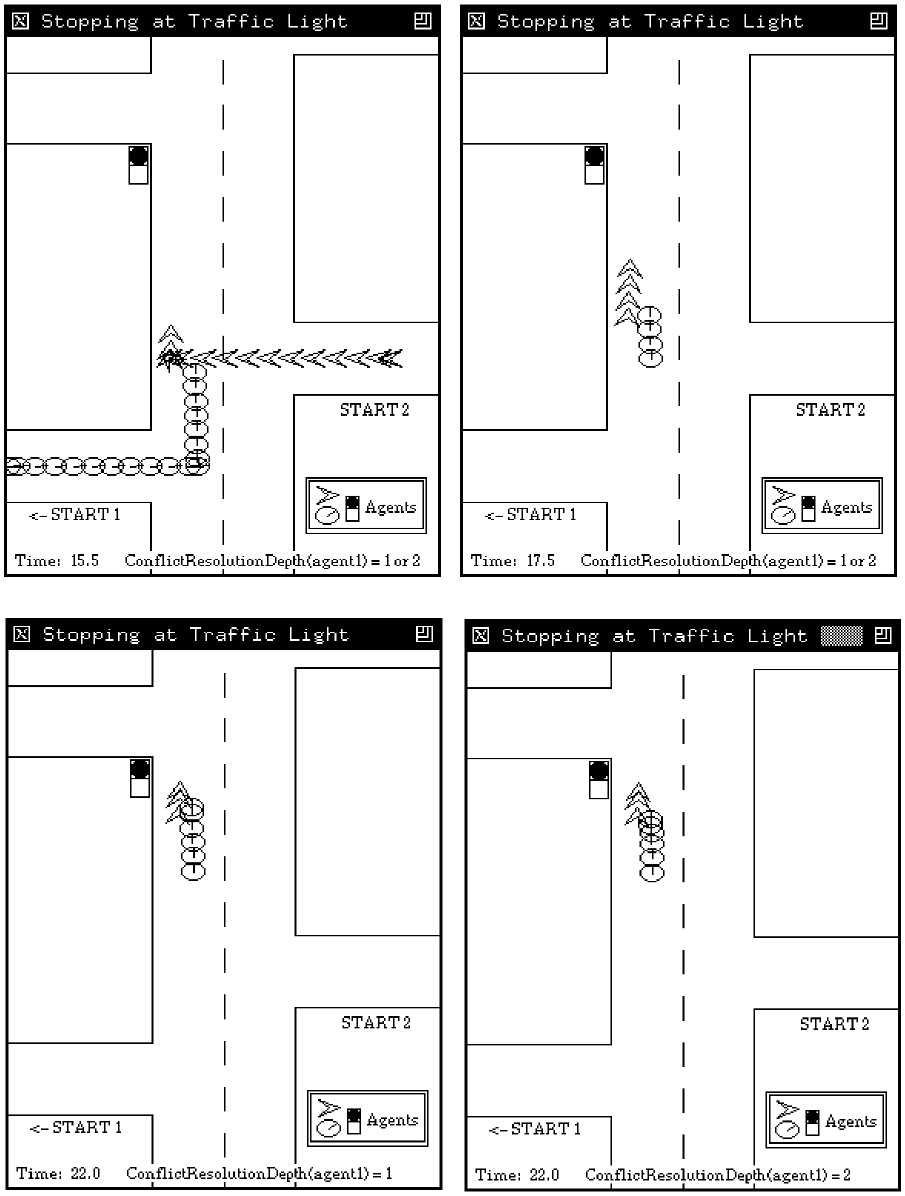
Stopping at Traffic Light
START 2
<- START 1
Agents
Time: 15.5 ConflictResolutionDepth(agent1) = 1 or 2
Stopping at Traffic Light
START 2
<- START 1
Agents
Time: 17.5 ConflictResolutionDepth(agent1) = 1 or 2
Stopping at Traffic Light
START 2
<- START 1
Agents
Time: 22.0 ConflictResolutionDepth(agent1)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

Fig. 4. Altering the value of an agent's **ConflictResolutionDepth** parameter can affect the timeliness and effectiveness of any predictions it might make.

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.[3] 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 Fig. 3, 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 `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.

4.3 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 conflicts.

In the scenario of Fig. 4, two TouringMachine agents can be seen following

3. 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.

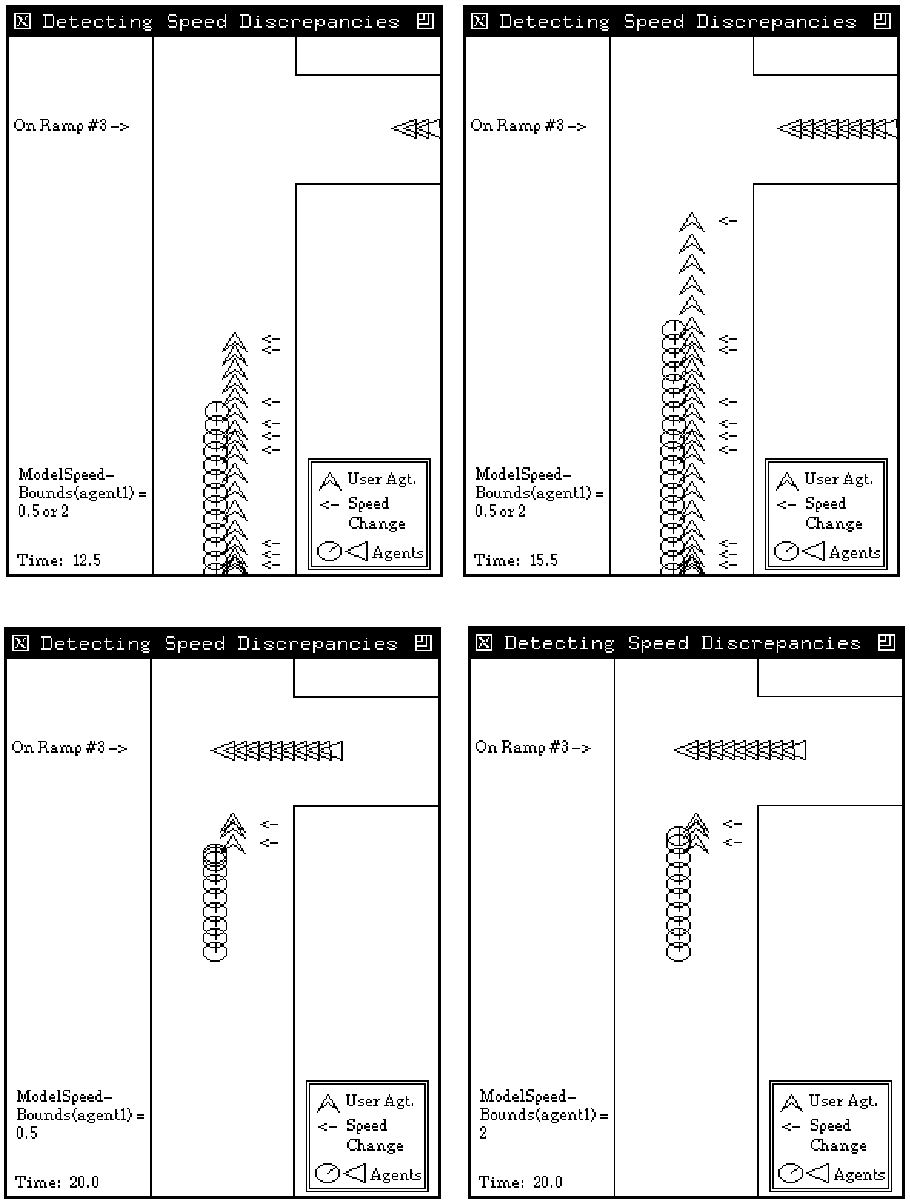
Detecting Speed Discrepancies
On Ramp #3 ->
ModelSpeed-Bounds(agent1) = 0.5 or 2
Time: 12.5
User Agt.
<- Speed Change
Agents
Detecting Speed Discrepancies
On Ramp #3 ->
ModelSpeed-Bounds(agent1) = 0.5 or 2
Time: 15.5
Detecting Speed Discrepancies
On Ramp #3 ->
ModelSpeed-Bounds(agent1) = 0.5
Time: 20.0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

Fig. 3. Varying the value of an agent's **ModelSpeedBounds** parameter can affect the agent's level of sensitivity to environmental change.

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.

4.2 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 Fig. 3 (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 (Fig. 3, 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 (Fig. 3, 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 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 non-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 [7], 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 [16,19]), 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 [17]) — combine to make detailed *quantitative* comparisons 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 [16, page 187], a number of 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 [14] 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 environment [15], the Tileworld [16], and MICE [17]).

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 [15].

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.* [15] 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 [18] refers to as the *MAD* methodology, for *modelling*, *analysis*, and *design*.[2] This methodology aims to justify system design (and re-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 [16], 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 [18]. 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 [7].

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

2. 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 [18, pages 29—32] for details.

example, an agent's own internal goals) that may constrain the possible actions that the agent can take [7].

4 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 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. 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 (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 well-known *extended prediction problem* [13]). 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.

4.1 Some Methodological Issues

One useful approach toward understanding the reasons for the behaviors exhibited 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 [11].

The mental 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 model.

Reasoning from a model of an entity essentially involves looking for the "interaction of observation and prediction" [12]; 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.

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 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

```
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.

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

3 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 [9] 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 [10].

The potential gain from incorporating causal 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 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 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 [8]. Both types of rules are of the *if-then* condition-action type. In the case of censor rules, the conditional parts are conjunctions of statements that test for the presence of particular sensory objects recently stored in the agent's Perception Subsystem (see Fig. 1). Censor rules' action parts consist of operations to prevent particular sensory objects from being fed as input to *selected* control layers. In Fig. 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 programmed 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 Fig. 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.

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.

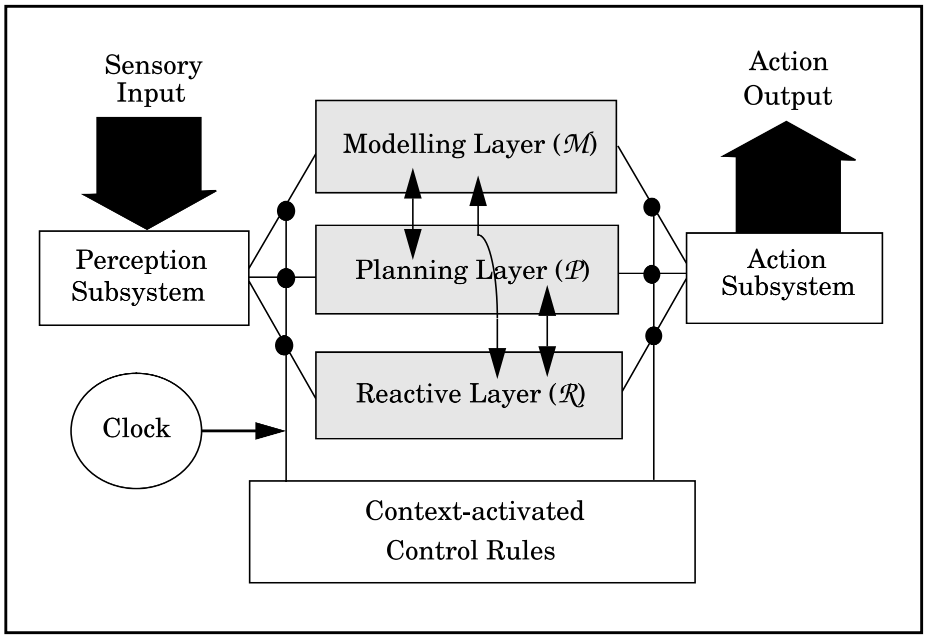
Sensory Input
Action Output
Modelling Layer ($\mathcal{M}$)
Perception Subsystem
Planning Layer ($\mathcal{P}$)
Action Subsystem
Reactive Layer ($\mathcal{R}$)
Clock
Context-activated Control Rules

Fig. 1. A TouringMachine's mediating control framework.

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 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 Fig. 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 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 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 [1,2] nor purely deliberative [3,4] 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 [5—7], 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 (see Fig. 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

Appears in M. Wooldridge and N. Jennings (eds.)Intelligent Agents - Theories, Architectures, and Languages. LNCS Series, Springer-Verlag: Berlin, Jan. 1995

Integrated Control and Coordinated Behavior: a Case for Agent Models

Innes A. Ferguson[1]

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

Abstract. This paper presents a new architecture for controlling autonomous agents in dynamic multi-agent worlds, building on previous work addressing reactive and deliberative control methods. The proposed multi-layered architecture allows a rationally bounded, goal-directed agent to reason predictively about potential conflicts by constructing causal theories which explain other agents' observed behaviors and hypothesize their intentions; 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 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.

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,

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