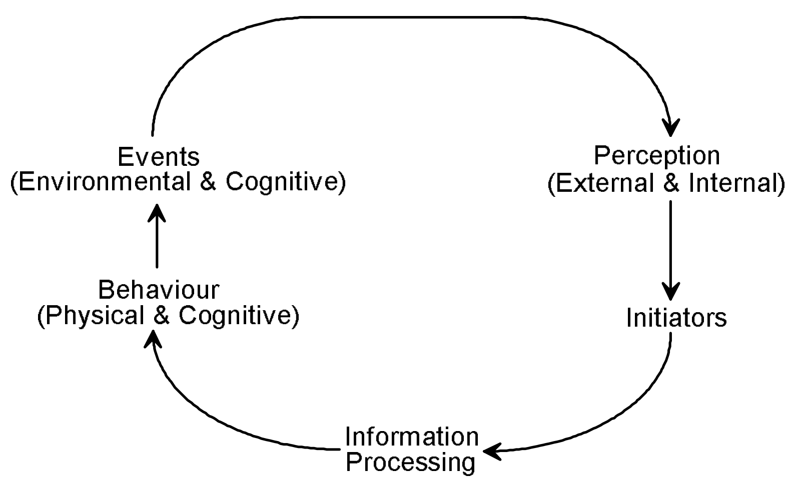
APPENDIX A

THE ROUTES TO BEHAVIOUR

The routes open for utilisation by an agent in such a production are of crucial importance to the viability of that agent and can be seen to be determined via a complex interaction between the agent's domain niche, and the nature of that agent's command and control system. Indeed, the relationship can be viewed as being somewhat circular in form - the intended niche determining the desired routes and thus the required command and control system, and conversely, the command and control system determining the available routes and the type of niche the agent can exist viably within (e.g. Hayes-Roth 1993b). It is therefore useful to take a look, even if only a somewhat cursory one, at three methods by which action may be instigated within an autonomous agent.

Fig. A.1 - From Perception to Action (and Back)



Effectively behaviour within an autonomous agent can be classified as originating from the following three routes, with perceptual events (environmental or cognitive) being termed *initiators*, and the resultant output (motor or cognitive[a]) being termed *behaviour* (see Fig. A.1):

i. Innate reflexive behaviour (Type I).

[a] It is important to note that the results of processing perceptual data may give rise not only to effector output by an agent, but also to other cognitive events.

ii. Goal mediated behaviour (Type II).

iii. Learnt reflexive behaviour (Type III).

Let me clarify the distinction. The term *reflexive* is used to imply that in response to a given initiator the behaviour generated is done so:

- Unavoidably, i.e. once an initiator is 'triggered' its response will occur without fail, even if that response happens to be adverse.
- Without recourse to deliberative processes, i.e. no planning, scheduling, or problem solving of any kind is carried out.
- Non-introspectable, i.e. the manner by which the behaviour is generated is not open to inspection by cognitive processes.

It can thus be seen that the term *reflexive* covers a wide range of non-goal mediated behaviour from the innate physical reflexes (Type I behaviours), e.g. the knee jerk reflex, to what might be termed learnt cognitive reflexes (a subset of Type III behaviours), e.g. the disgust one feels at rape. However, in addition to these types of non-goal mediated behaviours, the term can also be applied to certain instances of goal mediated behaviour, where the required behaviour has become so 'ingrained' through learning it has become reflexive, the majority of so called Type III behaviours - reflexive *goal mediated* behaviours. These behaviours, although goal mediated, become unavoidable, executable without recourse to deliberative processes, and non-introspectable. A good example of this would be riding a bicycle; you cannot forget how to ride once you have learnt, nor do you need to deliberate about how to ride, indeed attempting such deliberation is pointless: the process is not introspectable.

In contrast to reflexive behaviour is non-reflexive, or *goal mediated* behaviour (Type II); behaviour which is brought about as a result of the application of deliberative processes by a system in its attempt to satisfy certain objectives, or what I term goals. However, the representation within a system of such goals may vary from an explicit symbolic form (see Chapter 4 on Traditional AI), to one which is non-symbolic (see Chapter 6 on Nouvelle AI). The crucial point being, that regardless of whether goals are represented explicitly or not, they are generated by the system as it operates within its environment, and their achievement is determined by some form of deliberative process, the results of which effect how the system then behaves. A good example of this would be the formation of a goal to eat in response to hunger signals, and the achievement of this goal by cooking - a task which is not achievable reflexively (although some of the sub-tasks involved in cooking dinner may be). Although this idea that autonomous agent behaviour is purposeful, or goal directed, appears self evident, psychologists have for many years eschewed such explanations. Nonetheless, in more recent years this trend has been redressed, and the mechanisms by which goal directed behaviour can be produced are now widely recognised as being central to our understanding of the origins of voluntary behaviour (e.g. Nelson 1993).

Within the animal behaviour and psychology literature models of command and control systems have tended to typically be discussed in terms of self-regulation, e.g. Power's *control theory*[b] 1973; Carver & Scheier's *feedback model*[4] 1981 & 1988; Kuhl & Kraska 1989; and Kuhl 1992. Power's control theory was the first to put forward the idea that purposive behaviour results from processes by which an agent self-regulates itself through changing its perceptions to match a reference value. This reference value may be imposed internally, by the agent itself, or externally, e.g. by an examination board. Using models of this genre leads to behaviour being conceptualised as resulting from a hierarchically organised plethora of these self-regulating negative feedback loops, with the higher loops having a more 'abstract' influence upon behaviour, e.g. self ideals, and principles. As such, models of this type are seen to equate well with Sloman's view of behaviour generation (e.g. Sloman 1992c). However, this 'basic' conceptualisation fails to account for the tendency of behaviour to be generated in accordance with the control layer most salient for the individual at a given time. While appearing to be plausible such basic feedback models lack a notion of attention - the ability for the agent to alter the relative influences of control layers when so required (Nelson 1993).

Carver & Scheier extended these feedback models by introducing such a notion. They viewed attention as a bi-directional process, being able to focus internally or externally, that allowed a system to focus on individual layers in its control hierarchy. A control layer (feedback loop) so 'attended' influenced behaviour almost to the complete neglect of other higher up layers.

---

b Taken from Nelson (1993).

Furthermore, they also introduced the notion of *expectancy* - the value that influences the degree to which an agent will continue its attempts to achieve and maintain a reference value. According to this model, one possible cause of depression may thus be expectancy values which are set at an unattainable point (showing how design investigation can give explanations for behaviour).

However, there are problems with such a view of behaviour production. If such a conceptualisation is taken to be a specific model of behaviour it fails to account for many of the processes that we would ascribe as being necessary for intelligence, e.g. abstract deliberation, and goal mediated (type II) behaviour. Yet, if the conceptualisation is taken to be a more general framework of behaviour it loses a large amount of its significance as a scientific theory. For such a theory to be of use in the understanding of intelligent agent behaviour it must specify not only the hierarchy of control, but also the computational processes within each layer. Consequently, it should be seen that the application of formal control theory to the understanding of agent behaviour is of limited use because the complexity of biological systems precludes such complete model specification (Shing 1994).

APPENDIX B

CLASSICAL PLANNING TECHNIQUES

An analysis of the planning problem reveals that there are two main influences that impinge upon the efficiency with which a plan can be produced by a planner:

i. Search space size and complexity.

ii. Search Efficiency.

Given the above considerations, there is an obvious need for methods by which the size of the search space can be pruned, e.g. constraint posting, and search itself guided, e.g. via the use of heuristics, and other evaluation functions. Consequently, a number of techniques have been developed to alleviate these problems.

1 Sub-Goals

The division of a goal into sub-goals often results in an analogous division of the search required for each of those sub-goals. Since the total complexity of the overall problem is a sum, rather than a product, of those multiple sub-goal searches (Korf 1987), it can be seen that sub-goaling can provide an effective method for reducing potentially exponential search to one which is at worst polynomial. However, the degree that this use of sub-goals has in reducing the required search is dependent upon the maximum distance between any two successive sub-goals, i.e. the largest number of actions that are required to move the system from one state to the next (see point iii at the beginning of this chapter). This is analogous to crossing a stream via stepping stones - the measure that determines the difficulty of the problem is not the width of the stream (the optimal solution length), but the largest distance between any two successive stones (the maximum sub-goal distance). It should be noted that while sub-goals generally reduce the degree of the search required by reducing the search space size, they do so at the expense of producing plans which are typically longer than the optimal solution length (Korf 1987). However, as will be shown in Appendix B Sections 2 & 4, sub-goals can also reduce the complexity of search, thereby further increasing planning efficiency.

2 Independence

A we have seen, the progression of search through the problem space should be viewed as the result of the application of system operators at each intermediate state within any developing plan. The applicable operators for each of these intermediate states will form a subset of those available to the system overall, and if the applicable operators within each subset affect only the distance to the next state, then it is said that the states are independent. When sub-goals are independent it can be seen that sub-goaling allows the problem solver to focus in on only a subset of possible operators at any one time. This not only brings about a reduction in the branching factor (the number of possible choices that can be made at each intermediate goal state), but also as a result of this the complexity factor for the overall problem solution (Korf 1987), i.e. sub-goal independence leads to an increase in search efficiency (this is the main reason partial order planning is more efficient than means-ends planning - see Appendix B Section 5).

An example of the efficiency imparted by having goals which are independent can be seen in human day-to-day life. It is unlikely that we would be able to service the multitude of goals that require satisfying throughout the course of a day, if these goals were not relatively independent of each other. However, a number of global constraints apply to limit the effective independence of such goals, e.g. time and money. In general though, such global constraints only reduce independence when the limits that they impose are neared. However, where independence is lost, and problematic interactions either occur, or are likely to occur, the interactions should effect the manner in which the system creates plans prior to actually carrying out any planning and then having to waste time correcting incorrect plans (e.g. Hayes 1987, Pryor 1991; Collins et al. 1991; Krulwich 1991; Birnbaum et al. 1991).

3 Serialisation

However, not all sub-goals are, or can be made to be (e.g. via partial order planning - see Appendix B Section 5), independent, and for sub-goals that are not independent a reduction in problem solving complexity may still be possible if they can be made serializable. Korf (1987) defines serialisability as the phenomenon where by the satisfaction of each sub-goal within an ordered series of sub-goals does not violate any of those previously solved. This definition has recently been extended by Barrett & Weld (1994) with the introduction of two new terms (Fig. B.1):

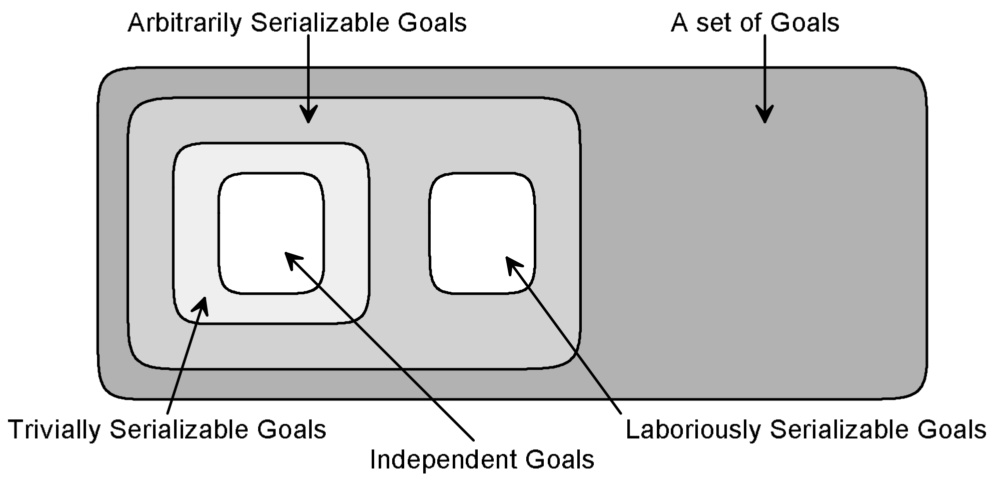
i. If each sub-goal in a series can be solved sequentially in any order without violating past progress then that series is *trivially serializable*.

ii. If there exists an inadequate percentage of orders in which a series of sub-goals may be solved without ever violating progress then that series is *laboriously serialisable*.

When making sub-goals serializable certain operators at certain points may not be applicable due to the need to protect previous sub-goal solutions. This tends to result in an increase of the overall solution length. Fortunately, the reduction in the applicable operators also results in a reduction in the branching factor at each point. Thus, for example, if the sub-goal of solving a 2 x 2 x 2 corner cube of a Rubik's cube is achieved the overall solution length may increase by up to 25%, while the branching factor is reduced by 50% (Korf 1987).

Unfortunately it is often the case that given a sequence of sub-goals, it is only possible to progress towards the main goal, by violating previous sub-goals, regardless of their order within such a sequence. Such sub-goals are termed non-serializable. Examples of this phenomena abound, e.g. Sussman's Anomaly within the Blocks world (Sussman 1975), and local minima in hill climbing systems (e.g. Thornton 1992). The degree to which non-serializable sub-goals aid the problem solver in moving towards the final goal state is less than that for those which can be termed serializable or independent. However, the solutions to such sub-goals still tend to produce resultant states which are closer to the final goal state than just purely arbitrary sub-goals. It is also often possible to group sub-sequences from a sequence of non-serializable sub-goals into a single sub-goal, making each such sub-sequence serializable. This is termed block-serialization.

Fig. B.1 - Categorizing Goals



From Barret & Weld (1994)

4 Abstraction Hierarchies

Closely linked with the utilisation of sub-goals by a planner, is that of abstraction hierarchies, as first used within the planning version (1972) of GPS (Newell & Simon 1963). The creation of an abstraction hierarchy allows a planner to improve the efficiency with which it solves complex problems by allowing it to focus in on the more essential features of those problems, i.e. the higher level goals. As a result of not constraining a system to make arbitrary choices about lower level goals, a proportion of backtracking is thereby removed, and lower level goals can be 'filled in' as and when future decisions specify both the need for, and the information applying to, such goals (e.g. Sacerdoti 1977; Dean et al. 1988; Fehling et al. 1989). This idea has recently been extended by Pryor with her use of explicit information gathering steps within a semi-formed plan (Pryor 1993a & 1993b).

Korf (1987) has shown that generally a deep abstraction hierarchy will reduce the search more than a shallow one. However, for abstraction hierarchies to improve search efficiency at all, the problem solver should have available to it the operators that apply to move the system between any such specified abstract states. If these abstract operators must be found by search among all the possible operators in the original solution space then no efficiency is realised (See also Appendix B Section 7 & 8).

5 Partial-Order Planning

Prior to the work carried out on NOAH (Sacerdoti 1977), the process of planning had been viewed with the light that Newell & Simon had 'turned on' (1963) - *means ends analysis* - and this occurred within a search space of possible world states. The result of such search, even part way through the planning process, was always a totally ordered plan, one where each step followed another in a precise fashion. However, Sacerdoti made two major advances into planning efficiency.

The first was the introduction of the notion that search should be carried out through a space of plan states rather than possible world states, and this shift meant that the structure of the search space could be determined both by the planner as well as the domain. Such a shift resulting in partial-order planning being exponentially faster than other forms of search because it gives rise to search spaces where goals are generally independent or 'trivially' serializable (Barrett & Weld 1994).

The second advance Sacerdoti made was the eloquent demonstration that during plan construction if the majority of sub-goals within plans were viewed as effectively independent, then a partial ordering was not only possible, but computationally more efficient. By not forcing a system to decide upon the ordering of any sub-goals, both until necessitated within the 'growing' plan, and until the information necessary was available, it removed from the planning process some of the assumptions that had previously plagued such sub-goal ordering decisions. Decisions concerning the order of sub-goals could be made in the 'light' of relevant information rather than being in the dark. The result; the degree of backtracking that a system was required to make was reduced, with a commensurate gain in planning efficiency.

6 Constraint Posting

This can be defined as the creation of a plan through the problem space, from a starting state to a final goal satisfying state, via the progressive application of constraints, such that at each iteration areas of the search space are removed from consideration, i.e.

INCOMPLETE PLANS + CONSTRAINTS ~> COMPLETE PLAN

In keeping with this, during the planning process the system maintains, at all times, an incomplete specification of a plan aimed at solving the problem in question. Plan completion, i.e. the production of a totally ordered finite number of steps, is seen to occur in a myriad of different ways depending upon the constraints that the planner applies to it. Thus, plans within a constraint posting planner represent a class of complete plans. A good example of this is Chapman's TWEAK system (1985 & 1987), a non-linear constraint posting planner, planning being achieved via the progressive removal of chunks of the search space until all remaining plan alternatives are viable. As such, it can be seen to have an advantage over other planners, e.g. those using means-ends analysis, in that low level details need not be searched for until a reasoned decision can be made about them. This reduction in arbitrary choice often resulting in a commensurate reduction in the amount of backtracking a system needs to carryout while planning. Where TWEAK differs from other planners of its genre is in its use of the *Modal Truth Criterion* to determine which constraints to apply to complete any given plan, and this is defined as follows:

i. A proposition p is necessarily true in a situation s if two conditions hold:

- **a.** There is a situation t necessarily equal or previous to s in time in which p is necessarily asserted.

- **b.** For every step c possible before s and every proposition q possibly codesignating with p which c possibly denies, there is a step w necessarily between c and s which asserts r, a proposition such that r and p codesignate whenever p and q codesignate.

Thus plans within TWEAK could be incomplete for two reasons:

i. The time ordering for the steps may be incompletely specified.

ii. The steps themselves may be incompletely specified.

The actual process of planning within TWEAK is non-deterministic, the system choosing an unachieved goal and then adding either temporal or codesignation constraints in accordance with this Modal Truth Criterion, such that either:

i. The planning task is successful.

ii. The planning task is not successful.

iii. The planning task never ends, the plan growing larger and larger without converging on a solution.

This method of planning has been proven correct and complete (Chapman 1985), although it suffers from a number of problems:

i. A similar method was propounded by McAllester & Rosenblitt (1991) called SNLP. While this used Chapman's ideas concerning co-designation constraints, it did away with the use of the Modal Truth Criterion. In this sense, SNLP casts a shadow as to whether the Modal Truth Criterion was in fact of any great importance within TWEAK as a non-linear constraint posting planner.

ii. There is no notion of time, and the search process not being time bounded may take a polynomial amount of time.

iii. The Modal Truth Criterion cannot cope with situation dependent actions.

iv. There is no learning within TWEAK.

7 Macro Operators (Korf 1987)

When a non-serializable sub-goal does not represent progress towards the main goal its achievement may still be beneficial to a problem solver, due to the problem solver's knowledge concerning a sequence of primitive operators (called macro-operators) that tell it how to get to the next sub-goal from that state. Of particular use in the satisfaction of non-serializable sub-goals are those macro-operators that leave previously satisfied sub-goals intact, even if they temporarily violate them. The use of a macro network can have a large computational advantage when problem solving for arbitrary pairs of initial and goal states. In this case problem solving can be viewed as a three stage process:

i. Finding a path from the initial state on to an arbitrary point on the macro network.

ii. Doing the same for the goal state.

iii. Finding a path between these two specified points within the network.

Earlier evidence that macro's do give rise to planning efficiency gains comes in the form of an MORRIS (Minton 1985), a heuristic problem solver that selectively learns two types of macro's: *S-Macro's* and *T-Macro's*. The former represent common sequences between new problem solutions and previously acquired problem solutions, hence they are relatively short, reducing applicability costs, and relatively few in number, reducing indexing (search) costs. The latter, T-Macro's, result from the system's use of a heuristic evaluation function (H_{diff}), that checks whether macro's make any progress towards the current goal(s). However, such a heuristic function can give rise to hill climbing problems and these are what the formation of T-Macro's are designed to help with, i.e. they are macro's that represent non-obvious solutions. Hence, MORRIS is important because it shows that although S-Macro's are used more frequently than T-Macro's, it is the latter which provide the greatest gains in search efficiency.

However, Minton (1985) has also shown that although macro-operators can potentially give rise to increased planning performance (by compressing the search space associated with a particular domain), this is not guaranteed. Indeed, the utilisation of such macro's can have the opposite effect, slowing the system down. Consequently, macro-operators and other types of plan 'chunks' should only be used when they offer a clear increase in planning efficiency.

8 Case-Based Planning

The Case-Based planning paradigm is one where new plans are devised via the retrieval and adaptation of old plans, e.g. SOAR (Laird et al. 1987 & 1991), CHEF (Birnbaum et al. 1991 - see Appendix D Section 1); RUNNER & TRUCKER (Hammond 19xx). However, due to the fact that such adaptation makes use of the same type of complex reasoning as planning does, potentially the adaptation of a retrieved plan may require search which is as expensive as planning from scratch. Obviously, a Case-Based planner should be more efficient than planning from scratch, and so constraints are placed upon the retrieval process. Planning efficiency is crucially dependent upon the plan retrieved from memory being the plan that requires the least adaptation to fit the current circumstances. Furthermore, it has been shown (see Appendix B Section 7 above) that within even simple domains the explosion of stored plan chunks (or macro-operators) may result in search through the space of possible plans becoming as expensive as search through the original problem space. Thus, to avoid 'swamping' finding the right plan in a cost effective manner is the central issue in Case-Based planning, where a trade-off exists between plan retrieval and finding the most easily adapted plan, i.e. in some cases the time required to obtain the most easily adapted plan may be so large that it outweighs any additional cost incurred in adapting a less appropriate, but more easily retrieved plan, or building a new plan from scratch.

APPENDIX C

REACTIVE PLANNING TECHNIQUES

1 Universal Planning

Put forward by Schoppers (1987), in contradistinction to traditional Classical Planning techniques the quintessence of Universal Planning is that a system in pursuit of any given goal can determine what to do next by locating its current situation within a large table: the Universal Plan. From such a table an executable primitive will always be prescribed which furthers the system's pursuit of its goal(s) in relation to its current situation. As such Universal Planning is the apotheosis of Reactive Planning.

In effect, the Universal Plan forms a direct functional mapping between perceptual input and motor output, removing the need for many of the deliberative, but highly time consuming processes that other planning paradigms require, especially the problems associated with building complete domain representations (see Section 6.1). This gives Universal planning an advantage over more traditional Classical Planning techniques in that they allow a system to react quickly to environmental events, but at a cost.

Whilst Universal Planning may work in simple domains, it is infeasible (in the practical sense) that a system's designer would be capable of explicitly enumerating and mapping all of the inputs possible for anything like a real-world domain. Yet, even if this were feasible, the size of the resultant table would pose severe indexing problems; the search required for the system to map itself onto the table could take longer than planning from scratch in some situations (cf. Minton 1985 - see Appendix B Section 7 & 8). Indeed, Ginsberg (Ginsberg 1989) argues that the size of a Universal Plan would grow exponentially with the complexity of the intended domain.

Ignoring the fact that the creation of an exact Universal Plan for a real-world domain would be a perversely difficult task of Herculean proportions, Universal Plans still suffer from two onerous disabilities:

i. They are precompiled.

ii. They invariably suggest primitive actions.

Having the input-output mappings precompiled gives rise to the fact that whilst the majority of the combinatorial problems that would, and do, arise from ascertaining what to do next, and then how to do it, have been removed from the system, they have unfortunately been shifted onto the designer; i.e. there is no actual planning by the system. Thus, in its purest form Universal Planning is a prime example of AI researchers producing misnomeric terms for their command and control paradigms. Furthermore, enhancements to performance are limited; there is no way the system can change the Universal Plan after its instantiation in the light of past performance(s). In effect the system, by design, is unable to learn.

Ironically, the inflexibility brought about by having plans that always prescribe an executable system primitive, is also the fundamental corner-stone of Universal Planning. The Universal Plan in its purest form can never suggest that 'some time should be spent thinking about what to do'. In effect the optimality of the system is reliant upon, and fixed, by the correctness of the precompiled plan. There is never any way that enhancements to performance can be gained with greater expenditures in computational resources traded against a reduction in reactivity; a trade-off which is usually one of the defining characteristics of what is commonly termed 'cognition'.

While it may be that for non-cognitive activities, such as locomotion, a lack of flexibility to enhance performance could be acceptable (indeed for robot systems such an improvement may not be possible without changing whatever effectors the system has access to), the same cannot be said for cognitive tasks. Here one would expect both an improvement in performance over time (learning), and the ability for the system to dynamically alter the balance between its reactivity to environmental events, the time (and resources) it expends in deliberation, and the optimality of its behaviour. Such a loss can only be viewed as overly restrictive for any agents intended to be embedded within real-world domains.

Such short comings have, however, not stopped 'watered down' versions of the Universal Planning paradigm from deluding some researchers into thinking that they have postulated better solutions for situated activity than previous attempts have done. One such example would be the use of Approximate Universal Plans in the guise of Routines and a Deictic Representation by Agre & Chapman in their Pengi system (Agre & Chapman 1987 - see Appendix E Section 1). The term Approximate Universal Plan is used to imply that while it is clear that attempting to use complete Universal Plans in anything but a simple domain is unrealistic, it is reasonable to concede that approximations to complete Universal Plans will be available. In theory, such approximations would provide many of the benefits of more exact plans with only marginal performance degradations in terms of occasionally suggested inappropriate, or sub-optimal, actions (Agre & Chapman 1987). Furthermore, such degradations in performance could be countered if Approximate Universal Plans were allowed to specify that one of their 'actions' be to plan when unanticipated situations were encountered, e.g. Nilsson's Action Nets (Nilsson 1988).

While it is clear that such an approach (Approximate Universal Plans + Explicit Planning) would be a way around some of the problems confronting the use of Universal Planning in situated activity, it is at the selling out of the original idea that only primitive actions be specified. The use of a planner in certain situations quite clearly re-invokes the problems associated with planning per se, yet crucially it does so only when required to do so by the unanticipated situation at hand. Hence, the paradigm becomes one of look, do, and think if you need to. Where the Universal Plan becomes superfluous when it becomes nothing more than a repository for storing precompiled plans that the system has access to at run time (Ginsberg 1989). However, it could conceivably be argued (e.g. Schoppers 1989) that in fact the use of cognitive actions merely extends both the number of possible actions and sensors available to the system, and thus represents a broadening of the Universal Planning paradigm rather than a selling out. But this still does not address the fact that in such a case Universal Planning is still reduced to nothing more than a mechanism for caching precompiled plans, or that such plans cannot be altered by learning. The introduction of such learning mechanisms, as well as the allowance of non-primitive actions would effectively complete the transition of the paradigm full circle back into classical planning techniques.

As expounded above, Universal Planning as a paradigm for situated activity suffers from a number of problems. Indeed, even the use of Approximate Universal Plans would give rise to these problems along with a degradation in performance optimality unless explicit planning was allowed for. In such a situation Universal Planning is reduced to a paradigm for caching plans without the required flexibility that would allow learning as well. It seems unlikely in the light of this that Universal Planning as a paradigm of situated activity within real-world environments will succeed where others have failed.

2 RAPs

In clear distinction to the Universal Planning method of transcribing perception into action, Firby (1987) takes a more traditional approach to situated activity. Goals, plans and actions are embodied within entities termed Reactive Action Packages (RAPs), independent resource competing units, that pursue goals without recourse to any potentially time-unbounded prediction, in accordance with three principles:

i. All decisions concerning what action to do next are based solely on the current state of the environment.

ii. The successful termination of a RAP guarantees that the goal in question has been achieved and all sensory tasks needed to note this have been carried out.

iii. The unsuccessful termination of a RAP guarantees that there is no possible way of achieving the goal in question.

To reduce the search required when attempting to satisfy a goal, each RAP contains a set of pre-defined plans (task nets) by which the goal that it embodies may be achieved. These task nets are chosen between when the RAP is executed in accordance with the current environmental model (point i above). If no task net can be applied the RAP has effectively guaranteed that there is no way to achieve the goal (point iii above). Finally, in keeping with point (ii) above, some task nets include specific sensory tasks. Thus, the RAP control algorithm, designed to address the two key points of execution monitoring and low level replanning, is achieved in a uniform way within a control system that consists of three structures, a hardware interface, an execution queue, and an interpreter. These are used by the control algorithm in the following cycle:

i. The interpreter selects an item from the execution queue on the basis of approaching deadlines and ordering constraints specified by the various 'active' task nets.

ii. If the chosen item is an action then it is passed to the hardware interface for execution. If, on the other hand, it is a goal the appropriate RAP is executed where this takes the following form:

  a. The world model is consulted to check whether the RAPs success test is satisfied. If it isn't the world model is used to choose a task net from those that are associated with the RAP.

  b. The commands within the task net are all placed into the execution queue.

  c. The RAP itself is suspended until the chosen task net terminates.

iii. After action or RAP execution, feedback is acquired - directly from the hardware interface in the case of actions, or indirectly in the case of an activated sensory task.

iv. Finally, on the basis of feedback changes to the world model are made.

Firby notes that there are three main problems with RAPs as a means of producing situated activity:

i. A task net might fail without changing the world enough for it not to be chosen by the interpreter on the next cycle. Such an occurrence could feasibly lead to 'futile looping' behaviour.

ii. There is no way to prevent plans for conjunctive goals clobbering each other.

iii. By design RAPs lacks any means of look-ahead and strategic planning.

Furthermore, analogous to the problems suffered by Universal Planning, because task nets are pre-compiled into RAP structures there may be difficulty both with elucidating all the required task nets for a specific domain, but more importantly, with the acquisition of new task nets, the system having to be fully re-compiled at present. In conjunction with this last point, not only must the system be re-compiled to acquire new task nets, but task net selection does not take account of past experience. Hence, by design there cannot be any learning within RAPs, a major failing in any situated agent.

APPENDIX D

INTEGRATED PLANNING TECHNIQUES

1 CHEF & CASTLE

CHEF (Birnbaum et al. 1991) is a case-based planning system that devises plans to cook meals, where such plans must take into account the effects foods have on each other when cooked together. This is achieved via the use of Case-Based planning, where new plans are devised via the retrieval and adaptation of old plans, to cook meals. However, potentially the adaptation a retrieved plan requires may be as expensive as planning from scratch due to the fact that such adaptation makes use of the same type of complex reasoning as planning does (Minton 1985). Obviously, a Case-Based planner should be more efficient than planning from scratch, and so constraints are placed upon the retrieval process. Planning efficiency is crucially dependent upon the plan retrieved from memory being the plan that requires the least adaptation to fit the current circumstances. The trade-off in finding the right plan in a cost effective manner is the central issue in Case-Based planning. My main criticism of the CHEF system is that it is embedded within a non-dynamic, non-complex kitchen domain. In fact, CHEF is purely a classical planner, without the need for any reactivity, unconstrained in its deliberation time, and without the need to handle the problems associated with unpredictable domains.

Converse to this is CASTLE (Collins 1991a, 1991b & 1992; Krulwich 1991a & 1991b; Krulwich et al. 19xx), a system embedded within a chess domain, and consisting of the following set of functionally defined 'execution' components, each dedicated to a particular cognitive task and implemented as a set of rules:

i. Threat detector.

This monitors the environment for situations that match known threats as specified within a rule base. When a rule is triggered the threat description associated with that rule is passed on to the plan selection component. To reduce cognitive load, threat detection rules are evaluated at intervals rather than continuously, and focusing rules are used to delimit the domain areas over which these detection rules are applied.

ii. Detection focusing component.

Uses focus rules to delimit the set of bindings over which detection rules are evaluated, i.e. delimit the Threat Detector to check only the areas of the domain in which new threats may have been enabled. This prevents rechecking of those areas where no events have recently occurred.

iii. Plan selection component.

Attempts to formulate a response to any threats that are posted to it by the Threat Detector. Presumably this involves the creation of a goal.

iv. Schema applier component.

Retrieves schemata (generalised sequences of actions) that will achieve a particular goal in a particular situation. In CASTLE these schemata are called 'offensive strategies'. Since no indexing system is mentioned, it is not obvious (a) how the retrieval of such schemata occurs, and (b) how the system avoids costly search through the database of applicable schemata. Furthermore, although within the CHEF system it is specified how such applicability is judged, this is not the case within the CASTLE system, and one can only presume that the roles of indexing, schemata retrieval, and schemata adaptation are carried out in a similar manner to CHEF.

Once a schema is chosen it is assigned a deadline, along with a priority, and placed onto a queue. On each cycle of the system this queue is checked and the highest priority schema is chosen for execution. Thus, a schema will only be executed if there exists a time prior to its deadline when it is the highest priority schema on the queue.

v. Offensive strategy component.

Encodes the actions that go to make up an offensive strategy (within the Chess domain), along with the conditions of applicability and the goal that the strategy is to satisfy.

vi. Learning component.

Learning within this system is failure driven, and involves finding the right control parameters to manipulate in response to any such perceived plan failures (cf. blame assignment problem). In many cases the parameters needing to be altered are in the form of rules within various rule bases rather than just simple numerical quantities. Learning in this approach is thus a matter of (a) noticing a planning failure, (b) identifying the control parameter of the planner that is most relevant to the cause of the failure, and (c) adjusting that parameter. It is here that Model-Based Reasoning is used to provide a unified approach to these three tasks.

vii. Plan execution component.

An agent's plans will often be carried out under conditions that it cannot generally anticipate, and as such much of the responsibility for determining the particular actions that the agent will carry out must lie in this component rather than resting exclusively with the plans themselves. However, it is implied within the Offensive Strategy Component section above, that the plans that this component retrieves are already preperformed down to the level of actions. Thus, it is (a) unclear whether this component actually does carry out any form of 're-planning', and (b) points towards the system being more classical than reactive.

There are a number of criticisms that can be levelled at the CASTLE system. Firstly, the domain. The trouble with the chess domain is that it is only dynamic in a very structured manner, and hence there is a lack of true reactivity within the system. As such, the system lacks both the ability, and need, to act asynchronously with respect to the environment, and asynchronously internally (i.e. between its planning and execution components). Furthermore, the agent has full knowledge concerning the location of the pieces, and knows that an intended action will always be executed successfully. This is to say that, for example, moving a pawn diagonally onto an opponent piece will always end up with that opponents piece being taken.

This predictability and completeness of domain knowledge alleviates many of the problems associated with more unpredictable domains. Thus, while the theory aims to account for a theory of domain-independent planning the implementation falls short of fulfilling this promise by placing the architecture within a domain that is highly defined, dynamic only in a highly structured manner, and relatively uncomplex. Secondly, the myopic view taken. While the chess domain offers a planning system the perfect opportunity for long term goal directed 'strategic' play, along with shorter reactive behaviour, the CASTLE system does not appear to support any method by which such longer term strategic play can occur (unlike other Chess playing programs). Thirdly, the mechanics concerning goal generation and plan priorities are very vague, leaving a number of questions unanswered, e.g:

i. How are goals generated?

ii. How are goal or plan priorities determined?

iii. Are these priorities fixed or variable, i.e:

- Do they alter in relation to approaching deadlines?
- Do they alter in relation to domain changes?
- Do they alter with respect to other posted goals?

2 PRS

PRS consists of a database that holds known facts concerning the world represented in first order predicate calculus (its beliefs), a set of goals to be achieved (its desires), an intention structure, that holds adopted partially formed plans (its intentions), and a set of such plans (Knowledge Areas, or KAs).

In keeping with the idea that much of a system's knowledge concerning goal achievement is in the form of action sequences, information within KAs is expressed using procedural logic. This extends many of the logics already used within embedded agents, being able to specify not only a sequence of steps, from a start node to an end node, but also condition selection, non-deterministic choice, iteration, and hierarchical abstraction (Wright 1994b). Associated with each KA is also an invocation condition specifying when the KA is of potential use. Thus, KAs "express a declarative fact about the results and utility of performing certain sequences of actions under certain conditions" (Georgeff & Ingrand 1989, pp. 973).

Based upon the beliefs and goals held by the system, on each cycle of the PRS Interpreter a subset of the system's KAs will be applicable for execution. Deciding which single KA from a set of applicable KAs to post to the Intention Structure is achieved in one of two ways. By default the Interpreter embodies certain fixed decision making processes that are time bounded in that no deduction is used, selection being based upon undefined heuristics. This guarantees speed and reactivity, but as with all heuristic mechanisms will not be infallible to mistakes such as selecting a KA of lesser 'importance' in preference to one of greater 'importance'. However, as well as the KAs described above, PRS also maintains a number of meta-level KAs. These provide the interpreter with methods for more informed choices than would otherwise be possible, but at the expense of longer decision times. Because of this potential time unboundedness, along with the need to maintain Interpreter reactivity, processes specified within meta-level KAs are interruptible. Finally, in terms of triggering, meta-level KAs are invoked in the same manner as normal KAs, but usually due to internal rather than external states. For example, failure to achieve a certain goal, re-activation of a suspended intention, or multiple applicable KAs. With respect to this latter point, meta-level KAs encode information concerning various methods for deciding between multiple KAs in a non-heuristic deliberative manner. This provision of and for meta-level KAs within a system is an important point, it means that a designer can construct a relatively simple Interpreter safe in the knowledge that the system's behaviour will not always be constrained by this simplicity.

Upon selection of a single KA, it is posted onto the Intention Structure, where if it provides a sub-KA for an existing intention, it expands that intention. This occurs for each point in the path between an intention's start and end node until the sub-KAs posted specify a system executable action rather than a path (so called Primitive KAs). If the selected KA is not part of an extant intention it instigates the creation of a new one. Using this formalism, as with other planning algorithms, successful execution of a KA consists of achieving each sub-goal along the path from its start to end node. Hence, the Intention Structure may contain any number of partially achieved intentions, each consisting of a top-level KA and the sub-KAs that have been posted in its service.

This set of intentions forms a partial order within the Intention Structure, with the root intention being the one executed, and other intentions 'waiting' their turn. However, earlier intentions must be realised, or dropped, prior to later ones being executed. In this respect PRS may in fact be limiting the degree to which it can dynamically re-order the achievement of intentions, and hence falling foul of one of the requirements for intelligent autonomous behaviour (requirement 3.5.ii). A partial solution to this lies in the fact that intentions can exist in several states within the Intention Structure. They may be:

i. Active

An intention which is active will be executed when it is made the 'root' of the Intention Structure.

ii. Suspended

As determined by meta-level KAs, this is an intention that has been adopted, but without a decision concerning when to execute it.

iii. Conditionally Suspended

As determined by meta-level KAs, this is an intention which is suspended until a certain condition is met.

In the case of suspended (or conditionally suspended) intentions, by default upon re-activation they are placed at the root of the Intention Structure for immediate execution (although this can be over-ridden by certain meta-level KAs). The final step for PRSs Interpreter is the execution of the intention at the root of the Intention Structure. If this intention specifies an action then this is carried out, affecting the internal and/or external states, where changes to the internal state can operate directly on any of the beliefs, desires, or intentions of the system. Conversely, if the intention specifies a sub-goal this is posted as a new operational goal for the system.

In keeping with the BDI paradigm (see Section 5.2), when goals fail they are re-tried either until the system believes them to be achieved, or until the system no longer believes that they are readily achievable. This latter point is judged via the application of meta-level KAs, or when all the possible KAs for that goal have been attempted. This is unlike RAPs (Firby 1987) which continually re-attempts task-networks even to the point of looping behaviour. One possible problem with the manner in which PRS attempts to achieve a failed goal is that at no point are previously attained goals re-asserted, it being argued by Georgeff & Ingrand (1989) that such re-assertion is extraneous. However, in saying this Georgeff and Ingrand seem to have ignored the possibility that changes in the world caused by failed attempts may well have altered the world such that goal achievement is no longer possible by any KA, even if theoretically it was prior to those attempts.

PRS provides a formalism whereby both planning and execution are carried out in a manner which guarantees responsiveness to environmental changes, is coherent with respect to the agents goals, and can be said to be 'intentional'. The procedural representation of goal achieving knowledge provides for a richer control than other logic formalisms, and the use of meta-level KAs provides a means of enhancing the degree of deliberation carried out by the system. It is argued by Georgeff & Ingrand (1989) that such meta-level KA control can change a system from exhibiting 'cautious' to 'bold' behaviour depending on how often they are invoked.

In summary, the attraction of PRS is that while being simple, it presents a flexible architectural framework for the development of embedded agents, addressing some, but not all of the problems impinging upon the basic BDI formalism (see Section 5.2). Most notably, there are gaps in the system concerned with the generation of goals and intentions, as well as possible problems concerned both with the degree to which PRS can deliberatively re-order the achievement of goals (point iii in Section 5.2), and in the manner in which failed goals are dealt with. In addition to this however, PRS does not appear to support any learning concerning the behavioural characteristics of KAs, for example KA1 is better than KA2 in situation X.

3 GUARDIAN

The GUARDIAN system has primarily been implemented as a system for intensive care monitoring, although recent extensions have been used to study the role of 'motivation' in behaviour production (Morignot & Hayes-Roth 1994), as well as methods for evaluating integrated architectures (Hayes-Roth 1991b).

It consists of sub-systems for perception, action and cognition, able to influence each other via the passing of information along a communications interface which feeds information 'packets' into limited capacity, best-first retrieval and worst-first overflow, i/o buffers. Perception sub-systems comprise of sensors (for data collection) and pre-processors (for data abstraction, annotation and filtering). Perceptual information is placed into output buffers for transfer to cognitive or action sub-systems, and in return, instructions from the cognitive subsystem may be sent that alter pre-processing. Similarly, drivers within action sub-systems retrieve intended actions from input buffers, translates these into executable programs of effector commands, and feedback success or failure to output buffers. However, what makes GUARDIAN special is not the manner in which it co-ordinates data retrieval and action execution, but the manner in which problem solving and control reasoning are carried out within the cognitive sub-system.

Representing information in a conceptual graph formalism, the cognitive sub-system holds all of an agent's knowledge and performs all of its reasoning, both in terms of problem solving and control reasoning, within a global memory structure. Hence, this contains the results not just of reasoning operations, but also of observations, inferences, predications and plans. The most important part of this structure with respect to dynamic control reasoning is that of the Control Plan.

The Control Plan presents, within a temporally organised pattern of control decisions, the classes of operations that the agent intends to perform over some period of time, as well as the manner in which such performance is to take place. Construction is incremental via the execution of control operations suggested in response to perceptual or cognitive events. These control operations are handled in exactly the same manner as other reasoning operations, but result in the generation, or modification, of control decisions, rather than new inferences (for instance). Since different control operations can embody different reasoning methods, control decisions may be generated in response to a number of criteria. For example, perceived events (bottom up generation), a need to elaborate more general control decisions (top down generation), or in response to new goals (Hayes-Roth 1990). Lastly, the Control Plan supports the coexistence of multiple control decisions regarding a particular time interval, thus there exists the possibility that control decisions may be either complementary or competing.

GUARDIAN extends dynamic control planning as implemented within BB1 (see Section 5.4), while retaining an execution cycle (now termed a satisficing cycle) which is in principle the same:

i. The Agenda Manager

This identifies and rates executable KSs (now termed reasoning operations) which are added to the agenda, a limited capacity sub-section of the global memory. However, the time this identification task requires is a product (among other things) of the number of known reasoning operations, and thus within a time-bound agent it is unreasonable to expect the Agenda Manager to identify all executable operations. Instead it must act in "an incremental, non-exhaustive fashion, identifying and rating a subset of the executable operations at any a time" (Hayes-Roth 1990, p. 113), if the satisficing cycle time is to be bound and system responsiveness maintained. Furthermore, since all active control decisions whose time intervals include the current time, are used by the Agenda Manager to rate executable operations, it can be seen that the Agenda Manager in conjunction with the Control Plan provides the agent with a means of following strategic plans to which it has committed. In other words the Agenda Manager must be seen as pivotal for the effective operation of GUARDIAN.

ii. The Scheduler

Determines which of the identified reasoning operations held within the agenda to execute, and in what order, based on their ratings.

iii. The Executor

Selects the 'next operation' as specified by the Scheduler, instantiating and then executing the program defined for that operation type, and producing associated changes in a limited capacity section of the global memory called the Cognitive Buffer. These changes can be a number of types:

a. Inferences for extant reasoning tasks.

b. Perceptual pre-processing instructions or intended actions for output buffers.

c. Importantly, using control parameters the control plan is able to modulate information-focusing, resource-filtering, and load-balancing within sensory sub-systems, thereby effecting both the type and rate of information entering cognition in a manner which is dynamic, and in accordance with both the control plan and available resources.

d. Changes in the control plan, i.e. modifications of the Control Plan occur such that an agent may construct strategic plans that are appropriate to an evolving task environment, or change those plans as is appropriate.

APPENDIX E

NOUVELLE AI SYSTEMS

1 PENGI

Pengi (Agre & Chapman 1987) plays the commercial video game Pengo, which consists of a 2-d maze constructed from blocks of ice, a penguin, and a number of bees. The penguin, controlled by the Pengi System, must crush the bees by pushing the ice blocks into them, while the bees can kill the penguin by stinging it. Hence, the domain shares many of the essential characteristics of a real-world domain; it is complex, uncertain, dynamic, and operates in real-time[c]. In antithesis to the majority of more traditional situated activity command and control systems Agre & Chapman eschew the use of explicit representational schemes, taking the stance that the world is its own best model and that activity is first and foremost a function of how the world is now. As such, this is an ideological position which accords closely with that taken by Brooks 1986, Maes 1989, and Dennett 1994 (among others). Explicit representation is viewed as more of a hindrance than a necessity and is thus avoided. But this is at the expense of, among other things, system scalability (as can be seen within Section 6.2).

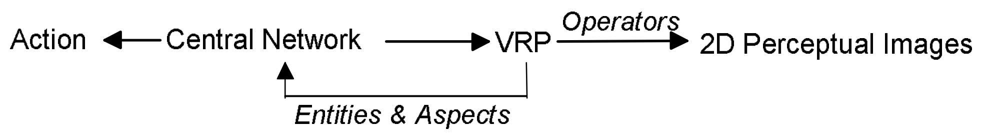
Clearly contrasting with more traditional representational schemes where individual entities are represented with their own identifier, e.g. (at block-213 417 654), Agre & Chapman propose that entities are represented using a description which is dependent upon the system's immediate situation; so called *Indexical-Functional Entities*, e.g. the-block-I'm-pushing. Representing the world in this way reduces the combinatorial problems associated with building and manipulating complete domain representations, where such representations may require many hundreds or thousands of entity descriptors. Furthermore, Indexical-Functional-Entities registered by routines within Pengi's visual system Visual Routine Processor (VRP), in conjunction with Pengi's current goals, give rise to *Indexical-Functional Aspects* where these take the form, e.g. the-block-I'm-going-to-kick-at-the-bee-is-behind-me- (so I have to backtrack).

---

[c] The term *real-time* is used to denote that the time course of events within a simulation is equivalent to the time course of analogous events within the real-world.

The outputs from the VRP form the inputs to a central combinatorial network system which carries out the deliberative task of deciding what actions are appropriate given Pengi's current circumstances. This appears to be carried out in a manner which is extremely similar to Maes's Spreading Activation command and control System (Maes 1989), although elucidating the actual mechanisms involved from Agre & Chapman's paper (1987) is not possible. Nodes within the network register particular aspects, and the network's outputs go to the motor control systems. Hence, as the domain changes and new aspects are propagated through the network different behavioural actions result. Furthermore, the network feeds back to the VRP, and as such the aspects are always dependent upon the system's current goals, and activities (Fig. E.1).

Fig. E.1 - Pengi's Internals



It is claimed that resultant from the utilisation of domain regularities via simple heuristic mechanisms, the above system can give rise to patterns of activity between the system and its environment. These patterns of activity, or behavioural routines, are not typically represented within the system, and a system so engaged does do not typically have any preconceived notion of what will happen. Indeed Agre & Chapman write:

> "..we believe that activity mostly derives from very simple sorts of machinery interacting with the immediate situation. This machinery exploits regularities in its interaction with the world to engage in complex apparently planful activity without requiring explicit models of the world."
>
> *Agre & Chapman 1987, p.268*

Since routines are not plans, but simply the result of the system in conjunction with its environment, a system engaged in executing a routine is free both to capitalise upon unexpected opportunities, or deal with unanticipated contigencies. This is unlike more traditional classical planning techniques, where the execution of a plan proves to be inflexible in the face of such unanticipated environmental changes (however, it should be noted that such behaviour could be handled by any conventional rule-based program, or a planner that allowed condition expressions). Thus, when circumstances dictate that a new routine is applicable the system does not need to register this change as an unexpected event, since no future events were predicted in the first place. An example of this would be the routine of running behaviour resultant from the rule 'run from a bee', in conjunction with a situation where a bee is present and chasing the penguin. If the bee stops chasing the penguin then the penguin need not continue to run, conversely if a block is in the way the penguin can kick it out of the way.

Thus, routines can be said to be both opportunistic and robust in the face of uncertainty. Yet, this is misleading, not least because (as Agre & Chapman are keen to point out) routines are epiphenomena - they arise from the conjunction of the system and its environment. But even ignoring this shows that it is not the routines that are opportunistic or robust, but the fact that any particular routine is engaged for only so long as the environment dictates so. If a system failed to notice changes in the environment necessitating their termination, or failed to terminate them, it would suffer from the same problems a more traditional planning system, e.g. inappropriate behaviour, cf. the Greylag Goose egg rolling phenomenon[d].

Hence, in a manner which is clearly reminiscent of Universal Planning, sensory data is transformed directly into motor output without recourse to high level cognitive processes such as planning, and thus without any need for explicit representations. However, one crucial difference between this approach and that of Universal Planning is that unlike Universal Planning, where each input state is dealt with in isolation and without consideration of any other states, Chapman (1989) argues that a maintenance of state within the system is both desirable in terms of coherence, and in terms of computational feasibility[e]. He goes on to provide proof of this in the form of research carried out by Tsotos (1989 - from Chapman 1989), where it is argued that such a maintenance of state can simplify the problems associated with perception

---

[d] A well documented phenomenon in which the Greylag Goose, while sitting in its nest, rolls eggs back into the nest with its beak when they fall out. This behaviour is caused via the triggering of a fixed action pattern (FAP), a sequence of behaviours which must run to completion even when inappropriate. Consequently, the goose will continue to execute the beak actions to roll an egg even when the egg in question has been removed after FAP onset.

[e] Such a maintenance is state is supported with all the Nouvelle AI systems, from Brooks (Appendix E Section 2), to Kaelbling (Appendix E Section 3), to Maes (Appendix E Section 4).

from exponential to linear solutions. It can be seen that this maintenance of state within their Pengi architecture is achieved via a feedback loop from the central cognitive system to the visual system. Furthermore, the maintenance of states with causative properties means that, unlike Universal Planning (and the majority of other reactive planning systems, e.g. Firby's RAPs system - Appendix C Section 2), identical environmental events will not give rise to identical system actions. However, it is my view that such a use of state in either system (central or visual) constitutes a form of representation and that this weakens Agre & Chapman's arguments against the need for explicit representations rather than strengthening it. A change in form does not provide a formal basis for a denial of existence (e.g. Sloman 1992d & 1994c). Furthermore, the system must have explicit representations of what a bee is in the first place for it to be able to know both how to recognise one, and how it should then react, cf. Plato's forms.

This aside, although Agre & Chapman allow previous states to effect the behavioural output of the system, this maintenance of state within the system is achieved via feedback between the central network and the VRP, not from any form of explicit representation of that state. As such, the system is unable to learn from past performance failures, or successes, and in this respect, is therefore essentially no different from many more traditional reactive planning systems. Furthermore, in an attempt to reduce the computational problems associated with building, maintaining, and reasoning about an explicit domain representation, Agre & Chapman have reduced the system to the use of representations that only gain significance in terms of the system's situation at any one time, i.e. representations whose full semantic significance are not explicit in the representation. This leads the paradigm into a number of problems, as outlined within Section 6.2.

I do not believe that the approach to situated activity suggested by Agre & Chapman offers significant advantages to that of Universal Planning, RAPs, or any other reactive planning system (see Chapter 4 & Appendix C). As with the majority of these it cannot learn from its experiences, and lacks the flexibility to reason about events and entities within the domain. Thus, while this approach to situated activity command and control may be sufficient for the domain chosen, it is by no means the only viable approach to situated activity command and control as Agre & Chapman seem to allude. Indeed, this partisan attitude is a failing which also afflicts followers of Behaviour Based approach to situated activity command and control, e.g. Brooks and Maes.

2 Subsumption Architectures

One of the key proponents of Behaviour-Based AI is Rodney Brooks, to whom the initiation of the approach can be ascribed[f]. Indeed, up until Brooks formulated his Subsumption Architectures (1986), AI was firmly engrossed with the traditional paradigm of achieving autonomous behaviour, an approach which seemed at the time to have sunk into a quagmire of mediocrity. The Subsumption architecture was, however, a radical departure from this, advocating not least that a design should be founded on the desired manifest behaviours of the agent, should operate without the need for explicit representations, and should be physically instantiated.

In general, within Behaviour-Based approaches each behavioural sub-system (termed a *competence level* by Brooks) is viewed as being independent, with its own input from perception, its own processing capabilities, and its own output to effectors. Indeed, Brooks outlines eight levels of competence (levels of control if you like) that a physically instantiated intelligent autonomous agent should posses, and be designed in accordance with:

i. Avoid contact with objects.

ii. Wander aimlessly.

iii. Explore the world by sensing and moving to distant places.

iv. Build a map and plan routes using it.

v. Notice changes in the static environment.

---

[f] Although, it could be argued that originally the concept that intelligent autonomous behaviour is a product of being situated within a domain, and cannot be abstracted from that domain without losing meaning, was that of Heidegger.

vi. Reason about the world in terms of identifiable objects and perform tasks related to certain objects.

vii. Formulate and execute plans.

viii. Reason about the behaviour of objects and modify plans accordingly.

However, where Brooks' Subsumption Architectures differ from other Behaviour-Based approaches, e.g. Maes and Kaelbling, is the manner in which these various behaviour producing layers are implemented and co-ordinated. The principal ideas propounded by Brooks being that:

i. Each layer should exist as a real physical entity, not just a simulation (Brooks 1991a & 1991c).

ii. Each layer should be connected to its relevant input and output areas via real wires (Brooks 1991a).

iii. Each layer should not have any direct access to the internal workings of any other layer (Brooks 1986).

iv. In addition to each layer's perceptual input and effector output, layers may also be connected to each other. The idea being that some layers may inhibit[g], or be inhibited by, other layers (Brooks 1986).

While it should be noted that point three above is of crucial importance in behaviour co-ordination for Brooks, let me just spend a few moments arguing against points one and two. Brooks is incorrect is stating that a physically instantiated agent would experience computational problems which a simulated agent wouldn't (Brooks 1991a). If the simulated environment is made such that it shares the same essential characteristics as the intended physical environment then that simulation will also provide the agent with the same computational problems that the real environment would. Furthermore, simulation can be used to provide the agent either with problems that it is unlikely to encounter within the real world, or problems

[g] The term inhibition is used here, but Brooks not only talks about layers inhibiting each other, but also suppressing each other, but the distinction is a minor one.

which cannot safely be experienced within a real world environment. For example, power station control systems, autonomous anti-aircraft systems, to name but a few. Using simulation for the processes of prototype testing and refinement should therefore be seen as advantageous over physical instantiation, not least in terms of time, cost and safety.

But let us move on to the point of note concerning Brooks' approach to Behaviour-Based AI, that of co-ordination of behaviours. The idea that certain layers may inhibit others via the use of inhibition-suppression forms an effective hierarchy of control, from low level behavioural layers to those higher up, and this *subsumption* of a lower level behaviour into a higher level synthesis is the manner in which co-ordination is achieved between layers within the Subsumption paradigm. Let me give a simple example:

Let us make a robot's behaviour of avoiding objects its lowest level behaviour. Thus, when something comes close to it moves away slightly (like a magnet near another magnet of equal polarity) but otherwise the robot will remain stationary. The next layer up, let us say a movement layer, would allow the robot to move aimlessly around, but crucially the behaviour of this layer and the lower level layer would be synthesised such that the robot could move while avoiding obstacles. This process, it is argued by Brooks, can continue further and further, building more and more complex behaviours until truly intelligent autonomous behaviour results. Indeed, as evidence that such an approach is viable for the creation of intelligent autonomous agents Brooks calls to mind the manner in which evolution has 'appeared' to operate; first designing methods of movement, then refining those methods, and adding high layers of control (Brooks 1991a & 1991c).

Yet this is a somewhat shallow proof. One has only to look at the countless millions of designs that nature has evolved throughout history, and the relatively few successes it has had at evolving animals with complex reasoning capabilities, to note that such an approach may be more difficult than it initially appears. Even more telling is the fact that depending upon the criteria used to judge success, in many respects it could be argued that the most successful design nature has evolved so far is that for an insect; with vastly more species than any other design, covering vastly more environmental niches than any other design, including man. The important point being that evolution appears to have stopped at a particular point for insect designs. Of course, the question is, did it stop because an increase in 'cognitive' abilities was unnecessary, or because it was not possible given what it had already put in place? It is tempting to use the latter argument, but this would appear unlikely given the fact that there does exist variety in the 'cognitive' capabilities of insects, culminating in the bee's neuronal system. This, would argue that the reason further 'cognitive' capabilities were not added to insects was the fact that such capabilities did not carry with them a high enough utility to be evolutionary viable. In other words, given this form of (evolutionary) bottom up approach to the design of an autonomous agent the addition of high level cognitive abilities is generally not seen to occur because the difficulty in instantiating such capabilities is more costly than it is beneficial. Such an argument does not say therefore that Brooks' approach will not culminate in an intelligent system, but that it will do so only at great, possible insurmountable, expense (in terms of time, money, design complexity, and a large number of failures). As such, Brooks' argument that such an approach to agent design is 'better' than more traditional approaches is weakened. Leaving this aside, let us look at how Brooks goes about instantiating Subsumption Architectures.

Brooks' first paper (1986) put forward the specifications for a language that could be used to design Subsumption Architectures: the Subsumption Language. This allowed a designer to specify *Augmented Finite State Machine* (AFSM) descriptions, where AFSMs tightly connected perception to action, operated asynchronously with each other, and were not able to monitor other processors directly (such monitoring being done indirectly via the input lines from these other processors). The subsumption language thus allowed individual AFSMs to work on their own goals independently of each other, co-ordination being achieved via an inhibition-suppression mechanism mediating the actions to be taken. For this reason new AFSMs could be added to those already extant within a system, allowing a designer to incrementally build competence layers into a system using a bottom-up approach, where such inhibition-suppression can be seen to occur between AFSMs in three ways:

i. New AFSMs can inhibit the output of old AFSMs.

ii. New AFSMs can suppress the input to old AFSMs.

iii. New AFSMs can provide data for previously unused inputs to old AFSMs .

Notably there does not appear to be any method for new AFSMs to amplify the inputs and outputs for old AFSM. However, this is still clearly a marked departure from traditional approaches to command and control where modules worked on goals in conjunction with each other, co-ordination being explicit within the design. Lastly, there is no shared global memory between processors, and no central processor. Using this language Brooks designed and built a number of robots exhibiting a variety of relatively low level behaviours.

This Subsumption Language has recently (Brooks 1990) been extended to form the New Subsumption Language, which adds the ability for multiple processes (usually AFSMs) to be grouped into behavioural groups, such that those processes can share input/output registers, various timing mechanisms, and a form of shared memory. However, these behavioural groups are much like the old AFSMs, acting as abstraction barriers across which direct monitoring cannot occur, but allowing 'message passing' and inhibition-suppression within and between groups. In many respects, what the New Subsumption Architecture does is to specify the internal workings of AFSMs in a more rigorous manner, where previously the actual implementation of AFSMs was open to the designer.

While I admire the work done by Brooks on Subsumption Architectures primarily because they show how low level behaviour can be bought about within real-world environments, I do not believe that such a paradigm will allow for more complex behaviours. Not only are there are a number of problems stemming from the use of physically grounded representations (Section 6.2), as well as the structural makeup of Subsumption Architectures (Section 6.4), the lack of any learning ability is a major omission for agents of the type Brooks is attempting to design, i.e. intelligent, autonomous and real-world embedded.

3 Situated Automata

In response to the problems of information fusion, and processing redundancy (outlined in Section 6.4) Kaelbling (1986) proposes a hybrid approach to the design of an agent architecture. In keeping with more traditional approaches, at a gross level an agent is viewed as being internally divided on a functional rather than behavioural basis, having a major vertical division between two separate, but interacting automata (finite state machines) - the perception and action components (Kaelbling & Rosenschein 1990). However, in keeping with Nouvelle approaches and the philosophy of hybridisation, each of these two main components is then divided 'horizontally', the perceptual component in to layers of abstraction (from raw sensory input to abstract world models), and the action component in to behaviour producing modules.

Utilising an approach where the components of an agent are viewed as Situated-Automata allows certain points to be made about the information states within those agents. An agent "*x* is said to carry the information *p* in world state *s*... if for all world states in which *x* has the same value as it does in *s*, the proposition *p* is true" Kaelbling & Rosenschein 1990, p.36). As such, the Situated-Automata theory of embedded computation is "a specification of the information content of the internal states of the machine in terms of the external states of the environment" (Kaelbling 1991, p.85), and this allows the information states within a automata to be differentiated into two types:

i. Static Information - Information that is to all intents and purposes invariant within the environment or the agent. For example, the sun always rises in the east, when it rains things get wet, the M40 goes between London and Birmingham.

ii. Dynamic Information - Information that changes over variable periods of time. For example, the cat is on the chair, the bath is half run, the milk is fresh, I am not hungry.

Such a distinction is important because it suggests that the manipulation of information within an agent needn't be stereotypical. Whilst the acquisition[h] of static information is a process that need only occur once, the same cannot be said for dynamic information - information which must be acquired and updated at a rate which is in-keeping with both the temporal aspects of the information itself, and its eventual use by the system. Within the Situated-Automata approach this distinction is used to determine what information must be acquired at run-time by the agent (dynamic information), and what information can be hard-wired into it during 'agent compilation' (static information).

---

[h] The term *acquisition* is used in preference to *sensing* because information gathering need not be the sole preserve of sensory input, but may occur through deliberative processes as well. A large proportion of the knowledge we have at our disposal should be seen as being derived through logical inference rather than direct sensory input.

In support of this differentiation, and the perception-action split, two high-level languages have been developed for the encoding of static facts - Ruler for the perceptual component, and Gapps for the action component. Ruler consists of three parts:

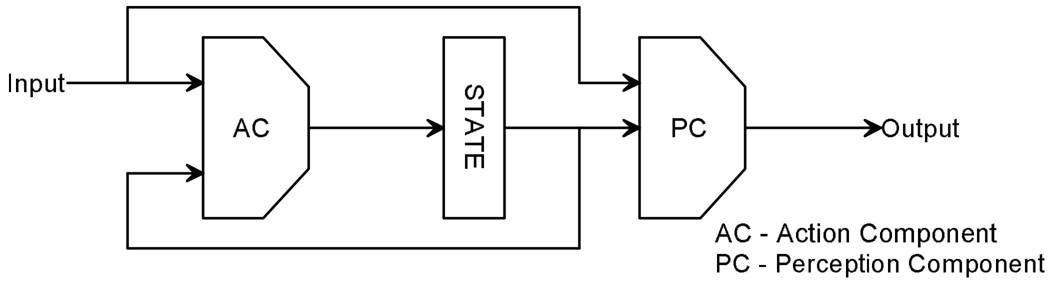
i. A specification for the semantics of the inputs.

ii. A set of static facts.

iii. Specifications of state transitions within the world.

In designing the perception part of an agent, once the above has been specified, the designer then specifies the output semantics. The program then backchains through the provided static facts and world dynamics synthesising a program. Gapps is simpler in form, consisting of a set of goal-reduction rules that map high-level goals into more specific goals. These rules, together with top-level goal specifications, are compiled into a program that maps perceptual conditions into actions, much like Universal Plans. To produce a complete agent both of the resultant programs from the Ruler and Gapps languages are compiled using Rex, a Lisp-like language used to implement fixed sequential circuits of the form shown in Fig. E.2.

Fig. E.2 - Decomposition Of A Sequential Circuit



The result of this compilation is "an undifferentiated mass of circuitry" (Kaelbling 1991, p. 86), destroying any modularity that might have been present within the Ruler, Gapps or Rex programs, i.e. the approach does not support, as it appears, a truly hybrid architecture at run-time. Instead, hybridisation is viewed as being a purely conceptual construct that aids in the designing of agents which must adhere to strict run-time time constraints. Utilising such a view has two obvious advantages over a purely behavioural sub-dividing of an agent's architecture, even if it is at a conceptual rather than functional level. Behaviour producing modules can access information from the relevant perceptual layer, removing the problems of processing duplication, and sensory fusion is made more efficient. The exciting part is that once this is achieved at a conceptual level, by allowing the majority of complex, potentially time unbounded, computations to occur at compile time, the approach guarantees reactivity at run-time.

The approach should be seen as a more rigorous specification for producing behaviour-based architectures than either Brooks, Maes or Agre & Chapman have produced, allowing at different conceptual levels the integration of both symbolic and non-symbolic systems in the design of a situated agent. In this respect it is extremely biologically plausible, deservedly attracted much interested. However, there are a number of important points which must be raised, not least the fact that such a compilation of symbolic to non-symbolic functioning occurs for any computer program, not just those following a Situated Automata approach:

i. Static vs. Dynamic Information.

Utilising a differentiation between static and dynamic facts in the design of an agent ignores the fact that the distinction is neither clear cut, nor constant. An example of this is provided by radioactive decay. A 1 kg block of Uranium 238 viewed on a year-to-year, or even decade-to-decade basis does not change in weight. However, viewed using a larger time scale it can be seen that this is not the case. Given 4500,000,000 years the 1 kg block will have decayed into a 0.5 kg block, 9000,000,000 years 0.25 kg, and so on[i]. Using this level of abstraction the weight of the Uranium block is clearly not static. This is an extreme example, and not one which is particularly pertinent to autonomous agent design. One that is, however, might be the evaporation of water from a heated saucepan. Using this it can be seen that not only is the division between dynamic and static dependent on the abstraction level utilised, but also upon the agent's physically characteristics as well as its current goals[j]. The agent may be fully 'aware' that water is evaporating from the saucepan, but if the changes involved have

---

[i] Other isotopes of Uranium have half life's from equivalently long time spans down to the order of seconds. However, only Uranium 238, 235, and 234, are naturally occurring.

[j] The choice about what level of temporal abstraction to utilise when classifying information is analogous to the problems concerning what level of ontological abstraction to use (Section 4.1).

no effects of the agent's current goals then it need not worry about them, i.e. the water level can be viewed as static even if it isn't in the truest sense of the word. In many domains it will be clear to a designer what facts can be taken as invariant, and what may be taken as dynamic. However, this gives room for errors not least in terms of the incorrect classification of information, and the restriction of an agent's adaptability to change.

ii. Poorly specified goals.

In an analogous fashion to the treatment of information, goals are also viewed as being differentiated into those which are static, being 'statements' about the global behaviour of the agent (i.e. design specifications), and those which are dynamic, being created at run-time (Kaelbling & Rosenschein 1990). The problem lies not in this differentiation but in the fact that once compiled an agent does not possess any abstract symbolic representations of the world. Given this, the mechanisms by which dynamic goals are both generated and dealt with by the system are unclear. A system that does not utilise abstract symbolic representations cannot have abstract run-time goals, it can have perceptually driven action, but this is not quite the same (see Section 6.2)

iii. No ability to learn.

In exactly the same way that a maintenance of state within Pengi constituted learning only in its weakest form, so too does it within Situated-Automata. The lack of abstract symbolic representations removes learning of the sort embodied within statements such as 'The shop opens at 9 am'. Furthermore, even if this was not the case, it is likely that for static facts which are hard-wired alterations required by learning mechanisms would prove difficult to achieve at run-time.

iv. Information specification and compile time.

The majority of the agent's information is compiled off-line and dependent upon the designer. As with Universal Planning (Appendix C Section 1) such a provision may be difficult in complex domains.

v. Does not make use of abstract symbolic representations (see Section 6.2).

vi. Co-ordination is simplistic.

At the conceptual level an agent is dividing into two main components, and within the action component behaviour producing modules operate semi-independently, producing output which must be mediated between. Mediation is seen to occur simplistically on the basis of 'urgency', and a 'desire' to grab resources, and it is difficult to see how co-operation and co-ordination can occur in a non-simplistic fashion.

4 Spreading Activation Networks

While traditional AI has tried to demonstrate sophisticated reasoning in rather impoverished domains, nouvelle AI has attempted the reverse, demonstrating less sophisticated tasks operating robustly in complex domains (Brooks 1990). Consequently, higher level cognitive functions such as planning, learning, and communication are the just the types of problems that previous behaviour-based systems have avoided, preferring instead to focus in on the lower level aspects of behaviour, e.g. walking while avoiding obstacles (cf. Brooks). Such an omission is of grave concern if the two paradigms are to stand on an equal footing for comparison, and for this reason, Maes (1989, 1990b, 1991) has focused in on these issues, but crucially remained within the behaviour-based school of thought. In doing so Maes has not only addressed the problems of module co-ordination, but also the integration of basic learning.

In contrast to the manner in which Brooks and Kaelbling have implemented behavioural co-ordination, Maes has taken an altogether different route, borrowing ideas from neural networks, to form, what she terms, the *Spreading Activation Network* model for the command and control of intelligent autonomous behaviour. An autonomous agent is viewed as a distributed collection of competence modules, each with its own specific behavioural competence (hence the term), and each co-operating locally. Such an agent, embedded within a real-world, will have a variety of goals, but be limited in how it can achieve them by a multitude of resource constraints imposed upon it by that domain. Consequently, the problem is how action selection within such a distributed system can result in globally rational behaviour while meeting the constraints placed upon it, i.e. in this paradigm rational behaviour is seen as a module selection problem (Maes 1991). Maes (1989) puts forward two hypotheses for such a system, namely that:

i. Rational behaviour can emerge from activation/inhibition dynamics.

ii. No bureaucratic agents are needed.

In service of these is an architecture constructed from two types of competence modules, those concerning actions and those concerning beliefs (although the distinction is not always clear), interconnected via three types of link, successor, predecessor, and conflictor. Let me elaborate. Unlike Brooks' Subsumption Architecture, the workings of such modules are left implicit, only their expected effects being explicitly represented within the system via STRIPs like operators. Indeed, modules are defined as a tuple {p a d l}, where:

- p - List of preconditions.
- a - Add list.
- d - Delete list.
- l - activation level.

Action modules are those that are connected on their output side to system effectors, i.e. they make the system carry out certain actions. Belief modules are those whose activation levels elicits a 'belief' within the system, e.g. object x is at location y. Both of these module types are divided into incompatibility groups, e.g. where modules make use of the same effector, or elicit contradictory 'beliefs'. However, what is of importance within Maes's architecture is not the instantiation of the modules themselves, but the manner in which they are connected and activated. Implicit within a module's tuple are the specifying descriptions that give rise to the various links between modules, the so-called successor, predecessor, and conflictor links, which, together with the modules themselves, form an interconnecting control network. A predecessor link will exist between two modules if a precondition (or more) of one module is specified within the add list of another module (i.e. one module precedes the other), and similarly, for every predecessor link there must exist the reverse successor link. Hence, a predecessor link implies that a module's preconditions will be partially satisfied by another module if that other module is made active, while a successor link implies the reverse, namely that an active module will partially satisfy another's preconditions. Lastly, a conflictor link will exist between any two modules if the delete list of one module contains descriptors that undo a proportion of the descriptors within the precondition list of another.

These links are of great importance to the control of an agent because it is through them that activation energy spreads, such spread being seen as continuous, and originating both from modules partially matching the current environmental state (as determined by a module's pre-condition list), and modules that promise to realise one or more of the system's global goals (as determined by a module's add list). From these, positive activation spreads forwards from executable agents (i.e. data driven influences on behaviour) and backwards from non-executable ones (i.e. goal driven influence on behaviour), with negative activation spreading in an equal fashion via conflictor links. The amount of activation passed on is a function both of the module's activation level, and a number of global system parameters (as set out below). Finally, as activation accumulates within a module there comes a point when a system set threshold is reached and that module becomes active, spreading activation forwards to other modules, or carrying out some effector action (as outlined above). However, crucially, module selection is not specific to activation levels, being probabilistic instead. This prevents the system from deterministic in module selection and thus allows new behaviours to be tried. One downside of this use of thresholds is that deadlocks may occur where no agent is strong enough to become active, although within a dynamic environment this is unlikely. The idea is thus both simple and relatively easy to implement, the calculation cycle taking the following form; at each time step do:

i. For each agent:

  a. Calculate input from goals and environment.

  b. Calculate spread of activation.

  c. Locally compute a forgetting (decay) factor.

ii. For all agents within a competence group, see whether:

a. It is executable.

b. Its activation is above threshold.

c. Its activation is above the other agents meeting points a & b (above).

iii. Activate (probably) strongest agent from II for each competence group.

iv. Reset to 0 the activation level of activated agents.

The spread of activation, and thus the overall behaviour of the system itself, can be tuned via the alteration of a number of global system parameters. While within earlier design implementations these were altered by hand, in more recent implementations (1991+) a meta-level network has been utilised such that the system maintains a dynamic balance between reactivity, deliberation, and action execution, in the face of varying environmental situations and goals[k]. These parameters are as follows:

i. Activation Threshold.

ii. Percentage of activation spread forward.

iii. Percentage of activation spread backwards.

iv. Relative amounts of influence from goals vs. the environment.

v. Conflict sensitivity.

Although it is difficult to analyse spreading activation via differential equations, a more qualitative formal theory can be produced, and it can be seen that:

i. Activation threshold level is inversely proportional to system speed.

ii. Activation threshold level is proportional to behaviour optimality.

---

[k] Neither the mechanisms for goal generation, nor the structure of goals themselves are made explicit within the literature (e.g. Maes 1991). However, I believe that goals act in the same manner as other modules, specifying preconditions which must be met, and having predecessor and conflictor links to other modules and goals.

iii. The percentage of back spread is proportional to the system's goal- orientedness.

iv. The number of agents realising a goal is inversely proportional to the time necessary for that goal's achievement.

v. The number of goals is inversely proportional to the system's goal- orientedness.

vi. The number of propositions required to describe the environment is inversely proportional to the system's data-orientedness.

In many respects the spreading activation process can be viewed as implementing a notion of relevance for each competence module, activation levels being inversely proportional to the distance to goals and sensory data, while being proportional to the distance to protected goals (those which must not be undone). Thus, it is argued that although the notion of a plan is different to the norm, in that there are no explicit representations utilised within the system (in keeping with the Behaviour-Based philosophy), such a system can exhibit certain planning capabilities - societies of agents (modules) being able to both express intention and consider the effects of action sequences via activation levels. Indeed, if a sequence of agents (modules within the system) exists which transforms the current state into the goal state then this sequence becomes highly active, but if the sequence potentially implies negative effects it is weakened by inhibition. Furthermore, there exist no centralised search process or search tree, the agents themselves selecting the sequence of agents that are activated in non-heirarchical distributed way.

It is argued that problems, such as local maxima in module selection, are avoided by allowing the temporal length of activation spread to exclude such maxima (effected by the activation threshold). However, crucially, because the spread of activation is partially dependent upon the environmental circumstances on each cycle, continuous replanning is effectively carried out. Consequently, unforeseen situations can be dealt with at each time step, i.e. behaviour is both flexible and adaptive, and yet is always biased towards ongoing plans (activation levels of a module never getting reset unless it is made active, instead being reduced on each cycle).

One limitation that was present within earlier implementations (and indeed with most of Brooks' robots) was the degree to which the system could learn, in fact early versions could not learn at all. However, more recently (1991+) mechanisms for learning have been included within the system, such that a spreading activation network can now exhibit two forms of learning, that of acquiring new knowledge, and that of chunking old knowledge (cf. SOAR), as outlined below:

i. Learning new knowledge.

Achieved via an improvement of 'reliability weights' associated with each link contained within a network (previously unmentioned), and carried out in a distributed manner, every module attempting to learn when it should become active (via its condition list), and what its expected effects are (via its add and delete list). Such calculations are based upon the use of statistical analysis of the network prior to and post activation. However, the manner that this analysis takes is unclear, being unspecified by Maes (1991). This ability of the system is clearly advantageous in that it allows the system to determine the reliability of modules on the basis of past performance, and this in itself is useful not just in improving already extant links but in integrating new modules (as added by the system's designer).

ii. Chunking existing knowledge.

When any goal is satisfied, the system can utilise a trace of the module path used to satisfy it for compilation into a macro module. Such a module, when executed, increases the activation of those same modules that were specified in the initial trace. This effectively leads to a increased probability that that same path will then be utilised again in the satisfaction of the goal.

However, while the system has been given certain learning abilities it still lacks any means of storing information concerning past system states. In some respects it can be viewed as suffering from chronic short term memory failure (cf. Milner's case study of patient H.M. - 1966), and consequently may enter into behavioural loops from which it cannot recover until the environment or system goals change.

I believe that such a problem arises not out of the design stance of utilising activation/inhibition dynamics for the selection of modules, but in the lack of an explicit abstract symbolic representation. Without the means of abstracting away from the here and now the system is effectively stuck in the present, unable to recall the actual actions it took in the past, the order of those actions, and their outcomes. Furthermore, such a lack of representation gives rise to other problems (see Section 6.2). Not least, concerning the system's ability to carry out explicit reasoning tasks of the sort that epitomises human-like cognitive activity. Indeed, Maes herself states that...

> "the architecture is not designed for environments requiring very sophisticated inference and planning."
>
> *Maes 1991, p.120*

In my view this is a crucial loss. Without explicit representational mechanisms the system is denied the ability to reason both about other agents within that environment, and about possible future events, i.e. the system must always act in a reactive (albeit goal directed) manner, rather than one which is prevenient. As Maes writes...

> "the external influence during the 'planning' or spreading activation phase is so important that plans are only formed as long as the influence from the environment is present"
>
> *Maes, 1989, p.995*

Now, although this is posited as a mechanism by which the system is able to adapt 'plans' to changing environmental demands, one must remember that all extant goals influence the activation spread. As such it is worrying to think that 'plan' adaptation could result in the system effectively abandoning its current goal in favour of another one (even if only temporarily). Although such an interleaving of goals is often very useful, it brings with it numerous questions concerning the manner in which the system views the independence of such goals, and the actions required to achieve them. If goal x requires the effects of a module to be protected then should that module remain protected when goal x is abandoned in favour of goal y? If it is, then that might preclude the system from achieving y, but if it isn't then the converse may hold. Furthermore, would such a swap in 'active' goals be desirable?

Although Maes does not mention how a system lacking an explicit domain representation would generate new goals in the face of environmental events, she does claim that the system would be able to cope with such new run-time goals (1989). Given the likelihood that many new goals may possibly be generated at any moment in time within a real-world domain, the lack of any stated mechanisms by which the number of such goals could be reduced is clearly problematic. Indeed, earlier we saw how the speed of the system decreases proportionally with the number of goals extant within the system. This would imply that a system of this sort needs to deliberate about when to achieve extant system goals, rather than simply bundling them all together. While such an implicit ordering of goals is made possible via the use of activation levels, the ordering is very myopic, being entirely driven by environmental rather than cognitive events. There can be no sense of goal importance, urgency, or any other factor which might influence the order system goals are placed in, and as such there are likely to be times when the system attempts to address goals which are inappropriate given the circumstance. I believe the proposed AIMAE system's (Chapter 7) use of ratings is in many ways analogous to Maes' spreading activation, but crucially does allow effective goal management of this sort.

In summary therefore, it can be seen from the above that the use of spreading activation patterns in the co-ordination of action selection provides a system with a process which is capable of selecting actions such that the overall system behaviour is seen to be goal directed, reactive, and 'rational'. Furthermore, as we have seen, the combination of competence module usage, reliability learning, and the fact that inter-module links need not be explicitly specified by a designer (being compiled by the system itself), is clearly advantageous. Firstly, it means that at any time new modules may be run-time added to (or deleted from) the system, effecting its behaviour immediately, the system compiling the requisite links automatically. But secondly, it allows the system to determine, on the basis of past performance, the reliability of modules with concern to their various links with other modules. As such, the system is not only able to learn the best 'ways' of achieving various goals given certain starting conditions, but also to integrate new modules, or adjust for 'deleted' modules. Finally, with chunk learning the system is provided with a means of learning plans once they have been created.

However, as with all other behaviour-based systems, the lack of explicit representations, and the reasoning mechanisms that utilise such representations, proves to be a major hurdle which this and other systems of its genre fail to clear. Consequently, I see it as extremely unlikely that without adding additional functionality to the system in terms of reasoning and goal management mechanisms, it will prove viable as an embedded agent command and control system. For instance, Maes suggests that the model supports implicit reasoning concerning beliefs, yet it is difficult to see how beliefs such as 'Joe is not in the kitchen', or 'all students drink too much' can be accommodated. Thus, while it does eloquently show that behaviour-based systems are capable of behaving at a more complex level than that so far exhibited by Brooks' systems, sadly it so far falls short of the complexity of deliberation more traditional knowledge-based systems can achieve.

APPENDIX F

THE AIMAE ARCHITECTURE

1 Perception Within AIMAE

The Perceptual Module forms the input interface between an agent's environment and its cognitive system, transcribing domain events into goals. Although I hold to the belief that an agent must represent such goals in a symbolic form, it is possible that their generation can be as a result of search carried out on the perceptual input (or a processed version of), rather than on its abstraction into an explicit symbolic representation, i.e. from physically grounded representations a system can abstract symbolic goals. However, such a transcription of domain events into symbolic goals raises a number of points for consideration, not least the manner in which data is acquired and processed prior to goal generation, and the manner in which generated goals are then dealt with.

6.1.1 Data Acquisition

Within Chapter 3 it was shown that there are a number of requirements concerning data acquisition that an autonomous agent's perceptual system must address. Firstly, and most importantly, in accordance with requirement 3.2.i, it is infeasible to presume that an agent will be capable of processing all environmental data available to it. Consequently, all sensory modalities (but especially vision) need to be active rather than passive data acquisition processes. Specific areas within sensory fields should be singled out for perceptual processing, and this 'focusing' should occur in accordance with the agent's currently active goals, as well as its environment. Hence, the areas 'focused' on need not simply be spatial areas, such as the fovea of human-like eyes, but may also include events, such as loud noises, or entities, such as fast moving objects. However, it should be noted that in accordance with requirement 3.2.iii, such 'focusing' must not remove complete sensitivity to areas outside of that which are being processed. The reason for this is two-fold. Firstly, it prevents an agent becoming 'blind' to certain critical events. Secondly, it means that a modicum of very low level processing must be carried out on all data, consequently providing the agent with a means of 'keeping track' of potentially interesting events, and giving rise to phenomena such as the cocktail party effect (e.g. requirement 3.2.iii).

There exist a variety of methods by which such active perception may be accomplished, but the method I intend to utilise within AIMAE is one where all information within the perceptual field is processed at a low level, with more in-depth processing being directed towards areas of interest within that field in accordance with higher level cognitive processes. For example, an agent whose currently active goal is to check up on babies, may focus in on just the babies within its visual field, partially, but not completely, ignoring other objects and entities.

6.1.2 Data Processing

In addition to the above and in accordance with requirement 3.2.ii, perceptual information will need to be pre-processed prior to any goal generative search. Such preprocessing taking the form of data abstraction (e.g. statistical analysis), data amalgamation (e.g. within or between sensory modalities), or a combination of the two. It should also be noted that in much the same way that the combining of two tonal frequencies within the human auditory system can sometimes give rise to the perception of a third, or more, tonal frequency[I], it may also be the case that the amalgamation and abstraction of low level information may sometimes result in goals other than those that would be produced if the information was left unprocessed. As with perceptual focusing, such data processing should be at the behest of higher level cognitive mechanisms. For example, while focusing on babies within the visual field, data processing may be directed towards gaining information pertaining just to their apparent health, e.g. energy levels.

6.1.3 Goal Generation

I So-called combination tones, a phenomenon termed Otto Acoustic Emissions.

The final step between raw sensory data and explicit symbolic goals, is the generation of those goals as determined by processed sensory data. Goal generation is a task which is seen to occur within AIMAE via the use of specific sub-mechanisms, the *goal generators* (produced by the Deliberation Module - see Appendix F Section 2.1.c). Each of these is effectively an independent sub-unit operating in parallel and asynchronously from the rest of the Perceptual Module, and each other. As discussed earlier within Chapter 2, one of the concerns that effects an agent embedded within a real-world environment is the fact that it will always be resource bounded. Consequently, goal generator mechanisms should not require recourse to high level cognitive processing, and hence, they operate heuristically, searching the data resultant from perceptual processing for specific 'triggers', as specified within each generator's rule-base. However, even if such generators operate heuristically, and therefore do not themselves 'tie up' high level cognitive resources, within a real-world environment they could potentially produce a huge number of resource utilising goals with catastrophic effects. Thus, in keeping with requirement 3.3.i, an agent must have mechanisms for preventing a flood of conflicting goals becoming extant within it, and one way of achieving this is via the filtering of newly generated goals prior to any form of high level cognitive activity upon them (see Section 7.2). As with NML1, goals which pass such filtering are described as having *surfaced*, while goals that fail such filtering are described as being *suppressed*. Such suppressed goals are not immediately lost, but remain within a Perceptual Module Goal Store, where their insistence levels may either decrease with time, or be increased via Generator re-activation (see later).

If filtering is viewed as a means by which a system may reduce the loads placed upon its own cognitive resources, then it should also be seen as occurring as 'cheaply' as possible (requirement 3.3.ii). Consequently, when a generator is triggered by perceptual data, or the Deliberation Module (see Appendix F Section 2), the goal that it produces is assigned a cheaply computed heuristic *insistence* value. This is a value which represents a rough approximation to that goal's importance[m] and urgency[n] upon which

---

[m] This is in relation to an agent's dispositions (see Footnote 31) only. Consequently, over time as dispositions change the relative importance of goals will alter as well.

[n] It is difficult to specify exactly how urgency can be computed in a heuristic fashion; possibly its value

filtering can occur. When a goal is first produced this value is set at a default level, but it can increase with future generator activation. However, because of this activation driven increase in insistence level, there exists the equivalent of a neural refractory period for an activated generator, within which perceptual data cannot re-activate that generator (Fig. F.1). This is not a requirement as laid out within Chapter 3, but rather a design decision. Without the use of a refractory period a generator could be re-triggered repeatedly until perceptual data dictated otherwise. Such re-triggering could cause the insistence value of a goal to increase at a rate greater than its rate of decrease. An occurrence that could lead to the rapid spiralling of a goal's insistence value.

Fig. F.1 - Goal Generation & Activation Levels

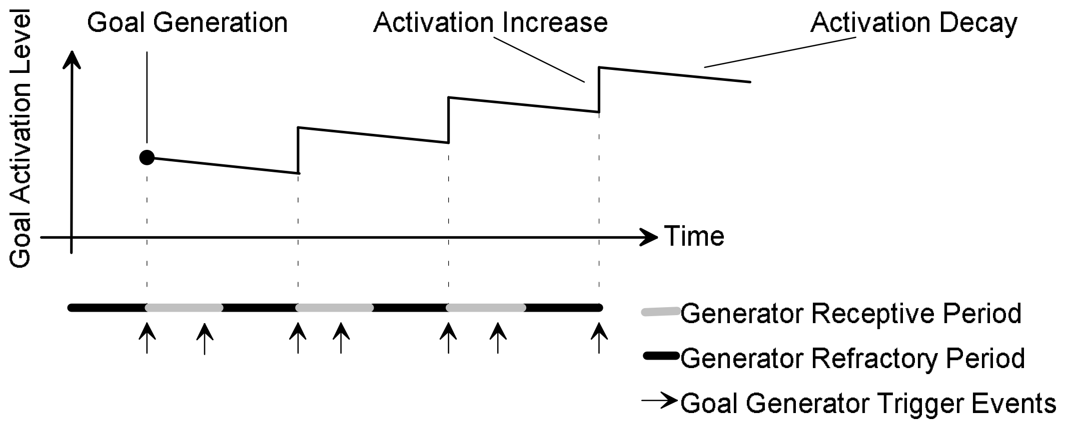
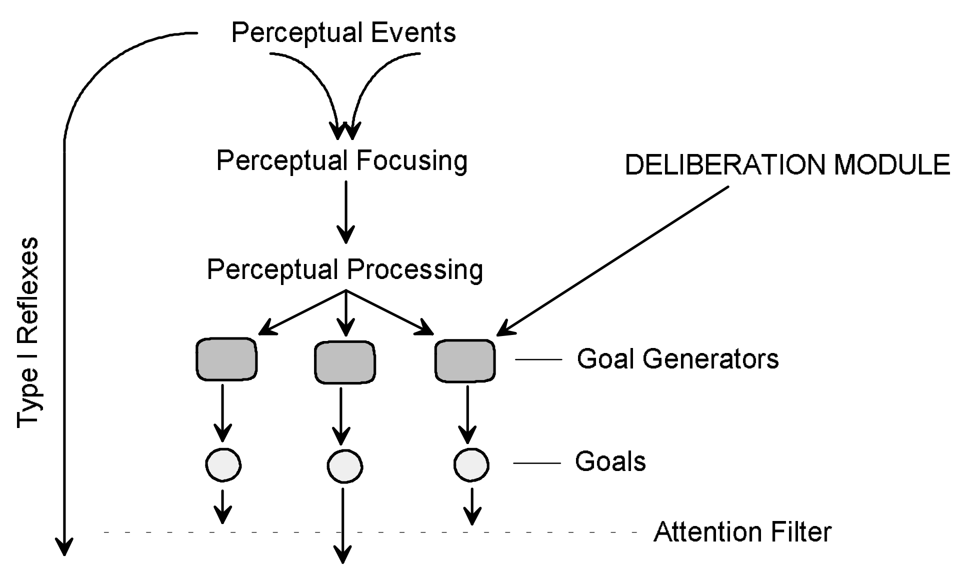


Fig. F.2 summarises the above. Perceptual data (or the Deliberation Module) triggers the activation of a generator, which produces a goal whose insistence value while set initially at a default level is not fixed (in fact Deliberation Module triggered goals are assigned insistence values as determined by that module not the generators, but this is a special case of goal triggering and so can be ignored for now). The goal is placed within a storage area used by the Perceptual Module, termed the Goal Store, and the generator after producing this goal then goes into a refractory period, within which it cannot be re-triggered. After this period, it may be re-triggered, and if the goal that it produces is already extant within the Goal Store it increase the insistence value of that

would be based on the intensity of the triggering stimuli.

extant goal rather than creating a new instance. Similarly, if the goal is instead extant within the Goal Management Module generator activation increases the goal's *intensity* value within this module rather than creating a new instance of the goal within the Perceptual Module (see Section 7.2.1). This has important consequences for the agent's behaviour. Finally, in addition to pre-filter insistence values being able to increase, they may also decrease. Such a decrease occurring at a system set rate, no faster than the system set refractory period. When a pre-filter goal's insistence value reaches zero it is removed from the Perceptual Module Goal Store and subsequent activation of its generator re-initialises the whole process described above.

Fig. F.2 - From Domain Events To System Goals



Finally, certain perceptual events may trigger reflexive mechanisms, either of the Type I (innate) variety, or of the Type II (learnt) variety. Both types by-pass the more normal route to action, i.e. from events to goals to actions, tending to elicit action directly (Fig. F.2).

6.1.4 Goal Filtering

As touched upon above, and in accordance with requirement 3.3.i, to prevent goals from swamping the system, an agent's command and control system must have a mechanism for filtering out a proportion of newly generated goals prior to any high level deliberation concerning those goals. This is achieved within AIMAE in a manner first posited by Sloman & Humphreys (1992) via the use of a simple heuristic *filter* mechanism, operating on the insistence values of newly generated goals (see Fig. F.2 above). Goals whose insistence values do not pass the filter threshold are not admitted into higher level cognitive modules and remain within the pre-filter stage of the perceptual module, either until they are admitted, or until their insistence value drops to zero. However, this simplistic view of filtering was extended by Beaudoin (1994) with NML1, where the notion that filter mechanisms should be context sensitive was introduced, e.g. blocking certain 'insistent' goals from cognitive resources, while permitting certain 'uninsistent' goals access to those resources. By carrying out such context sensitive filtering a system is able to more flexibly control the flow of goals entering its cognitive mechanisms, for example, not permitting hunger goals to surface while carrying out a delicate medical operation. In contrast to this, AIMAE does not utilise such idiosyncratic filtering, instead control extra to basic filtering is provided by the goal adoption/rejection mechanisms based within the Goal Management Module. It will be interesting to see whether such mechanisms are adequate at protecting cognitive resources from goals which while insistent are not congruent with currently active goals, or whether the filter mechanisms used should adopt a more flexible type of operation.

Goals can be seen to fail filtering at one point in time only to pass at a later point for either of (or a combination of) the following two reasons:

a. The goal's insistence value is raised by re-activation of the appropriate generator.

b. The filter's threshold decreases.

Point (b) is important, and accords with requirement 3.3.iii. The filter threshold value is not a static fixed point, but one which may move dynamically as determined by the agent's higher level cognitive modules (see Appendix F Section 2.2). Raising the threshold has the general effect of reducing the flow of new goals into the system, while lowering it has the opposite effect, increasing the flow of new goals into the system. This is thus one of the 'tools' that the system possess for controlling its own responsiveness to environmental events (requirement 3.6.vi).

Finally, within the Affect and Attention project this filter is known as the *attention filter*, due to the following reasoning: One of the characteristics of higher level cognitive mechanisms within humans (and possibly other animal species) are that they are partially introspectable by an agent's management systems, unlike lower level mechanisms. The question is of course why? One seemingly plausible answer lies in the reasoning behind the need to limit input into those higher level mechanisms in the first place. If cognitive resources are limited, and hence require protection from overloading, then it is likely that one possible protective measure is to prevent their direct introspection of the complex workings of lower level mechanisms. Take as an analogy the management structure for a large company - any single manager on that company's board would very quickly get swamped down in the 'nitty gritty' day-to-day workings of that company if he or she tried to take a direct role in those workings. Thus, to prevent this happening, and to keep the managers free for the 'important' work, it is sensible if they are dissuaded from directly examining and 'tampering around' with the lower levels. But this does have a downside. If managers are not directly involved in the day-to-day running of the company they cannot know what occurs on a day-to-day basis. Similarly, within an agent, if its higher level cognitive mechanisms are not permitted to directly view lower levels then they cannot know what is going on in those lower levels. Thus, an agent will not be 'aware' of newly generated goals until they pass the 'attention' filter (hence its name), cf. Freud's use of the sub- and pre- conscious. It can be seen from this that a modular decomposition of an agent's command and control system (as opposed to a Brooksian type of approach) allows a clearer understanding of why in certain circumstances 'hiding' information from the system can be beneficial. Furthermore, it clearly illustrates how the specifying of an actual design provides a means of gaining an understanding concerning psychological phenomena.

2 The Deliberation Module

As has been mentioned at various points throughout the Chapter 7, certain decisions require computational processes more complex than that which may be delivered via heuristics. It is these decisions that form the raison d'etre of the Deliberative Module, and although outlined as follows the means by which they are achieved are not as yet detailed:

6.2.5 Perceptual Module

a. Data Acquisition.

One of the central characteristics of a real-world environment concerns the manner in which information is accessible to an agent. An environment of this type potentially contains far more information than an agent could process, or even need to process. Consequently, it is desirable that an agent collect information that is relevant to its current goals, and hence perceptual focusing is necessary (see Appendix F Section 1.1). Such focusing whether internally directed or externally directed should be at the behest of the Deliberative Module with respect to the agent's current goals and dispositions.

b. Data Processing.

Further to the above, potentially useful information is likely to be distributed both in terms of time and space, as well as varying in quality. Additionally, perceptual systems may not be wholly accurate in their transduction of environmental data into a form of use to an agent. As such, an agent requires mechanisms for abstracting information away from the raw perceptual input, such that the degradation of data from noisy inputs is minimised, and data distribution is accounted for (see Appendix F Section 1.2). This is achievable via the processing of data in accordance with an agent's current goals and dispositions, as directed by the Deliberative Module, e.g. continuous statistical analysis.

c. Creation of Goal Generators.

The generation of goals within the Perceptual module is achieved via sensory and abstracted data triggering rules associated with various Goal Generators. Such goals are given an insistence value by the appropriate generator, and all generators have a default refractory period (see Appendix F Section 1.3). Consequently, the production of Goal Generators by the Deliberative Module has a large bearing upon the agent's behaviour; determining not only the goals the agent may form, but also what triggers them, their initial insistence values, and the potential for such values to increase.

d. Triggering of Goal Generators.

In addition to environmentally triggered Goal Generators, generators may be triggered at the behest of the Deliberation Module, as and when required. Goals triggered in this manner are given an insistence value which by default is equal to what they would have gained had they been triggered environmentally, but which may potentially be set by the Deliberative Module.

6.2.6 Attention Filter

The Attention Filter is one of the main mechanisms that an agent's command and control system has for controlling the flow of goals into 'cognitive' behaviour producing modules. It is a heuristic filter which operates on the insistence values of goals within the Perceptual Module's goal store, allowing those with a higher than threshold value to gain entry into the Goal Management Module. The threshold value of this filter is set by the Deliberative Module in accordance with the number of current goals, their importance, and urgency.

6.2.7 Goal Management Module

a. Prediction & Feasibility Testing.

While the Control Plan carries out actual goal expansion, it cannot carry out such expansion in a 'mental' sense to determine questions of a 'what if...?'

nature. This form of expansion is seen to be the providence of the Deliberative Module which should be capable of taking a copy of the Control Plan contents and manipulating it to determine possible future outcomes. As such the Deliberative Module, must either have access to Control Plan mechanisms for its calculations, or have those mechanisms itself. Either way, the outcome of such predictive feasibility testing should determine the workings of the Control Plan with respect to the appropriate goal or goals. It must be noted however, that generally behaviour is not guided in such an explicit manner, prediction tends to be utilised in deciding what to do next when heuristics are not capable of doing so (see later).

b. Goal Directed Information Gathering.

When initial goal assessment takes place, it does so heuristically on the basis of ratings associated with a heuristically chosen strategy. These ratings can be utilised in conjunction with domain knowledge to provide a rough assessment as to whether the goal should be adopted or not. However, when information extra to, or more accurate to, that which is currently known is required the Deliberation Module can be requested to gain this information (see Section 7.2.1). In effect the Control Plan asks the Deliberation Module to trigger information gathering goals, for example, looking at a clock.

c. Goal Assessment.

Once a goal has been placed within the Control Plan re-assessment of it may take place such that more accurate and up-to-date measures of the goal's importance, urgency, and utility are produced (see Section 7.2.9). Such re-assessment is likely to be more accurate for a number of reasons, not least because as the goal is processed more information becomes available for assessment, and because importance can now reflect goal congruence as well as dispositional congruence.

d. Termination Point Computation.

Re-assessment of goals may give rise to extra or more accurate information for use in Termination Point calculations. Furthermore, such information is more up-to-date, allowing Termination Point re-calculation to not only be more accurate, but also context sensitive.

e. Operator Assessment.

In a manner similar to goal assessment, when an operator is chosen by the Control Plan mechanisms for use in service of a goal the conditions associated with that operator must be verified as being met. Deciding which operators should be met involves consideration not only of the goal in question, but also its relationship to other extant goals and system dispositions. For this reason, such verification is seen to occur within the Deliberation Module in conjunction with domain knowledge. Furthermore, when conditions are seen to fail a decision about what should then be done is also a Deliberation Module task, and may take three general forms:

i. Attempt to meet the condition via the addition of behavioural operators prior to that in question. This option is open to all operator types.

ii. Reject the operator in favour of a viable alternative. This option generally only occurs for 'failed' strategies, i.e. swap one strategy for another. However, if a goal only has one associated strategy then option (iii) must be used, i.e. reject the goal that required the strategy.

iii. Reject the strategy/goal that specified the operator. This option is used instead of option (ii), i.e. when no viable alternatives for the operator in question exist.

These three options give rise to varying degrees of plan alteration, or backtracking; the first option giving rise to the least alteration, and the third, the most alteration. When the Deliberation Module is unable to carry out such assessment, the operator in question is selected without condition checking, and this may give rise to a number of actions slips. Lastly, operator assessment is also utilised by the Control Plan when it is notified by the Effector Module of failed actions.

f. Rating Selection.

Heuristically deciding which strategy should be utilised in the service of a particular goal is carried out with respect to ratings associated with those strategies (see Section 7.2.6). However, not only is it likely that a system may make use of many different types of ratings depending upon the goal in question, but it may also utilise various ratings sets depending upon the current situation (requirement 3.6.vii). Consequently, deciding how to achieve a goal requires a choice not only about which strategy to adopt, but how to decide between those strategies. Since condition checking is seen as being a Deliberative Module task, deciding which rating set to utilise is seen as being so as well, and is carried out with respect to the current environmental situation. Once a rating set has been chosen determining which rating type(s) from that set to utilise, and how must be decided upon. Depending upon the 'busyness' of the Deliberation Module, such a decision may be simple, e.g. use the fastest strategy, or more involved, e.g. use the fastest reliable strategy. It should be seen that in cases of extreme 'busyness' the rating chosen defaults to the frequency rating. Hence, when the Deliberation Module is otherwise occupied, e.g. carrying out prediction computations, the strategy chosen for a goal in need of expansion will be the one most often chosen before. Such default expansion may give rise to certain types of action slips.

g. Strategy Selection.

Once the rating set and types have been determined the Deliberation Module then has to decide which method of strategy selection should be used (see Section 7.2.8). Lastly, although strategy selection should proceed automatically once this has been done, there is a possibility that if non-probabilistic methods are used no clearly winning strategy will be available. At this point the Deliberation Module must be utilised to decide which of the possible strategies to use (see Section 7.2.8). Such a decision may involve prediction computations, however, where these are not possible, or the Deliberation Module is otherwise engaged, the module defaults to selecting the strategy in a best first manner without any difference threshold.

h. Goal Re-evaluation.

The last main task of the Deliberation Module is the re-evaluation of goals once their intensity vales have dropped below a threshold point. Such re-evaluation is initially carried out via the same Control Plan mechanisms that carried out heuristic goal assessment (see Section 7.2.9). However, if a goal is not rejected at this point it must be assigned a new intensity value, and this is determined by the Deliberation Module with respect to other extant goals and system dispositions. Once, this has been achieved a new Termination Point is computed and the goal is replaced in the Control Plan.

i. Finally, in addition to carrying condition checking at the specific behest of the Control Plan (option e above) the Deliberative Module should be capable of monitoring the extended conditions for 'active' operators. When such conditions fail the operator in question must terminate prematurely, and the module must decide what to do, as in point (e). However, if possible the module should also keep track of pre-conditions - their failure may signal successful premature termination (see Appendix F Section 4). The degree to which extended and pre-conditions are monitored by the Deliberation Module should be determined by the Intensity of the goal in question. Hence, the more intense a goal the more processing time the Deliberation Module will give to monitoring its conditions and the less time it will be able to give to other tasks. Hence, the reason why certain tasks may effectively be ignored, the module return default values rather than carrying out actual computation.

While the above outlines the various tasks that the Deliberative Module should perform it does not detail these. The reason for this is two-fold. Firstly, and most importantly, I wish to examine the types of behaviour produced by a system which does not carry out any form of non-heuristic deliberative processing. Such a study will give a better understanding of what it is that deliberation must account for within real-world embedded agents, allowing the processes of deliberation to be more clearly specified, and possibly shedding light on human psychological disfunctioning. Secondly, it is not yet clear how the above tasks could/would be achieved within the AIMAE system, although most of them will need to be implemented in a basic form for AIMAE to operate. Thus, this module forms part of the 'shallow' aspect of the overall broad but shallow architecture.

3 The Effector Module

The last module that makes up the AIMAE command and control system is the Effector Module. I envisage this as being constructed in a manner analogous to Brooks's Subsumption Architectures, with hierarchically organised layers of control, where higher layers subsume lower layers. Since it is likely that a system will have more than one form of behavioural effector, it is also likely that it will be capable of performing more than one task at a time (see Footnote 30). It is the role of the Effector Module to co-ordinate behavioural actions such that they do not conflict, to notify the Control Plan when an action has been achieved, or not, and whether an action can be carried out, or not. Hence, on every cycle of the Control Plan it may attempt to pass to the Effector Module a new action. The Effector module must either accept that action and attempt to execute it, or bounce it back at the Control Plan. In the latter case a new action will be chosen, or the Control Plan will wait until the chosen action is executable. This decision will be based upon the 'strength' of the intended action (see Section 7.2.9). When an action fails the Control Plan is notified and must act accordingly. A decision concerning what to do in this situation is passed to the Deliberation Module which carries out operator assessment. If all conditions are seen to be met, the action is re-attempted, however, if certain conditions have become unmet a decision must be made by the Deliberation Module as to what to do to rectify the situation.

4 Behavioural Conditions Within AIMAE

All behavioural 'operators' (e.g. goals, strategies, actions, cycles, and process initiators) within the AIMAE system are associated with conditions, some of which need only be met prior to the initiation of the operator, and some of which must met for the entire length of that operator's execution, i.e. *invocational pre-conditions*, and *temporally extended conditions* (extended conditions for short), respectively. Invocational pre-conditions are those which must be met for a behavioural operator to be initiated, but need not necessarily hold for the whole of that operator's execution, and indeed in some cases cannot hold for the whole of the operator. Conversely, an extended condition is one which must be met for operator's initiation, and then continue to hold for that operator's entire 'existence'. For example, the action of picking up an object requires that your hand be empty; a pre-condition which cannot remain true. This is opposed to the extended condition specifying that an object must be present and 'pick upable' throughout the action. Furthermore, as commented upon later, condition failure during an operator's execution may be one of the indicators available to an agent for determining when it has failed or succeeded. Hence, it should be seen that the conditions attached to operators can play a dual role within an agent's command and control system, determining whether operators may be initiated, and determining whether operators are to terminate early (or not), and with success (or not).

One problem pervading the subject of operator conditions is their ontology; what structure do they take, how do they interact, and what factors effect those interactions? Structurally, conditions are seen as logical expressions, thus they may include operators such as AND, OR, >, <, true, false, etc. For example, {X must be true}, {Y OR Z must be true}, {Z must be greater than X} etc. However more subtly, they may also take the form of probability expressions, or expressions denoting various system states, or a combination of all three. Consequently, the evaluation of individual conditions may be highly problematic, and the evaluation of a set of conditions even more so, especially when they interact with one-another, e.g. some conditions may fully preclude others, while some conditions may alter the relative importance of others. Such evaluation problems may be further compounded when conditions take not only boolean forms, but also qualitative and quantitative metrics.

Determining whether a behavioural operator can potentially be initiated may be a task of immense proportions, involving highly complex inference over a set of conditions; inference that may be computationally and time intensive. Thus, the greater the proportion of inference that can be achieved heuristically, the better. Poli & Brayshaw (1994) are examining ways in which the need for such inference may be circumnavigated via the use of neural networks trained to give the same results as inference, but without the run-time costs. Given the assumption that humans at least seem to be able to remove the need for complex inference through learning, at a cursory level their approach does seem biologically plausible. However, it has yet to be seen how the learning of such condition inference results can cope with the binding of variables - possibly it can't.

So far the discussion has centred on the types of conditions and the effects they have both on each other, and the instantiation of behavioural operators. However, it must also be seen that the manner in which a system deals with failed or unmet conditions is just as important. When conditions are not met, or fail, the system is faced with a difficult choice concerning whether to:

i. Attempt to establish (or re-establish) those conditions, usually via the introduction of additional operators prior to that in question, but not always.

ii. Abandon the operator altogether in favour of another which can also potentially serve the same purpose.

iii. Abandon the line of behaviour which resulted in the problematic operator.

iv. Suspend the line of behaviour which resulted in the problematic operator.

For example, if a pre-condition takes the form of {X must be greater than Y, OR X must be greater than Z} and neither aspect of this is true, then the system must decide not only whether to attempt to meet the condition at all, but which value it should attempt to alter in doing so. Consequently, it should be noted that the order in which such pre-conditions are evaluated may have an effect on the resultant behaviour depending upon the manner in which they are handled. AIMAE will address this problem by evaluating all pre-conditions prior to any decision concerning the adoption or rejection of the operator in question. Furthermore, such decisions will be made only within the Deliberative Module, as they are likely to be non-trivial, involving among other things time available, potential costs, potential benefits, etc., i.e. the decision as to whether particular operators should be used when their pre-conditions are not met is seen as being identical in form to the non-heuristic decisions concerning goal assessment (as posited within Section 7.2.9, these decisions are also identical to those concerning what to do when strategy components fail).

Similarly, if an extended condition fails during an operator's execution, the system must decide whether to attempt the re-establishment of the failed condition and if so how, or whether to suspend, or abandon the operator. Indeed, the only difference between an unmet pre-condition and a failed extended one, is that by definition the failure of the latter must also lead to a premature (if only temporary) termination of the behavioural operator in question. Thus, a decision concerning whether to attempt to re-establish an extended condition should effectively be seen as a re-affirmation of the system's commitment to the operator that required that condition. A good example of how a system may address a failed extended condition in accordance with the four points above can be seen within the Nursery Scenario (Chapter 8). When a baby gets hungry the nurse may form the goal to feed that baby. An extended condition which must hold true throughout the duration of any resultant behaviour is that the baby is still hungry. Thus, if the baby dies this condition will fail, and the nurse may (should!) decide that rather than attempting to re-establish the failed condition, it is 'better' to abandon the goal altogether. Conversely, however, an implicit extended condition may be that the goal to feed the baby is the most insistent one that the nurse currently has. Upon failure of this condition baby feeding behaviour terminates prematurely, at which point the nurse must decide whether to attempt to re-establish the condition (possibly by arbitrarily raising its intensity), or again terminate the goal. However, in this latter case goal termination should take the form of suspension, rather than complete abandonment.

As well as signifying problems, premature condition failures may signify more positive results. For example, when filling a glass with water, if the extended condition that the glass is not full fails then the goal is in effect achieved. The question that must therefore be asked is, if the failure of extended conditions may be indicative both of the successful termination of a process (or cycle), as well the unsuccessful termination of a goal (note the difference in effect between the failure of an extended condition for a goal and a process/cycle), might not this be true for pre-conditions? It is my belief that while the failure of a pre-condition after operator initiation should never be seen as problematic (if it was then the pre-condition should have been an extended condition!), its failure may indicate the successful premature termination of that operator. Take for example, the pre-condition that the hand must be empty prior to picking up an object. Failure of this condition may indicate that the object the hand attempted to grasp is now being held (of course it could also indicate that something else entirely was grasped by mistake). What of course remains to be seen is whether this assumption is correct, or whether it is necessary to specifically state a set of termination conditions as used by Loyall & Bates (1991) within their HAP system. I believe that the use of termination conditions within AIMAE may prove to be excessive given the other conditions used.

As with everything in life, there is one problem with this use of failed conditions in the determination of terminating operators; noticing such failures involves perceptual information. Consequently, it is feasible to imagine times when the perceptual and deliberative mechanisms are 'focused' on environmental information relevant to other tasks, such that termination goes unnoticed. Indeed, this may be the cause of many action slips, e.g. over filling a cup when listening to a particularly 'attention grabbing' piece of news, attempting to write with a straw rather than a pen, etc. This example appears to indicate that within humans perceptual information must be routed through the equivalent of AIMAE's Deliberative Module for use within planning mechanisms, e.g. the Control Plan. Furthermore, it also lends support to the idea that precondition checking occurs in conjunction with the Deliberative Module, and hence not only are action slips of the above type possible (failing to terminate behaviours correctly), but also of a type characterised by performing behaviours which are not congruent with the current goal(s) and situation. This is a prime example of how the design based approach to command and control within real-world environments can help us understand the possible causes for behavioural problems within humans.

REFERENCES

Agre & Chapman (1987)
Pengi: An Implementation Of A Theory Of Activity.
AAAI - vol. 1, pp. 268 - 272

Allen, J.F. (1984)
Towards a General Theory of Action and Time.
AI Magazine - vol. 23, pp. 123 - 154

Allen, Johannes, & Koomen (1983)
Planning Using a Temporal World Model.
In Proceedings: 8th IJCAI

Barrett & Weld
Partial-order planning: evaluating possible efficiency gains.
AI Magazine - vol. 67, no. 1, pp. 71 - 112

Basye, K. (1991)
A Decision-Theoretic Approach To Robot Control Systems.
SIGART Bulletin - vol. 2, no. 4, pp. 34 - 37

Bates & Reilly (1991)
Broad Agents.
SIGART Bulletin - vol. 2, no. 4, pp. 38 - 40

Beaudoin & Sloman (1991)
A Proposal For A Study Of Motive Processing.
TR: #CSRP-91-6, School Of Computer Science, Birmingham University.

Beaudoin & Sloman (1993)
A Study Of Motive Processing & Attention.
In Proceedings: AISB-93 - pp. 99-125

Beaudoin, L.P. (1994)
Goal Processing in Autonomous Agents.
PhD Thesis. Cognitive Science Department. Birmingham University.

Birnbaum, Collins, Brand, Freed, Krulwich, & Pryor (1991)
A Model-Based Approach to the Construction of Adaptive Case-Based Planning Systems.
In Proceedings: The Case-Based Reasoning Workshop, Washington - pp. 215 - 224

Bratman, Israel, & Pollack (1988)
Plans and Resource-bounded Practical Reasoning.
Computational Intelligence - vol. 4, pp. 349 - 355

Brooks, R.A. (1986)
A Robust Layered Control System For A Mobile Robot.
IEEE Journal Of Robotics & Automation - vol. ra-2, no. 1, pp. 14 - 23

Brooks, R.A. (1990)
Elephants Don't Play Chess.
Robotics & Autonomous Systems - vol. 6, pp. 3 - 16

Brooks, R.A. (1991a)
Intelligence Without Representation.
Artificial Intelligence - vol. 47, pp. 139 - 159

Brooks, R.A. (1991b)
Integrated Systems Based On Behaviours.
SIGART Bulletin - vol. 2, no. 4, pp. 46 - 50

Brooks, R.A. (1991c)
Intelligence Without Reason.
In Proceedings: 12th IJCAI, Sydney, Australia - vol. 1, pp. 569 - 595

Chapman, D. (1985)
Non-linear Planning: A Rigorous Reconstruction.
IJCAI - vol. 2, pp. 1022 - 1024

Chapman, D. (1987)
Planning for Conjunctive Goals.
AI Magazine - vol. 32, pp. 333 - 377

Chapman, D. (1989)
Penguins Can Make Cake.
AI Magazine - vol. 10, no. 4, pp. 45 - 50

Collins, Birnbaum, Krulwich, & Freed (1991a)
Model-Based Integration of Planning and Learning.
SIGART Bulletin - vol. 2, no. 4, pp. 56 - 60

Collins, Birnbaum, Krulwich, & Freed (1991b)
Plan Debugging in an Intentional System.
12th IJCAI, Sydney - vol. 1, pp. 353 - 358

Collins, Birnbaum, Krulwich, & Freed (1992)
The Role Of Self-Models In Learning To Plan.
In Machine Learning: Induction & Discovery, Meyrowitz (ed.), Kluwer Publishing

Correa & Coehlo (1992)
Around The Architectural Agent Approach To Model Conversations.
TR. - INSEC, Lisbon, Portugal

Dean, Firby, & Miller (1988)
Hierarchical Planning Involving Deadlines, Travel Time, and Resources.
Computational Intelligence - vol. 4, pp. 381 - 398

Dennett, D.C. (1994)
The Practical Requirements For Making A Conscious Robot.
Presented at the Royal Society meeting, AI and the Mind.

Fahlman, S.E. (1973)
A Planning System For Robot Construction Tasks.
TR.: #283, MIT

Fehling, Altman, & Wilber (1989)
The Heuristic Control Virtual Machine: An Implementation of the Schemer Computational Model of Reflective, Real-Time Problem-Solving.
Blackboard Architectures & Applications - pp. 191 - 217

Fikes & Nilsson (1971)
STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving.
AI Magazine - vol. 2, pp. 189 - 208

Fikes & Nilsson (1993)
STRIPS, a retrospective.
AI Magazine - vol. 59, pp 227 - 232

Firby, R.J. (1987)
An Investigation Into Reactive Planning In Complex Domains.
AAAI - pp. 202 - 206

Garvey, Cornelius, & C. Hayes-Roth (1987)
Computational Costs Versus Benefits Of Control Reasoning.
AAAI - vol. 7, pp. 110 - 115

Gat, E. (1991)
Integrating Reaction And Planning In A Heterogeneous Asynchronous Architecture For Mobile Robot Navigation.
SIGART Bulletin - vol. 2, no. 4, pp. 70 - 74

Georgeff & Ingrand (1989)
Decision-Making In An Embedded Reasoning System.
In Proceedings: 11th IJCAI.

Georgeff & Lansky (1987)
Reactive Reasoning And Planning.
In Proceedings: 6th National Conference on AI. Seattle - pp. 677 - 682

Ginsberg, M.L. (1989)
Universal Planning: An (Almost) Universally Bad Idea.
AI Magazine - vol. 10, no. 4, pp. 40 - 44

Green, C. (1969)
Application of Theorem Proving To Problem Solving.
IJCAI - pp. 741 - 747

Hammond, Converse, & Martin (19xx)
Integrating Planning and Acting In A Case-Based Framework.
Cognitive Modelling - pp. 292 - 297

Hanks & Firby (1990)
Issues and Architectures for Planning and Execution.
In Proceedings: Innovative Approaches to Planning, Scheduling and Control Workshop. San Diego.

Hayes-Roth, B. (1990a)
Dynamic Control Planning In Intelligent Agents.
Workshop: Planning In Uncertain, Unpredictable Or Changing Environments, Stanford University.

Hayes-Roth, B. (1990b)
Architectural Foundations For Real-Time Performance In Intelligent Agents.
The Journal Of Real-Time Systems - vol. 2, pp. 99 - 125

Hayes-Roth, B. (1991a)
An Integrated Architecture For Intelligent Agents.
SIGART Bulletin - vol. 2, no. 4, pp. 79 - 81

Hayes-Roth, B. (1991b)
Evaluation Of Integrated Agent Architectures.
SIGART Bulletin - vol. 2, no. 4, pp. 82 - 84

Hayes-Roth, B. (1993b)
Intelligent Control.
Artificial Intelligence - vol. 59, pp. 213 - 320

Horvitz, E.J. (1989)
Reasoning About Beliefs And Actions Under Computational Resource Constraints.
Uncertainty In AI - no. 3, pp. 301 - 324

Izard, C.E. (1993)
Four Systems For Emotion Activation: Cognitive & Non-cognitive Processes.
Psychological Review - vol. 100, no. 1, pp. 68 - 90

Johnson-Laird, P.N. (1989)
The Computer and The Mind.
Fontana Press.

Kaelbling & Rosenschein (1990)
Action And Planning In Embedded Agents.
Robotics & Autonomous Systems - vol. 6, pp. 35 - 48

Kaelbling, L.P. (1986)
An Architecture For Intelligent Reactive Systems.
In Proceedings: Reasoning About Actions And Plans Workshop. San Mateo, Ca. - pp. 395 - 410

Kaelbling, L.P. (1991)
A Situated-Automata Approach To The Design Of Embedded Agents.
SIGART Bulletin - vol. 2, no. 4, pp. 85 - 88

Korf, R.E. (1987)
Planning as Search: A Quantitative Approach.
AI Magazine - vol. 33, pp. 65 - 88

Krulwich, B. (1991a)
Determining What to Learn in a Multi-component Planning System.
In Proceedings: 13th Annual Conference of The Cognitive Science Society, Chicago - pp. 102 - 107

Krulwich, B. (1991b)
Learning From Deliberated Reactivity.
In Proceedings: 8th International Workshop on Machine Learning, Evanston, IL. - pp. 318 - 322

Krulwich, Birnbaum, & Collins (19xx)
Learning Several Lessons From One Experience.
Unknown Source - pp. 242 - 247

Kuhl & Kraska (1989)
Self-Regulation And Meta-motivation: Computational Mechanisms, Development, And Assessment.
In Kanfer, Ackerman & Cudek (eds.), Abilities, Motivation, and Methodology: The Minnesota Symposium - pp. 343 - 374

Kuhl, J. (1992)
A Theory Of Self-Regulation: Action Versus State Orientation, Self-Discrimination, And Some Applications.
Applied Psychology: An International Review - 41 (2), pp. 98 - 129

Laird, Hucka, Huffman, Rosenbloom (1991)
An Analysis of Soar as an Integrated Architecture.
SIGART Bulletin - vol. 2, no. 4, pp. 98 - 103

Laird, Newell, & Rosenbloom (1987)
SOAR: An Architecture for General Intelligence.
AI Magazine - vol. 33, pp. 1 - 64

Lesser, Pavlin, & Durfee (1988)
Approximate Processing In Real-Time Problem Solving.
Artificial Intelligence - vol. 9, no. 1

Loyall & Bates (1991)
Hap: A Reactive, Adaptive Architecture For Agents.
TR.: #CMU-CS-91-147, Carnegie Mellon University.

Lyons & Hendriks (1991)
Reactive Planning.
TR. - Autonomous Systems Department, Philips Laboratories, North American Philips Co., New York.

Maes, P. (1989)
The Dynamics Of Action Selection.
In Proceedings: 11th IJCAI, Detroit, MI.

Maes, P. (1990a)
Guest Editorial: Designing Autonomous Agents.
Robotics & Autonomous Systems - vol. 6, pp. 1 - 2

Maes, P. (1990b)
Situated Agents Can Have Goals.
Robotics & Autonomous Systems - vol. 6, pp. 49 - 70

Maes, P. (1991)
The Agent Network Architecture (ANA).
SIGART Bulletin - vol. 2, no. 4, pp. 115 - 120

McAllester & Rosenblitt (1991)
Systematic Non-linear Planning.
Mathematical Foundations In Planning - pp. 634 - 639

McCarthy & Hayes (1969)
Some Philosophical Problems From The Standpoint of AI.
Machine Intelligence - no. 4, pp. 463 - 402

Miller, Firby, & Dean (1985)
Deadlines, Travel Time, And Robot Problem Solving.
In Proceedings: IJCAI - vol. 2

Minton, S. (1985)
Selectively Generalising Plans For Problem Solving.
Computer Science Department, Carnegie-Mellon University

Morgenstern & Stein (1990)
Why Things Go Wrong: A Formal Theory Of Causal Reasoning.
From Allen, Hendler & Tate (eds.) 1990. Readings In Planning. Morgan Kaufmann Publishers - pp. 641 - 646

Morignot & Hayes-Roth (1994)
Why Does An Agent Act?
TR. - Knowledge Systems Laboratory, Stanford University.

Morignot, P. (1994)
Embedded Planning.
TR.: #KSL 94-23, Computer Science Department, Stanford University.

Nelson, T.D. (1993)
The Hierarchical Organisation Of Behaviour: A Useful Feedback Model Of Self-Regulation.
Current Directions In Psychological Science - vol. 2, no. 4, pp. 122 - 126

Newell & Simon (1963)
GPS, A Program That Simulates Human Thought.
Simulation of Cognitive Processes - pp. 279 - 293

Nilsson, N.J. (1988)
Action Networks.
In Weber, Tenenberg & Allen (eds.), From Formal Systems To Practical Systems - pp. 21 - 52

Perry, J. (1985)
Self-Knowledge and Self-Representation.
IJCAI - vol. 2, pp. 1239 - 1242

Peterson, D. (1992)
Re-representation or No! No! No!
Cognitive Science Seminar Notes, Birmingham University.

Pfeifer, R. (1992)
The New Age of The Fungus Eater: Comments on AI and Emotion.
Extended version of a talk given at AISB-91

Poli & Brayshaw (1995)
A Hybrid Trainable Rule-Based System.
Submitted to AISB95

Pryor & Collins (1991)
Information Gathering As A Planning Task.
TR. - The Institute For The Learning Science, Northwestern University, Evanston IL.

Pryor & Collins (1993a)
Cassandra: Planning For Contingencies.
TR.: #41, The Institute For The Learning Sciences, Northwestern University, Evanston IL.

Pryor & Collins (1993b)
Cassandra: Planning With Contingencies.
In Proceedings: 13th IJCAI.

Rao & Georgeff (1991)
Modelling Rational Agents Within A BDI-Architecture.
Technical Note: #14, The Australian Artificial Intelligence Institute.

Rao & Georgeff (1992)
An Abstract Architecture For Rational Agents.
In Proceedings: 3rd International Conference On Knowledge, Representation, And Reasoning. Boston.

Sacerdoti, E.D. (1977)
A Structure for Plans and Behaviour.
N.J. Nilsson (ed.) Artificial Intelligence Series, Elsevier Computer Science Library.

Schoppers, M.J. (1989)
In Defence Of Reaction Plans As Caches.
AI Magazine - vol. 10, no. 4, pp. 51 - 60

Shing, E. (1993)
Computational Mechanisms For Constraining Learning.
In Proceedings: WAUME93.

Shing, E. (1994)
PhD. Thesis In Preparation.
Cognitive Science Department. Birmingham University.

Shoham & McDermott (1988)
Problems in Formal Temporal Reasoning.
AI Magazine - vol. 36, pp. 49 - 61

Simmons, R. (1991)
Co-ordinating Planning, Perception, And Action For Mobile Robots.
SIGART Bulletin - vol. 2, no. 4, pp. 156 - 159

Simon, H.A. (1967)
Motivational and Emotional Controls of Cognition.
In Models of Thought (1979).

Sloman & Croucher (1981)
Why Robots Will Have Emotions.
In Proceedings: IJCAI, Vancouver

Sloman & Croucher (19xx)
You Don't Need A Soft Skin To Have A Warm Heart: Towards A Computational Analysis Of Motives And Emotions.
TR. Cognitive Science Department, Birmingham University.

Sloman & Humphreys (1992)
The Attention & Affect Project.
Appendix To JCI proposal - Cognitive Science Department, Birmingham University.

Sloman, A. (1987)
Motives, Mechanisms And Emotions.
Emotion And Cognition - 1-3, pp. 217 - 234

Sloman, A. (1991a)
Developing Concepts Of Consciousness.
Behaviour And Brain Sciences - 14-4, pp. 694 - 695

Sloman, A. (1991b)
Why Consciousness Is Not Worth Talking About (Yet).
In Proceedings: International Conference In Cognitive Science, San Sebastian.

Sloman, A. (1991c)
Prolegomena To A Theory Of Communication And Affect.
In Ortony, Slack & Stock (eds.), Communication from an AI perspective: Theoretical and applied issues - pp. 229 - 260

Sloman, A. (1992b)
Silicon Souls: How To Design A Functioning Mind.
Inaugural Lecture, Birmingham University.

Sloman, A. (1992c)
The Emperor's Real Mind: Review Of Roger Penrose's The Emperor's New Mind.
Artificial Intelligence - vol. 56, pp. 355 - 396

Sloman, A. (1992d)
The Role Of Representations In Intelligent Systems.
Seminar Notes. Cognitive Science Department, Birmingham University.

Sloman, A. (1993a)
Prospects For AI As The General Science Of Intelligence.
In Proceedings: AISB93, Oxford IOS Press.

Sloman, A. (1993b)
Why We Need To Study Architectures.
In Proceedings: WAUME93.

Sloman, A. (1994a)
Explorations In Design Space.
In Proceedings: 11th European Conference On Artificial Intelligence.

Sloman, A. (1994b)
Semantics In An Intelligent Control System.
In Proceedings: British Academy And Royal Society Conference.

Sloman, A. (1994c)
Towards A General Theory Of Representations.
TR.: #CSRP-94-13, School Of Computer Science, Birmingham University.

Sloman, A. (1994d)
Autonomous Motive-Driven Agents With Broad Architectures.
SERC Research Grant Proposal, School Of Computer Science, Birmingham University.

Sloman, A. (1994e)
Management and Meta-Management.
Internal Report Based on Beaudoin's (1994) Thesis

Sussman, G.J. (1975)
A Computer Model of Skill Acquisition.
Artificial Intelligence Series, Elsevier Publishing.

Thornton, C.J. (1992)
Techniques in Computational Learning.
Publishers: Chapman & Hall Computing

Torras, C. (1993)
From Geometric Motion Planning To Neural Motor Control In Robotics.
AICOM - vol. 6, no. 1, pp. 3 - 17

Wooldridge & Jennings (1994)
Intelligent Agents: Theory & Practice.
Submitted to Knowledge Engineering Review

Wright, I. (1994a)
Architecture as Interpretive Ground: Towards a Design-Based Analysis of Emotional Episodes.
Cognitive Science Internal Report, Birmingham University

Wright, I. (1994b)
An Emotional Agent.
Thesis Report 3, Cognitive Science Department, Birmingham University