TouringMachines: Autonomous Agents with Attitudes[*]

Innes A. Ferguson[†]
Computer Laboratory, University of Cambridge
New Museums Site, Cambridge CB2 3QG
England, UK. Internet: iaf@cl.cam.ac.uk

Technical Report 250 – April 1992

Abstract

It is becoming widely accepted that neither purely reactive nor purely deliberative control techniques are capable of producing the range of behaviours required of intelligent robotic agents in dynamic, unpredictable, multi-agent worlds. We present a new architecture for controlling autonomous, mobile agents – building on previous work addressing reactive and deliberative control methods. The proposed multi-layered control architecture allows a resource-bounded, goal-directed agent to react promptly to unexpected changes in its environment; at the same time it enables the agent to reason predictively about potential conflicts by constructing and projecting theories which hypothesise other agents' goals and intentions.

The line of research adopted is very much a pragmatic one. A single, common architecture has been implemented which, being extensively parametrized, allows an experimenter to study functionally- and behaviourally-diverse agent configurations. A principal aim of this research is to understand the role different functional capabilities play in constraining an agent's behaviour under varying environmental conditions. To this end, we have constructed an experimental testbed comprising a simulated multi-agent world in which a variety of agent configurations and behaviours have been investigated. Some experience with the new control architecture is described.

[*]This paper expands on the author's article "TouringMachines: Autonomous Agents with Attitudes" which appears in *IEEE Computer*, 25(5), May, 1992.

[†]Contact address as from September 1992: Bell-Northern Research Ltd. P.O. Box 3511, Station C, Ottawa, Ont., Canada K1Y 4H7. Internet: uiiferg@bnr.ca.

1 Introduction

As operating environments such as automated factories, nuclear power plants, and space stations continue to grow in complexity, it will become increasingly more difficult to control these with centralised scheduling policies which are both robust to unexpected events and flexible at dealing with operational changes that might occur over time. One solution to this problem which has growing appeal is to distribute the control and scheduling of operations to a number of intelligent, task-achieving *computational* or *robotic agents*.[1] Most of today's robotic agents, however, are limited to performing a relatively small range of well-defined, pre-programmed or human-assisted tasks.

In order to survive and thrive in complex, real-world domains, future agents will need to be made considerably more robust and flexible than they are at present. Such domains are likely to be populated by multiple agents, each pursuing any number of tasks. Because agents will have incomplete knowledge about the world and will compete for shared and limited resources, it is inevitable that some of their goals will conflict. In real-world domains agents will typically perform complex tasks requiring some degree of attention to be paid to computational resource bounds, temporal deadlines, and the impact their shorter-term actions might be having on their longer-term goals. On the other hand, time won't stop or slow down for them to deliberate upon all possible courses of action for every world state. Intelligent agents will thus require a range of skills to respond promptly to unexpected events, while simultaneously being able to carry out pre-programmed tasks and resolve unexpected conflicts in a timely and efficient manner.

In this article we present a new multi-layered, integrated architecture for controlling autonomous, mobile agents or TouringMachines which combines capabilities for producing a range of reactive and deliberative behaviours in dynamic, unpredictable domains. This new approach is influenced on the one hand by recent work on reactive and behaviour-based agent architectures [Bro86, Fir87, Kae87], and on the other by more traditional AI endeavours such as planning, diagnostic theory formation [PGA86], resource-bounded reasoning [BIP88, HR90], and cognitive modelling of propositional attitudes such as beliefs, desires, and intentions [BIP88, GI89, Bra90].

Our research adopts a fairly pragmatic approach toward understanding how complex, dynamic environments might constrain on the design of agents and, conversely, how different functional capabilities within agents might combine to generate different behaviours. To evaluate the TouringMachine architecture we have implemented a multi-agent simulation testbed. By varying

[1]For our purposes, we shall consider an *agent* to be any autonomous, goal-directed, computational process capable of robust and flexible interaction with its environment.

parameters constraining agents' functional capabilities (e.g. sensing characteristics, attentional powers, degree of reactivity, world modelling powers) or parameters characterising the environment itself (e.g. number of agents and obstacles, ratio of CPU time to simulated-world time), we can study a number of tradeoffs vis-à-vis how much reacting, planning, and predicting resource-bounded agents should be doing in order to behave rationally with respect to their goals.[2] In many ways, our approach to evaluating agent designs resembles the empirical approaches used in the DVMT [LC83], Phoenix [CGHH89], MICE [DM89], and Tileworld [PR90] projects.

In our example domain we consider one or more agents, each with the task of following a different route from some starting location to some goal location within certain time bounds and/or spatial constraints. Each agent starts with some geographical knowledge of the world (e.g. locations of paths and path intersections), but has no prior knowledge regarding other agents' locations or goals or static obstacles it might encounter along its route. 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 simulated world time. We consider this domain interesting because it presents our agents with a series of challenges including having to cope with multi-agent interactions, unpredictability, uncertainty, resource-constrained tasks, and environmental change. Before discussing specifics of the TouringMachine architecture, its implementation, and its simulation testbed, we consider some important requirements for intelligent agency.

2 Intelligent Agency

In recent years there has been considerable growth of interest in the design of intelligent agent architectures for dynamic, unpredictable domains. One popular design approach – whose resulting architectures we'll call *deliberative* – attempts to endow agents with sophisticated control by embedding in these a number of general AI capabilities such as means-end reasoning, epistemic modelling [BIP88], plan recognition [Woo90], or natural language understanding [VB90].[3] Influenced principally by the fruits of classical AI planning re-

[2]The definition of rational behaviour used here is borrowed from Bratman *et al*. [BIP88, page 349] and corresponds to "the production of actions that further the goals of an agent, based upon [its] conception of the world."

[3]More generally, by *deliberative* we mean that the agent possesses reasonably explicit representations of its own beliefs and goals that it uses in deciding which action it should take at a given time. Conversely, by *non-deliberative* (see below), we mean that the agent's goals are implicitly embedded or pre-compiled into the agent's structure by its designer.

search, deliberative architectures have been designed both to handle complex goals (e.g. those involving action-at-a-distance, resource constraints, or multiple agents) and to operate flexibly in unpredictable or novel situations (e.g. by performing contingency planning or analogical reasoning). This generality, however, exacts a price; by virtue of having to maintain complete, up-to-date world models, deliberative architectures can be resource-intensive and are usually slow at making critical decisions in real-time situations.

Breaking with the traditionally held belief that "complex" architectures are required to produce intelligent agent behaviours, a number of *non-deliberative* (e.g. reactive [Fir87], situated [AC87, Mae90], and behaviour-based [Bro86, Kae87]) architectures have recently been proposed. These architectures are characterised by a more direct coupling of perception to action, increased decentralisation of control, and relative simplicity of design. Because they perform localised search, the time spent deciding which action to effect in any given situation can be minimised. At the same time, however, these architectures run the risk of generating sub-optimal action sequences *precisely because* they operate with minimal memory or state information [Fir87]. Also, because non-deliberative agents are essentially *hardwired* to effect a particular action sequence in each given situation, they can be ineffective when confronted with situations which are either novel or which do not provide immediate access to the complete set of environmental stimuli needed for determining subsequent action sequences. Indeed, as other researchers have noted [GI89, Mae90, Kir91], there has been little evidence to date to suggest that pure non-deliberative architectures are capable of handling multiple, complex, resource-bounded goals in any sophisticated manner. Like their deliberative cousins, non-deliberative agents will require that their environments be reasonably cooperative if they are to achieve their goals satisfactorily [Bro86].

Operating in the real world means having to deal with multiple events at several levels of granularity – both in time and space. So, while agents must remain reactive in order to survive, some amount of strategic or predictive decision-making will be required if agents are to handle complex goals while keeping their long-term options open. Agents, however, 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 nor purely deliberative control techniques are capable of producing the range of robust, flexible behaviours desired of future intelligent agents. What is required, in effect, is an architecture that can cope with uncertainty, react to unforeseen events, and recover dynamically from poor decisions. All of this, of course, on top of accomplishing whatever tasks it was originally assigned to do.

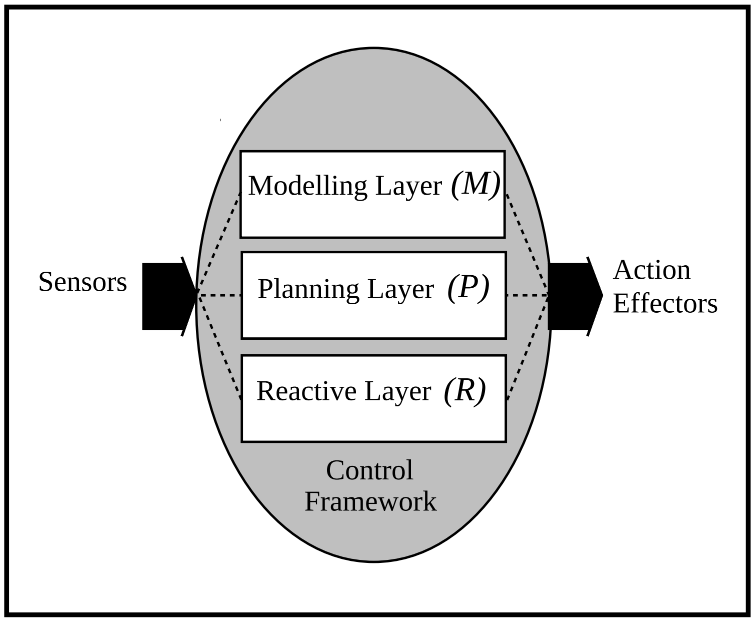


Figure 1: The TouringMachine architecture.

3 TouringMachines

To operate successfully in our chosen multi-agent domain, an autonomous robotic agent must be both *robust* and *flexible* – it must be capable of carrying out its intended goals in dynamic, unpredictable environments. To do this, we believe, the agent must be capable of exhibiting a range of different behaviours. First, it will need to be reactive to deal with events which it might not have had sufficient time or resources to consider. Secondly, since the agent's main task, in our case, will be to get from some starting location to some target location in some specified time, it should be capable of rational, resource-bounded, goal-directed behaviour. And thirdly, since it will inhabit a world populated by other entities (about which very little will be known in advance) it must be able to reason about what events are taking place around it, determine what effect these events could have on its own goals and, where possible, predict what is likely to happen in the near future so as to be better informed when choosing and effecting subsequent actions. Because these skills have such disparate characteristics and requirements, the most sensible way of realizing them, it would seem, is as separate activity-producing behaviours in a layered framework. We have adopted this approach in designing and implementing TouringMachines.

TouringMachines comprise three concurrently-operating, independently motivated, activity-producing layers: a *reactive* layer $\mathcal{R}$, a *planning* layer $\mathcal{P}$, and a reflective-predictive or *modelling* layer $\mathcal{M}$ (see Figure 1). Each mod-

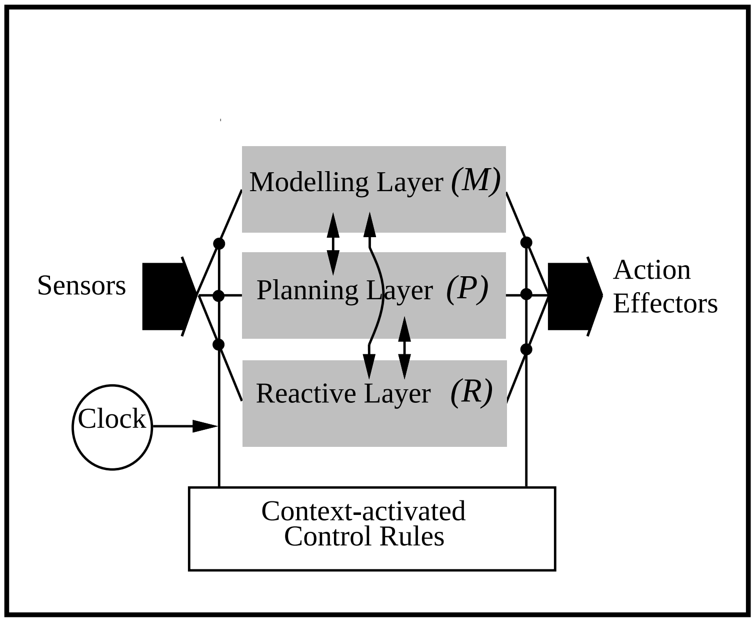


Figure 2: A TouringMachine's mediating control framework.

els the agent's world at a different level of abstraction and each is endowed with different task-oriented capabilities. The TouringMachine framework is, in fact, *hybrid*, as it may incorporate several functional or *horizontal* faculties within a given task-achieving or *vertical* layer. For example, hypothetical reasoning and focus of attention are both realized in layer $\mathcal{M}$.

The main principle behind vertical decomposition is to create activity-producing subsystems each of which directly connects perception to action and which can independently decide if it should or should not act in a given world situation. Frequently, however, one layer's proposed actions will conflict with those of another: a layer is an *approximate* machine and thus its abstracted world model is necessarily incomplete. Because of this, layers need to be mediated by an enveloping *control framework* if the agent, as a single whole, is to 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 2), 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, 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.

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 synchronisation point. The rules, thus, act as filters between the agent's sensors and its internal layers, and between its layers and its action effectors. 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 embodies a scheduling regime which, while striving to service the agent's high-level tasks (e.g. `plan-a-route`) is sensitive also to its low-level, high-priority behaviours such as avoiding kerbs or obstacles.

The TouringMachine layered framework is strongly influenced by Brooks' *subsumption* architecture [Bro86]. This comprises several concurrently-operating, task-achieving behaviours which are implemented as fixed-topology networks of finite-state machines along with various registers and timers. Layers communicate via fixed-length messages over "wires" and are mediated by *suppression* and *inhibition* mechanisms which can alter the flow of inter-layer messages to produce the correct action for the situation at hand.

Besides several technical differences, the main distinction between the two architectures is that TouringMachines store and manipulate explicit representations of, among other things, propositional attitudes such as beliefs, desires, and intentions in order to perform such cognitive tasks as reflection and prediction (see below). Brooks' agents have not to date been used to solve such high-level tasks, and it's not at all clear whether his architecture could be scaled up indefinitely without ever resorting to the use of internal representations [Kir91].

Most designs for integrated agent architectures share the common aim of enabling autonomous agents to interact flexibly and robustly in more or less dynamic environments. The brevity of the current description of TouringMachines belies the numerous design, implementational, performance, and (even) philosophical issues which have to be considered and traded-off when creating a new control architecture. Establishing when and how an agent should reason versus act, deciding which goals or behaviours should be explicitly planned for or embedded in the agent's structure, are open questions, the answers to which depend heavily on the agent's specific task requirements and environmental influences. At the risk of over-simplifying the argument, we believe the main strength of TouringMachines lies in their ability to operate flexibly in dynamic environments while interacting with and reasoning about other agents with complex goals and intentions. A surprisingly small number of previous architectures have addressed, and more importantly, investigated, issues pertaining to coordination in multi-agent environments. Where these issues have been considered, the agents involved were either given relatively simple goals [AC87, SH88] or operated with little autonomy under the control of a supervisory agent [CGHH89]. Also, by modelling agents' desires and intentions in a more principled way and by not requiring that all actions be generated by a planning module, we believe TouringMachines are more powerful and robust than Wood's AUTODRIVE agents [Woo90]. Our approach to multi-agent coordination also differs from that of Durfee and Montgomery by placing more emphasis on the autonomous modelling capabilities required by complex agents, rather than on the mechanisms and protocols needed by agents to communicate and exchange information about their goals [DM90]. Again, achieving the right balance between modelling and communicating is a complex issue which, although worthy of further investigation, has not been addressed at present. The following sections describe each layer in more detail.[4]

3.1 Layer $\mathcal{R}$ (reactive)

The purpose of this layer is to provide an agent with fast, reactive capabilities for coping with events its higher layers haven't previously planned for or modelled. A typical event, for example, would be the sudden appearance of some hitherto unseen agent or obstacle. Layer $\mathcal{R}$ provides the agent with a series of *situation-action* rules for avoiding obstacles, walls, kerbs or other agents, and for preventing it from straying over path lane markings (see Figure 3). Thus for example, the two rules for avoiding collisions with other agents are:

```
rule-3:  if is-in-front(Other, Observer) and
            velocity(Other) < velocity(Observer) and
            separation(Other, Observer) < FrontalAgentThreshold
         then
            change-velocity(Observer, FrontalAvoidanceVelocity)

rule-4:  if is-behind(Other, Observer) and
            velocity(Other) > velocity(Observer) and
            separation(Other, Observer) < RearAgentThreshold
         then
            change-velocity(Observer, RearAvoidanceVelocity)
```

where **FrontalAgentThreshold**, **FrontalAvoidanceVelocity**, **RearAgent-Threshold**, and **RearAvoidanceVelocity** are parameters associated with

[4]Due to space restrictions much detail will, in fact, be omitted and presented elsewhere [Fer92].

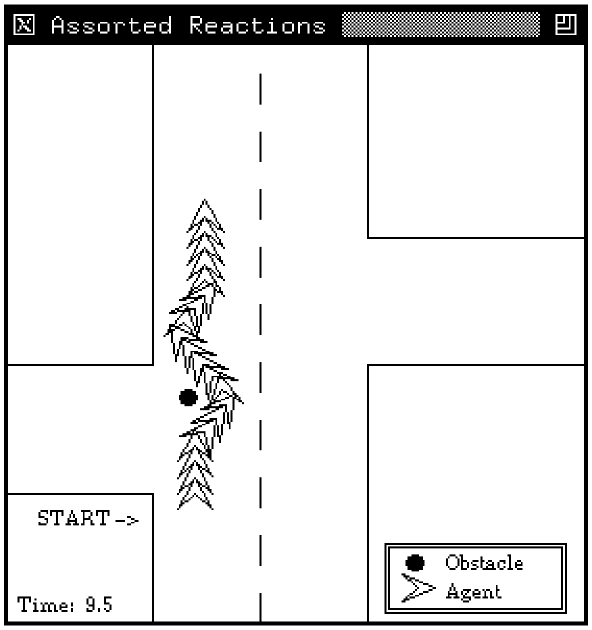


Figure 3: Appropriate situation-action rules can enable an agent to avoid obstacles and kerbs, prevent it from straying over lane markings, and, when no other events require attention, adjust the agent's orientation to one of four orthogonal directions (0°, 90°, 180°, or 270°).

the agent `Observer`. As we shall see below, an agent can be made variably reactive or inert by choosing appropriate values for such parameters.

Rules are stimulated *solely* and directly by input they receive from the agent's sensors. When a given rule fires, an appropriate action (`change-velocity` or `change-orientation`) is sent to the agent's effectors, subject, of course, to "approval" by the agent's mediating control framework.[5] Clearly, actions effected at this level cannot be guaranteed to be rational since rules are memoryless and fire on the agent's sensory information alone. Consequently, each time a reactive rule fires, layer $\mathcal{M}$ (modelling) must be flagged (sent a message by layer $\mathcal{R}$) so that it can assess whether the resulting unplanned state change will require further processing. In particular, layer $\mathcal{M}$ will need to determine if any actions effected by layer $\mathcal{R}$ are likely to prevent the agent from achieving its planned tasks.

[5]Several reactive rules could fire simultaneously but only one is allowed to submit its corresponding action; currently the rule triggered by the (spatially) nearest environmental stimulus is chosen. Other selection policies may be considered in the future.

3.2 Layer $\mathcal{P}$ (planning)

The purpose of this layer is to generate and execute plans. Since an agent's main task typically involves relocating to some destination within certain prespecified time bounds, it makes sense for the agent to do some amount of forward planning before setting out (e.g. `determine-route`, `determine-cruise-speed`). In essence, we take Bratman's view [BIP88] that plans are useful for constraining the amount of subsequent deliberation an agent will need to perform. Nevertheless, since the agent is very likely to encounter other entities unexpectedly, complete, detailed plans are undesirable if replanning is to be kept to a minimum. Layer $\mathcal{P}$, therefore, is realized as a hierarchical, partial planner which can interleave plan formation and execution, and defer committing to specific subplan execution methods or temporal orderings of subplans until absolutely necessary. Also, since TouringMachines have limited computational resources, the planner is *embedded*; in other words, it is designed so that its operation can regularly be pre-empted and its state suspended for subsequent use [Kae91]. The plan elaboration scheme employed is akin to the partial elaboration method of PRS [GI89] and the lazy skeletal expansion scheme used in Phoenix agents [CGHH89]. These in turn appear to operate in a manner similar to NASL's control scheme for interleaving plan generation and execution [McD90].

The planner manipulates and instantiates template plans or *schemata* which it retrieves from a *schema library* (Figure 4). Schemata are procedural structures consisting of a body, a set of preconditions, a set of applicability conditions (e.g. temporal ordering constraints), a set of postconditions, and an associated cost in terms of computational resources. Schemata are either *primitive* or *composite*. Primitive schemata can either submit physical actions to be effected (e.g. `turn-to-angle`, `signal-left`) or perform various arithmetic or geometric calculations (e.g. `calculate-stopping-distance`). Composite schemata trigger library searches and subplan expansion. The planner also has access to a database of topological facts about its task domain.

The planner uses a fixed, combined earliest-first depth-first search strategy for constructing *single-agent* plans. Apart from occasionally generating sensory acts to determine the location of, say, some particular landmark, the planner remains largely "unaware" of what's going on around it. In particular, it does not consider what other agents are doing, this task being left to layer $\mathcal{M}$ which, in effect, is the only part of the agent that has any *reasoned* view of what other events are taking place in the world. So, while the planner is capable of some limited backtracking (e.g. to try an alternative execution method if the one initially chosen has failed or to try to re-satisfy a given applicability condition), initiation of *dynamic (re-)planning* (e.g. `stop-at-light`) is the

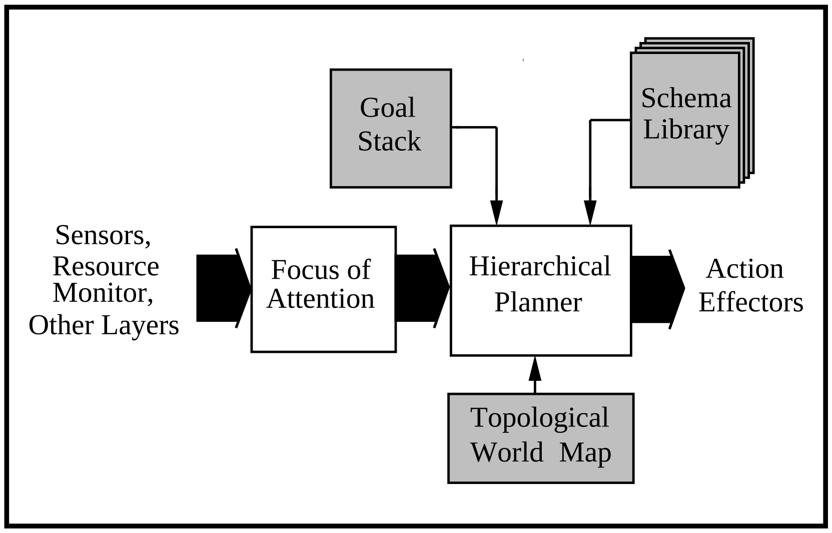


Figure 4: Top-level view of layer $\mathcal{P}$.

responsibility of layer $\mathcal{M}$. Layer $\mathcal{P}$, then, is able to take on new intentions and abandon old ones if layer $\mathcal{M}$ so dictates. In this manner, layer $\mathcal{P}$ keeps abreast of changes in the agent's environment.

3.3 Layer $\mathcal{M}$ (modelling)

The main purpose of layer $\mathcal{M}$ is to provide an agent with reflective and predictive capabilities. The agent realizes such capabilities by constructing cognitive models of world entities, including itself, which it uses as a platform for explaining observed behaviours and making predictions about possible future behaviours.[6] The potential gain in this approach 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 materialise. Goal conflicts can occur within the agent itself (e.g. 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 (e.g. 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 (e.g. accelerate by some amount to ensure the

[6]We assume TouringMachines can readily identify various physical properties of world entities such as type, size, Cartesian location, speed, acceleration, orientation, and signalled communications. This concords with most other simulated agent environments [CGHH89, DM90, PR90, SH88, VB90, Woo90].

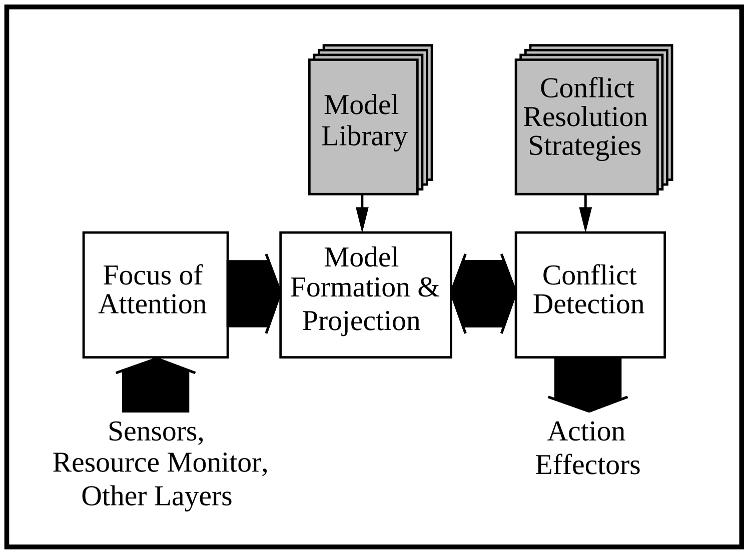

Model Library
Conflict Resolution Strategies
Focus of Attention
Model Formation & Projection
Conflict Detection
Sensors, Resource Monitor, Other Layers
Action Effectors

Figure 5: Top-level view of layer $\mathcal{M}$.

task deadline can still be met) or adopting some new intention (e.g. stop at the intersection to give way to an approaching agent).

Functions made available to the agent through this layer (see Figure 5) include a heuristic focus of attention module for creating closures within which to perform inferencing and a goal conflict detection/resolution facility for dealing with intra- and inter-agent conflicts. Like every module in the TouringMachine architecture, each function in layer $\mathcal{M}$ is resource-bounded. Because everything from sensing an object to inferring another agent's intentions has an associated cost measured in domain-specific computational resource units, and because each agent can only use a given number of these resources per unit of world time (after which the agent's three layers must submit their results for possible action taking), we can guarantee an upper-bound on the agent's inter-operation latencies [HR90] and thus ensure a degree of reactivity in the agent as a whole.

The structures used by an agent to model an entity's behaviour 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 prioritised goals or *Desires*; and I is its ascribed plan or *Intention* structure.[7] Using the terminology of Covrigaru and Lindsay [CL91], a TouringMachine's desires can either be *achievable* – with well-defined initial and final conditions (e.g. `reach-destination`), or *homeostatic* – to be achieved continuously

[7]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 [PGA86].

over time (e.g. `avoid-collisions`). The models used by an agent are, in fact, filed-in instances of model templates which the agent obtains from a Model Library (Figure 5). While all templates have the same basic 4-way structure, they could be made to differ in such aspects as the depth of information that can be represented or reasoned about (e.g. a particular template's B component might dictate that modelled beliefs are to be treated as hypothetical), initial default values provided, and 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 *discrepancies* between the entity's *actual* behaviour and that *predicted* by its model or, in the case of a self-model, between the agent's actual behaviour and that *desired* by the agent. 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 intentions. Noticing a discrepancy between actual and predicted (or desired) behaviours, however, need not on every occasion force the agent into a wholesale revision of its "faulty" model. This is because associated with each of the parameters of a model's C-vector are upper- and lower-bounds whose sizes can be chosen by the testbed user. The agent doing the modelling, then, will become "aroused" only if the entity's observed configuration parameters fall outside the corresponding C-vector bounds in its model of the entity. Clearly, different settings for these parameter bounds will affect both the amount of environmental change perceptible to the agent and the amount of time the agent will need to spend revising its models. Studying such tradeoffs in TouringMachines is a focus of current study. Achieving the optimal level of sensitivity to environmental change has also been recognised as a critical issue in Sanborn and Hendler's Traffic World system [SH88] and – through the use of plan-monitoring *envelopes* – in the Phoenix project [CGHH89].

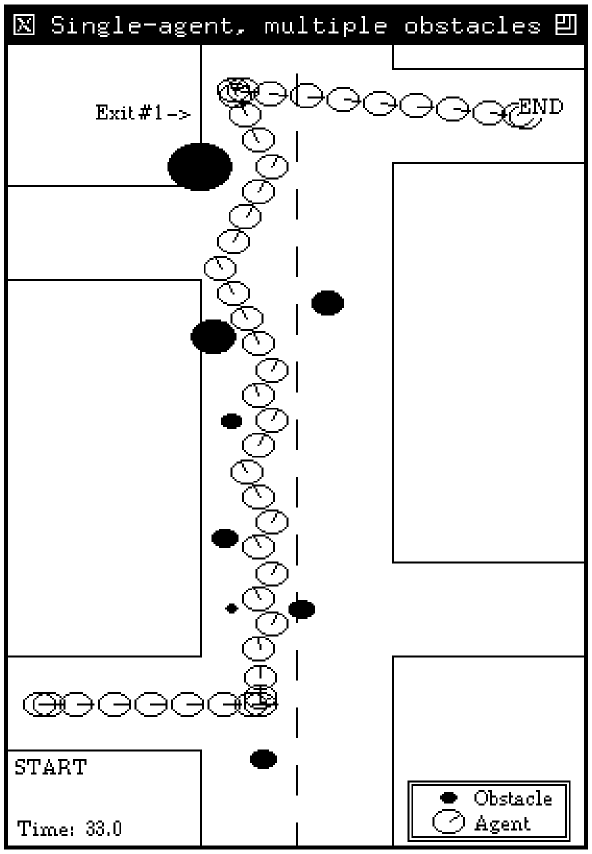
4 The TouringWorld Experimental Testbed

To validate TouringMachines, we have implemented our control architecture in SICStus Prolog and are experimenting with it in a simulated 2-dimensional world – the TouringWorld – occupied by, among other things, other 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 scenario parameters, a statistics-gathering package for agent performance analysis, and several text and graphics windows for displaying output.

Our testbed also provides a scenario definition facility which allows us to generate scenario instances from a fairly rich collection of *agent-* and *environment-level* parameters. So, for example, we can configure a TouringMachine to be variably reactive by altering parameters defining such things as the distribution of computational resources within its three control layers, the amount of forward planning it performs, the sensitivity of its reactive rules, or the frequency with which it senses or models the world. In a similar fashion, we can experiment with a TouringMachine's tolerance to environmental uncertainty by adjusting its sensing horizon, by tightening its initial goal deadline, by populating its world with many other fast-moving agents, or by varying the ratio of CPU to simulated world time used in the scenario. This last one affects the amount of time the TouringMachine has to deliberate between clock ticks.

In Figure 6, for instance, we can see the effect on the agent's behaviour of modifying the size of the bounds used to constrain allowable deviations from the parameter $\theta_{desired}$, the agent's desired heading. With wider bounds the agent fails to notice any discrepancy in its orientation and so does not take any corrective action; on the other hand, resolving goal discrepancies comes at a price in terms of computational resources, so finding the right size of bounds will typically require empirical validation. Figure 7 shows a pair of agents arriving at a light-controlled intersection. The two TouringMachines coordinate their activities by reasoning about actual and potential interactions between world entities, in this case, each other and the two sets of traffic lights. Agents ascribe intentions or plans to other agents as a way of explaining current and predicting future behaviour. By reasoning about the relationship between these hypothesised plans and each agent's goals, an agent can detect and resolve conflicts between entities before the situation can get out of hand (i.e. before the agent is prevented from achieving one of its goals). Again, success at predictive conflict resolution – in this particular domain – can be seen to depend on a number of internal agent parameters such as over what distance and how often sensing is performed, and how far into the future each agent projects for potential collisions.

The TouringMachine testbed has been designed to enable controlled, repeatable experimentation and to facilitate the creation of diverse agent scenarios for subsequent user analysis. Based on a number of single- and multi-agent experiments which we have performed, we are satisfied that our agents can behave robustly in the presence of unexpected obstacles while successfully accomplishing time-constrained, relocation-type goals. Just as we have



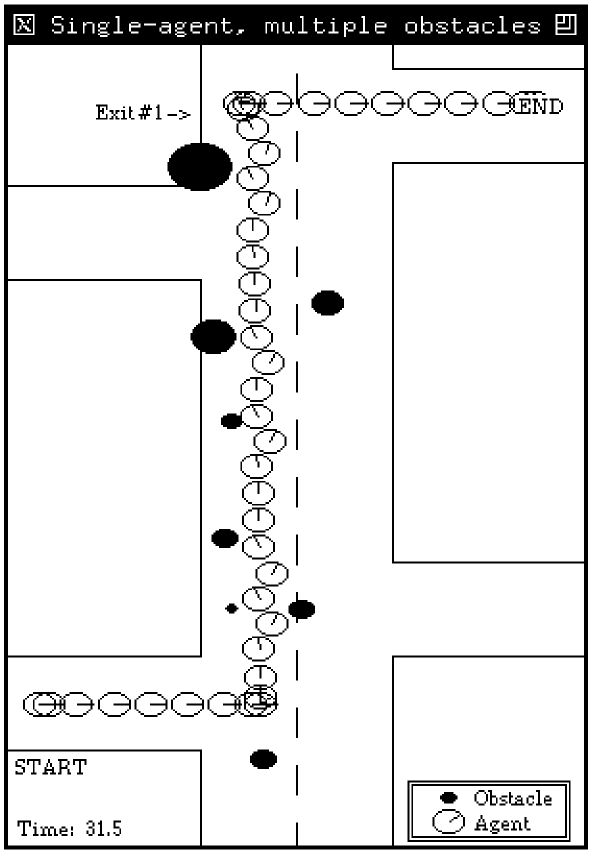


Figure 6: With the bounds around the parameter $\theta_{desired}$ set to +/-40° (left-hand frame), the agent covers more distance and so takes longer (1.5 time units) to arrive at its target than when its bounds are set to +/-0° (right-hand frame).

witnessed the strong influence environmental and task constraints can have on an agent's behaviour, we have also been convinced by the added value gained from causal modelling when agents with complex intentions are required to coordinate their activities with other agents in an effective and efficient manner. But this is just the beginning. Ultimately, through the design and analysis of more complex scenarios, we hope to gain more insight into the *behavioural ecology* – to use Cohen's terminology [CGHH89] – of TouringMachines. In other words, we are interested in studying, and eventually discovering general rules that describe, the relationships and tradeoffs that exist between an agent's design (in other words, the particular configuration of its functional capabilities and knowledge sources), its environment, and the repertoire of demonstrable behaviours that the agent is capable of. So, for example, we are interested in understanding how well a given TouringMachine configuration might perform across a wide range of environments and also how the behaviours of different configurations of TouringMachines compare when placed in a single common environment. Some criteria with which to

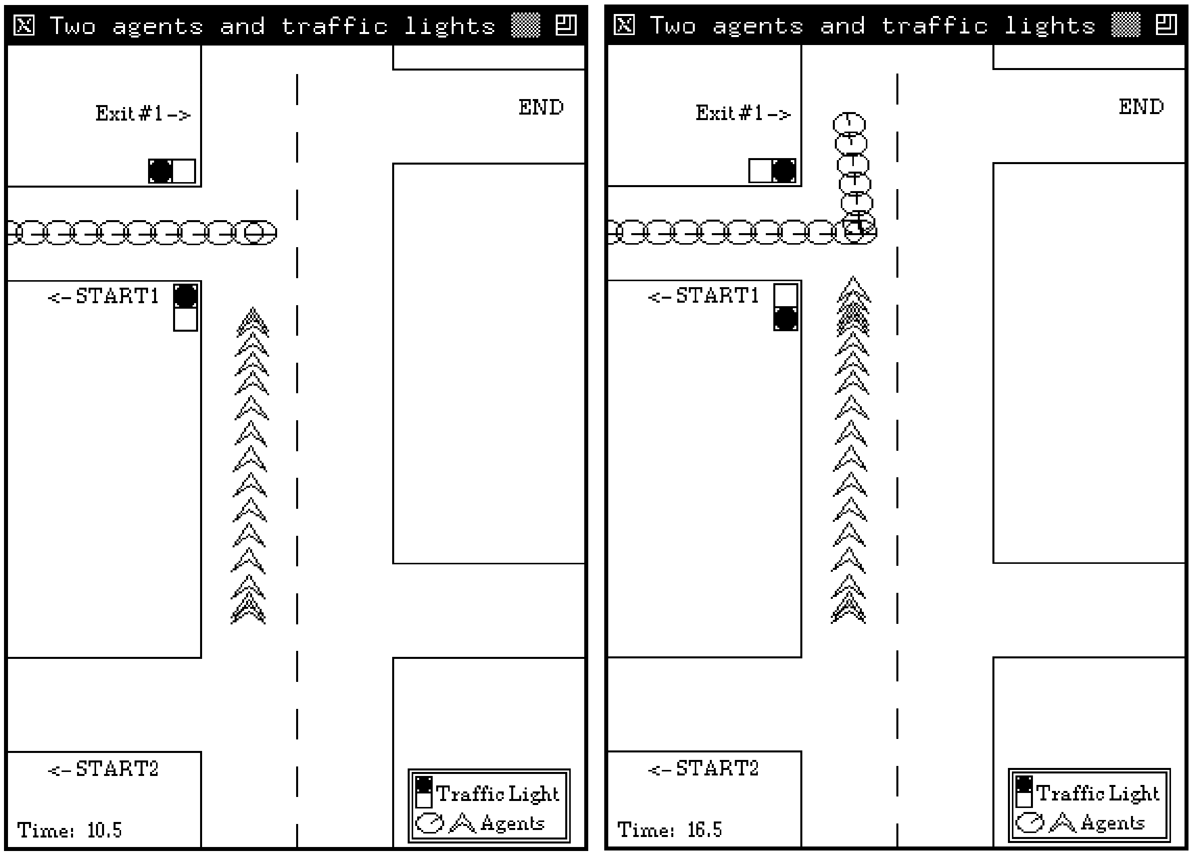


Figure 7: By reasoning about the interactions between themselves and the relevant traffic lights, the two agents coordinate their activities accordingly: in the left-hand frame, the chevron-shaped agent has stopped at its red light and the round agent has proceeded into the intersection; once the chevron-shaped agent's light has turned green (right-hand frame), it sets off to complete its initially intended task.

evaluate the performance of our agents have already been employed and include, among others, resource consumption and utilisation, wasted planning effort (e.g. amount of backtracking or replanning required), number of successful/unsuccessful actions effected, ratio of successful to unsuccessful model-based predictions, and delay in arriving at a target destination.

5 Conclusions

We have presented a new control architecture for resource-bounded, goal-directed, mobile agents operating in dynamic, multi-agent environments. Our layered, activity-producing architecture integrates both deliberative and non-deliberative control features enabling a TouringMachine to produce a range of reactive, goal-oriented, reflective, and predictive behaviours as demanded by the agent's goals and environmental situation. This empowers agents to deal robustly and flexibly with events and tasks at different levels of granularity (e.g. avoiding collisions, accomplishing complex, resource-bounded goals, and predicting the behaviour of other agents with equally complex goals). We have also briefly described a feature-rich simulation testbed within which we have started to study design-behaviour-environment tradeoffs.

By using a highly parametrized, layered architecture and testbed we have benefited greatly in terms of our effort to design, implement, and test a series of different agent configurations. Our experience so far has demonstrated that TouringMachines can be configured to behave with a high degree of robustness and flexibility in dynamic situations. The work presented here is ongoing: future work will consider the implementation of a fourth layer $\mathcal{L}$ to assist the agent in self-tuning or learning internal parameters, as well as experimenting with a variety of *heterogeneous* agent configurations in multi-agent settings. Making an agent tune its own parameters is non-trivial as optimal parameter values are to a large extent task- and situation-specific, thus requiring that the agent be able to classify different operational contexts. Handling heterogeneous agents will require that agents be able to model and predict the behaviour of agents whose beliefs or goals differ substantially from their own (e.g. obeying versus not obeying traffic regulations). At present only agents' intentions or plans are hypothesised and subject to revision; beliefs and goals are assumed identical (although target destinations and goal deadlines can differ). A more general modelling layer would need to deal with the possibility that unexpected behaviours are the result of fellow agents having substantially different "insides" from the observer. We believe these investigations will provide us with further clues about how best to design dynamic, rational, autonomous agents.

6 Acknowledgements

This work was supported by a Bell-Northern Research Postgraduate Scholarship and a CVCP Overseas Research Student Award. I would like to thank William Clocksin, Julia Galliers, Han Reichgelt, and Jane Dunlop for their helpful advice and support, and Barney Pell for many fruitful and enjoyable discussions.

References

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

[BIP88] Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349–355, November 1988.

[Bra90] Michael E. Bratman. What is intention? In P.R. Cohen, J. Morgan, and M.E. Pollack, editors, *Intentions in Communication*. MIT Press: Cambridge MA, 1990.

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

[CGHH89] 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.

[CL91] Arie A. Covrigaru and Robert K. Lindsay. Deterministic autonomous agents. *AI Magazine*, 12(3):110–117, 1991.

[DM89] Edmund H. Durfee and Thomas A. Montgomery. MICE: A flexible testbed for intelligent coordination experiments. In *Proceedings Ninth Workshop on Distributed Artificial Intelligence*, Rosario Resort, Eastsound, WA, 1989.

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

[Fer92] Innes A. Ferguson. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, Computer Laboratory, University of Cambridge, Cambridge, UK, 1992. Forthcoming.

[Fir87] James R. Firby. An investigation into reactive planning in complex domains. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 202–206, 1987.

[GI89] Michael P. Georgeff and François Felix Ingrand. Decision-making in embedded reasoning systems. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 972–978, 1989.

[HR90] Barbara Hayes-Roth. Architectural foundations for real-time performance in intelligent agents. *The Journal of Real-Time Systems*, 2:99–125, 1990.

[Kae87] Leslie Pack Kaelbling. An architecture for intelligent reactive systems. In M.P. Georgeff and A.L. Lansky, editors, *Reasoning about Actions and Plans - Proceedings 1986 Workshop*, pages 395–410. Morgan Kaufmann Publishers Inc: Los Altos, CA, 1987.

[Kae91] Leslie Pack Kaelbling. Specifying complex behaviours for computer agents. In Luc Steels and Barbara Smith, editors, *Proceedings Eighth Conference on Artificial Intelligence and the Simulation of Behaviour*. Springer-Verlag: London, UK, 1991.

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

[LC83] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed: A tool for investigating distribution. *AI Magazine*, 4(3):15–33, 1983.

[Mae90] Pattie Maes. Situated agents can have goals. *Robotics and Autonomous Systems*, 6(1&2):49–70, 1990.

[McD90] Drew McDermott. Planning and acting. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 225–244. Morgan Kaufmann Publishers Inc: San Mateo, CA, 1990.

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

[PR90] 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.

[SH88] J. Sanborn and J. Hendler. A model of reaction for planning in dynamic environments. *International Journal of Artificial Intelligence in Engineering*, 3(2):95–102, 1988.

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

[Woo90] Sharon Wood. *Planning in a Rapidly Changing Environment*. PhD thesis, University of Sussex, Brighton, UK, 1990.