

Modular Models of Intelligence - Review, Limitations and Prospects*

Amitabha Mukerjee
Center for Robotics
I.I.T. Kanpur, India 208016
(amit@iitk.ac.in)

Amol Dattatraya Mali
Department of Elec. Engg & Computer Science
University of Wisconsin-Milwaukee
Milwaukee, WI 53211, USA
Phone: 1-414-906-8942, Fax: 1-414-229-2769
(mali@miller.cs.uwm.edu)

Abstract

AI applications are increasingly moving to modular agents, i.e. systems that independently handle parts of the problem based on small locally stored information (Grosz & Davis 1994), (Russell & Norvig 1995). Many such agents minimize inter-agent communication by relying on changes in the environment as their cue for action. Some early successes of this model, especially in robotics (“reactive agents”), have led to a debate over this class of models as a whole. One of the issues on which attention has been drawn is that of conflicts between such agents. In this work we investigate a cyclic conflict that results in infinite looping between agents and has a severe debilitating effect on performance. We present some new results in the debate, and compare this problem with similar cyclicity observed in planning systems, meta-level planners, distributed agent models and hybrid reactive models. The main results of this work are:

- (a). The likelihood of such cycles developing increases as the behavior sets become more useful.

*Appears in Artificial Intelligence Review Journal, Vol. 17, No. 1, March 2002, pp. 39-64

- (b). Control methods for avoiding cycles such as prioritization are unreliable, and
- (c). Behavior refinement methods that reliably avoid these conflicts (either by refining the stimulus, or by weakening the action) lead to weaker functionality.

Finally, we show how attempts to introduce learning into the behavior modules will also increase the likelihood of cycles.

Keywords : Intelligent agents, Modularity, Reactivity, Behavior, Cyclic

Conflicts

1 Introduction

From web browsers to intelligent houses to devices that “learn about different users’ behavior”, agents are rapidly capturing the mainstream processes in AI. According to the AAAI position paper (Grosz & Davis 1994) the charter for AI in the twenty-first century is to build such agents for use in simulation, human assistance, robot interaction etc. In general, agent systems attempt to reduce or eliminate the centralized shared memory, relying instead on parameter passing and communication between individual agents.

Part of the allure of the agent model is the assumption, sometimes known as the *modularity hypothesis* that agents are modular, that more complex systems can be built up simply by putting a number of agents which are each tuned to solve some part of the problem. In this paper, we seek to investigate this modularity assumption in the light of one type of conflict that can disrupt its functioning severely. In addition to the well-known case of resource or goal conflicts, a temporal-cycle type of conflict may arise due to the temporal sequences in agent actions. Say an action of agent A triggers agent B. Agent B’s action may trigger C, which, unfortunately may trigger A again. This results in an unending chain, which we call a *cyclic*

conflict (Figure 1). Some existing agent systems have already exhibited such behavior and many system builders have experienced it themselves.

In this work we investigate this type of conflict and study its effects on the modularity hypothesis. We also propose some performance measures and show how these cycle removal procedures come at a cost to the performance. As an example domain, we study the subsumption type of agents, a specific class of agents as they have been implemented in robotics. However, the results hold implications for all areas involving agent architectures, and we review the effects of these results on several architectures of autonomous agents and the AI paradigms used in these implementations.

This paper is organized as follows. In the rest of this introduction section, we describe different types of agents and review the debate on the utility of reactive systems. In section 2, we describe how reactive systems differ from traditional systems and report some implementations of reactive agency, along with the problem of infinite cycles in some of the implementations. Section 3 constructs a model for the notion of behavior and develops some metrics to compare different behavior sets. In section 4, we discuss the current approaches to resolving the cyclic conflicts and show why these approaches do not guarantee that the cycles will be eliminated. In section 5, we propose approaches that reliably remove the cyclic conflicts and show

how they adversely affect the performance of a behavior set. In section 6, we review several architectures of autonomous agency and discuss how they handle cycle removal and discuss some of the important fundamental problems that the architectures face. In section 7, we show how learning agents may be more likely to exhibit cycles. Conclusions are reported in section 8.

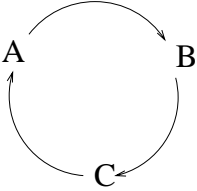


Figure 1: *Cyclic Conflict*. Agent A’s action triggers Agent B. B triggers C, which calls for action by A again, endlessly.

1.1 Reactive Agents

There are many different systems operating under the rubric of “agent” today. One way of looking at these is to consider the degree of autonomy of each agent (to what extent are its decisions affected by other agents or *boss* agents), and the degree of situatedness (to what extent the actions are a result of direct environmental interaction vs planned or deliberative actions). A conceptual categorization of some of the current agent work can be shown as in figure 2, in which the class of *reactive* agents like the robots of Brooks (Brooks 1986) may be thought of as those that are largely autonomous

with actions that are triggered by the stimulus received. Furthermore, these agents are characterized by a simpler mode of communication, where there is no global memory, and since there is a cost to communicating information, inter-agent messages are also minimal, limited mostly to inhibitive signals.

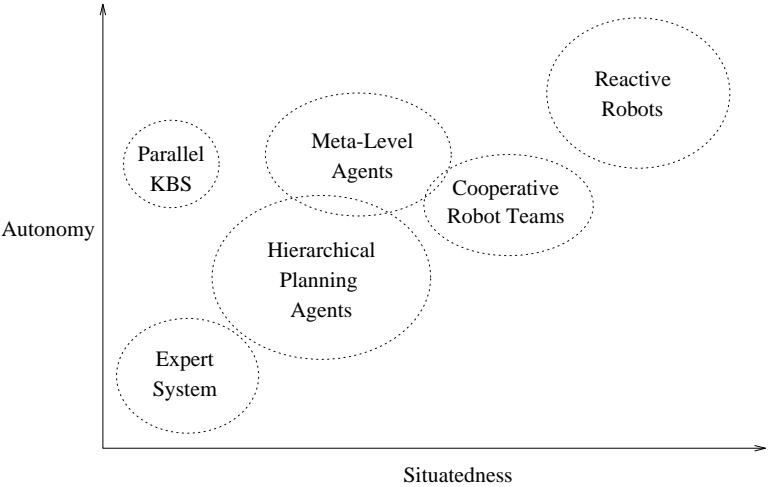


Figure 2: *Dimensionalities of Agent models.* Reactive systems are highly situated and autonomous.

Many biological behaviors, even some fairly complex ones, can be viewed in reactive terms, as a set of stimulus-response systems. Only in mammals and other highly evolved species is there evidence of non-reactive behavior, (often called *deliberative*), based on a causal and functional representation of the world. In contrast with biology, the early history of AI has been almost entirely deliberative – representing the world and its rules has taken precedence over attempts to model the sensorimotor interactions. The current

growth in reactive systems has engendered debate over many of the fundamental aspects of AI, such as the need for embodied vs cerebral systems.

1.2 The evolution of agents - the deliberative-reactive debate

In order to illuminate the debate between reactive and deliberative approaches to intelligence, it is perhaps useful to take a moment to consider the reasons for the dominance of deliberative processes in early AI. In particular, it was assumed that sensory processes “could be done”, and the hard problems were those of reasoning based on symbolified sensory data. This may have been because

- (a). Sensory processes are intrinsically difficult, i.e. the complexity of the sensing process is so much beyond our powers of modeling it. Or it may be that

- (b). Symbolic processing is biologically newer, and therefore closer to the surface of our conscious intelligence. The naive meaning of the term intelligence is associated with prowess in solving symbolic puzzles and playing chess. There is no intelligence in sensing, actuation or mundane tasks such as navigating between obstacles and recognizing faces.

The attraction of symbolic activity is that we are still not masters of it: it takes many years to learn that 7 times 3 is 21. The tools of this knowledge are explicit, and ready for us to wield, whereas the mechanism of speech recognition or motor control are ingrained in our biology, and how we learn these in first years of life is a book that yields its secrets but grudgingly.

However, voices were heard that began to raise other points of view:

- Synthetic biologists (Braitenberg, and Grey-Walter), were talking about building simple “creatures” and actually taking them in a real environment according to the laws of evolution. These creatures had no representations, but only a collection of stimulus-response behaviors.
- Roboticists argued for more sensory relevance Some went further (Moravec 1984), to claim that intelligence could not possibly be pre-programmed through huge symbolic knowledge bases, but must evolve in a synthetic manner through actual interaction with a rich and responsive environment. Mobility, it was argued, has often been the impetus for greater intelligence, as when the apes acquired extra degrees of freedom through the two upper limbs.
- Philosophers, like Searle, or Dennett, challenged the effectiveness and philosophical validity of pure symbol systems. Dennett suggested cre-

ating a complete artificial creature, albeit simpler than humans, such as cockroaches or the iguana.

At the same time, cognitive scientists, animal ethologists, and neuroscientists, were working on understanding biological nervous systems. One of the ideas from this area was that of stimulus-response type of behavior systems. Intelligent or not, the majority of biologically successful creatures seemed to be little more than a bunch of stimulus-response behaviors. These ideas formed the core of the reactive robot paradigm. A related idea from psychology was that of situated action, which proposed that intelligence is not encoded in symbols but is rather better understood as a model of interactions between the agent and its environment and other agents (Vera & Simon 1993).

In planning also, it was found that simple linear plans were running into infinite cycles even on very small planning tasks. One of the solutions to emerge from this debacle was reactive planners (Georgeff & Lansky 1990), which modify their plans in response to changes in the environment. Different plan modules are pre-defined depending on possible contingencies that may arise while attempting to achieve a certain goal. Thus the actual behavior may be modulated based on sensory cues.

2 Reactive Agents in Robotics

Reactive systems became popular in robotics with the work of Rodney Brooks (Brooks 1986), who challenged the deliberative paradigm in AI by building stimulus-response based robots, also known as behavior-based robots. A typical reactive robot is a collection of several independent task-achieving modules, with a simple distributed control mechanism. Each behavior mediates directly with the external world and is in a parallel control structure, as opposed to the traditional serial structure where interaction with the world is processed serially through sensors, reasoners, planners, actuators, etc. This paradigm eventually came to be known as *subsumption* since sometimes some behaviors may subsume or deactivate other, “lower” behaviors (Figure 3).



Figure 3: *Traditional vs. Reactive*. Reactive robots are based on behaviors that mediate directly with the outside world and communicate very little between themselves.

Within this basic structure, a number of models evolved using the stimulus-

response paradigm. One great advantage is the ease of modeling and debugging each *behavior module* at its own task, (e.g. “avoid-obstacles”) as opposed to the larger and more integrated centralized controllers. Each module is fully debugged before other modules are added, with suitable inhibitive interconnections where conflicting. Early impressive results were obtained using this strategy in a can collection robot (Connell 1990), navigation of mobile robot (Arkin 1992), a prototype airplane controller (Hartley & Pipitone 1991), a coaching system based on reactive coaches (Masthoff & Hoe 1995), box-pushing robots (Mahadevan & Connell 1992), etc.. One of the characteristics of these models is the minimal communication between the parallel controllers, in effect relying on the environment to serve as an external memory for the process. That the reactive model has had a deep impression on Artificial Intelligence is clear from the fact that several journal issues in the '90s have been devoted to debating this approach (Artificial Intelligence v.47, 1991, Robotics & Automation, v.6:1, 1990, Cognitive Science v.17, 1993, Artificial Intelligence v.73 1995). New journals “Autonomous Robots” and “Autonomous Agents and Multi-agent Systems” were launched to publish papers on autonomous agents, many of which are entirely or partially behavior-based.

In the debate on reactive models, traditional AI researchers have argued

that control cannot serve as a complete substitute for representation (Kirsh 1991). Others have attacked the parallel psychological model of situated action, showing that the reactive models were actually symbol systems, or that they required significant internal representations which were, or could be, represented using symbolic systems (Vera & Simon 1993). One of the motivations behind moving from centralized planners to reactive approaches was the computational hardness of planning which is a combinatorial problem. Muscettola et al (Muscettola et al 1998) argue that reactive approaches are not useful for remote agents and in fact the recent advances in efficiently solving combinatorially hard problems make hard search problems like planning solvable in real time. For example, truth assignments that satisfy a formula in propositional logic that contains 10,000 variables and 1 million constraints can be found in a few seconds. Traditional planners were able to synthesize plans containing only 5 to 7 steps in real time, each step being mapped to an action. However, recent planners can synthesize plans containing 100 steps in real time. Despite this broad debate, no metrics have been proposed for comparing the overall functionality of different reactive agent systems.

Another aspect - the “modularity” claim that other modules can be added to them “later” – has gone largely uninvestigated, though a number

of skeptics have challenged this claim. For example, questions of implementation such as which behavior modules should be at the lower levels in the hierarchy, or which ones should one debug first - have not been answered. Mitchell (Mitchell 1990) reports that though his robot becomes increasingly reactive (since it stores successful plans constructed by a planner for use in future, by just retrieval and execution), its decisions do not become increasingly correct. Kube & Zhang (Kube & Zhang 1997) claim that in simple tasks such as foraging and box-pushing, reactive behaviors carefully chosen by hand did work well, however it is not clear how the behavior-based approach can be extended to accomplish tasks requiring several ordered steps. These are some of the aspects that are addressed in the current work, in the context of analyzing temporal cycles that may develop in reactive systems.

2.1 Temporal Cycles in Reactive Agents

An instance of a temporal cycle is recorded by Connell where a can collecting robot attempts to re-pick the can it has just deposited (Figure 4); this conflict was detected only after a full implementation (Connell 1990). Laird & Rosenbloom (Laird & Rosenbloom 1990) argue that a disadvantage of reactive behaviors is that a very limited piece of information triggers the execution of motor actions, thus the faster responses come at the cost of

controllability. Cyclical wandering and cyclic conflict of going back and forth between two obstacles is widely experienced in potential field based systems (Anderson & Donath 1990). Cyclic behavior in multi-robot systems for box pushing has been reported by (Kube & Zhang 1997). Such cycles are potentially very costly in real systems. Even when they are detected, it is not clear how to fix these “bugs”. We feel that this issue actually has a major ramification on the issue of modularity in agents, and this is the aspect that we set out to explore.

Note that the conflicts we are addressing are not control conflicts but temporal sequence conflicts for which it is necessary to define a temporal structure. This is often sequential since one behavior usually provides the stimulus for another, so that the behaviors are executed in sequence. This temporal sequentiality allows us to use some of the mechanisms of situation calculus to analyze this problem. We show that cycles occurring in this temporal sequence can be eliminated with certainty only by modifying the behaviors, either specializing the stimulus or restricting the action of a behavior. One of the key results of the paper is that any such modification reduces the utility of the behavior structure. Another result is that attempts at introducing robot learning, which will result in more useful behaviors, will also increase the likelihood of cyclic conflicts occurring in a behavior set.

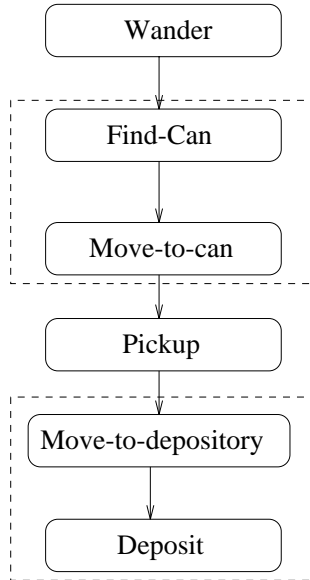


Figure 4: *Subsumption structure the robot HERBERT* . After Deposit, sometimes the robot attempts to Pickup the same can.

3 What Is a Behavior?

We consider a system of agents in this work. All changes to the world are caused by the action of one of these agents, which are interchangeably called behaviors. Nobody changes the world except these agents.

The essential notion of a behavior is a mapping from a stimulus to a consequence. In this work we followed the psychologists and adopted a *3-tuple* model of behavior: stimulus, action, consequence – behavior β_i is denoted by $\langle s_i, a_i, c_i \rangle$. In the examples and proofs that follow, the stimulus s and the consequence c are defined using First Order predicates as in the sub-

sumption architecture: these are constructed from a set of literals with the conjunction, disjunction and negation operators. This leads to a structure much like the standard situation calculus models.

Over time, any behavior has two states - *waiting-for-stimuli*, which may be called the dormant state, and the actual action, which may be called the dominant state. We define the *dominant period* of a behavior as that period when the behavior is active. This term is analogous to the duration of an event. It is read as “precedes”: module β_1 precedes module β_2 *iff* there exist time instants $\theta_1, \theta_2, \theta_3, \theta_1 < \theta_2 < \theta_3$ such that at θ_1 , β_1 is dominant but β_2 is not and at θ_2 , β_1 and β_2 are both dominant and at θ_3 , β_2 is dominant but β_1 is not.

3.1 Behavior Chain

We define a behavior chain and the corresponding task as follows.

- **Behavior Chain:** a temporal sequence of behavior modules $\{\beta_1 : \beta_2 : \beta_3 : \dots : \beta_n\}$. Here the action of the earlier module changes the situation in such a way that the newly changed part of the situation is in turn a stimulus for the next module in the sequence. Behaviors can be chained together to generate action-streams such as trajectories for a robot, question sequences in coaching, etc.

• **Task:** A task is defined as a transition from an initial state of the world to a new state, achieved through a temporal chain of behaviors. Note that the chain of behaviors by itself is executable *only* when the world is in certain configurations. However to perform the task from these configurations, this chain of behaviors would need to be executed.

In particular we are interested in defining a measure of the number of tasks that are potentially fulfillable. These correspond to all possible temporal chains of behaviors. Where this is achieved by executing behaviors in a temporal sequence, tasks can be enumerated by the total number of chain fragments that are possible, such that these chains carry out different state transitions. Thus the chain $\{\beta_1 : \beta_2 : \beta_3\}$ represents a task that is different from $\{\beta_1 : \beta_2\}$.

At this point, we need to address an issue relating to all situation calculus models: the finiteness of the universe. Typically, the set of predicates required for any set of behaviors is finite. Of this set, only a few will be affected by the actions in a behavior chain; the rest constitute the static *universe*, which in the rest of this discourse, is considered to be a finite set of entities U . When we write $(c_i \Rightarrow s_{i+1})$, c_i contains the entire finite universe U . For compactness in the explicit statements below, we do not list all predicates from U and limit ourselves to those whose change affects the

firing of various stimuli in the chain.

What we mean by the finite universal state can be clarified by an example. Let the state of the Universe be $(X \wedge Y \wedge Z)$ and $s_2 = (X \wedge A)$. Let us say that the execution of β_1 makes A true. However c_1 is assumed to contain X which is a part of the universe. Then β_1 leads to β_2 and $(c_1 \Rightarrow s_2)$. Thus when we say that $\beta_1 : \beta_2$ we mean that a part of s_2 was true in the Universe and execution of β_1 causes the rest of s_2 to come true. This is another version of the frame problem, which arises in any model of such temporal sequencing, such as the add and delete list model used widely in planning and knowledge representation. Including the state of the universe allows us, in finite domains, to proceed with the core of our argument. Also, the model chosen can be reformulated in terms of the successor state axioms that Reiter (Reiter 1991) has shown to be derivable from the positive and negative effect axiomatic structure.

We define a *behavior set* B as a set of distinct behavior modules, (i.e. no two modules have the same stimulus and consequence). A temporal chain of behaviors C is said to be composable from B (written as $C \triangleleft B$), if the elements of C are also elements of B .

3.2 Power vs Modularity of Behaviors

To compare different behavior sets, we define some metrics that relate to the effectiveness of a behavior set to deliver the desired level of functionality. Intuitively, it is desirable that a behavior be capable of being used in a wider range of situations, without weakening the effect of the action. This is captured in the $\langle stimulus, consequence \rangle$ formalism by the notion of power defined next.

- **Power**: A behavior $(\beta := \langle s, a, c \rangle)$ is more powerful than $(\beta' := \langle s', a', c' \rangle)$ iff $((s' \Rightarrow s) \wedge (c \Rightarrow c'))$. In other words, it can be triggered at least as frequently as a less powerful behavior and results in a state that subsumes the older one (figure 5). A behavior set B is more powerful than the behavior set B' if B' can be obtained from B by replacing some module $\beta \in B$ by less powerful module β' .

The following metric attempts to evaluate an answer to the question - how economical is a behavior set?

- **Span**: Behavior set B **spans** task space τ if and only if $\{\forall (t \in \tau) (\exists (C \triangleleft B) \text{fulfills}(C, t))\}$.

- **Greatest Potential Task Space** $\tau_G(B)$: Is the set of all tasks that can be fulfilled by behaviors in B , i.e. the largest task space that is spanned by

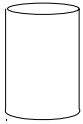
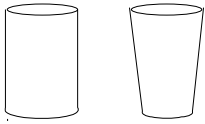
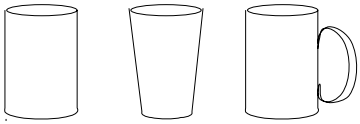
Stimuli for behaviors of increasing power		
(A)	(B)	(C)
		
$graspable(x) \wedge can(x)$	$graspable(x) \wedge (can(x) \vee cup(x))$	$graspable(x)$

Figure 5: *Power*: Behavior B can pick up cups as well as cans (has a more general stimulus). C is even more general. The notion of power is used to capture this, provided all three behaviors have the same consequence.

the behavior set B .

- **Modularity**: A behavior set is more modular if the different modules in that set are more independent, in the sense of minimal interference between modules. One measure of interference in a behavior set is the incidence of cyclic behavior. We therefore define the modularity of behavior set B as the inverse of the likelihood that cyclic conflicts will arise in B .

4 Detection of Temporal Cycles

In the broad sense of the word conflict, any behavior chain leading to non-fulfillment of the desired objectives can be said to contain a conflict. Let a chain $C = \{\beta_1 : \beta_2 : \dots : \beta_n\}$ be the behavior sequence that achieves a desirable outcome. There are three types of conflicts that can cause the

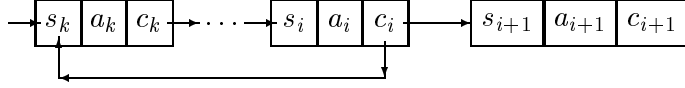


Figure 6: *Cycle in a temporal chain of behaviors.*

chain C not to be executed. In each case, some subsequence $\beta_i : \beta_{i+1}$ must be broken. (a) **Extraneous behavior Conflict**: $\beta_i : \beta', \beta' \notin C$. Here the conflict is with a module outside the current chain; it can be inhibited. (b) **Cyclic Conflict**: $\beta_i : \beta_k, \beta_k \in C, k \leq i$. This is the type of conflict that we are focusing on (Figure 6). (c) **Skipping Conflict**: $\beta_i : \beta_k, \beta_k \in C, k > (i+1)$. This type of conflict can be treated in a manner analogous to extraneous behavior conflicts.

Terminated Cycles

However, not all cycles result in a conflict, e.g. there may be repeated tasks, or even life-cycle-maintenance-tasks that are repeated many times. An observer with the capability for identifying cycles (similar to plan recognizers) can learn the “good” cycles over a number of successful runs (cycles in these runs are flagged as task-fulfilling ones), so that in later runs, all other cycles

would be flagged. Such an observer may be implemented as a meta-level for the purpose of detecting cycles in the temporal sequence of actions. Next we investigate the question of removing a cyclic conflict after it is detected.

4.1 Prioritization

A popular method, suggested originally as part of the subsumption architecture, and also widely used in biological systems, is that of **inhibition**. In general, these methods attempt to avoid conflicts by changing the priority of various modules. Let $\alpha : \gamma$ be the desirable sequence and $\alpha : \beta$ be the disruptive sequence. Then in *inhibition* or *suppression* the action of β may take place, but only after γ is no longer dominant. Here the chain $\alpha : \beta$ has the module γ inserted, so $\alpha : \gamma : \beta$ may still occur, if the stimulus for β is not affected by γ . *Delayed action* is a special case of inhibition, where the inhibiting link remains effective for some time t_{delay} even after the inhibiting module is no longer dominant. Connell uses the term *retriggerable monostable* to capture this sense (Connell 1990). In his case, the robot pauses after depositing the can before it starts looking for a new one. In the meanwhile, it is in *wander* mode, and it is expected, though not assuredly, that the robot will move away and not see the freshly deposited can. These mechanisms are not guaranteed to remove the stimulus of β_k , hence β_k may become active

after the dominant period of the suppressing module (or the delay) is over. Thus within the scope of the prioritization schemes discussed here, it is not guaranteed that cyclic conflicts will be avoided.

5 Behavior Refinement

Here our objective is to break the $\beta_i : \beta_k$ link in the cyclic conflict without disturbing the $\beta_i : \beta_{i+1}$ or $\beta_{k-1} : \beta_k$ triggerings which are essential to the successful execution of the chain. Thus we seek to modify the behaviors such that $(c_i \Rightarrow s_{i+1})$ and $(c_{k-1} \Rightarrow s_k)$ will be maintained whereas $(c_i \Rightarrow s_k)$ would be negated. We develop two methods for achieving this: in stimulus specialization, s_k is specialized, and in consequence generalization, c_i is generalized.

5.1 Stimulus Specialization

Let us consider the conflict in picking up the soda cans, where the freshly deposited can is picked up. If we were to add the condition “not-deposited-just-now (\mathbf{x})” to the stimulus predicate for pickup, then this might be achieved. Here the stimulus for β_k becomes more specialized. However, in doing this, one must be careful so as not to disturb the rest of the chain, i.e. $(c_{k-1} \Rightarrow s_k)$ should still hold but $(c_i \Rightarrow s_k)$ must be broken. Clearly this will not be possible where $(c_i \Rightarrow c_{k-1})$, then any changes we make in s_k such that $\neg(c_i \Rightarrow s_k)$

will also result in $\neg(c_{k-1} \Rightarrow s_k)$. Thus stimulus specialization can be used only if $(c_i \Rightarrow c_{k-1})$ is not true. Conflict arises when, for $k \leq i$, $(c_{k-1} \Rightarrow s_k)$, $(c_i \Rightarrow s_{i+1})$ and $(c_i \Rightarrow s_k)$.

If $(c_i \Rightarrow c_{k-1})$ does not hold and we assume that the stimuli and consequences are expressed in purely conjunctive form, then there must be some literal p in c_{k-1} which does not occur in c_i . ANDing s_k with p gives new s_k that is not implied by c_i .

In pick-move-drop-pick cycle, pick is the behavior β_k and drop behavior is β_i . $s_k = (\text{can}(x) \wedge \text{graspable}(x))$ and $c_i = (\text{can}(x) \wedge \text{graspable}(x) \wedge \text{at_repository}(x))$

If we use stimulus specialization, $\neg \text{at_repository}(x)$ serves as literal p .

5.2 Consequence Generalization

We can also break the cycle by generalizing c_i to c' so that $(c_i \Rightarrow c')$, but $\neg(c' \Rightarrow c_i)$. For example, we can modify the action of the module *drop* so that while dropping the can the robot crushes it so it is no longer recognized as a can. The original consequence was $(\text{can}(x) \wedge \text{graspable}(x) \wedge \text{at_repository}(x))$ and the modified consequence is $(\text{graspable}(x) \wedge \text{at_repository}(x))$. Otherwise, we may modify the consequence by covering the can to make $\text{can}(x)$ false, then this leads to addition of a new behavior

module or modifying the action part of the original module, both of which require considerable re-programming, and are expensive. Addition of new behavior modules also requires a verification that no new undesirable interactions are introduced. Implementors of CYC (Lenat et al 1990) have observed that inference becomes harder as the size of the knowledge base increases because of the increase in the total number of axioms and the number of possible bindings for each of the variables in these axioms.

In consequence generalization, $(c_i \Rightarrow s_k)$ must be negated, while $(c_i \Rightarrow s_{i+1})$ must hold. Hence consequence generalization can be used only when $(s_{i+1} \Rightarrow s_k)$ does not hold.

5.3 Effects Of Behavior Refinement

First, we note that removing a cycle from chain C by stimulus specialization or consequence generalization cannot introduce a cycle in any other chain C' , that did not have a cycle before. Let us examine other effects.

Behavior Modification Theorem. Given two behavior sets B and B' such that B is more powerful than B' (i.e. B' is obtained from B by replacing some behaviors β of B by the less powerful ones β') then:

(a) The greatest potential task space of behavior set B' is smaller than that of B , i.e.

$$|\tau_G(B')| < |\tau_G(B)|$$

(b) Usefulness of B is larger than that of B' i.e.

$$\frac{|\tau_G(B)|}{|B|} > \frac{|\tau_G(B')|}{|B'|}.$$

(c) Likelihood of a cycle in B is at least as large as that for B' .

Proof (a, b) :- First, let us consider the case where a single behavior β has been replaced by the less powerful β' . The set of chains of behaviors composable from a behavior set represents a tree with the root corresponding to the availability of the right initial stimulus and each node in this tree represents a world state which may correspond to the desired state, indicating an existence of a task fulfilling chain. The greatest potential task space is proportional to the total size of this tree of behavior chains (this size is defined in terms of the number of all possible chains contained by the tree and the number of such trees, since different trees can be constructed for different initial world states. Such trees can be combined into a single tree, the root of which is a node that represents virtual initial state. This root can then be connected to the actual initial states (roots) of the individual trees by links). Now, either the behavior β will have more applicability due to a weaker stimulus as compared to the behavior β' , or the behavior β will have stronger consequence resulting in more behaviors being triggerable after it or both. In terms of the task tree, either β will have more parent nodes, or

it will have more children. In either case, the branching factor in B is larger than that in B' and the size of the task tree will be larger. Hence the result (a). Since $|B|$ has not changed, the usefulness of the behavior set, $\frac{|\tau_G(B)|}{|B|}$ decreases when β is replaced by β' which proves part (b). This treatment can be extended to multiple instances of replacing a stronger behavior by a weaker one. \square

Proof (c) :- Let $\beta_i \in B$ and $\beta'_i \in B'$ be two behaviors s.t. β_i is more powerful than β'_i , i.e. $(s'_i \Rightarrow s_i)$ or s_i is weaker than s'_i . The sets $B - \{\beta_i\}$ and $B' - \{\beta'_i\}$ are the same. Now consider any chains of length n composable in B and B' , which differ only in that the module β_i is replaced by β'_i . Consider all behaviors $\beta_j \in C$, $j \geq i$, with consequence c_j . The likelihood of a cycle in B , denoted by $L_{cycle}(B)$ (which we do not restrict to 0-1 range like probabilities) is

$$\sum_{j \geq i}^n (prob(c_j \Rightarrow s_i)),$$

and $L_{cycle}(B')$ is

$$\sum_{j \geq i}^n (prob(c_j \Rightarrow s'_i)).$$

Clearly, since $(s'_i \Rightarrow s_i)$, $\forall c_j (prob(c_j \Rightarrow s_i) \geq prob(c_j \Rightarrow s'_i))$. Similarly $(c_i \Rightarrow c'_i)$ for which similar analysis can be carried out. Thus $L_{cycle}(B) \geq L_{cycle}(B')$. If there is a cycle in B' , there will be a cycle in B too. \square

5.4 Modularity

One of the key questions that we set out to answer is that of modularity. We had defined modularity as the extent to which behavior sets are modular - i.e. free of destructive interactions between the modules. Here we ask a more direct question. Given two cycle free behavior sets, if we wish to add a new module to them, can there be some estimate of the likelihood that this new module will cause a temporal cycle? In an interesting result, we find that the likelihood of such a cycle *is greater for the more powerful behavior set*. This is stated more formally as the following theorem, which is based directly on part (c) of the Behavior Modification Theorem.

Modularity Theorem. Given cycle free behavior sets B and B' (B' is obtained from B by replacing some behaviors of B by less powerful ones) and a module λ not belonging to B and B' is added to both of them, then the set $B \cup \{\lambda\}$ is at the most as modular as the set $B' \cup \{\lambda\}$.

6 Conflicts and Modular Agent Architectures

We have shown that for reactive models of intelligence, the more powerful one tries to make a system, the more difficult it is to avoid destructive

cycles. In connection with the debate referred to in the introduction, this is an important result, for it shows that reactive models may be limited in the complexity of tasks that they can achieve.

One important question that this raises, however, deals not with behavior-based robots alone, but with any distributed knowledge representation in general. The key difference between DAI and the robot behavior model studied here is that of shared global memory (similar to the notion of Blackboard), which is a repository for all the results learned by all the agents. The difficulties of implementing this model are similar to the skills that humans acquire in kindergarten (Durfee 1992) - issues of social interaction that arise between human beings, including that of using a common language, defining and maintaining a shared memory, and resolving conflicts in a manner appropriate to the achievement of the shared overall goal as opposed to individual agent subgoals. This last is the key issue addressed in this paper, and we showed that in reactive systems such as behavioral robots, the likelihood of fatal conflicts increase as the system tries to achieve more complex tasks.

How do the arguments change if one uses a large global memory, as in DAI? The answer is not very clear. DAI deals with issues of large scale open systems which are always subject to unanticipated outcomes in their

operation and which can receive new information from outside themselves at any time. Certainly a long history of past actions is useful in avoiding cycles; but even this cannot be infinite, or infinitely detailed. The other type of shared memory that systems may find useful to have are symbolic representations of the problem domain which are always very succinct and also useful. One of the arguments is that reactive systems have always used such symbolic models, but instead of having one model shared by the task-achieving modules, each module has its own instance of the same symbolic description (Vera & Simon 1993). Autonomous agents, if used to set up schedules, reserve hotel and meeting rooms, arrange transportation and even outline meeting topics, all without human intervention, will have to take into account the temporal relationships with other activities, distributed in space and time, not under control of the agent.

Cyclicity issues can have critical ramifications in the autonomous agents arena. Already agents are widely used in a number of financial back-end applications. Consider an agent that makes payments by supplying credit card numbers. When credit card A's monthly statement comes, it pays by another credit card B and when B's statement comes, it pays by A. This cyclicity will appear to work well for a limited time until the interest rates overwhelm the credit balance of the user, resulting in possibly serious conse-

quences. Asynchronous coordination makes the problem of cycle detection harder.

6.1 Parallel Systems

In case of sequential programs, cyclicity is easier to handle; it is adequate to seek for correctness and termination properties, but in the case of parallel programs, additional issues like mutual exclusion, deadlock absence, livelock absence and individual starvation have to be considered. If two operations are dependent on the outputs of each other, then deadlock may result. For example, the two processes $f(a, b) = c$ and $g(c, d) = a$, wait for output of each other.

A circular chain of tasks may exist such that each task holds one or more resources that are being requested by the next task in the chain (“circular wait ” condition). In nested transactions, locks not released by a transaction are inherited by its parents, making the problem severe. Yiu and Shyamsundar (Yiu & Shyamsundar 1990) report that a process terminates properly if and only if it reaches the end of the process body. This however is not sufficient for termination, because end of the process may switch control to start of the same process. Testing has to be carried out at several levels to ensure absence of deadlocks.

In a distributed system, without a global control and lack of shared variables, it is not possible for all processors to have the same global view of the state of the system (Samadi & Muntz 1988). To detect deadlocks in distributed database systems, a transaction-wait-for graph is maintained. This graph is continually updated as transactions request and release resources. If this graph does not contain a cycle, then the system is not in a deadlock state, but if there is a cycle then it may or may not be in deadlock state. The issue of distinguishing between deadlocks that terminate and undesirable infinite deadlocks is important there as well. Two types of deadlock are common there, resource type (where a process waits for resource) and communication type (where a process waits for message). Distributed deadlock detection algorithms exchange state information periodically among all sites. Though these algorithms are more fault tolerant than centralized ones, they are more prone to detect false deadlock, have complicated deadlock resolution and are difficult to prove correct (Abonamah & Elmagarmid 1994).

Task scheduling algorithms require a mechanism for determining when a search is complete, otherwise, idle workers will never stop requesting work from other workers. This termination detection operation is straightforward in centralized schemes because the manager can easily determine when all

workers are idle. It is more difficult in decentralized algorithms because not only is there no central record of which workers are idle, but also messages in transit may be carrying tasks, even when all workers appear to be idle (Foster 1995). Randomization provides some relief to reflex agents caught into infinite loops, but only at the expense of risking many fruitless actions. Hickman & Shiels (Hickman & Shiels 1991) report a cycle in which two agents assembling parts contend for the same part and exchange it endlessly. Such agents that compete rather than cooperate will be less effective than a single agent that will take longer time but will be free of cycles. Mataric (Mataric 1997) advocates the use of communication to reduce undesirable effects of locality in fully distributed multi-agent systems where the agents learn in parallel, while interacting with each other. Wellman (Wellman 1992) presents a market-oriented programming environment for the construction and analysis of distributed planning systems, based on general equilibrium theory. He reports that despite a careful design of distributed decision making structure according to economic principles, an effective decentralization is not always guaranteed.

Is it the lack of memory that causes cycles? Indeed many robot behaviorists are now moving in the direction of providing some global memory; even Brooks's group has been building map-learning robots, and a number

of other systems provide for some form of “internal state.” In the approach to multi-robot coordination (Simmons et al 2000), different robots independently learn maps and a central mapper module integrates the local maps into a consistent global map. To ensure that the entire floor gets cleaned efficiently, the *clean-floor* behavior set in the implementation (Parker 1996) utilizes a grid map, to keep a track of the areas already cleaned, those yet to be cleaned and those that are inaccessible due to obstacles. This memory allows the robots to avoid cyclic navigation. We feel however, that memory alone is not sufficient to avoid this type of cyclicity, although it may certainly make many operations more efficient. What one does with this memory is critical. For example, even if we maintain a list of recently accomplished events, it is not easy to distinguish cycles from repeated tasks such as repeated visits to the same location to drop off different cans. To do that requires much more knowledge; of the overall objectives, of the plan being followed, of the possible outcomes, etc. - a problem that situation calculus has been struggling with since the early days of AI.

6.2 Meta-Level Reasoning

One of the approaches to this problem, suggested originally in the work in planning and situational calculus, is to use a *meta-level reasoner*, or a

central overseer which decides priorities between modules based on global objectives. A number of researchers have built agent behavior systems using some form of higher-level mediation (Gat 1993), and there is also considerable DAI literature using global mediation strategies. Meta level reasoning refers to reasoning about reasoning, e.g. rules about rules in expert systems, a meta-rule for resolving conflicts could be: once a rule has fired, it may not fire again until the elements that match its conditions have been modified (*refraction*). Corkill & Lesser (Corkill & Lesser 1983) describe a decentralized approach to network coordination with each node guided by a high-level strategic plan for cooperation among the nodes in the network. This high level strategic plan is a meta-level control. Augmenting stimuli of behaviors with conditions that allow composition of behaviors for complex tasks possible is proposed in (Nicolescu & Mataric 2000). These additional conditions are different from conditions that describe world state. It is acknowledged in (Nicolescu & Mataric 2000) that this is a way of specifying plans within the behavior-based approach.

However, meta-level processes have their own problems of implementation and modeling, and it is not clear how these are to be solved. Meta-level processes also raise the question of meta-meta levels, i.e. even higher levels that keep the meta-level from incurring conflicts. Thus, a lot of research

issues remain open on the meta-level approach to conflict resolution. A variation of the meta-level approach, called *hybrid systems*, offers a compromise by employing a reactive system for low-level control and a planner for higher level decision making (Payton et al 1990).

While this is probably the general direction in which a solution may lie, cycles are handled here using meta-knowledge, and hence these are susceptible to the same problems as meta-level planners. Also the question of tradeoffs between the reactive and the deliberative parts is crucial. These have been considered in the work by Simmons in his Task Control Architecture (Simmons 1994). Reactive and deliberative approaches have some similarity with the greedy search and look-ahead based search in constraint satisfaction respectively. Greedy search methods try to satisfy constraints by choosing those values for variables that are locally best as per some criteria, e.g. maximizing the number of constraints satisfied. Look-ahead methods do reasoning to recognize the dead ends and examine a larger part of the search space. Both (Schaerf 1997) and (Rish & Dechter 1996) report that hybrid approaches to constraint satisfaction that combine both local search and look-ahead based search have not proved to be a universal winner, and it is yet not clear on what kinds of problem are the hybrid methods better.

6.3 Collaboration through shared goals

Recently, the explicit specification of the objectives of an agent team has made it possible to avoid some of the mechanisms in which conflicts may arise. Mostly the types of conflict studied in these systems are not temporal or cyclic; they typically involve issues related to the sharing of limited resources. Mechanisms for collaboration in this domain range from explicit demarcation about situations which may have potential conflicts, and obtaining permission or ownership prior to accessing such resources (Durfee et al 1998), or those in which the agents negotiate indicating the degree of preference for a given resource (Chu-Carroll & Carberry 1995) or a set of general “good citizen” rules, such as to defer globally constraining choices, and to use predictable models of behavior that are known to others. Jennings (Jennings 1995) describes experiments in the electricity transportation management, using a multi-agent cooperative problem solving system. In this system, the pre-conditions which must be attained before collaboration can commence are specified and the way the agents should behave when joint activity is progressing satisfactorily is prescribed as well, along with how they must behave in times of difficulty. In the system of simulated robotic soccer playing agents (Stone et al 2000), to deal with the common

challenge of predicting other agents' actions in multi-agent systems, agents model results of other agents' future actions. Multiple types of models and storing traces of behavior of multi-agent teams so that learning can be performed on them, is advocated in (Tambe 2000). Some of these strategies may also apply to agents in a temporal cyclic conflict, especially if it arises due to inadequate communication or understanding of shared needs. For example, in the can-picking example, the pick-up agent may have some knowledge about the sites at which it has recently deposited cans. However, it is not at all clear how such situations can be avoided in general or what types of communication needs would be needed for this purpose. One possible mechanism may be to use complex representations for the task domain, such as the modal logic model in which predicates that are inconsistent can be de-asserted in a possibilistic world, while others are asserted either as necessarily P, or as possibly P (Wooldridge & Jennings 1995).

6.4 Environmental Engineering

Brooks (Brooks 1986) argued that robots like Shakey cheated since they operated in simplified environments where the sensing and perception processes were not challenging at all. The environment was engineered to help the robot in succeeding. However environments continue to be engineered,

as exemplified by the mobile robot competitions of AAAI (American Association of Artificial Intelligence). Though this environmental engineering is carried out to simplify sensing and perception, it also serves the purpose of avoiding undesirable cycles in navigation and manipulation, since markers are put on objects to help the vision systems in distinguishing between multiple identical objects. Hammond & Converse (Hammond & Converse 1991) suggest that agents that interact with an environment over an extended period of time can in fact create regularities in their environments and maintain them, to reduce the amount of reasoning they have to do.

Several AAAI mobile robot competition entries have used behaviors whose input is not local information, but similar to global information, e.g. the occupancy grid used by Flakey (Congdon et al 1993). In the 1992 AAAI mobile robot competition described in this article, one team attached an omnidirectional bar-code tag to each pole and designed the vision algorithm to extract the bar codes from an image. Using the competition rules, these robots are engineered to optimize the performance at the cost of generality. Konolige (Konolige 1994) says, “In the first competition, SCARECROW was engineered and built specifically for the contest and had a very little utility in even slightly different environments”. Konolige also says, “A set of uniform perceptual markers was added to the environment, like black crosses,

bars, X's and O's on a white background. These markers were placed in key areas in the events: in event 1, on each door, in event 2, near selected doorways and the coffee pot, and in the event 3, on the boxes." However cycles were exhibited despite the use of markers in some cases, for example, one robot was programmed to ignore markers that were outside its model of the room and this ad-hoc strategy did not avoid cycles. Konolige remarks that **this environmental engineering is symptomatic of the nonautonomy in the mobile robot systems.** Some teams took advantage of the unique features of the map in hand coding strategies. Because of only one room, exploration was kept to a minimum in one event, avoiding going back and forth between the rooms. Kortenkamp et al (Kortenkamp et al 1993) say, "Until a general computer vision system is perfected, the environments in which robots operate will have to be engineered in order to simplify the computer vision portion of the tasks".

Identifying the situations under which reactive behaviors work, Konolige also warns, "For avoiding obstacles and accomplishing goals in tightly constrained contexts, reactive architectures have proved to be a winner. Using reactive behaviors guarantees that the robot will exhibit at least a modicum of intelligent behavior, staying out of trouble and performing well when the situation is favorable. Still there are problems with the reactive behaviors,

they can take a long time to program and test and debug.”

6.5 Representations: Precise vs. Fuzzy

The other question that arises is that of the representation. Could it be that the symbolic logic used in most of the behavioral robots causes these conflicts, i.e. the conflicts are an attribute more of the representation than the problem domain? This has been a source of significant debate among cognitive psychologists, for instance - see (Vera & Simon 1993) for a guide. A quick look at the problem here, however, shows this to be unimportant in the current problem. The cycle that occurs in the can dropoff task is due to the nature of the task itself, and is not at the representation level. Behaviorists claim that symbolically motivated behaviors lack the “strength of stimulus” concept so critical to biological systems. Most predicates such as $can(x)$, are really based on low level sensory information that reflect the *can-ness* of an object, a modality that can be captured by using a different representation such as fuzzy logic or a neural network. The representations associating potential fields with objects in the environment (Arkin 1992) also achieve such a strength of stimulus notion. However in the can pickup conflict for example, it is clear that a change of representation in itself will not eliminate the problem. After dropping off the can, if the robot happens

to “see” the same can, there is nothing in the fuzzy or other representation that would cause it to stop: whatever mechanisms are needed to avoid cycles in the boolean logic framework would still be needed in these other representations.

Potential Fields are multi-dimensional fuzzy functions. Instead of returning true or false, the function returns a fuzzy measure - e.g. “degree-of-can-ness”. Systems using such functions can be built based on fuzzy logic, or may use a global function which is optimized to find a suitable course of action. Using this approach in the can-cycle problem above, if the same can is sensed from same location, the same motion vector (for picking up the can) will be found by summation of attractive and repulsive fields, leading to cycle. To increase robustness of behavior systems by enhancing them to be able to work with different levels of resource availability, multi-fidelity behaviors are used in (Winner & Veloso 2000). Behaviors at different fidelity levels can act with variable state information. Though this can avoid some of the problems with original behavior set, this approach just augments the original behavior set, leaving the fundamental problems we raised in this paper unsolved. Thus, the conclusions of our work remain valid across different representational schemes, since they reflect underlying properties of the problem and not the representation.

7 Learning and Behavior Modification

Another modification of the reactive approach to intelligence is to *learn* the behavior modules instead of trying to predefine them (Mahadevan & Connell 1992). The basic claims here are that the learning proceeds directly from the sensory interaction, and that the feedback for learning also comes directly from its performance in the task.

The work by Mahadevan and Connell (Mahadevan & Connell 1992), supported by both experimental and simulated results on a simple box-pushing task, particularly merits a more detailed investigation. This work differs from Brooks's paradigm in one significant respect: some state history information is used - a recency memory that may conceivably be useful in avoiding cycles. The basic learning mechanism, based on reinforcement learning, involves assigning each module a reward function, which evaluates the effects of the stimulus-consequence behavior. The more successful actions are reinforced. Thus each module is modified independently, based on the pre-programmed reward function. One of the weaknesses of this work is the use of the same reward functions to evaluate the learning process, as opposed to some other criteria based on the overall performance of the actual task. Let us see how our basic model can be modified to handle this

problem.

7.1 Effect of learning in agents

We consider the effect of learning to be that of modifying a behavior β to β' , where β' is more successful at a task than β . Here we define success not in terms of higher reward values, but in terms of two task-specific attributes:

- (a). It performs the task under a wider range of initial conditions, and
- (b). It delivers results to a more exact level of precision.

We feel that such a model better reflects the capability of a robot to learn than the self-fulfilling use of the reward functions, which are hard to define (Mahadevan & Connell 1992).

The first criterion we have used, that of a wider range of initial conditions, may be called a generalization - i.e. β' will be capable of performing the task whenever β would have been. Hence, the stimulus s' is more general: ($s \Rightarrow s'$). Similarly, the second criteria, of performing the task more precisely, is seen to be a specialization: ($c' \Rightarrow c$).

These are the same conditions that make β' more powerful than β . To rephrase it in the context of our example of can-pickup, if a robot can pick up a soda can, it should be able to pick up a coffee cup or other similar object. If a robot can drop off a soda can, it should be able to do the same

for a kitchen knife or a wad of currency. Note that in all these examples, the refined behavior is more powerful in the sense of section 3.2. This property, together with the Modularity theorem from section 5.4 indicates that *the more a robot learns, the more likely it is to exhibit cycles*. Clearly then, if a robot, originally programmed with behavior set B refines some of its behaviors through behavior learning resulting in the more powerful behavior set B' , then the latter set will be more prone to cyclic conflict.

This question did not arise in the work of Mahadevan and Connell, due to the simplicity of the task chosen, which was to identify something as being pushable, and then to push it, in no particular direction. While a laudable effort in introducing learning into reactive systems, this direction will also have significant problems in overcoming the effect of the modularity theorem. Indeed, even in their simple models, some cycles are reported (Mahadevan & Connell 1992), such as turning back and forth repeatedly. Cycles are also reported in other attempts at robot learning (Kube & Zhang 1997).

Others have raised concerns about the lack of modularity in AI: that the prevailing AI research paradigm rarely produces reusable components, because it pays too little attention to context and more attention to heuristics like “divide and conquer”, “functional specialization” and “domain specialization” (Hayes-Roth 1996). Highly useful systems have been developed

using this approach, but they are less general and less reusable.

8 Conclusion

We reviewed several paradigms of designing autonomous agents and the conflicts that arise in them, with a special emphasis on the infinite cycles. Developing metrics to evaluate the effectiveness of behavior-based reactive systems, we showed that procedures that reliably eliminate cyclic conflicts adversely affect the power, usefulness and modularity of behavior sets. We have raised these fundamental problems that arise in reactive models of distributed intelligence. The answers to these questions are crucial to the future direction of artificial intelligence as a whole.

Acknowledgments

We would like to acknowledge the support from the Center for Robotics, IIT Kanpur and the college of Engineering and Applied Sciences, Univ. of Wisconsin, Milwaukee for this work. Valuable discussions with Lilavathi Krishnan, P.R.K. Rao, Mohini Mullick, Sudipto Mukerjee, Somnath Biswas and Harish Karnick at IIT Kanpur, as well as feedback from Randall Beer, Reid Simmons and Frederick Hayes-Roth helped formulate some of the ideas presented here. We also thank Nicolas Kushmerick, Paul Rosenbloom, David Kirsh, B. Chandrasekaran, Henry Hexmoor, Robin Murphy, John Blicht,

Matt Mason, Avi Kak, Anthony Barrett, Douglas MacKenzie, Subbarao Kambhampati, John Connell, John Bay, Melinda Gervasio, Anavai Ramesh, Neil Murray, Marvin Minsky, Erann Gat, Pattie Maes, Praveen Bhatia, Benjamin Kuipers and Pete Bonasso for useful feedback and remarks by e-mail. We thank Ichiro Suzuki, Michael Girmscheid, Randy Ellis and James Dabrowski at University of Wisconsin, Milwaukee for useful discussions on autonomous agents.

References

- Abonamah A. A. and Elmagarmid A. K. (1994). A survey of deadlock detection algorithms in distributed database systems, In *Advances in distributed and parallel processing, Vol. 1 - system paradigms and methods*, ed. H.W.Tyrer, Ablex publishing corp., pp. 310-341
- Anderson, T.L. and Donath M. (1990). Animal Behavior As A Paradigm For Developing Robot Autonomy, *Robotics and Autonomous Systems*, 6(1 & 2), pp. 145-168.
- Arkin R.C. (1992). Behavior-Based Robot Navigation for Extended Domains, *Adaptive Behavior* 1(2), pp. 201-225.
- Brooks R.A. (1986). A robust layered control system for a mobile robot,

IEEE journal on robotics and automation, 2(1), pp. 14-23.

Chu-Carroll Jennifer and Carberry Sandra (1995). Communication for Conflict Resolution in Multi-Agent Collaborative Planning, *Proceedings of the First Int'l Conference on Multiagent Systems (ICMAS)*, pp. 49-56.

Congdon Clare, Huber Marcus, Kortenkamp David, Konolige Kurt, Myers Karen, Saffiotti Alessandro and Ruspini Enrique H. (1993). CARMEL versus FLAKEY: A comparison of two winners, *AI Magazine*, Spring issue, pp. 49-57.

Connell J. (1990). *Minimalist mobile robotics, A colony style architecture for an artificial creature*, Academic press Inc.

Corkill Daniel D. and Lesser Victor R. (1983). The use of meta-level control for coordination in a distributed problem solving network, *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 748-756.

Durfee E.H. (1992). What Your Computer Really Needs to Know You Learned in Kindergarten, In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pp. 858-864, San Jose, CA.

Durfee Edmund H., Kenny Patrick G., and Kluge Karl C. (1998). Integrated Permission Planning and Execution for Unmanned Ground Vehicles, *Autonomous Robots*, 5, pp. 1-14.

Foster Ian (1995). *Designing and building parallel programs*, Addison-Wesley.

- Gat E. (1993). On the Role of Stored Internal State in the Control of Autonomous Mobile Robots, *AI Magazine*, 14(1), pp. 64-73.
- Georgeff M. P. and Lansky A. L. (1990). Reactive Reasoning and Planning, In *Readings In Planning*, Eds. James Allen, James Hendler & Austin Tate, Morgan Kaufmann Publishers, Inc., pp. 729-734.
- Grosz Barbara J, and Davis Randall (1994). AAAI Report to ARPA on 21st century intelligent systems, *AI Magazine*, Fall issue, pp. 10-20
- Hammond Kristian J. and Converse Timothy M. (1991). Stabilizing environments to facilitate planning and activity: An engineering argument, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Vol. 2.
- Hartley R. and Pipitone F. (1991). Experiments with the subsumption architecture, In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, pp. 1652-1658.
- Hickman Stephen and Shiels Martin (1991). Situated action as a basis for cooperation, In *Decentralized AI 2*, Yves Demazeau and Jean-Pierre Muller (eds.), Elsevier Science Publishers B.V., pp. 35-47.
- Jennings N. R. (1995). Controlling cooperative problem solving in industrial multi-agent systems using joint intentions, *Artificial Intelligence* 75, pp. 195-240.

- Kirsh, D. (1991). Today the earwig, tomorrow man?, *Artificial Intelligence*, 47(1-3): 161-184.
- Konolige Kurt (1994). Designing the 1993 robot competition, *AI Magazine*, Spring issue, pp. 57-62.
- Kortenkamp David, Huber Marcus, Choen Charles, Raschke Ulrich, Birlack Clint, Congdon Clare Bates, Koss Frank and Weymouth Terry, (1993). Integrated mobile robot design: Winning the AAAI'92 robot competition, *IEEE Expert*, August issue, pp. 61-73.
- Kube C. Ronald and Zhang Hong (1997). Task modeling in collective robotics, *Autonomous Robots* 4, 53-72.
- Laird John and Rosenbloom Paul, (1990). Integrating execution, planning and learning in Soar for external environments, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 1022-1029.
- Lenat D.B., Guha R. V., Pittman K., Pratt D. and Shepherd M. (1990). CYC: towards programs with common sense, *Communications of the ACM* v.33(8), pp. 30-49, August issue.
- Mahadevan S., and Connell J. (1992). Automatic programming of behavior-based robots using reinforcement learning, *Artificial Intelligence*, 55, pp. 311-365.

Masthoff J. and Hoe Van R. (1995). A View on the Architecture and Design of Highly Autonomous and Situated Agents, *Proceedings of the First International Conference on Multiagent Systems (ICMAS)*, ed. Victor Lesser, p. 458 (Poster)

Mataric Maja J. (1997). Using communication to reduce locality in multi-robot learning, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 643-648.

Mitchell Tom (1990). Becoming increasingly reactive, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 1051-1058.

Moravec H.P (1984). Locomotion, Vision, and Intelligence, *Proceedings of the First International Symposium on Robotics Research*, Bretton Woods, NH, edited by Michael Brady and Richard Paul, MIT Press, Cambridge, MA, pp. 215-224.

Muscettola Nicola, Nayak Pandurang P., Pell Barney and Williams Brian C. (1998). Remote agent: To boldly go where no AI system has gone before, *Artificial Intelligence* 103, 1-2, pp. 5-47.

Nicolescu Monica N. and Mataric Maja (2000). Deriving and using abstract representation in behavior-based system, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Student abstract, pp. 1087.

Parker Lynne E. (1996). On the design of behavior-based multi-robot teams,

Advanced Robotics, 10(6), pp. 547-578.

Payton, D. W., Rosenblatt J. K. and Keirse D. M. (1990). Plan guided reaction, *IEEE Transactions on Systems, Man and Cybernetics*, 20(6), pp. 1370-1382

Reiter R. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz V., editor *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, Academic Press, NY, pp. 359-380.

Rish Irina and Dechter Rina (1996). To guess or to think? Hybrid algorithms for SAT, *Proceedings of the Principles and Practices of Constraint Programming (PPCP)*.

Hayes-Roth Frederick (1996). AI: What works and what doesn't? *Invited talk at the Innovative Applications of Artificial Intelligence conference (IAAI)*, Portland.

Russell Stuart and Norvig Peter (1995). *Artificial Intelligence: A Modern Approach*, Prentice-Hall, NJ.

Samadi Behrokh and Muntz Richard (1988). A distributed algorithm to detect a global state of a distributed simulation system, In *Distributed processing*, eds. M.H. Barton, E.L. Dagless and G.L.Reijns, North-Holland, pp. 19-34

- Schaerf Andrea (1997). Combining local search and look-ahead for scheduling and constraint satisfaction problems, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1254-1259.
- Simmons Reid (1994). Structured control for autonomous robots, *IEEE Transactions Robotics & Automation*, February issue.
- Simmons Reid, Apfelbaum David, Burgard Wolfram, Fox Dieter, Moors Mark, Thrun Sebastian and Younes Hakan, (2000). Coordination for multi-robot exploration and mapping, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 852-858.
- Stone Peter, Riley Patrick and Veloso Manuela, (2000). Defining and using ideal template and opponent agent models, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 1040-1045.
- Tambe Milind (2000). Agent assistants for team analysis, *AI Magazine*, Fall issue, pp. 27-31.
- Vera A.H. and Simon Herbert A. (1993). Situated Action: A Symbolic Interpretation, *Cognitive Science* 17, pp. 7-48.
- Wellman Michael P. (1992). A general-equilibrium approach to distributed transportation planning, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 282-289.
- Winner Elly and Veloso Manuela (2000). Multi-fidelity robotic behaviors:

Acting with variable state information, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 872-877.

Wooldridge Michael and Jennings Nick (1995). Agent Theories, Architectures, and Languages: A Survey, In *Intelligent Agents - Theories, Architectures, and Languages*, Michael Wooldridge and Nicholas R. Jennings (ed.) Springer-Verlag Lecture Notes in Artificial Intelligence January issue, pp. 1-32.

Yiu Leo and Shyamsundar R.K. (1990). Static analysis of deadlock, distributed termination and timing properties in real-time distributed systems, In *Decentralized systems*, M. Cosnard and C. Girault (eds.), Elsevier science publishers B.V. (North-Holland), pp. 411-425