Modularity Assumptions in Situated Agency

Amol Dattatraya Mali
Dept. of computer science and engg.
Arizona state university
Tempe, AZ 85287-5406
e-mail: mali@tahoma.eas.asu.edu

Amitabha Mukerjee
Center for Robotics,
I.I.T. Kanpur, India 208016,
e-mail: amit@iitk.ernet.in

Abstract

This research focuses on a specific class of agents called "situated agents", which use minimal communication and rely mostly on changes in the environment as their cue for action. Some early successes of this model, especially in robotics, have led to an intense 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 situated models. Some of our results are - 1. The probability of such cycles developing increases as the situated agents become more useful (phase transition). 2. Control methods for avoiding cycles such as prioritization are unreliable. 3. Behavior refinement methods that reliably avoid these conflicts (either by refining the stimulus, or by weakening the action) lead to weaker functionality. 4. Conflicts place a bound on the power of new behavior that can be added to a situated system. We use these and some other results to examine the assumption of modularity in situated agents. Viewing chess playing as a situated activity, we discuss how our results hold relevance to the design of chess players.

1 Introduction

Artificial Intelligence paradigms today are moving towards a more distributed agent-based architecture. The charter for AI in the twenty-first century is to build such agents for use in simulation, human assistance, robot interaction etc. From web browsers to intelligent houses to devices that "learn about different users' behavior", agents are rapidly moving into diverse types of applications. These agents can be many modules of a single system e.g. a robot or simulation system, or may be many agents themselves e.g. a group of robots. They may be in hardware devices or as softbots i.e software modules. However, all agent systems attempt to reduce or eliminate the centralized shared memory, relying instead on parameter passing and communication between individual agents.

Many traditional AI practitioners are however, worried about reliability. How does one ensure that such a system of independent modules, each performing its own task, can deliver the appropriate overall functionality? In particular, how does one analyze the conflicts and interactions between such agents? One of the problems well known in agent systems is that of resource or goal conflicts. In this work we look at another type of conflict which arises due to the temporal chaining of 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*. These are difficult to foresee during agent design. In this work we investigate this type of conflict and propose some mechanisms which may be used to handle it. We also propose some performance measures and show how these cycle removal procedures result in a cost to the performance.

The paper is organized as follows - In section 2, we discuss the achievements of situated agency and instances of cyclic conflicts reported. In section 3, we develop a formal model of situated systems and use it to prove our results in sections 5 and 6. In section 4, a classification of behavior conflicts is provided and the role of prioritization in eliminating cycles is discussed. In section 5, we discuss our behavior refinement mechanisms for reliably eliminating cycles and prove our results on effects of behavior refinements. In section 6, our results on modularity are discussed. In section 7, we discuss the implications of our conclusions for situated agency, chess players and distributed AI. Section 8 presents our conclusions.

2 Situated 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 effected 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).

In this work we will first build a temporal model for situated systems that will follow the temporal sequencing of the actions. Next we will investigate the issue of cycles in such temporal chains, and identify some techniques for avoiding such cycles in practice. Finally, we shall show that there is a tradeoff between the possibility of such conflicts arising and the amount of global memory (e.g. history of recent actions) that is maintained.

Situated systems became popular in robotics with the work of Brooks[4], who challenged the deliberative paradigm in AI by building stimulus-response based robots, also known as behavior-based robots. A typical situated 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.

Within this basic structure, a number of models have 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. Impressive results have been achieved using this strategy in a can collection robot [5], navigation of mobile robot [2], a prototype airplane controller [7]. That the situated model has had a deep impression on Artificial Intelligence is clear from the fact that several journal issues have recently been devoted to debating this approach (Artificial Intelligence v.47, 1991, Robotics & Autonomous Systems, v.6:1, 1990, Cognitive Science v.17, 1993, Artificial Intelligence v.73 1995).

In the debate on situated models, traditional AI researchers have argued that control cannot serve as a complete substitute for representation [8]. Others have attacked the parallel psychological model of situated action, showing that the situated models were actually symbol systems, or that they required significant internal representations which were, or could be represented using, symbolic systems [14].

Despite this broad debate, no metrics have been proposed for comparing the overall functionality of different situated 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 choosing an order in which to add or debug behavior modules have not been answered. These are some of the shortcomings that are addressed in the current work.

Our primary objective here is to analyze temporal cycles in situated systems and examine the promises made by modularity. An instance of such a cycle is recorded by Connell where a can collecting robot attempts to re-pick the can it has just deposited. This conflict was detected only after a full implementation [5]. Cyclical wandering and cyclic conflict of going back and forth between two obstacles is widely experienced in robotic systems, as in [1]. Cyclic behavior in multi-robot system for box pushing has been reported in [9]. Such cycles are potentially very costly in real systems. Even when they are detected, it is not clear how to fix these "bugs". These are some of the questions we set out to answer.

The temporal structure of situated behaviors is often sequential since one behavior usually provides the stimulus for another, so that the behaviors are executed in sequence. 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.

3 Evaluation Metrics

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. We model a behavior by a *3-tuple*: stimulus, action, consequence, thus behavior β_i is denoted by $< s_i, a_i, c_i >$. In the examples and proofs that follow, the stimulus s and the consequence c are defined using First Order predicates as in Brooks' subsumption architecture. This leads to a structure much like the standard situation calculus models. In the notation in 3.1, the operator : in $\{\beta_1 : \beta_2\}$ means that the module β_1 precedes module β_2 and this happens iff there exist time instants θ_1, θ_2, θ_3, $\theta_1 < \theta_2 < \theta_3$ such that at θ_1, β_1 is in execution but β_2 is not and at θ_2, β_1 and β_2 are both being executed and at θ_3, β_2 is executed 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 $\{\beta_1 : \beta_2 : \beta_3 : \ldots : \beta_n\}$. 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 executable. 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. 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 ***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 $X \bigwedge Y \bigwedge Z$ and $s_2 = X \bigwedge A$. 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.

We define a *behavior space* 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 $C \lhd B$), if the elements of C are also elements of B.

3.2 Power, Usefulness, and Modularity of Behaviors

To compare different behavior systems, we define some metrics that relate to the effectiveness of a behavior system 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 $< stimulus, response >$ formalism by the notion of power defined next.

• **Power:** A behavior (β := <s, a, c>) is more powerful than (β' := <s', a', c'>) iff $(s' \Rightarrow s) \bigwedge (c \Rightarrow c')$. 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. A behavior space B is more powerful than the behavior space 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 the behavior space? (similar terminology has been defined in [10])

• **Span:** Behavior space B **spans** task space τ if and only if $\{\forall(t \in \tau)\,(\exists(C \lhd B) fulfills(C, t))$.

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

• **Usefulness:** The ratio $\frac{|\tau_G(B)|}{|B|}$.

• **Flexibility:** A behavior space B is at least as flexible as behavior space B' if $\{\forall t \in (\tau_G(B) \cap \tau_G(B'))\, \exists(C \lhd B)\{fulfills(C,t) \bigwedge \forall(C' \lhd B')\,\{fulfills(C', t) \Rightarrow |\ C\ | \leq |\ C'\ |\}\}\}$. This means that B can fulfill tasks with shorter chains (which can be composed from fewer behaviors).

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

4 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 $C = \{\beta_1 : \beta_2 : ... : \beta_n\}$ be the behavior sequence that achieves a desirable outcome. There are three types of conflicts that can cause the chain C from not being executed. In each case, some sequence $\beta_i : \beta_{i+1}$ 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. (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.

In this paper, we are focusing on *cyclic conflicts*, where, both β_{i+1} and β_k may be triggered and the triggering of β_k would lead to a cycle (Figure 1).

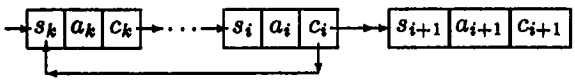

Figure 1. *Cycle in a temporal chain of behaviors.*

However, all cycles do not result in a conflict, e.g. there may be repeated tasks, or even life-cycle-maintenance-tasks. Next we investigate the question of removing a cyclic conflict after it is detected. One of the methods widely noted in natural behavior systems is that of behavior inhibition, where one behavior inhibits another from performing its normal task. The inhibitive effect may or may not continue beyond the end of the execution period of the inhibiting behavior. But this mechanism does not necessarily kill the stimulus of β_k and the cycle may still persist.

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 response 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 (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 $k \leq i$:, $(c_{k-1} \Rightarrow s_k)$ and $(c_i \Rightarrow s_{i+1})$, and $(c_i \Rightarrow s_k)$.

In Stimulus Specialization, we seek to specialize s_k s.t $(c_i \Rightarrow s_k)$ is not valid. 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.

5.2 Response 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 *(visible(x)* $\bigwedge$ *can(x)* $\bigwedge$ *graspable(x))* and the modified consequence is *(visible(x)* $\bigwedge$ *graspable(x))*. Otherwise, we may modify the consequence by covering the can to make *visible(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. In response generalization, $(c_i \Rightarrow s_k)$ must be negated, while $(c_i \Rightarrow s_{i+1})$ must hold. Hence response generalization can be used only when $(s_{i+1} \Rightarrow s_k)$ does not hold.

5.3 Effects Of Behavior Refinement

Let us now investigate the effects of stimulus specialization and response generalization.

Lemma 1(a). Whenever a behavior space B is modified to B' to eliminate cyclic conflicts by specializing stimulus s of some behavior $\beta \in B$ to s' so that $(s' \Rightarrow s) \bigwedge \neg(s \Rightarrow s')$, then $\frac{|\tau_Q(B)|}{|B|} > \frac{|\tau_Q(B')|}{|B'|}$.

Proof :- Let Σ be the stimulus space and s $\subset \Sigma$ and s be a stimulus that is a conjunction of its members. Now let s be specialized to s' so that $s' \subset s$. Now tasks or subsequent behaviors fulfillable or triggerable in (s - s') (this difference corresponds to the states of the world in which s holds but s' does not) will no longer be so. Thus we need a new behavior β'' for some $s'' \subseteq$ s, such that $(s'' \cup s')$ = s, so that β' and β'' together serve the stimulus set s which implies that $\mid B \mid$ increases and the usefulness of B decreases. If the new behavior is not added, the usefulness decreases because of decrease in the greatest fulfillable task space. The notion of this proof can be generalized to non-conjunctive stimuli, where also a similar set of unserviced stimuli can be found.□

If a new behavior is added, length of the chain required to fulfill unfulfilled tasks will not necessarily increase, hence we do not claim that stimulus specialization does reduce flexibility. However this is not the case with response generalization. This leads us to the next result.

Lemma 1(b). Whenever a behavior space is modified to eliminate cyclic conflicts by response generalization, the flexibility and usefulness of the behavior space decreases.

Proof :- If the consequence c of β is generalized to c' so that $c' \supset$ c. Thus $(c - c')$ is not being made available by β. Let the behavior space containing β' be denoted by B'. Hence other new behaviors are needed to complete the tasks requiring (c - c') which increases

| B |. This implies that the usefulness of B decreases. Also, addition of new modules increases the lengths of the chains composed to fulfill these tasks resulting in decrease in flexibility. If this not done then some tasks requiring (c - c') (this should be interpreted as a pure difference of literals) cannot be performed which implies that $| \tau_G(B') | < | \tau_G(B) |$ which means that the usefulness of the behavior space decreases. The notion of this proof can be generalized to non-conjunctive consequences, where also a similar set of unserviced stimuli can be found.□

If *pick_up* is specialized so that it does not pick up a can it has dropped, then to drink coffee, we will have to design another "multiple_pick_up". In some cases, it may not be possible to design an action such that it will fulfill these conditions. This discussion brings us to our most important results, about the power and usefulness of behavior spaces.

Behavior Modification Theorem. Given two behavior spaces 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 fulfillable task space of behavior space B' is less 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 space represents a tree with initial point 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 existence of a task fulfilling chain. The greatest fulfillable 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 actual initial states of the world by links). Now, either the behavior β will have more applicability due to smaller stimulus length as compared to the behavior β', or the behavior β will have stronger consequences 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 space, $\frac{|\tau_G(B)|}{|B|}$ decreases when β is replaced by β' which proves part (b). This treatment can be extended to multiple instances of replacing a strong behavior by a weak one.□

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'. Now consider any chain of length n composable in B and B', which differ only in that the module β_i is replaced by β_i'. Now 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 is not restricted to 0-1 range like probabilities) is

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

and $L(B')$ is

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

Clearly, since $(s_i' \Rightarrow s_i)$, $\{\forall j[prob(c_j \Rightarrow s_i) \geq prob(c_j \Rightarrow s_i')]\}$. 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. □

6 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 systems are free of destructive interactions among the modules. Here we ask a more direct question. Given two cycle free behavior systems, 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 system.* This is stated more formally in the following theorem, which is based directly on part (c) of the Behavior Modification Theorem.

Modularity Theorem. Given cycle free behavior spaces 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 space B$\cup\lambda$ has a higher likelihood of cycle than the space $B'\cup\lambda$.

Proof - The theorem follows from the part (c) of behavior modification theorem. As B contains more powerful behaviors, the chance of occurrence of a chain $C = \{\beta_1 : \beta_2 : ...\beta_n\}$ such that $(c_i \Rightarrow s_k)$, $i \geq k, 1 \leq i, k \leq n$ is higher in $B\cup\lambda$ and from part (c) of behavior modification theorem, $L_{cycle}(B) \geq L_{cycle}(B')$. Thus,

$L_{cycle}(B \cup \lambda) \geq L_{cycle}(B' \cup \lambda)$. Hence, $\frac{1}{L_{cycle}(B \cup \lambda)} \leq \frac{1}{L_{cycle}(B' \cup \lambda)}$. This decreases modularity as defined in section 2.3. This holds irrespective of the power of the behavior λ. These results imply that more powerful behavior spaces are consequently less modular.□

Usefulness of a behavior space $B(i)$ is denoted by $U(B(i))$, the greatest potential task space of $B(i)$ is denoted by $\tau_G(B(i))$.

We define two disjoint behavior spaces $B(i)$ and $B(j)$ to be *scalable* together if $\tau_G(B(i) \cup B(j)) \supset (\tau_G(B(i)) \cup \tau_G(B(j)))$

Lemma 2. If disjoint behavior spaces $B(1), B(2), ..., B(n)$ are scalable together, then $U(B)* \mid B \mid > (U(B(1))* \mid B(1) \mid + ... + U(B(n))* \mid B(n) \mid)$ where $B = B(1) \cup B(2) \cup ... \cup B(n)$.

Proof - By definition of usefulness, $\mid \tau_G(B(i)) \mid = U(B(i))* \mid B(i) \mid$. Since the behavior spaces $B(1), B(2), ..., B(n)$ are scalable together, $\mid \tau_G(B) \mid > \mid \tau_G(B(1)) \mid + \mid \tau_G(B(2)) \mid + ... + \mid \tau_G(B(n)) \mid$ The result follows from these two equations. Hence the proof.□

Lemma 3. When p new behaviors are added to a behavior space B, $\tau_G(B)$ must increase by a factor of at least $(\frac{p}{|B|} + 1)$ for the usefulness to increase.

Proof - For this, it is necessary to have $U(B \cup P) > U(B)$, where P is the set of p new behaviors to be added to B. Hence, we need $\frac{|\tau_G(B \cup P)|}{|B|+p} - \frac{|\tau_G(B)|}{|B|} > 0$. The result follows from this.□

Lemma 4. If a behavior β' is added to behavior space B such that there exists $\beta' \in B$ which is less or more powerful than β', such that β and β' cannot occur concurrently, there exists a control conflict in $(B \cup \beta')$.

Proof - Let us say that β' is more powerful than β. When β is triggered, β' will also be. If β' is less powerful than β, then when β' is triggered, β will also be. Hence in any of the two cases, β, β' will be triggered at the same time, resulting in a control conflict. Hence the proof.□

Theorem 3. Elimination of cyclic conflicts places a bound on the power of behaviors to be added to a behavior space.

Proof - Let the new behavior to be added to the behavior space B be β_k. There should not exist a chain $\{\beta_{i_1} : \beta_{i_2}, : \beta_{i_3} : ... : \beta_{i_{j-1}} : \beta_{i_j}\}$ in the tree corresponding to $\tau_G(B)$ such that new behavior can come after this chain and the chain can succeed the new behavior, resulting in the chain $\{\beta_{i_1} : \beta_{i_2}, : \beta_{i_3} : ... : \beta_{i_{j-1}} : \beta_{i_j} : \beta_k : \beta_{i_1} : \beta_{i_2}, : \beta_{i_3} : ... : \beta_{i_{j-1}} : \beta_{i_j}\}$ This requires that $\neg((c_{i_j} \Rightarrow s_k) \bigwedge (c_k \Rightarrow s_{i_1}) \bigwedge \neg \exists a, i_p ((1 \leq p \leq j) \bigwedge (c_k \Rightarrow \neg a) \bigwedge (\beta_k \prec \beta_{i_p}) \bigwedge (s_{i_p} \Rightarrow a) \bigwedge \neg \exists i_q ((c_{i_q} \Rightarrow a) \bigwedge (\beta_{i_p} \prec \beta_{i_q}))))$. Intuitively, s_k should be stronger and c_k should be weaker, so that it cannot both occur immediately after the existing chain and immediately precede the existing chain. These conditions should hold for all chains in the current $\tau_G(B)$. This places a bound on the strength of s_k and c_k which is not very desirable since we want a stimulus to be as weak as possible and the consequence to be as strong as possible. Hence the result.□

Ideally, a behavior space should exhibit same performance irrespective of the order in which the behaviors are added. The following result shows that this is not always true.

Lemma 5. If a set of behaviors $\{\beta_{i_1}, \beta_{i_2}, \beta_{i_3}, ..., \beta_{i_k}\}$ (such that these behaviors form a cycle free behavior space), is to be added to a behavior space B such that for all $1 \leq m, n \leq k, m < n$, β_{i_n} is more powerful than β_{i_m}, then there exist orders in which the behaviors can be added to detect and avoid cycles in the resulting behavior space.

Proof - One can add the most powerful behavior β_{i_k} first and check if the behavior space $(B \cup \{\beta_{i_k}\})$ contains a cycle. If the new behavior space is cycle free, adding one or more less powerful behaviors will not introduce a cycle and behaviors from the remaining set $\{\beta_{i_1}, \beta_{i_2}, \beta_{i_3}, ..., \beta_{i_{k-1}}\}$ can be added in any order. If addition of β_{i_k} results in a cycle, one can avoid adding it and add remaining behaviors in decreasing order of power. This ensures that the largest subset of $\{\beta_{i_1}, \beta_{i_2}, \beta_{i_3}, ..., \beta_{i_{k-1}}\}$ that does not cause a cycle when added to B, is added to B. One can add the least powerful behavior β_{i_1} first and check for existence of cycle in the space $(B \cup \{\beta_{i_1}\})$. If $(B \cup \{\beta_{i_1}\})$ contains a cycle, adding any other behavior will also result in a cycle. In that case one can just discard the set $\{\beta_{i_1}, \beta_{i_2}, \beta_{i_3}, ..., \beta_{i_k}\}$. Hence the proof.□

Corollary 1. Cycles occur even if individual modules are combined into more complex behaviors.

Proof - We consider two ways of combining behaviors - chaining and conditional combination. In chaining, behaviors $\beta_{i_1}, \beta_{i_2}, \beta_{i_3}, ..., \beta_{i_k}$ forming the chain $\{\beta_{i_1}, \beta_{i_2}, \beta_{i_3}, ..., \beta_{i_k}\}$ are replaced by a single behavior β_x such that $s_x = s_{i_1}$ and c_x is a conjunction of consequences of individual behaviors of the chain, with literals whose truth is changed by latter behaviors and never changed after that, being absent in c_x (e.g. if behaviors β_1 and β_2 are replaced by a single behavior that is essentially the chain $\{\beta_1 : \beta_2\}$ and $c_1 = a_1 \bigwedge a_2 \bigwedge a_3$ and $c_2 = \neg a_2 \bigwedge a_4 \bigwedge a_5$, then consequence of this chain is $a_1 \bigwedge \neg a_2 \bigwedge a_3 \bigwedge a_4 \bigwedge a_5$). But this combination does not do behavior refinement in the form of stimulus specialization and response generalization, which is required to eliminate cycles. Hence cycles persist (though the lengths of cycles defined in terms of number of behaviors involved change).

In conditional combination, set of k behaviors can be combined into a single behavior whose stimulus is a disjunction of stimuli of individual behaviors and consequence is a disjunction of consequences of individual behaviors. Even this does not prevent the cycles from occurring. These kinds of reorganization or restructuring of existing behaviors really does not solve the knowledge level problem. Hence the result.□

7 What do these results mean?

One important question that this work raises, however, deals with the entire problem of distributed knowledge representation. The key difference between DAI and the situated 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 arise, as Durfee so elegantly stated in his AAAI-92 invited lecture, in much the same issues of social interaction that arise between human beings, *What your computer really needs to know, you learned in kindergarten.* These include 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 situated systems, 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. 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 situated 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 [14]. 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.

One of the approaches to this problem, suggested originally on 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 robot behavior systems using some form of higher-level mediation as in [6], and there is also considerable DAI literature using global mediation strategies. 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,* offer a compromise by employing a reactive system for low-level control and a planner for higher level decision making as in [11].

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 are crucial. These have been considered in the work by Simmons in his Task Control Architecture [13].

The other question that arises is that of the representation. Could it be that the symbolic logic used in situated agents cause 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 [14] for a guide. A quick look at the problem here, however, shows this to be unimportant. 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 as in [2] 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 reduce 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. Thus, the conclusions of this work, since they reflect underlying properties of the problem and not the representation, and remain valid across different representational schemes.

One other question that our results open up is that of the *task limitations* of situated systems. While the probability of conflicts will increase with more pow-

erful behaviors, it may be possible, with much fine-tuning, to design systems that work well even for complex tasks. Can we set some limitations on the tasks themselves? This would be a stronger result, no doubt. However, distinguishing this class of tasks seems to be a difficult problem.

Wilkins describes a program called PARADISE (pattern recognition applied to directed search) in [15] that finds moves in chess in tactically sharp middle game positions using a large body of production rules. Wilkins claims that human masters whose play is still much better than the best programs seem to have a huge number of stored patterns and analyze position by matching these patterns to suggest plans for attack or defense. Such knowledge should be incrementally modified and expanded. To achieve this, PARADISE has a knowledge base of largely independent rules. Hence we can view PARADISE and similar chess players as being situated agents that have a modular structure. PARADISE also carries out static analysis to produce plans that help in focusing on critical part of new position, making it similar to situated agents having a hybrid computational structure that contains a planner and a reactor.

[12] describes CHINOOK, an automated checkers player that derives its power from perfect information on all positions with seven pieces or less. CHINOOK's evaluation function has 25 heuristic components, weighted sum of which gives a positional evaluation. These heuristic components are like agents whose output is combined by a meta-level agent. CHINOOK has been reported to have stuck in deadlock, being unable to make next move. [3] describes a backgammon program which uses some pre-computed tables and heuristic functions, rather than carrying out search (backgammon domain involves 10^{20} positions). It carries out more computation for evaluating each alternative in current position rather than doing brute force search and can be viewed as being a situated agent. Hence our results on modularity apply to a number of game players.

8 Conclusion

Our analysis assumes minimal communication among agents, that occurs largely through the world. Real situated agents however allow "finite inter-agent communication" while still maintaining modular structure. However our arguments that use cyclicity to challenge the assumption of modularity continue to apply to agents with finite communication bandwidth, since this bandwidth cannot eliminate internal state. Having finite communication just means that some modules can maintain internal state that can be queried by other modules to eliminate cycles. In chess playing, certain board patterns correspond to certain moves. Some moves can be used more frequently than others. The moves can be looked upon as situated agents that modify board patterns creating different consequences. Hence our results have implications not only for the future of situated agency, but the design of chess players as well.

References

[1] **Anderson, T. L. & Donath, M.** Animal Behavior As A Paradigm For Developing Robot Autonomy, Robotics and Autonomous Systems, 6(1 & 2): 145-168, 1990.

[2] **Arkin, R. C.** Behavior-Based Robot Navigation for Extended Domains, Adaptive Behavior 1(2): 201-225, 1992.

[3] **Hans Berliner**, Backgammon computer program beats world champion, Artificial Intelligence 14, 205-220, 1980.

[4] **Brooks, R. A.** Intelligence without representation, Artificial Intelligence, 47(1-3): 139-159, 1991.

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

[6] **Gat, E.**, On the Role of Stored Internal State in the Control of Autonomous Mobile Robots, AI Magazine, 14(1): 64-73, 1993.

[7] **Hartley, R. & Pipitone, F.** Experiments with the subsumption architecture, In Proceedings of the IEEE Conference on Robotics and Automation, 1652-1658, 1991.

[8] **Kirsh, D.** Today the earwig, tomorrow man?, Artificial Intelligence, 47(1-3): 161-184, 1991.

[9] **C. Ronald Kube & Hong Zhang.** Collective Robotics: From Social Insects To Robots. Adaptive Behavior, Vol. 2, No. 2, 189-218, 1994.

[10] **Mali, Amol D., & Amitabha Mukerjee**, Robot Behavior Conflicts: Can Intelligence Be Modularized?, Proceedings of AAAI-94, Seattle, 1279-1284.

[11] **Payton, D.W., Rosenblatt J. K., & Keirsey, D. M.**, Plan guided reaction, IEEE Transactions on Systems, Man and Cybernetics, 20(6): 1370-1382, 1990.

[12] **Schaeffer J., Treloar N., Lu P. and Lake R.**, Man versus machine for the world checkers championship, AI Magazine 14(2): 28-35, 1993.

[13] **Simmons, Reid**, Structured control for autonomous robots, IEEE Trans. Robotics & Automation, February 1994.

[14] **Vera, Alonso H., and & Herbert A. Simon**, Situated Action: A Symbolic Interpretation, Cognitive Science 17: 7-48, 1993.

[15] **Wilkins D. E.**, Using patterns and plans in chess, Artificial Intelligence 14, 165-203, 1980.