Simulating agents and their environments

Darryl Davis, Aaron Sloman and Riccardo Poli

School of Computer Science
The University of Birmingham
Birmingham, B15 2TT
United Kingdom

Abstract

This paper describes a toolkit that arose out of a project concerned with designing an architecture for an autonomous agent with human-like capabilities. Analysis of requirements showed a need to combine a wide variety of richly interacting mechanisms, including independent asynchronous sources of motivation and the ability to reflect on which motives to adopt, when to achieve them, how to achieve them, and so on. These internal 'management' (and meta-management) processes involve a certain amount of parallelism, but resource limits imply the need for explicit control of attention. Such control problems can lead to emotional and other characteristically human affective states. We needed a toolkit to facilitate exploration of alternative architectures in varied environments, including other agents. The paper outlines requirements and summarises the main design features of a toolkit written in Pop-11. Some preliminary work on developing a multi-agent scenario, using agents of differing sophistication is presented.

1 Introduction

The term agent arises from a number of areas of AI and cognitive psychology. At one end of the spectrum, the impetus behind a theory of agents is provided by the desire to produce computational cognitive models that explain and/or simulate, to some degree, reported findings and theories in cognitive psychology. In doing so a computational theory and model of intelligence can be approached. Recently agents have been used to simulate and advance more complete models (e.g. SOAR [8] and the Birmingham cognition and affect project [4, 20]). At the other end of the spectrum sit the engineers, wanting to develop theories of agents (and robots) that allow specific problems to be automated.

Exploring the design space for complete agents is a daunting problem, particularly where the long term aim is the design of agents with human-like qualities.

An all embracing theory of agency is not as yet extant - a number of researchers believe that it is an attainable but very distant goal, along the lines of all encompassing theory of cognition. Theories of such sophisticated information processing entities are presently incomplete and tentative.

There are some attempts to describe complete architectures (e.g. [7]) but implementations are usually lacking, partly due to the difficulty of building complete implementations. There are some attempts to design and implement more complete architectures (e.g. [6, 5, 2]) though these projects are still at a relatively early stage, and those involving complete physical robots include only a small subset of the motive management processes that interest us. Even partial theories that address specific capabilities (e.g. attention, learning, perception etc.) are lacking in predictive and explanatory power. Therefore in pursuing our long-term goal, it would be unwise to constrain ourselves to particular hard or soft constraints arising from the adoption of specific architectures or formalisms.

SIM_AGENT is an experimental toolkit to support exploration of design options for one or more objects interacting in discrete time. The need for the toolkit arose out of a long term research project concerned with architectures for autonomous agents with human-like capabilities including multiple independent asynchronous sources of motivation and the ability to reflect on which motives to adopt, when to achieve them, how to achieve them, and so on.

Whereas most people building agent architectures aim for a single architecture, we wished to be able to explore a variety of architectures, and, moreover, to support simulation of interacting agents with different architectures. A generic toolkit is needed to support the exploration of alternative designs [14, 15]. In particular, a toolkit providing the following features is needed:

Minimal ontological commitment: i.e. many kinds of objects with very different architectures should be supported.

External behaviour: which is detectable by or which affects other objects or agents, e.g. movement and communication.

Internal behaviour: involving (possibly resource-limited) mechanisms for changing internal states that are not directly detectable by others, e.g. beliefs, motives, goals, plans, etc.

Rich internal architectures within agents: e.g. several layers of sensory interpretation, multiple interacting rule-based systems, 'cognitive reflexes', neural nets, and other trainable sub-mechanisms.

Use of classes and inheritance: e.g. one or more default classes of objects should be provided, but it should be easy to override defaults by defining subclasses with more specific forms of behaviour, without having to edit the generic code.

Control of relative speed: allowing easy variation of relative speeds of different agents and also relative speeds of different components within an agent (e.g. perception, planning).

Rapid prototyping: enabling the easy mapping from idea to specification to implementation.

This paper describes a toolkit supporting a mixture of symbolic (e.g. rule-based) and sub-symbolic (e.g. neural) mechanisms, which is being used both here in Birmingham and at DRA in Malvern. The paper ends with a brief description of how the toolkit is being used to further ongoing experiments with agent architectures.

2 The SIM_AGENT toolkit

This toolkit is implemented in Poplog Pop-11 [1], using the Objectclass and Poprulebase [17] libraries. Objectclass, designed by Steve Knight at Hewlett Packard provides CLOS-like object oriented facilities integrated with Pop-11 syntax and semantics, including classes, inheritance and generic methods that can be redefined for specific user-defined classes of objects or agents. The class hierarchy in the SIM_AGENT toolkit starts with the top level class **sim_object** which supports (at least) the following variety of object types:

Compound objects are composed of (or possibly 'own') two or more other objects, e.g. a forest (composed of paths, trees, etc.), a town (composed of roads, houses, parks, people, vehicles, etc.), a house (composed of rooms, doors, etc.) a family (composed of various people), etc. A **simple object** has no parts that are objects managed by the scheduler but may nevertheless have a complex *internal* architecture, involving several interacting subsystems. e.g. an intelligent agent.

Active objects change their state spontaneously, i.e. without the intervention of other objects. Some of the active objects merely obey physical laws, whereas others, which we call **agents** take in and manipulate information and vary their behaviour as a result of processing information. The division between agents and non-agents is not very sharp [15]. We are specially interested in agents that generate their own motivators. These are "autonomous".

Passive objects are objects which do not change their state spontaneously. Examples might be walls, ditches, ladders, roads. A battery whose charge runs down, a volcano that erupts from time to time or a tree that grows would be examples of *active* objects, though they are not *agents*. Some passive objects become active under the *control* of agents, e.g. cars, screwdrivers, and we can refer to these as **instruments**. Other passive objects

may *react* to impact or pressure (e.g. a wall). We call these **reactors**. These distinctions are all provisional and "fuzzy".

The SIM_AGENT toolkit provides a scheduler which 'runs' objects in a virtual time frame composed of a succession of time slices. In each time-slice every object in the world is given a chance to do three things:

(a) sense its environment, including creating internal representations derived from sensory data,

(b) run processes that interpret sensory data and incoming messages, and manipulate internal states, and

(c) produce actions or messages capable of influencing other objects in the next time slice.

In each time-slice the scheduler has two runs through the list of objects.

- First it allows each object to sense the state of the world and, if appropriate, detect communications from other agents, and then do as much internal processing as the time slice permits, e.g. changing internal databases and preparing external actions and communications.

- On the second pass, the scheduler transfers messages from sources to targets, and runs external action routines corresponding to each object with actions pending. Actions that need to be combined into a resultant may need special treatment.

This two-pass strategy makes behaviour (generally) independent of the order of objects in the list given to the scheduler, since nothing detects the state of the environment until everything has completed its actions in the previous time-slice.

The toolkit allows users to explore different forms of internal processing, but provides special support for agents whose internal processing can be implemented using interacting collections of condition-action rules, including rules that invoke non-symbolic 'low level' mechanisms. This is based on Poprulebase, a forward chaining production system designed by the authors which also provides many of the features associated with conventional rule interpreters as well as the ability to drop into Pop-11 where appropriate. In particular, Poprulebase supports the following:

- Each agent type has an associated collection of rulesets. Each ruleset is a collection of condition-action rules that interact via one or more 'databases' or working memories internal to the agent. Thus one ruleset might be concerned with interpretation of low level sensory information, another with generating motivators in response to the formation of new beliefs, another concerned with assessing the importance of motivators, another concerned with planning, and so on.

- Rules can switch between databases, push them onto a stack, restore them, etc., as in SOAR [8]. Rules can also transfer control to a new ruleset. Rulesets may be stacked then restored.

- Within an agent, learning or developmental processes may change individual rules, or introduce new rulesets, or introduce new interactions between rulesets, e.g. by adding new communication channels.

- For more sophisticated reasoning, it is possible to invoke Prolog, or some sort of theorem prover, since the rules can invoke Pop-11, and Poplog Pop-11 can invoke other languages.

- If 'sub-symbolic' mechanisms are required, they can be invoked by appropriate rules, e.g. using FILTER conditions and SELECT actions (see [19, 16]).

- Each ruleset corresponds to a sub-mechanism or 'context' in the agent. E.g. a context may be analysing incoming messages, or analysing sensory data, or deciding which goals to adopt, or planning, or executing goals, or constructing messages to transmit. The facility in Poprulebase to switch between rulesets or between databases permits rapid switching between these contexts.

- Parallelism between rulesets within an agent can be simulated by limiting the number of cycles allocated to each ruleset in each time-slice, and repeatedly running all the (runnable) rulesets.

- Varying relative speeds of different kinds of internal processing can be achieved by associating different numbers of interpreter cycles with different rulesets. Thus it is possible for one agent to have perceptual processes speeded up, per time slice, relative to planning processes, and another agent to have planning processes speeded up relative to perception.

It is worth noting that, in order to support exploration of design options, we have adopted a 'virtual' representation of time in the toolkit instead of real time. Real time or even cpu time measures are highly arbitrary and implementation dependent, and meaningless relative to the aims of typical simulations. This representation has the following features:

- The scheduler gives each object a chance to 'run' in each time-slice.

- What it means for an object to run is defined via methods which perform the internal actions, the sensory detection, and the external actions. They should all assume that the amount of *virtual* time available in a time-slice is fixed.

- The use of virtual time allows simulated speeding up of processing in a particular subset of agents by allowing them to do more in each time-slice. Similarly faster physical motion would be represented by larger physical changes in each time-slice.
- Specifying relative speeds is entirely up to the user, and can be determined by class of object, by ruleset, etc., and can be varied dynamically.
- The toolkit has been designed to support flexible design and exploratory development through rapid prototyping, rather than optimal speed or space efficiency.

3 Using the toolkit

In order to use the toolkit, the user must be prepared to do the following:

- Define the ontology, i.e. classes of objects and agents required, making all of them subclasses of the classes provided. Eventually a library of standard classes will be provided.
- Define the sensor methods and do_action methods for the classes. This includes defining internal formats for sensory information and action specifications.
- Define the send_message method for the classes which need it, and decide on the formats for different kinds of messages and the protocols for sending and receiving messages. (E.g. some may require an acknowledgement some not, and there may be requests, orders, questions, answers to questions, advice, permissions, and so on.) A library of useful formats will be developed.
- Define the (Poprulebase) rulesets for internal processing of the different classes of agents, and the rules for each ruleset. This involves defining the formats for the different kinds of information to be used in the internal databases, e.g. sensor information, beliefs about the environment, motivator structures, plan structures, management information, etc.
- Specify the initial databases for each type of agent.
- Specify which collection of rulesets should be used by each type of agent, and in what order (in each time slice), and the relative processing speeds associated with each ruleset.
- Create all the required initial instances of the agent classes and put them into a list to be given to the scheduler.

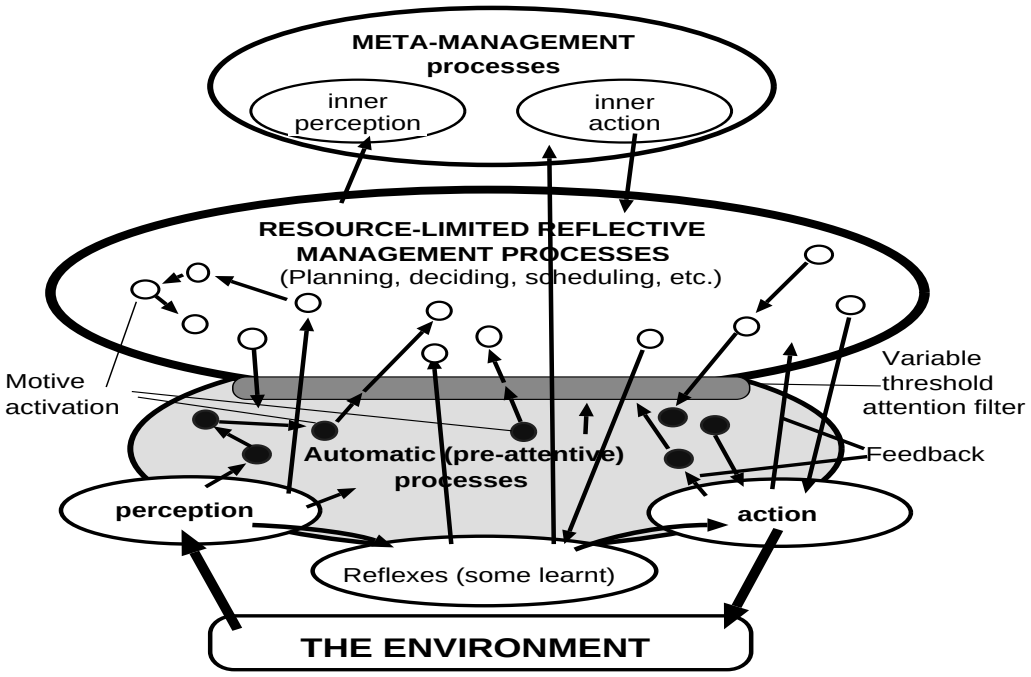

Figure 1: Towards an Intelligent Agent Architecture

- Create any other data-structures required, and the procedures to access and update them (e.g. a map of the world, if the world itself is not an object).
- Define any required object-specific tracing methods, e.g. graphical tracing methods.

By applying this procedure to different "worlds", collections of re-usable libraries corresponding to particular ontologies and applications will be developed over time, and shared between users. Simulation of physical movement and graphical projection are in part already available and will take the form of a re-usable library class, including graphical tracing.

3.1 The creche simulation

Figure 1 depicts approximately the type of architecture we have been exploring (described incrementally in [18, 11, 4, 3]).

The figure, based partly on ideas by Luc Beaudoin [3] and Ian Wright, is intended to be suggestive of an architecture in which there are many different coexisting components with complex interactions. Some processes are automatic

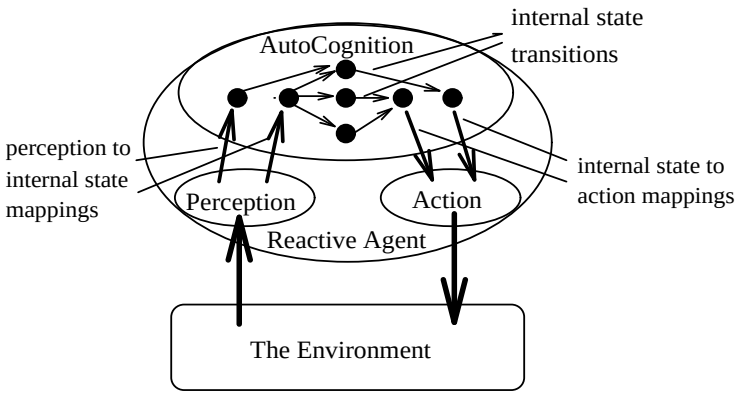

Figure 2: Simple Reactive Architecture

(pre-attentive) in the sense that all changes are triggered directly. Others, the 'management processes' are 'reflective' or 'attentive' knowledge-based processes in which options are explicitly considered and evaluated before selection. The latter are resource-limited and may sometimes require both attention filters to protect them from disturbance and meta-management processes to regulate and direct them, involving some sort of self-monitoring. Reflexes (learnt or innate) bypass 'normal' processing. The diagram should not betaken to imply any sharp division between processes concerned with perception or action and other cognitive processes.

The internal 'management' (and meta-management) processes involve a certain amount of (asynchronous) parallelism, but we wanted to show how resource limits restricting parallelism in high level processes can lead to emotional and other characteristically human states involving partial loss of control of attention and thought processes [10, 12, 13, 3]. This requires an architecture combining a wide variety of types of mechanisms. Like the OZ project [2] we are initially interested in 'broad and shallow' architectures, combining many sorts of capabilities, where each capability is initially simulated or implemented in a shallow fashion. Deeper, more complete, implementations will follow.

We are currently working towards a full implementation of the architecture sketched in figure 1, and hoping to compare its performance with other agent architectures using a creche scenario. This consists of several rooms with connecting doors, an infirmary, a variable number of babies and a minder (or nursemaid). The minder's task is to monitor the environment using its simulated (visual and auditory) perceptual processes, and ensure that no room becomes overpopulated, that sick babies are moved (using an effector attached to the camera) to the infirmary and to dispel babies that are too old.

Figure 2 depicts the simple agent architecture used to simulate the babies in the environment. Conceptually the babies can be considered as relatively

simple autonomous entities with limited internal processing (with no planning, explicit goal manipulation etc.). Three forms of babies are allowed: active, aggressive, and fearful. Active babies follow a naive physics model (obeying Newtonian dynamics). Aggressive babies have the potential to become thugs when the room they inhabit becomes overpopulated; thugs attack other babies. Fearful babies become timid once attacked and then flee from any other baby (irrespective of its character). Babies can be injured (through collisions) and have their vitality reduced (through being attacked). They can then become ill, and subsequently they die. Ill babies cannot move, while dead babies perform no processing!

We allow these simple agents a shallow perceptual system (they can sense other 'active' **sim_objects** in the same room and if they are within a certain distance). This environmental information (they are also given information stating whether the current room is over-crowded) is manipulated by one of three rulesets. This ruleset maps the environmental information into 'internal' information. The nature of this mapping is determined by the agent's character, which can be altered by this information. The second ruleset maps this internal information into a set of action potentials. Action potentials may take the form of simple goals, such 'avoid agentX' or 'attack agentY', or less directed behaviour such as the default action of continue moving in the current direction. No explicit planning occurs in the reactive agents, their internal state is (potentially) transformed by environmental factors and their actions are a response to their internal state. The third set of rules maps the action potentials into an effect in the simulated environment; i.e. it simulates the Newtonian physics of the babies' movement in the room (collisions with other babies, rebounds off walls, and room changes through *collisions* with doors). These simple dynamic objects could be implemented in a number of other ways, for example the avoidance behaviour associated with fleeing an aggressor could well make use of existing hybrid mechanisms discussed in [19].

The babies together with their five room environment provide a dynamic environment of sufficient complexity to gauge the performance of more sophisticated agents. Figure 3 depicts the architectural processes associated with a first implementation of the minder. Initially we shall provide a shallow implementation of the architecture, but aim to later imbue it with greater sophistication, for example learning and communication with other agents (a multi-minder, multi-baby scenario). We have taken a thin computational line through the architecture depicted in figure 1, simulating many processes and interactions with the environment (e.g. visual and auditory perception) in a very shallow manner (Broad but Very Very Shallow - BVVS). This adoption of a BVVS standpoint allows us to address issues such as what processes we need, how these different processes interact, the necessity for feedforward and feedback mechanisms, and how to represent information, belief and knowledge internally.

The minder is an abstract entity which interfaces to the environment through directed organs (typically sensory and effective processes). Implemented as a

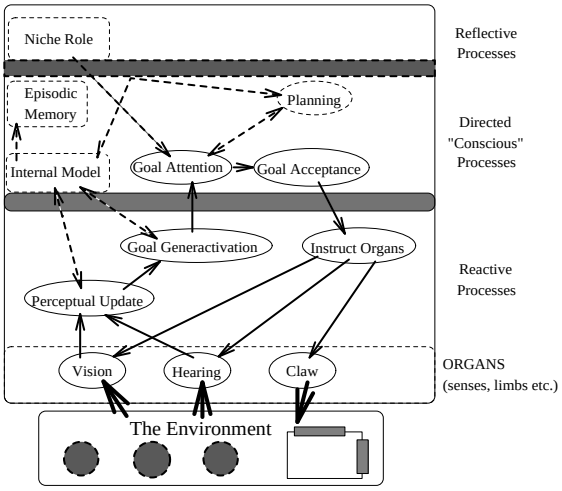

Figure 3: Prototypical Broad but Very Very Shallow Architecture

compound object, it currently consists of three organs (ears, eyes and arm), with two (the eyes and arm) occupying the same *physical* location, plus a three tier architecture (the reflective processes are currently a set of static niche roles used in goal generactivation, e.g. 'attend sick babies', 'move aggressive babies to less populated rooms' etc.). Within each time slice, the minder first runs its organs, before performing its own internal processing. Each organ first attends to any instructions from the minder (e.g. 'move to position (X,Y)' or 'grasp object O') before running any sensory processes. The resultant structured sensory information is used by the minder to update its internal model of the environment. The internal model, together with its niche role information, is used to generate goals. Goals with a high insistence value (we use a variable threshold attention filter) become attended to at the second level of the architecture. Here conflict resolution between competing goals and some rudimentary planning allow a compatible set of goals to be pursued. Pursued goals are typically mapped into a set of instructions which are planted directly in the relevant organs's database. Attended goals are 'carried' over to the next cycle when the attentive processes determine whether they have succeeded. Complex goals are similarly processed, with a reactive planner determining what current sub-goal may be currently acheived. More abstract reasoning processes will follow in due course.

4 Conclusion

Our toolkit is very general, but not so general as to be completely useless: a frequent consequence of generality. It is currently being used for a number of

experiments; these experiments themselves are being used to drive the development of the toolkit. These experiments typically involve the explicit design of architectures, including architectures that change themselves or use trainable sub-mechanisms. Evolutionary computation techniques can also play a role in it [9]. For example, genetic algorithms or similar techniques can be used to evolve designs instead of creating them explicitly. It should be easy to copy the ideas in any Lisp-like language rich enough to support rule-based processing, and including 'hooks' to invoke lower level languages.

5 Acknowledgements

The research reported in this paper was carried out in collaboration with Luc Beaudoin, Ian Wright, Glyn Humphreys, and other members of the Cognition and Affect project at the University of Birmingham [20]. This research is funded by a University Of Birmingham internal grant, the Renaissance Trust, the UK Joint Council initiative in HCI and Cognitive Science and DRA Malvern.

References

[1] J.A.D.W. Anderson, editor. *POP-11 Comes of Age: The Advancement of an AI Programming Language*, Chichester, 1989. Ellis Horwood.

[2] J. Bates, A. B. Loyall, and W. S. Reilly. Broad agents. In *Paper presented at AAAI spring symposium on integrated intelligent architectures*, 1991. (Available in SIGART BULLETIN, 2(4), Aug. 1991, pp. 38–40).

[3] L. P. Beaudoin. *Goal processing in autonomous agents.* PhD thesis, School of Computer Science, The University of Birmingham, 1994.

[4] L.P. Beaudoin and A. Sloman. A study of motive processing and attention. In A.Sloman, D.Hogg, G.Humphreys, D. Partridge, and A. Ramsay, editors, *Prospects for Artificial Intelligence*, pages 229–238. IOS Press, Amsterdam, 1993.

[5] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[6] B. Hayes-Roth. Intelligent control. *Artificial Intelligence*, 59:213–220, 1993.

[7] M. L. Minsky. *The Society of Mind.* William Heinemann Ltd., London, 1987.

[8] A. Newell. *Unified Theories of Cognition.* Harvard University Press, 1990.

[9] R. Poli, M. Brayshaw, A. Sloman. *A Hybrid Rule-Based System with Rule-refinement Mechanisms.* Procs. Expert Systems 95, (In prep).

[10] H. A. Simon. Motivational and emotional controls of cognition'. Reprinted in *Models of Thought,* Yale University Press, 29–38, 1979.

[11] A. Sloman. Motives mechanisms and emotions'. *Emotion and Cognition*, 1(3):217–234, 1987. Reprinted in M.A. Boden (ed), *The Philosophy of Artificial Intelligence*, 'Oxford Readings in Philosophy' Series, Oxford University Press, 231–247, 1990.

[12] A. Sloman. Prolegomena to a theory of communication and affect. In A. Ortony, J. Slack, and O. Stock, editors, *Communication from an Artificial Intelligence Perspective: Theoretical and Applied Issues*, pages 229–260. Springer, Heidelberg, Germany, 1992.

[13] A. Sloman. The mind as a control system. In C. Hookway and D. Peterson, editors, *Philosophy and the Cognitive Sciences*, pages 69–110. Cambridge University Press, 1993.

[14] A. Sloman. Prospects for AI as the general science of intelligence. In A.Sloman, D.Hogg, G.Humphreys, D. Partridge, and A. Ramsay, editors, *Prospects for Artificial Intelligence*, pages 1–10. IOS Press, Amsterdam, 1993.

[15] A. Sloman. Explorations in design space. In *Proceedings 11th European Conference on AI*, Amsterdam, 1994.

[16] A. Sloman. Filtering of rules in lib poprulebase, 1994. Available at URL ftp://ftp.cs.bham.ac.uk/pub/dist/poplog/prb/help/prb_filter.

[17] A. Sloman. Poprulebase help file, 1995. Available at URL ftp://ftp.cs.bham.ac.uk/pub/dist/poplog/prb/help/poprulebase.

[18] A. Sloman and M. Croucher. Why robots will have emotions. In *Proc 7th Int. Joint Conf. on AI*, Vancouver, 1981.

[19] A. Sloman and R. Poli. SIM_AGENT: A toolkit for exploring agent designs. In *ATAL95*, Agent workshop at IJCAI95. pages 304–316 Montreal, 1995.

[20] I. Wright. A summary of the attention and affect project, 1994. Available at URL ftp://ftp.cs.bham.ac.uk/pub/dist/papers/cog_affect in the file Ian.Wright_Project_Summary.ps.Z.