Robot Behaviour Conflicts: Can Intelligence be modularized?

Amol Dattaraya Mali and Amitabha Mukerjee
Center for Robotics
I.I.T. Kanpur 208016

Abstract

In this paper, we examine the modularity assumption of behaviour-based models: that complex functionalities can be achieved by decomposition into simpler behaviours. In particular we look at the issue of conflicts among robot behaviour modules. The chief contribution of this work is a formal characterization of temporal cycles in behaviour systems and the development of an algorithm for detecting and avoiding such conflicts. We develop the mechanisms of stimulus specialization and response generalization for eliminating conflicts. The probable conflicts can be detected and eliminated before implementation. However the process of cycle elimination weakens the behaviour structure. We show how (a) removing conflicts results in less flexible and less useful behaviour modules and (b) the probability of conflict is greater for more powerful behaviour systems. We conclude that purely reactive systems are limited by cyclic behaviours in the complexity of tasks they can perform. [1]

1 Introduction

Complex robot interactions are conveniently modeled in terms of stimulus-response sequences often called behaviours. It is easier to model and debug the *behaviour modules* as opposed to the larger and more integrated centralized controllers. Impressive results have been achieved using this strategy in a can collection robot (Connell 1990), navigation of mobile robot (Arkin 1992), a prototype airplane controller (Hartley & Pipitone 1991), office rearrangement robot in AAAI-93 robot competition etc. This behaviour-based intelligence paradigm propounded by Brooks and others has challenged the role of representation in AI. In a sense, these approaches treat the world as an external memory from which knowledge can be retrieved just by perception. Brooks argues that when intelligence is approached in such an incremental manner, reliance on representation disappears (Brooks 1991).

In response, traditional AI researchers such as Kirsh have argued that control cannot serve as a complete substitute for representation (Kirsh 1991). At the same time, behaviour systems have also been moving away from the purely reactive paradigm. A well-known extension of the Brooks' approach includes the SONAR MAP (Brooks 1986) which is a module that learns what looks suspiciously like a central representation. Some researchers (Gat 1993) are beginning to propose that the internal state be used, but only for modeling highly abstract information.

All modular designs (databases, architectures, factories) face the problem of intermodular conflict. In robot behaviour implementations, conflicts which do not result in cycles can be removed by prioritization schemes (e.g. suppressive links), but this is usually *ad hoc*, with the primary objective of demonstrating success in the current task objectives. Brooks stresses that additional layers can be added without changing the initial system - our results show that such a claim is most probably not tenable. Then how does one put the behaviour modules together and get useful performance? This depends on identifying the possible sources of inter-modular conflicts that are likely to arise in behaviour chains.

This paper is one of the first formal investigations on the issue of combining behaviours and inter-behaviour conflicts. Despite the attention such models have been receiving, the issue of inter-behaviour conflict, which challenges the fundamental assumption of modularity, has not been investigated. For example, if the consequence of a behaviour a triggers the stimulus for behaviour b and b precedes a, then we have an unending cycle. Such conflicts are also beginning to show up in the more complex behaviour implementations. For example, Connell records an instance where a can collecting robot attempts to

[1] AAAI-94, p.1279-1284

re-pick the can it has just deposited in the destination area as shown (Figure 1); this conflict was detected only after a full implementation (Connell 1990). Cyclical wandering and cyclic conflict of going back and forth between two obstacles have been reported (Anderson & Donath 1990) (Miller 1993). Can behaviour systems be constructed so that such conflicts can be detected beforehand? How can one modify the structure so as to avoid such conflicts? These are some of the questions we set out to answer.

Our analysis in this paper is dependent on a crucial observation regarding the temporal structure of purely reactive systems. The conflicts we are addressing are not control conflicts but temporal sequence conflicts for which it is necessary to define the temporal structure of behaviours, which is usually sequential since one behaviour usually provides the stimulus for another, so that there is often a clear temporal sequence in which behaviours are executed. In this paper we show that cycles occuring in this temporal sequence can be avoided only by modifying the behaviours themselves, and we introduce two such modifications, based on specializing the stimulus or restricting the action of a behaviour. One of the key results of the paper is that any such modification reduces the usefulness of the behaviour structure and makes it less flexible.

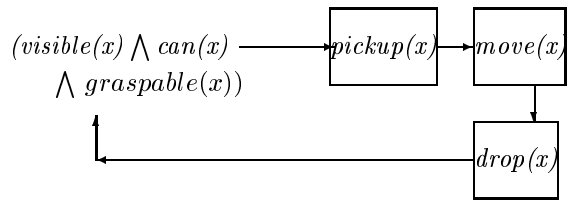

Figure 1. *Conflict in picking and placing the can.*

2 What Is a Behaviour?

AI researchers, psychologists, cognitive scientists, ethologists and roboticists, all use the term behaviour in senses that are related but are fundamentally different. At one end of the spectrum is Brooks who looks upon behaviours as a type of intelligent module, an input-output relation to solve small problems (Brooks 1986). Hopefully these modules can be combined to solve larger problems. There is no shared global memory. The stimulus to a behaviour is boolean and is tested by an *applicability predicate*. This is the model of behaviour investigated in this paper.

Minsky suggests thinking about goal directed behaviour as an output of a difference engine that measures the differences between the world state and the goal state and takes actions to reduce these differences (Minsky 1986). On the other hand, Simon feels that complex behaviour need not necessarily be a product of an extremely complex system, rather, complex behaviour may simply be the reflection of a complex environment (Simon 1969). Arkin proposes the *motor schema* as a model of behaviour specification for the navigation of a mobile robot (Arkin 1992).

Notation

Maes models a behaviour as a *4-tuple* <c, a, d, α> which represent pre-conditions, add list, delete list and level of activation respectively (Maes 1990). In this work we have followed the behaviourists and adopted a *3-tuple* model of behaviour: stimulus, action, consequence. An elemental behaviour module β takes the form <s, a, c>, although the action a is not directly referred to by us, and we sometimes abbreviate the notation to <s, c>. Both the stimulus s and the consequence c are commonly defined in terms of a predicate. We define the *dominant period* of a behaviour as that period when the behaviour is active. In most behaviour implementations, behaviours become dominant in a temporal sequence. We use the symbol ":" *(precedes)* to denote this. $\beta_1 : \beta_2$ implies that behaviour β_2 becomes dominant following behaviour β_1.

Behaviour Chain

We define a *behaviour chain* as a sequence of behaviour 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. If the consequence and stimulus include a finite universal state as well, then we can say that the stimulus s_{i+1} of the behaviour module β_{i+1} is logically implied by the consequence of the module β_i i.e. $(c_i \Rightarrow s_{i+1})$. What we mean by the finite universal state can be clarified by an example. Let *Universe* $= X \bigwedge Y \bigwedge Z$ and $c_1 = A$ and $s_2 = X \bigwedge A$. Then β_1 leads to β_2 but $(c_1 \not\Rightarrow s_2)$. Thus when we say that $(c_1 \Rightarrow s_2)$ we mean that a part of s_2 was true in the Universe and some literals in c_1 cause the rest of s_2 to come true. Thus in order for $(c_1 \Rightarrow s_2)$ to be true, both stimulus and consequence should always contain the "state of the universe predicate." This allows us to develop the argument effectively, skirting the philosophical debate on the frame problem (Georgeff 1987).

We define a *behaviour space* B as a set of behaviour modules. A temporal chain of behaviours C is said to be composable from B (written as $C \prec B$), if and only if $(C = ordered\,set\{\beta_i\} \bigwedge (\forall i)\, \beta_i \in B)$. A *stimulus space* Σ of a behaviour space B is the union of the stimuli of all behaviour modules in B.

Power, Usefulness and Flexibility of Behaviours

To compare different behaviour systems, we define a few relative measures.

Definition 1

• **Power:** A behaviour (β := <s, a, c>) is more powerful than (β^0:= <s^0, a^0, c^0>) iff ($s^0 \Rightarrow s$) $\bigwedge$ ($c \Rightarrow c^0$). In other words, it can be triggered at least as frequently as a less powerful behaviour and results in at least as strong a consequence. A behaviour space B is more powerful than the behaviour space B^0 if B^0 can be obtained from B by replacing some module $\beta \in B$ by less powerful module β^0.

• **Usefulness:** A behaviour space B spans the task space τ if and only if $\{\forall (t \in \tau)\ (\exists (C \prec B)\ fulfills\ (C, t))\}$. The *greatest fulfillable task space* $\tau_G(B)$ is the largest task space that is spanned by the behaviour space B. The *usefulness* of a behaviour space is defined as the ratio $\frac{|\tau_G(B)|}{|B|}$.

• **Flexibility:** A behaviour space B is at least as flexible as behaviour space B^0 if $\{\forall t \in (\tau_G(B) \cap \tau_G(B^0)) \exists (C \prec B)\{fulfills(C,t) \bigwedge \forall (C^0 \prec B^0) \{fulfills(C^0,t) \Rightarrow |C| \leq |C^0|\}\}\}$.

3 Detection of Conflicts

In the broad sense of the word conflict, any behaviour chain leading to non-fulfillment of the desired objectives can be said to have a conflict. Let a chain $C = \{\beta_1 : \beta_2 : ... : \beta_n\}$ be the desirable behaviour sequence that achieves a desirable outcome. There are three types of conflicts that can cause the chain C from not being executed, by breaking the sequence $\beta_i : \beta_{i+1}$.

Definition 2

(a)**Extraneous behaviour Conflict:** $\beta_i : \beta^0$, $\beta^0 \notin$ C.

(b)**Cyclic Conflict:** $\beta_i : \beta_k$, $\beta_k \in$ C, k $\leq$ i. (discussed later)

(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 behaviour conflicts.

The type of behaviour that we are investigating is the *cyclic conflict*, where, both β_{i+1} and β_k may be triggered and clearly the triggering of β_k would lead to a cycle (Figure 2).

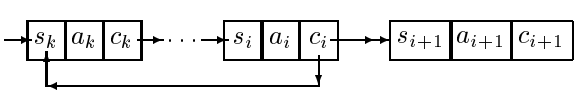

Figure 2. *Cycle in a temporal chain of behaviours.*

Terminated Cycles

All cycles do not result in a conflict. Let us say that a large block of butter needs to be cut into a number of smaller pieces. A module *cut* achieves it. Let $s_{Cut} = butter(x) \bigwedge cuttable(x) \bigwedge breadth(x,b) \bigwedge (b \geq 2\lambda)$. We specify the smallest acceptable size of the piece of butter by specifying the limit λ, which introduces the termination condition.

Prioritization

There are three types of prioritization used in robot behaviour control. If $\alpha : \beta$ is a possible but undesirable sequence of behaviours, the sequence can be modified. In *suppression*, a module γ suppresses the output of β, and instead of $\alpha : \beta$, $\alpha : \gamma$ occurs. In *inhibition* 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 occur, if the stimulus for β is not affected by γ. *Delayed action* is a special case of inhibition, where the inhibitive 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). These mechanisms are not guaranteed to kill the stimulus of β_k, hence β_k may be active after dominant period of the suppressing module is over. Thus within the scope of the three prioritization schemes discussed here, it is not possible to guarantee that cyclic conflicts will be avoided.

Detecting Cycles in Behaviour Graphs

Representing a behaviour chain as a graph, we present without proof the following lemmas:

Lemma 1(a). Whenever there is a cyclic conflict, there is a cycle in the temporal graph of behaviours.

Lemma 1(b). Whenever there is a cycle in the temporal graph of behaviours that is not terminated by a recursive condition, there is a cyclic conflict.

Thus detecting conflicts in a behaviour chain is equivalent to detecting cycles in the corresponding graph.

4 Behaviour Refinement

From definition 2, whenever there is a cycle in a behaviour chain $C = \{\beta_1 : \beta_2 ... \beta_k ... \beta_i : \beta_{i+1} ... \beta_n\}$, there must be a triggering of the kind $\beta_i : \beta_k$, where $k \leq i$. Then both $\beta_i : \beta_{i+1}$ and $\beta_i : \beta_k$ are possible at this point. Our objective is to break the $\beta_i : \beta_k$ link without disturbing the $\beta_i : \beta_{i+1}$ or $\beta_k \Gamma_1 : \beta_k$ triggerings which are essential to the successful execution of the chain. We have seen that priority based methods are not guaranteed to achieve this, so we look for behaviour modification approaches which will maintain ($c_i \Rightarrow s_{i+1}$) 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.

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 we would only need a small recency memory (recently dropped can). Thus 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. $(c_k\Gamma_1 \Rightarrow s_k)$ should still hold but $(c_i \Rightarrow s_k)$ must be broken. Clearly this will not be possible where $(c_i \Rightarrow c_k\Gamma_1)$, then any changes we make in s_k such that $\neg(c_i \Rightarrow s_k)$ will also result in $\neg(c_k\Gamma_1 \Rightarrow s_k)$. Thus stimulus specialization can be used only if $(c_i \Rightarrow c_k\Gamma_1)$ is not true. One model for this is to say that there must be a literal γ such that $(c_k\Gamma_1 \Rightarrow s_k \bigwedge \gamma)$ but $\neg(c_i \Rightarrow s_k \bigwedge \gamma)$. The conjunction of all such literals $\Gamma = (\gamma_1 \bigwedge \gamma_2 \bigwedge ... \gamma_m)$ is called the maximal difference between $c_k\Gamma_1$ and c_i. Stimulus specialization works only when $\Gamma \neq \emptyset$, and involves modifying s_k to $(s_k \bigwedge \gamma)$, $\gamma \in \Gamma$. It is advisable not to specialize it more than necessary (e.g. by adding more than one literal), since this adversely affects the power of the behaviour. A simpler understanding of the process is obtained if both c_i and $c_k\Gamma_1$ are in conjunctive form. Then Γ is nothing but the difference $(c_k\Gamma_1 - c_i)$ and s_k is modified by conjunction with one of the literals that is in $c_k\Gamma_1$ but not in c_i. Note that since the stimulus is specialized, any stimuli that are required by the action are still available to it.

Response Generalization

Here the action is modified so that the consequence of the action is weaker i.e. if the old consequence was c and the new one is c^0 then $(c \Rightarrow c^0)$ but $\neg(c^0 \Rightarrow c)$. For example, we can modify the action of the module *drop* so that while dropping the can on the ground, the robot puts it in an inverted position which prevents the robot from detecting that the object is a can. The original consequence was *(visible(x)* $\bigwedge$ *can(x)* $\bigwedge$ *graspable(x))* and the modified consequence is *(visible(x)* $\bigwedge$ *graspable(x))*, assuming the sensors cannot identify cans that have been inverted. Otherwise, we may modify the consequence by covering the can to make the predicate *visible(x)* false, then this leads to addition of a new behaviour 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. In fact, one could characterize the process of response generalization by saying that there must exist σ s.t. $(c_i \bigvee \sigma) \Rightarrow s_{i+1}$ but $\neg(c_i \bigvee \sigma) \Rightarrow s_k$. The disjunction of all such σ's is Σ. Again, if s_k and s_{i+1} are in conjunctive form, then a simpler understanding is obtained, since $\Sigma = \sim(s_k - s_{i+1})$ i.e. the negation of all the conjunctions that appear in s_k and not in s_{i+1}. This negation is a disjunction of many negative literals $(\sim\gamma)$. In this instance, modifying c_i is better understood as dropping the literal γ already appearing in c_i, written as $(c_i - \gamma)$. Since stimuli/consequences are often conjunctive, this difference notion is a useful concept in practice. Thus c_i is modified to $(c_i - \gamma)$, where $\gamma \in (s_k - s_{i+1})$.

Stimulus specialization is easier to do than response generalization, as response generalization requires that the action should be modified. However, stimulus specialization may not always be possible; e.g. with "not-deposited-just-now(x)" the robot may still pick up an older deposited can. Better solutions to this, such as "never-deposited-before(x)" or "not-at-depository(x)" would require considerable global memory. Therefore, stimulus specialization, while cheaper to implement, may not be available in many instances, since the actions require a minimum stimulus, and specializing it without memory may be counter-productive.

Effects Of Behaviour Refinement

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

Lemma 2. Removing a cycle from chain C by stimulus specialization or response generalization cannot introduce a cycle in any other chain C^0, that did not have a cycle before.

Proof :- Let β_k be the behaviour that was specialized. Now to introduce cycles when no cycles existed before, some link β^0: β_k must have become possible, i.e. $(c^0 \Rightarrow s_k)$ has become true. This is not possible since $c^0$is the same and s_k is more specific. Similarly, since c_k has not been modified, no new links $\beta_k : \beta^0$ could have been created. Hence no new cycle will be initiated. Similarly it can be shown that response generalization does not introduce new cycles. 2

Lemma 3. Whenever a behaviour space is modified to eliminate cyclic conflicts by response generalization and/or stimulus specialization, either the flexibility or usefulness of the behaviour space decreases.

Proof :- Let Σ be the stimulus space and s $\subset \Sigma$ and s is a conjunction of its members. Now let s be specialized to s^0so that $s^0 \subset s$. Now tasks or subsequent behaviours requiring the predicates in (s - s^0) will no longer be attended to by *B*. Thus we need a new behaviour β^{00}such that $(s^{00} \cup s^0)$ = s, so that β and β^{00}

together serve the stimulus set s which implies that |B| increases and the usefulness of B decreases. Similarly, if the response of β is generalized so that c has more literals than c^0. Thus $(c - c^0)$ is not being performed by β. Hence other new behaviours are needed to complete tasks requiring (c - c^0) which increases |B|, and also increases the chain lengths for performing tasks, reducing flexibility. Otherwise, some tasks requiring (c - c^0) cannot be performed which implies that $|\tau^0| < |\tau|$ which again means that the usefulness of the behaviour space decreases. 2

Let us say that we have a behaviour β whose consequence $c = p \bigwedge q$ leads to a cycle. If we use response generalization, we may have to design two behaviours β^0 and β^{00}such that $c^0 = q$ and $c^{00} = p$. If β has a stimulus $s = p \bigvee q$ which is triggered leading to a cycle and if we use stimulus specialization, we may have to design two more behaviours β^0 and β^{00}such that $s^0 = p$ and $s^{00} = q$. 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, which have to do with the power and usefulness of behaviour spaces.

Behaviour Modification Theorem. Given two behaviour spaces B and B^0such that B is more powerful than B^0(i.e. B^0can be obtained from B by replacing some behaviours β of B by the less powerful ones β^0) then:
(a) The greatest fulfillable task space for behaviour space $B^0$is less than that for B, i.e.
$|\tau_G(B^0)| \leq |\tau_G(B)|$
(b) Usefulness of B is greater than that of B^0i.e.
$\frac{|\tau_G(B)|}{|B|} \geq \frac{|\tau_G(B^0)|}{|B^0|}$.
(c)Probability of a cycle is also greater in B.
Proof (a) :- First, let us consider the case where a single behaviour β has been replaced by the less powerful β^0. The set of chains of behaviours composable from a behaviour 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 task. The greatest fulfillable task space is proportional to the total size of this tree of behaviour chains. Now, either the behaviour β will have more applicability due to smaller stimulus length as compared to the behaviour β^0, or the behaviour β will have stronger consequences resulting in more behaviours being triggerable. 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 is higher in B than in B^0 and the size of the task tree will be as large or larger. Since |B| has not changed, the usefulness of the behaviour space, $\frac{|\tau_G(B)|}{|B|}$ has decreased which proves part (b). This treatment can be extended to multiple instances of replacing a strong behaviour by a weak one. 2
Proof (c) :- Let $\beta_i \in B$ and $\beta_i^0 \in B^0$be two behaviours s.t. β_i is more powerful than β_i^0, i.e. $(s_i^0 \Rightarrow s_i)$ or s_i is weaker than s_i^0. Now consider any chain composable in B and B^0 of n modules, which differ only in that the module β_i is replaced by β_i^0. Now consider all behaviours $\beta_j \in C$, $j \geq i$, with consequence c_j. The probability of a cycle *prob-cycle (B)* is $\sum_{j}^{n}{}_{\cup i}\ prob(c_j \Rightarrow s_i)$ and *prob-cycle* (B^0) is $\sum_{j}^{n}{}_{\cup i}\ prob(c_j \Rightarrow s_i^0)$. Clearly, since $(s_i^0 \Rightarrow s_i)$, $\{\forall j[prob(c_j \Rightarrow s_i) \geq prob(c_j \Rightarrow s_i^0)]\}$. Similarly $(c_i \Rightarrow c_i^0)$ for which similar analysis can be carried out. Thus *prob-cycle (B)* $\geq$ *prob-cycle* (B^0). 2
Corollary :-If B and B^0have the same greatest fulfillable task space τ_G, but $\exists(\beta \in B) \bigwedge \exists(\beta^0 \in B^0)\{\beta$ is more powerful than $\beta^0\}$, but $\sim [\exists(\beta \in B) \bigwedge \exists\ (\beta^0 \in B^0)\{\beta^0$ is more powerful than $\beta]$, then $|B| \leq |B^0|$.

Residual

In this section we consider the parsimony of the logical chain underlying the behaviour chain. If $\beta_i : \beta_{i+1}$, then $(c_i \Rightarrow s_{i+1})$. There may be some literals in c_i which are not necessary in this implication, or there may be some disjunctive literals in s_{i+1} not referred to by c_i. We call this difference between c_i and s_{i+1}, the *residual*. If c_i and s_{i+1} are identical, then their residual is null. If θ is the most general matching string between c_i and s_{i+1}, i.e. θ is the most general construct for which $c_i \Rightarrow \theta$ and $\theta \Rightarrow s_{i+1}$, then we can write $c_i = \theta \bigwedge \gamma, s_{i+1} = \theta \bigvee \sigma$, then the residual consequence $= \gamma$, the remedial stimulus $= \sigma$ and the total residual between (β_i, β_{i+1}), R_i, is defined as $\gamma \bigwedge \neg\sigma$. Residuals are a measure of the degree of *coupling* in a behaviour chain. The intuition behind this is that as the residual increases, the flexibility as well as probability of cycles increase. Stimulus specialization as well as response generalization decrease the residual.

Lemma 4. If two chains C and C^0 have identical residuals except for some residuals R in C which are stronger than corresponding R^0in C^0, then the probability of a cycle is greater in C.
Proof :- Consider two behaviour chains C and C^0 where $C^0$is formed from C by replacing the single behaviour β_i by β_i^0. Then all residuals in the two chains are also identical except $R_i\Gamma_1$ and R_i. If the residuals in C are stronger, then $(\gamma_i \Rightarrow \gamma_i^0)$ and $(\sigma_i\Gamma_1 \Rightarrow \sigma_i\Gamma_1)$, i.e. behaviour β is more powerful than β^0. Hence by part (c) of the behaviour modification theorem, probability of a cycle is greater in C than in C^0. The same

arguments can be extended for multiple behaviour changes between C and C^{0}. 2

Conclusion

In this paper we have focussed on the *temporal* relations between behaviours as opposed to the control relations. This has highlighted an important similarity between behaviour-based modeling and the classical models of planning in AI. The effect of actions, which become new stimuli is similar to the postcondition - precondition structure in means ends planners such as STRIPS (Georgeff 1987). One of the approaches used to avoid cyclic conflicts in planning is the meta-level reasoner, idea which has also been used in behaviour systems such as in (Arkin 1992). But purists would not consider these to be true reactive behaviour systems. However, behaviour models differ from planning in some crucial aspects. Locality of behaviour programming makes opportunistic plan-generation automatic, since the relevant behaviour is triggered automatically when stimulus becomes very strong. Also, cycles are much more of a problem in behaviour models since unlike planners, a behaviour does not "switch-off-and-die" after execution; if the stimulus reappears, it may re-execute, causing a cycle.

One of the benefits of this work is that by testing for cycles, the designers will not have nasty surprises awaiting them after implementation. We also show that approaches such as prioritization will not avoid cycles. Thus the only guaranteed method for avoid cycles is to modify the behaviour itself, and this can be done either by specializing the stimulus or generalizing the response of some behaviour module. Unlike learning, which makes the behaviours more powerful, this reduces the usefulness of the behaviour module. If a robot can pick up a soda can, it should be able to pick up a coffee cup or other similar object. Using stimulus specialization, such a general behaviour would be split into many separate behaviour for picking up separate objects. The principal insight to be gained from this discussion is that in behaviour design, there is a tradeoff between the power of a behaviour and the likelihood of cycles. The crucial task of the behaviour designer is to achieve just the right amount of refinement, without involving conflicts and without sacrificing too much flexibility.

Can conflicts be avoided by using alternate architectures such as fuzzy logic (which allows behaviour designers to model *strength of stimulus*), meta-level reasoning (Yamauchi & Nelson 1991), or connectionist architectures (Payton, Rosenblatt & Keirsey 1990)? If we ported the can-pickup example into any of these representations, the conflict does not go away, since the conflict arises at the knowledge level and not at the representation level. Using internal state would not, in itself, be able to remove this type of conflict, although it would make it easier to modify the behaviours so that the conflict can be avoided. Another issue related to internal state is the intention of the robot (psychologists read *will*). Knowing the intention at some meta-level, it may be possible to construct tests for detecting conflicts, and even possibly of avoiding them. At the same time, models involving will or intention (as in Searle) are one of the most debated and difficult quagmires in AI today. Is there then some limit on the complexity of a system of behaviours before self-referential cycles develop? A deeper question raised by the presence of such cycles in behaviour based robotics, as well as in other branches of AI, is that of its significance to the entire search for artificial intelligence. Is there some bound on the complexity of *any* system claiming intelligence, before it begins to develop cyclic conflicts? This paper is a beginning of the search for these answers which are sure to affect the future of the behaviour-based robot modeling paradigm in particular and that of models for intelligence in general.

References

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

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

[3] Brooks, R. A. 1986. A robust layered control system for a mobile robot, IEEE transactions on robotics and automation, 2(1): 14-23.

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

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

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

[7] Georgeff, M. P. 1987. Planning, Annual Review of Computer Science, 2: 359-400.

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

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

[10] Maes, P. 1990. Situated Agents Can Have Goals, Robotics and Autonomous Systems, 6(1-2): 49-70.

[11] Miller, D. P. 1993, A Twelve-Step Program to More Efficient Robotics, AI Magazine, 14(1): 60-63.

[12] Minsky M. L. 1986. The Society of Mind, Simon and Schuster.

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

[14] Simon, H. A. 1969. The Sciences of the Artificial, The MIT Press.

[15] Yamauchi, B.; Nelson, R. 1991. A behaviour-based architecture for robots using real-time vision, Proceedings of the IEEE Conference on Robotics and Automation, 1822-1827.